

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

Приложение обработки и анализа
индивидуальных заказов на изготовление
лекарственных средств

Студент _____ Антипова Мария Максимовна
(подпись, дата) (ФИО полностью)

Руководитель _____ Шибанов С. В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«25» декабря 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Антипова Мария Максимовна гр. 20ВИ2 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы «Приложение обработки и анализа индивидуальных заказов на
изготовление лекарственных средств»

Тема утверждена приказом ПГУ № _____ от " ____ " _____ 20 ____

2. Руководитель работы _____ к.т.н., доцент кафедры «МО и ПЭВМ» Шибанов С. В.

3. Задание на работу (назначение разработки, исходные данные и т.п.) _____

Назначение: приложение для обработки и анализа индивидуальных заказов на
изготовление лекарственных средств

Средства выполнения: Операционная система Windows

Средства разработки: Visual Studio 2022, Framework 4.8, .NET

Язык программирования: C#

Технологии разработки: ООП, UML

4.	Перечень	подлежащих	разработке
вопросов			
<u>Анализ предметной области и постановка задачи на разработку приложения</u>			
<u>Проектирование приложения</u>			
<u>Разработка программного обеспечения</u>			
<u>Контроль качества программного обеспечения</u>			
<u>Оформление ПЗ</u>			

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи на разработку приложения	10%	08.04.2024 – 13.04.2024	
2	Проектирование приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств	20%	15.04.2024 – 25.04.2024	
3	Разработка программного обеспечения	30%	26.04.2024 – 12.05.2024	
4	Контроль качества программного обеспечения	20%	13.05.2024 – 22.05.2024	
5	Оформление пояснительной записки	20%	23.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 318 листов, 61 таблица, 44 рисунка, 29 использованных источников и 9 приложений.

УПРАВЛЕНИЕ ЗАКАЗОМ, СОСТАВ ЗАКАЗА, БД, СУБД, ИЗГОТОВЛЕНИЕ ЛЕКАРСТВА, ОФОРМЛЕНИЕ ЗАКАЗА, MYSQL, РЕЦЕПТ ПРЕПАРАТА, UML, C#.

Объектом разработки является процесс обработки и анализа индивидуальных заказов на изготовление лекарственных средств.

Цель работы: разработать приложение для автоматизации обработки и анализа индивидуальных заказов на изготовление лекарственных средств.

Технологии разработки: Visual Studio 2022, MySQL, C#.

В результате выполнения бакалаврской работы было разработано программное обеспечение для автоматизации обработки и анализа индивидуальных заказов на изготовление лекарственных средств.

Область применения разработки: для автоматизации ведения учёта индивидуальных заказов на изготовление лекарственных средств может быть применено в аптеках или организациях, занимающихся заказами изготовления лекарственных средств.

					ПГУ 09.03.02–8БР202.02 ПЗ			
Из м.	Лист	№ докум.	Подпись	Дата				
Разраб.		Антипова М.М.			«Приложение обработки и анализа индивидуальных заказов на изготовление лекарственных средств» Пояснительная записка	Лит.	Лист	Листов
Провер.		Шибанов С.В.					4	318
Реценз						Группа 20ВИ2		
Н. Контр.		Такташкин Д.В.						
Утверд.		Козлов А.Ю.						

Содержание

Введение	8
1 Анализ предметной области и постановка задачи на разработку приложения.....	10
1.1 Анализ бизнес-процессов обработки индивидуальных заказов на изготовление лекарственных средств	10
1.1.1 Действующие лица процесса обработки и анализа индивидуальных заказов на изготовление лекарственных средств.....	10
1.1.2 Бизнес-процессы обработки и анализа индивидуальных заказов на изготовление лекарственных средств	12
1.1.3 Классы информационных объектов и документов в процессе обработки и анализа индивидуальных заказов на изготовление лекарственных средств.....	16
1.2 Сравнительный анализ приложений-аналогов	19
1.2.1 Цели и задачи сравнительного анализа	19
1.2.2 Выбор приложений-аналогов для анализа	20
1.2.3 Обзор приложений-аналогов	20
1.2.4 Результаты сравнительного анализа	26
1.3 Постановка задачи на разработку приложения.....	27
1.3.1 Цели разработки и назначение приложения	27
1.3.2 Классы и роли пользователей приложения.....	27
1.3.3 Автоматизируемые функции приложения	28
1.4 Планирование процесса разработки приложения.....	30
2 Проектирование приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств	34
2.1 Моделирование функциональных возможностей приложения	34
2.2 Разработка концептуальной модели предметной области приложения..	43
2.3 Разработка программной архитектуры приложения	47

2.4	Разработка прототипа пользовательского интерфейса	50
2.5	Разработка требований к аппаратно-программному окружению	54
2.6	Обоснование выбора средств разработки	55
2.6.1	Обоснование выбора средств автоматизации	55
2.6.2	Обоснование выбора инструментальных средств разработки программного обеспечения	56
2.6.3	Обоснование выбора языковых средств разработки	57
2.6.4	Обоснование выбора средств управления данными	58
3	Разработка программного обеспечения	61
3.1	Разработка физической схемы базы данных	61
3.2	Разработка пользовательского интерфейса приложения	71
3.3	Разработка классов приложения	81
3.4	Разработка методов доступа к базе данных	83
3.5	Разработка запросов в базе данных	85
4	Контроль качества программного обеспечения	88
4.1	Планирование процесса контроля качества программного обеспечения	88
4.1.1	Мероприятия по контролю качества программного обеспечения	88
4.1.2	Обоснование выбора модели тестирования	89
4.1.3	Обоснование выбора метрик программного кода	91
4.2	Тестирование программных средств	93
4.2.1	Тестирование белым ящиком	93
4.2.2	Модульное тестирование	100
4.3	Оценка качества программного кода на основе метрик	103
4.3.1	Метрика SLOC	103
4.3.2	Метрика Холстеда	105
4.3.3	Метрика Чепина	107
	Заключение	109
	Список использованных источников	110
	Приложение А (справочное) Глоссарий	113

Приложение Б (рекомендуемое) Диаграмма вариантов использования приложения.....	117
Приложение В (рекомендуемое) Сценарии бизнес-процессов приложения	121
Приложение Г (рекомендуемое) Физическая модель предметной области	126
Приложение Д (рекомендуемое) Листинг SQL скриптов	128
Приложение Е (рекомендуемое) Листинг программы	135
Приложение Ж (рекомендуемое) Диаграмма последовательности предметной области	305
Приложение И (рекомендуемое) Результаты тестирования приложения	307
Приложение К (рекомендуемое) Листинг класса тестирования	315

Введение

Учет индивидуальных лекарственных средств ведется путем регистрации заказанных лекарственных средств по квитанциям или в Рецепттурном журнале за соответствующий отчетный период. За одно лекарственное средство, изготовленное в аптеке в порядке внутриаптечной заготовки и фасовки, принимается лекарство в единичной упаковке в готовом для отпуска виде, оформленное в соответствии с установленными правилами нормативных документов Минздрава России по оформлению лекарственных средств [1].

Изготовление лекарств возможно только в соответствии с промышленным регламентом, который разрабатывается на основе фармакопейных статей. Они в свою очередь содержат требования государственного стандарта качества лекарства к используемым реактивам, титрованным растворам, индикаторам, а также перечни нормируемых показателей (методов испытания) для конкретной лекарственной формы и физические, физико-химические, химические, биохимические, биологические и микробиологические методы анализа лекарственного средства.

Помимо требований к качеству, в промышленном регламенте содержатся сведения, важные для ведения бухгалтерского учета в части калькулирования себестоимости изготовления лекарства и ценообразования при его отпуске потребителю. А именно сведения об ингредиентном составе конкретного лекарства, а также о технических параметрах, сопутствующих его изготовлению (характеристики тары (упаковки), нормы технологических потерь).

При организации изготовления лекарств, помимо требований Государственной фармакопеи, следует руководствоваться и другими положениями действующего законодательства [2].

В данной работе объектом автоматизации является процесс обработки и анализа индивидуальных заказов на изготовление лекарственных средств.

В качестве объекта исследования будет рассмотрена организация обработки и анализа индивидуальных заказов на изготовление лекарственных средств.

Актуальность темы проекта обуславливается необходимостью автоматизации труда фармацевтов и провизоров, снижения временных и денежных затрат на выполнение стандартных рутинных операций. Практическая значимость работы определяется разработкой информационной системы, предназначенной для автоматизации организации ведения учёта индивидуальных заказов на изготовление лекарственных средств.

Целью данной работы является разработка приложения для организации обработки и анализа индивидуальных заказов на изготовление лекарственных средств.

Для достижения указанной цели в рамках выполнения бакалаврской работы необходимо выполнить следующие задачи:

- провести исследование предметной области;
- выполнить анализ бизнес-процессов обработки индивидуальных заказов на изготовление лекарственных средств;
- выполнить обзор и анализ существующих аналогов на рынке ПО;
- на основе анализа выполнить постановку задачи на разработку приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств;
- провести планирование бюджета разработки;
- определить функциональные возможности приложения;
- разработать приложение обработки и анализа индивидуальных заказов на изготовление лекарственных средств;
- выполнить тестирование разработанного ПО.

Практическая значимость разработки нового программного обеспечения для организации обработки и анализа индивидуальных заказов на изготовление лекарственных средств заключается в возможности внедрения данного программного продукта в медицинские учреждения.

В результате выполнения работы был реализован прототип программного обеспечения, который в дальнейшем будет дорабатываться и улучшаться.

1 Анализ предметной области и постановка задачи на разработку приложения

1.1 Анализ бизнес-процессов обработки индивидуальных заказов на изготовление лекарственных средств

1.1.1 Действующие лица процесса обработки и анализа индивидуальных заказов на изготовление лекарственных средств

Основными сотрудниками, которые выполняют функции по обслуживанию заказов лекарственных средств в аптеках и медицинских учреждениях, являются фармацевт и провизор. Рассмотрим подробнее их обязанности.

Деятельность фармацевтического работника оценивается по качеству, своевременности и полноте объема выполненных задач.

Основная обязанность фармацевта – обеспечить покупателя необходимым лекарственным препаратом и ответить на вопросы о его приеме и хранении в домашних условиях. Чтобы оставаться компетентным в своей области, работник аптеки должен постоянно углублять познания в фармакологическом деле, изучать актуальные нововведения и законы. Пройти курсы повышения квалификации можно в «Академии профессиональных стандартов». Обучение проходит с использованием дистанционных технологий, поэтому фармацевтам не нужно прерывать основную производственную деятельность.

В обязанности фармацевта компании входит:

- прием рецептов от покупателей;
- розничная торговля фармакологической продукцией, проведение кассовых операций;
- соблюдение правил хранения медицинских изделий и лекарственных средств, включая наркотические и психотропные препараты,

к работе с нервной системой (НС) и пищеварительным трактом (ПВ) фармацевт допускается распоряжением руководства аптеки;

- контроль сроков годности препаратов;
- прием фармацевтической продукции, ее распределение на места хранения и ведение соответствующих журналов, выкладка товара на витрины, маркировка отдельных групп лекарственных средств;
- регистрация показаний приборов контроля температуры и влажности в помещениях и холодильниках;
- инвентаризация, ведение предметно-количественного учета, определение потребностей аптеки и составление заявки провизору;
- консультации покупателей по выбору препарата, ведение санпросвет работы, в экстренных случаях – оказание доврачебной помощи;
- соблюдение санитарно-гигиенического режима, выполнение требований к личной гигиене и ношению спецодежды;
- четкое следование внутреннему распорядку труда, соблюдение правил эксплуатации оборудования и противопожарной безопасности, оповещение руководящего состава о выявленных проблемах;
- прохождение медосмотров и периодической аккредитации [3].

Провизор – это специалист с высшим профессиональным образованием, который может работать в частной или государственной аптеке, лечебно-профилактических учреждениях и научно-исследовательских центрах. В отличие от фармацевта со средним профессиональным образованием, провизор в аптечной организации имеет больше обязанностей и карьерных перспектив. Он может занять место заведующего отделом или менеджера по развитию торговой сети.

Основные обязанности провизора:

- контролирует качество поступающих аптечных товаров и препаратов, изготовленных фармацевтами, обеспечивает их хранение в соответствии с установленными правилами;

- принимает и ведет учет рецептов и требований лечебно-профилактического учреждения (ЛПУ) на лекарственные средства;
- отпускает препараты и медицинские изделия;
- изготавливает лекарства по индивидуальным рецептам;
- изымает невостребованные препараты по завершении срока хранения;
- информирует население и медицинских работников об ассортименте аптечных товаров и их применении;
- ведет учетно-отчетную документацию [4].

Потребителем и заказчиком лекарственных средств является клиент.

Основные обязанности клиента:

- оформление заказа на изготовление лекарственных средств;
- предъявление информации по рецепту;
- внесение предоплаты за заказ;
- оплата заказа.

В данном разделе были описаны задачи и функции, которые должны выполнять пользователи в соответствии с их должностными обязанностями, определенные в штатном расписании организации.

1.1.2 Бизнес-процессы обработки и анализа индивидуальных заказов на изготовление лекарственных средств

Проектирование диаграммы деятельности является важным этапом при разработке приложения. Диаграмма деятельности (Activity Diagram) является представлением алгоритмов неких действий (активностей), выполняющихся в системе.

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Диаграмма деятельности – это своеобразная блок-схема, которая описывает последовательность выполнения операций во времени. Их можно использовать для моделирования динамических аспектов поведения системы. Каждое состояние на диаграмме деятельности

соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии.

Диаграммы деятельности можно использовать на всех этапах разработки программного обеспечения и для различных целей. И поскольку они очень похожи на блок-схемы, они обычно более популярны, чем другие типы диаграмм UML [5].

В диаграммах деятельности используются пиктограммы «действие», «переход», «выбор» и «линии синхронизации». В языке UML действие изображается в виде прямоугольника с закругленными углами, переходы – в виде направленных стрелок, элементы выбора – в виде ромбов, линии синхронизации в виде горизонтальных и вертикальных линий.

Диаграмма деятельности представлена на рисунке 1.

После авторизации пользователя в системе ему доступен соответствующий его роли функционал.

Действующими лицами процесса обработки и анализа индивидуальных заказов на изготовление лекарственных средств являются:

- клиент;
- фармацевт;
- провизор.

Клиент может сделать заказ на изготовление лекарства в соответствии с информацией из рецепта, используя приложение, либо через фармацевта, и вносит предоплату. До получения заказа клиент может оформить отмену заказа. После того, как заказ будет готов к выдаче, клиент получает его у фармацевта, вносит оплату и получает чек об оплате.

Фармацевт принимает заказы у клиентов, работает с заказами клиента и отправляет заявку провизору на необходимые лекарственные средства. После готовности заказа фармацевт получает заказ, где клиент оплачивает и забирает его из пункта выдачи.

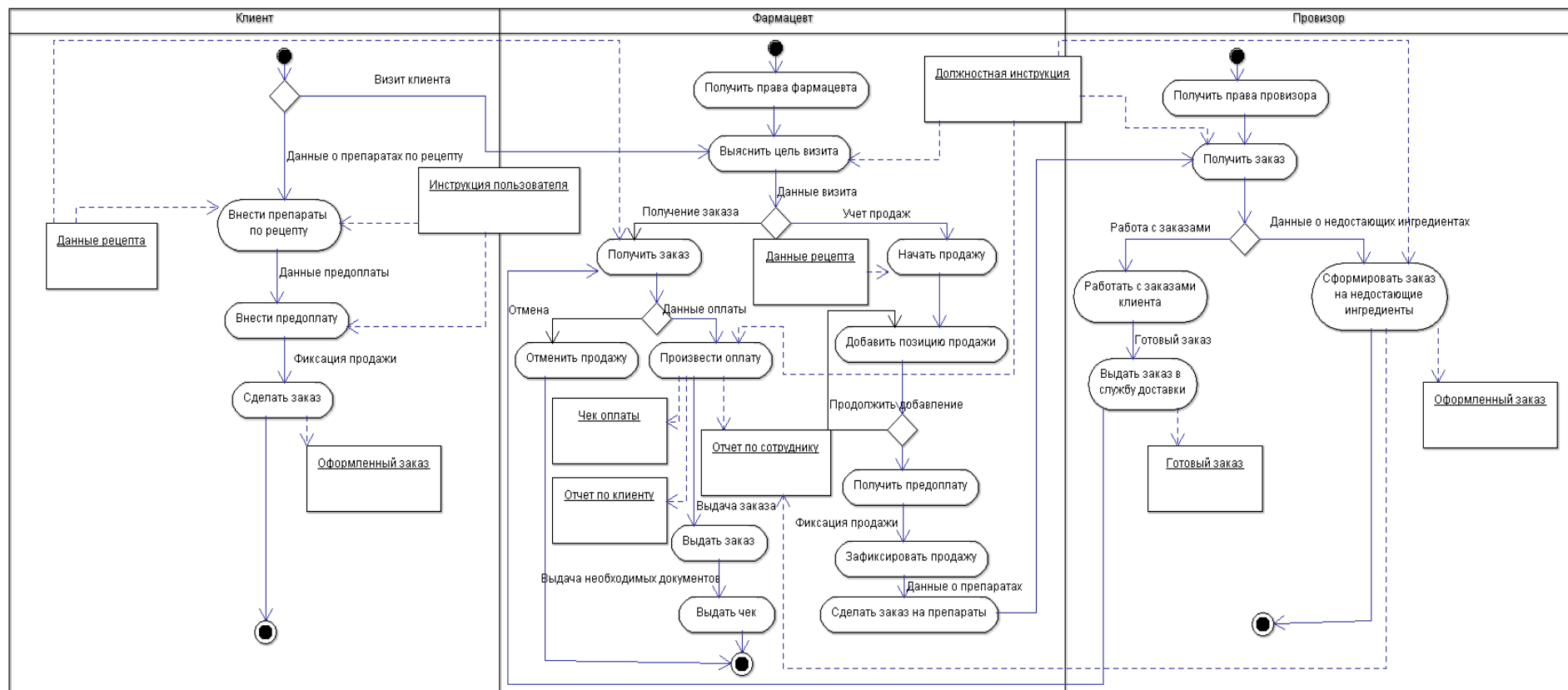


Рисунок 1 – Сценарий процессов выполнения функций пользователей (диаграмма деятельности)

Провизор работает с заказами клиентов и базой данных ингредиентов лекарственных средств, которые заказывает у поставщиков по мере необходимости. Изготовленные заказы провизор отправляет на пункт выдачи службой доставки, где фармацевт отдает заказ клиенту.

В процессе анализа предметной области были выявлены следующие основные бизнес-процессы:

- оформление заказа;
- оформление отмены заказа;
- изготовление заказа;
- оформление заявки поставщику.

Спецификации основных бизнес-процессов приведены в таблице 1.

Таблица 1 – Спецификации бизнес-процессов приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств

Бизнес-процесс	Исполнитель	Входные данные		Выходные данные		Ограничения
		Поставщик	Содержание	Поставщик	Содержание	
Оформление заказа	Фармацевт	Клиент	Информация из рецепта	Клиент	Оформленный заказ	Необходимо предоставить рецепт, внести предоплату
Оформление отмены заказа	Фармацевт	Клиент	Информация о заказе	Клиент	Подтверждение об отмене заказа	Отмену заказа необходимо сделать до получения препарата и полной оплаты
Изготовление заказа	Провизор	Фармацевт	Данные о заказе на изготовление	Провизор	Готовый препарат	Необходимо предоставить данные из рецепта об изготовлении и препарата
Оформление заявки поставщику	Провизор	Провизор	Информация о необходимых препаратах	Провизор	Полученные препараты	Необходимо посмотреть наличие препаратов

В ходе проектирования и разработки приложения будут реализованы все вышеуказанные бизнес-процессы.

1.1.3 Классы информационных объектов и документов в процессе обработки и анализа индивидуальных заказов на изготовление лекарственных средств

К классам информационных объектов предметной области относятся бумажные и электронные документы. К бумажным документам приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств относятся следующие документы, описание которых представлено в таблице 2.

Таблица 2 – Описание бумажных документов

Наименование документа	Описание документа	Атрибуты
Отчет по рецепту	На дату формирования отчета выгружается список рецептов по заказу клиента.	Информация, ФИО клиента, наименование файла рецепта.
Список ЛП в разрезе вида.	По указанному виду препарата выгружается список ЛП.	Название препарата, информация, цена, количество, название поставщика
Список ЛП в разрезе поставщика.	По указанному поставщику выгружается список ЛП, которые он поставил.	Название поставщика, информация, цена, количество
Отчет по клиенту	По выбранному клиенту из списка за указанный период дат выгружаются данные по заказам, которые он оформлял.	ФИО клиента, начало даты, конец даты, информация, сумма, наименование пункта выдачи, статус, ФИО работника
Отчет по сотруднику	По выбранному сотруднику из списка за указанный период дат выгружаются данные по заказам, которые он создал.	ФИО сотрудника, начало даты, конец даты, информация, сумма, наименование пункта выдачи, статус, ФИО клиента
Отчет по статусу заказов	По выбранному статусу из списка за указанный период дат выгружаются данные по заказам с этим статусом. Шаблон документа изображен на рисунке 2.	Наименование статуса, начало даты, конец даты, информация, сумма, наименование пункта выдачи, статус, ФИО работника, ФИО клиента

Продолжение таблицы 2

Наименование документа	Описание документа	Атрибуты
Отчет по оплатам	На дату формирования по выбранному заказу клиента выгружается список оплат.	Дата оплаты, сумма
Чек оплаты	Из таблицы с оплатами формируется чек оплаты. Шаблон документа изображен на рисунке 3.	Дата оплаты, сумма оплаты, код заказа, ФИО клиента

Дата: 10.6.2024

Отчет 'Информация по заказам (лекарства)'

Код	Дата(начало)	Дата(конец)	Клиент	Сумма	Статус	Пункт	Информация	Сотрудник
10	08.06.2024	08.06.2024	Иванов Н.Е.	5000	Открыт	Колбасный цех	Раствор от горла	Екин В.Д.
8	04.06.2024	04.06.2024	Макина И.Н.	6660	Открыт	Колбасный цех	Для лечения почек	Кумин Е.И.
7	01.06.2024	04.06.2024	Семенкина Е.А.	61800	Открыт	Солнечный	онкология 2 стадия	Кумин Е.И.
6	01.06.2024	08.06.2024	Макина И.Н.	4760	Открыт	Колбасный цех	Для лечения горла	Кумин Е.И.
1	28.05.2024	28.05.2024	Иванов Н.Е.	4500	Открыт	Солнечный	222	Мышин С.Н.

Рисунок 2 – Шаблон документа «Информация по заказам (лекарства)»

21.05.2024

Оплата по заказу
Код 3 (30-04-2024, Ефипова В.Г.)

Дата: 21-05-2024
Сумма: 800
Опатил: Ефипова В.Г.

Рисунок 3 – Шаблон документа «Чек оплаты»

К электронным информационным объектам относятся заказы на изготовление препарата и заказ поставщику для покупки необходимых ингредиентов для изготовления лекарства.

Для отражения данных по оформленному заказу клиентом на изготовление лекарственных средств используется печатная форма заказа. Шаблон печатной формы изображен на рисунке 4.

Дата: 21.05.2024

Заказ № 15

Состав заказа

№	Препарат	Описание	Стоимость
1	Порошок ППР-90-П	Для раствора	190
2	Порошок ЗФ-01-А	Для раствора	250

Данные клиента

Петров П.Н.
7-908-322-90-76
Документ: 45 09 354098

Данные рецепта

Врач: Николаев В.А. для лечение печени

Оформил

Курилов С.С.

Рисунок 4 – Шаблон ПФ «Заказ»

К классам информационных объектов также относятся следующие основные классы: Сотрудник, клиент, препарат, лекарство, рецепт, заказ. Описания классов информационных объектов и их основных атрибутов приведены в таблице 3.

Таблица 3 – Описание классов информационных объектов

Наименование	Описание	Атрибуты
Сотрудник	В этом классе прописываются данные о медицинских сотрудниках организации, а также их учетные данные	Фамилия, имя, отчество, логин, пароль, права.
Препарат	Данный класс содержит информацию о препаратах	Название препарата, цена, количество.
Лекарство	Хранится информация о лекарствах, которые изготавливаются в медицинской организации	Название лекарства, количество, цена.
Рецепт	В данном классе хранится информация о рецептах для изготовления лекарственных средств.	Информация, имя файла.

Продолжение таблицы 3

Наименование	Описание	Атрибуты
Заказ	Хранится информация о заказах клиентов	Сумма, информация, код клиента.

Рецепты на лекарственные препараты являются неотъемлемыми спутниками каждого фармацевтического работника. Несмотря на то, что оформление рецепта является обязанностью и ответственностью медицинского работника, провизорам и фармацевтам тоже необходимо знать порядок назначения лекарственных средств (ЛС), поскольку отпуск препарата по рецепту, оформленному с нарушением правил, может привести к административному штрафу. На рецепте указывается состав лекарственного препарата, лекарственная форма, обращение медицинского работника к фармацевтическому работнику об отпуске. Эта информация должна быть оформлена в родительном падеже на латинском или русском языке. Однако, электронные рецепты могут быть оформлены только на русском языке. Кроме этого, в рецепте обязательно прописывается способ применения ЛС, с указанием пути введения, дозы, частоты, времени приёма относительно сна (утром, на ночь) и его длительности, а для лекарственных препаратов, взаимодействующих с пищей, – времени их употребления относительно приёма пищи (до еды, во время еды, после еды). Способ применения должен быть указан на русском языке [9].

Все вышеуказанные классы информационных объектов и документов в процессе обработки и анализа индивидуальных заказов на изготовление лекарственных средств были разработаны в приложении.

1.2 Сравнительный анализ приложений-аналогов

1.2.1 Цели и задачи сравнительного анализа

Сравнительный анализ современных средств автоматизации обработки и анализа индивидуальных заказов на изготовление лекарственных средств позволит определить наиболее используемые системы, а также получить представление о том, насколько они распространены и актуальны.

Можно выделить следующие цели проведения анализа систем:

- выявление достоинств;
- выявление недостатков;
- выявление характерных базовых функций.

По итогам анализа главной задачей будет определение обязательных функций, а также уникальных возможностей, которые расширят функционал реализуемой системы и приведут к ее высокой конкурентоспособности.

1.2.2 Выбор приложений-аналогов для анализа

На сегодняшний день существует множество инструментальных средств, которые позволяют вести учет индивидуальных заказов на изготовление лекарственных средств. Приведем лишь некоторые из них:

- приложение «Больничная аптека»;
- приложение «Единая система учёта и контроля льготных рецептов»;
- приложение «Mobile SMARTS: МДЛП».

Среди представленных средств встречаются как бесплатные, так и платные, у которых возможно наличие on-line версий. В таблице 4 приведена данная информация.

Таблица 4 – Информация о приложениях

Название приложения	Доступность	Онлайн версия	Разработчик
Больничная аптека.	Платная.	НЕТ	1С
Единая система учёта и контроля льготных рецептов	Платная.	ДА	ЮНИКО
Mobile SMARTS: МДЛП	Платная.	НЕТ	Клеверенс

Для дальнейшего анализа были отобраны самые распространенные средства, которые обеспечивают возможность отслеживать индивидуальные заказы на производство лекарственных средств.

1.2.3 Обзор приложений-аналогов

Рассмотрим готовые решения для управления организацией работы с заказами на изготовление лекарственных средств в организациях, компаниях и

аптеках.

Приложение №1 «Больничная аптека». Разработчик «1С».

Продукт предназначен для детального аналитического учета аптечных товаров в больничной аптеке, отделениях и кабинетах, а также в розничных аптечных пунктах медицинской организации. Решение может использоваться как в государственном учреждении, так и в хозрасчетной медицинской организации [6].

На рисунке 5 приведен скриншот операции «Подбор товаров» в 1С.

Приложение поддерживает интеграцию с ФГИС «Мониторинг движения лекарственных препаратов» (МДЛП) в объеме, требуемом для больничных аптек и пунктов выдачи льготных лекарственных препаратов. Приложение позволяет регистрировать вывод из оборота лекарственных препаратов для медицинского применения в ФГИС МДЛП в соответствии с Федеральным законом 61-ФЗ «Об обороте лекарственных средств».

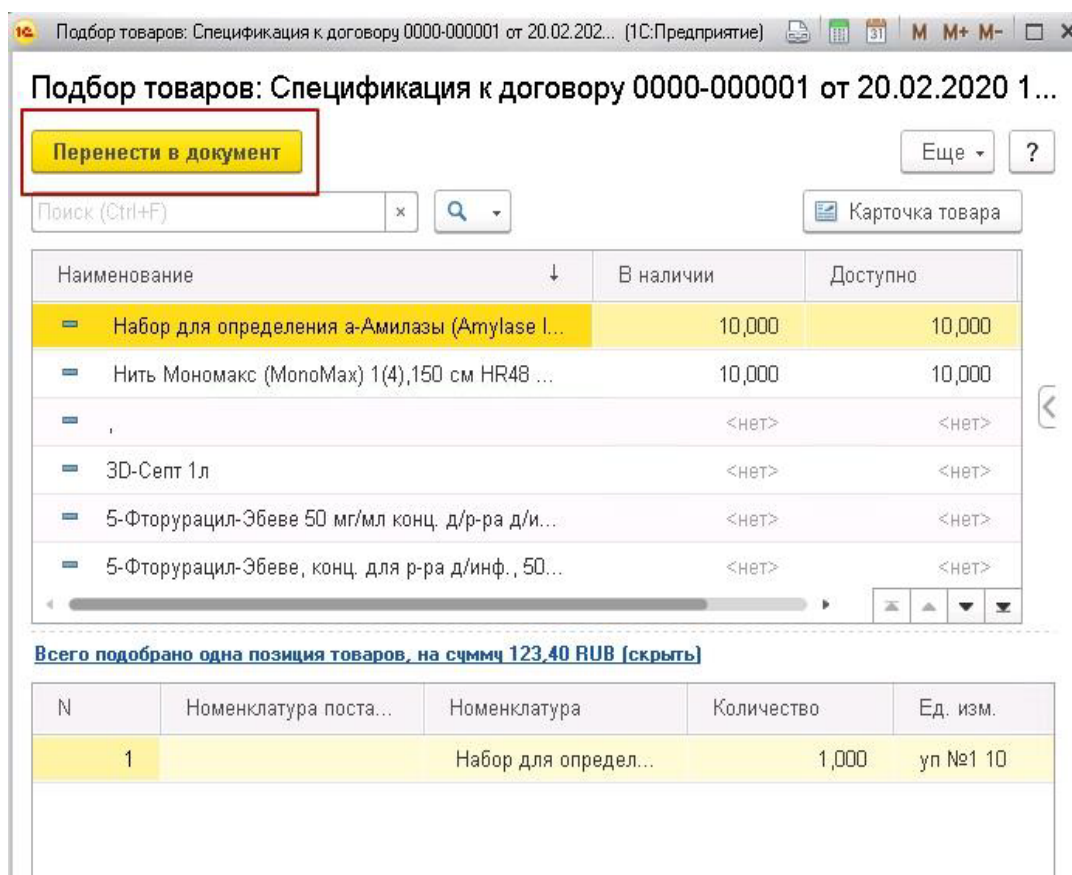


Рисунок 5 – Подбор товаров в 1С

Больничная аптека подходит:

- больничным аптекам медицинских организаций, в том числе с обособленными подразделениями (филиалами), оказывающими медицинскую помощь (разные места осуществления деятельности);
- аптекам, аптечным пунктам, аптечным киоскам и индивидуальным предпринимателям, осуществляющих отпуск лекарственных препаратов бесплатно;
- организациям, осуществляющим наряду с основной (уставной) деятельностью медицинскую деятельность (образовательные организации, учреждения науки, учреждения социального обслуживания, учреждения уголовно-исполнительной системы и иные организации), закупающих лекарственные препараты для использования в своей деятельности.

Приложение поддерживает ведение учета (в том числе складского и финансового) [6].

В таблице 5 приведена краткая информация о достоинстве, недостатке и стоимости продукта.

Таблица 5 – Показатели приложения «Больничная аптека»

Достоинства	Недостатки	Стоимость
Содержит очень много функций.	Сложный интерфейс, тяжело обучаться. Затраты на дополнительное оборудование.	Стоимость программы = 58 тысяч рублей.

Приложение №2 «Единая система учёта и контроля льготных рецептов». Разработчик «ЮНИКО».

«Единая система учета и контроля льготных рецептов» (далее «Система») предназначена для автоматизации учета льготных рецептов и контроля лекарственного обеспечения граждан, имеющих право на льготное обеспечение медикаментами и изделиями медицинского назначения [7].

На рисунке 6 приведен скриншот работы программы со справочником товаров.

Справочник товаров
более 50 тыс. наименований

Удобный поиск: по штрих-коду, коду и наименованию, ЖНВЛС, производителю, группам и т.д. Есть возможность хранить изображения и описания ЛС.

Тип груза: НДС 10%

По названию | По коду | По штрихкоду Доп. фильтры

Код	Наименование	Р	Ж	Штрихкод	Изготовитель	д. гр. зак.	Наименование гр. заказов	Код ЕПК	Наименование ЕП
17361	5-НОК 50мг №50 тб	✓	✓	1600000173613	Вектор/Лек Россия	2250	5-НОК 50мг №50 тб	32188	5-НОК тб п/о 50мг
2	5-НОК 50мг №50 тб	✓	✓	3638957090976	Вектор/Лек Россия	2250	5-НОК 50мг №50 тб	32188	5-НОК тб п/о 50мг
16232	5-фторурацил 1,0г-20мл конц. №	✓	✓	9088881314318	Евеве Австрия	2251	5-фторурацил 1,0г-20мл кон		
21313	5-фторурацил 5%-5мл №10 амл.	✓	✓	4823006400331	Дарница Россия	10900	5-фторурацил 5%-5мл №10		
26269	3 месяцае фольевая к-та 400мг			4602193010086	Валента Фармацевтика	20255	3 месяцае фольевая к-та 40		
25196	Нутрилак 6-12 400г			4600490612088	Нутрилек ЗАО Россия	19211	Нутрилак 6-12 400г		
25243	Нутрилак 6-12 с пребиотиками 4			4600490612359	Нутрилек ЗАО Россия	19530	Нутрилак 6-12 с пребиотика		
25197	Нутрилак безлактозный 400г			4600490612347	Нутрилек ЗАО Россия	19212	Нутрилак безлактозный 400		
25198	Нутрилак гипоаллергенный 400г			4600490613689	Нутрилек ЗАО Россия	19213	Нутрилак гипоаллергенный		
25199	Нутрилак низколактоз. 400г			4600490612330	Нутрилек ЗАО Россия	19214	Нутрилак низколактоз. 400		
25200	Нутрилон 1 с прибиот с 0 мес. 9			8712400715089	"NUTRICIA Суйк BV"	19215	Нутрилон 1 с прибиот с 0 м		
25675	Нутрилон 2 с прибиот с 6 мес. 90			8712400715096	"NUTRICIA Суйк BV"	19689	Нутрилон 2 с прибиот с 6 м		
25628	Нутрилон Безлактозный с 0 мес			8712400718486	ОАО "ДЕТСКОЕПИТАНИЕ" ИСТ	19644	Нутрилон безлактозный с 0		
25202	Нутрилон Комфорт 1-цимлюнофор			8712400732258	"NUTRICIA Суйк BV"	19217	Нутрилон комфорт 1-цимлю		
25201	Нутрилон комфорт 1-цимлюнофор			8712400715232	"NUTRICIA Суйк BV"	19216	Нутрилон комфорт 1-цимлю		

Рисунок 6 – Справочник товаров в «Единая система учета и контроля льготных рецептов»

Первый уровень применения «Системы» – аптечные организации, имеющие право на льготный отпуск лекарственных средств. Здесь устанавливается модуль «Учет льготных рецептов» с готовыми базами льготников, медикаментов, входящих в перечень льготных, врачей, лечебно-профилактических учреждений (ЛПУ) и др.

В аптеках при вводе параметров реализованного рецепта происходит сбор данных в следующем объеме: даты выписки и отпуска рецепта, категория льготности и ФИО больного, серия и номер его страхового полиса, адрес, диагноз, код ЛПУ, ФИО и специализация врача, наименование, количество и стоимость отпущенных медикаментов, а также отношение медикамента к формулярному списку или компетенции клинико-экспертной комиссии (КЭК). На основе этих данных автоматически формируются финансовые документы для компенсации затраченных средств, а также всевозможные отчеты:

- по категориям граждан в разрезе 50% и 100% скидок, по расходу медикаментов и денежных средств в разрезе больных, фармацевтических групп и медикаментов, врачей и их специализаций, нозологий, ЛПУ, плательщиков, отделов аптечных организаций, по экстенпоральной рецептуре;
- для органов социальной защиты по лекарственному обеспечению

инвалидов и ветеранов;

- лицевой счет больного и др.

Это позволяет руководителю аптеки прогнозировать номенклатуру и объем закупок, решать спорные вопросы с обеспечением медикаментами отдельных граждан из числа льготников. За отчетный период формируется также «акт экспертизы базы данных» позволяющий контролировать целостность и достоверность введенных данных.

Система выполнена по современной технологии в среде Windows, может работать в локальном и сетевом режиме, передача данных может быть осуществлена любым доступным для пользователя способом, подавляющее большинство первичных данных вводится из готовых справочников, технология синхронизации справочников позволяет пополнять их в реальном режиме времени.

В настоящее время модули учета льготных рецептов дорабатываются в соответствии с новыми требованиями согласно Закону о монетизации льгот [7].

В таблице 6 приведена краткая информация о достоинстве, недостатке и стоимости продукта.

Таблица 6 – Показатели приложения «Единая система учёта и контроля льготных рецептов»

Достоинства	Недостатки	Стоимость
Удаленное управление из дома.	Сложная в освоении, неудобная в использовании.	Стоимость продукта = 64 тысячи рублей.
Несложное создание отчетности.	Код программы поставляется не в открытом виде.	Расходы за внедрение.
Учет льготных препаратов.	Функционал системы не охватывает полностью потребности компании-заказчика.	

Приложение №3 «Mobile SMARTS: МДЛП». Разработчик «Клеверенс».

Новый продукт «Mobile SMARTS: МДЛП» для обратного акцепта и инвентаризации лекарств в аптеках и медицинских учреждениях с отправкой данных в учетную систему, либо напрямую в «Честный ЗНАК» [8].

На рисунке 7 приведен скриншот работы программы.

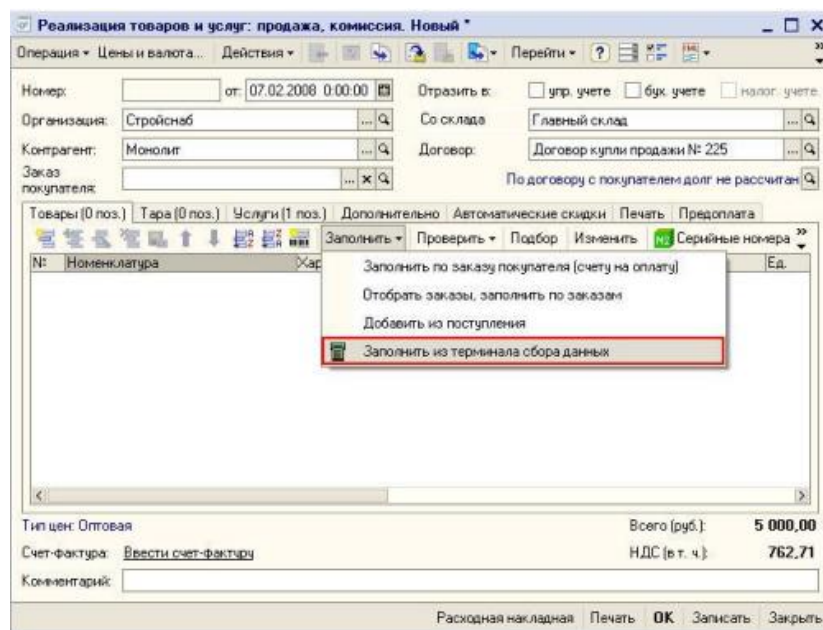


Рисунок 7 – Реализация товаров и услуг в «Mobile SMARTS: МДЛП»

Для аптек, а также для медицинских учреждений и мед. кабинетов, существует потребность в простой и легкой работе с Международной Доской Лекарственных Продуктов (МДЛП) для выполнения базовых операций по приему и учету лекарственных препаратов.

Частный случай такой потребности – организация обратного акцепта, когда аптека или медицинское учреждение обязуется самостоятельно сканировать коды маркировки поступивших лекарств для отправки этой информации поставщику.

Решение подходит для любых аптек, стоматологических клиник и кабинетов, медицинских клиник, больниц, поликлиник и врачебных кабинетов [8].

В таблице 7 приведена краткая информация о достоинстве, недостатке и стоимости продукта.

Таблица 7 – Показатели приложения «Mobile SMARTS: МДЛП»

Достоинства	Недостатки	Стоимость
Простая при использовании.	Сложность в установке и настройке	Стоимость = 39 тысяч рублей.
Стоимость не слишком высокая.	Очень мало функций. Для заказа доработок требуются дополнительные расходы.	

В данном разделе были рассмотрены инструментальные средства для управления организацией работы с заказами на изготовление лекарственных средств. Были выявлены их достоинства, недостатки и стоимость, как базовые функции.

1.2.4 Результаты сравнительного анализа

Результат сравнения рассмотренных аналогов по отношению к новому приложению приведен в таблице 8.

Таблица 8 – Результаты сравнения аналогов с новой программой

Критерии	№1	№2	№3	Новая программа
Установка (требования к оборудованию, серверу).	+	-	-	+
Стоимость лицензии и оборудования.	-	+	+	+
Внедрение (возможность установить ПО, обучить персонал своими силами).	-	+	-	+
Простота в обучении персонала.	-	-	+	+
Соответствие функциям в постановке задачи.	-	-	-	+
Доступный исходный код ПО.	-	-	-	+
Отчет по клиентам с выгрузкой в Excel.	+	-	-	+
Отчет по сотрудникам с выгрузкой в Excel.	+	-	-	+
Оформление заказа по заявленным требованиям.	-	-	-	+

Подведем итоги анализа готовых решений-аналогов на рынке программного обеспечения. Функциональные требования для разработки приведены в разделе постановки задачи. Анализируя достоинства и недостатки рассмотренных продуктов и сравнивая с требованиями, видим, что ни одна программа-аналог не подходит по заявленным требованиям. Все системы нужно будет дорабатывать под необходимые требования, это снова дополнительные расходы, некоторые системы представляют собой сложный интерфейс, сложная установка, отсутствует техническая поддержка программы.

Поэтому было принято решение разрабатывать новый программный продукт для обработки и анализа индивидуальных заказов на изготовление лекарственных средств.

1.3 Постановка задачи на разработку приложения

1.3.1 Цели разработки и назначение приложения

Разрабатываемое приложение предназначено для автоматизации процессов учета и анализа обработки индивидуальных заказов на изготовление лекарственных средств. Основными пользователями в разработке приложения являются фармацевт и провизор, для которых автоматизируется процесс. Это позволяет снизить временные и денежные затраты на выполнение стандартных рутинных операций пользователями. Новое приложение предназначено для фармацевтов, провизоров и клиентов.

Приложение должно выполнять основные функции для оформления заказов на изготовление лекарственных средств, хранить информацию в таблицах базы данных СУБД MySQL.

1.3.2 Классы и роли пользователей приложения

Новый программный продукт планируется разрабатывать для пользователей, которые будут управлять бизнес-процессами и вести учёт деятельности компании с помощью отчётов, а также оформлять индивидуальные заказы лекарственных средств по рецепту. Классы пользователей приложения и их роли определены в таблице 9.

Таблица 9 – Классы пользователей

Классы пользователей	Роли
Клиент	Человек, который обратился в организацию, для покупки необходимых ему лекарств. Он может обратиться к фармацевту за консультацией по выбору препаратов или оформить заказ на медикамент по рецепту врача.
Фармацевт	Специалист в области фармации, который консультирует клиентов по выбору медикаментов и их применению, контролирует соблюдение правил хранения и передачи медикаментов, оформляет заказы в аптеках и ведет учет деятельности компании с помощью отчетов.

Продолжение таблицы 9

Классы пользователей	Роли
Провизор	Специалист в области фармации, который контролирует качество поступающих аптечных товаров и препаратов, изготавливает лекарства по рецептам, ведет <u>учетно-отчетную</u> документацию.

В данной главе были рассмотрены классы пользователей приложения, такие как провизоры, фармацевты и клиенты, а также описаны роли для каждого класса пользователей.

1.3.3 Автоматизируемые функции приложения

Опишем процессы, которые требуется автоматизировать при оформлении индивидуальных заказов на изготовление лекарственных средств. Процессами в медицинских учреждениях, аптеках управляют сотрудники, а именно, фармацевт и провизор, а также необходимо предоставить доступ клиенту для формирования заказа, что поможет уменьшить нагрузку для сотрудников компании.

Для всех классов пользователей определена функция авторизации, которая обеспечивает вход пользователей в систему, используя свои учетные данные, предоставляя доступ к приложению.

Автоматизированные функции для фармацевта приведены в таблице 10.

Таблица 10 – Автоматизируемые функции для фармацевта

№	Функция	Входные данные	Выходные данные
1	Передача заказа провизору	Состав заказа с указанием, какие лекарства входят в заказ и в каком количестве, информация о пункте выдачи заказа	Отчет по пунктам выдачи, история по заказу
2	Оформление заказа на пункте выдачи	Информация о заказе на лекарственные средства по рецепту, информация о клиентах, которые заказывают лекарственные средства, информация о пунктах выдачи заказа	История по клиенту, история по заказу, отчет по клиенту
3	Подписание и выдача чека или документа об оплате	Оплата заказа	Печать чека оплаты

Перечень автоматизированных функций для клиента приведен в таблице 11.

Таблица 11 – Автоматизируемые функции клиента

№	Функция	Входные данные	Выходные данные
1	Регистрация в системе	Данные клиента (логин, пароль)	Созданная учетная запись
2	Оформление заказа на изготовление лекарственного средства в соответствии с рецептом	Состав заказа с указанием, какие лекарства входят в заказ и каком количестве	Оформленный заказ
3	Выбор из списка пунктов выдачи, где будет забирать заказанные препараты (аптечные пункты)	Информация о пунктах выдачи заказов	Выбранный пункт выдачи
4	Просмотр статуса заказа, просмотр истории заказа, оплата заказа при получении	Оформленный заказ	История по заказу, чека оплаты
5	Отмена заказа	Информация о заказе	Отмененный заказ

Для провизора автоматизированные функции приведены в таблице 12.

Таблица 12 – Автоматизируемые функции для провизора

№	Функция	Входные данные	Выходные данные
1	Просмотр заказов, поступивших от клиентов, сортировка	Состав заказа с указанием, какие лекарства входят в заказ и в каком количестве, информация о пункте выдачи заказа	Отчет по пунктам выдачи, история по заказу
2	Создание электронного заказа на недостающие препараты у поставщиков	Информация по поставщикам, у которых компания закупает лекарственные средства, информация о лекарственных средствах, производители лекарственных препаратов	Отчет о количестве лекарственных средств
3	Оперативная поддержка информации о медикаментах, заявках, пользователях и аптеках (добавление, редактирование, удаление)	Информация о пунктах выдачи заказов, информация о лекарственных средствах, информация о сотрудниках компании (фармацевты, провизоры), информация о заказе на лекарственные средства по рецепту	Отчет по сотруднику, отчет по пунктам выдачи

Продолжение таблицы 12

№	Функция	Входные данные	Выходные данные
4	Поиск информации о препаратах	Выбранный критерий или название препарата	Найденный препарат

К общим автоматизируемым функциям для провизора и фармацевта можно отнести обновление статуса заказа в системе. Фармацевт меняет статус заказа при оформлении заказа и передачи провизору, а также при выдачи изготовленного препарата при полной оплате заказа. Провизор меняет статус заказа при изготовлении заказа.

1.4 Планирование процесса разработки приложения

Планирование процесса разработки сервиса используется для управления поставленными задачами при реализации нового проекта. Планирование процесса позволяет грамотно распределять своё время, отслеживать и обновлять план разработки проекта от его начала и до самого конца.

Рассмотрим функционально-стоимостной анализ процесса разработки сервиса. В ходе анализа оценим себестоимость продукта через управление процессами при разработке приложения.

В качестве анализа был рассмотрен средний уровень заработной платы программиста в 2024 году в городе Пенза, который составляет 60 тыс. рублей в месяц при 8-часовом рабочем дне. Исходя из расчетов, программист в день получает 375 руб. в час.

В данной работе была разработана контекстная диаграмма бизнес-процессов в нотации IDF0 в программной среде BPwin. В IDEF0 планирование процесса представляется как совокупность взаимодействующих задач.

Процесс моделирования какой-либо системы в IDEF0 начинается с определения контекста, т. е. наиболее абстрактного уровня описания системы в целом. В контекст входит определение субъекта моделирования, цели и точки зрения на модель [10].

Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами, другими словами,

мы должны определить, что мы будем в дальнейшем рассматривать как компоненты системы, а что как внешнее воздействие [11].

Контекстная диаграмма является вершиной древовидной структуры диаграмм и представляет собой самое общее описание системы и ее взаимодействия с внешней средой.

Контекстная диаграмма модели AS-IS представлена на рисунке 8.

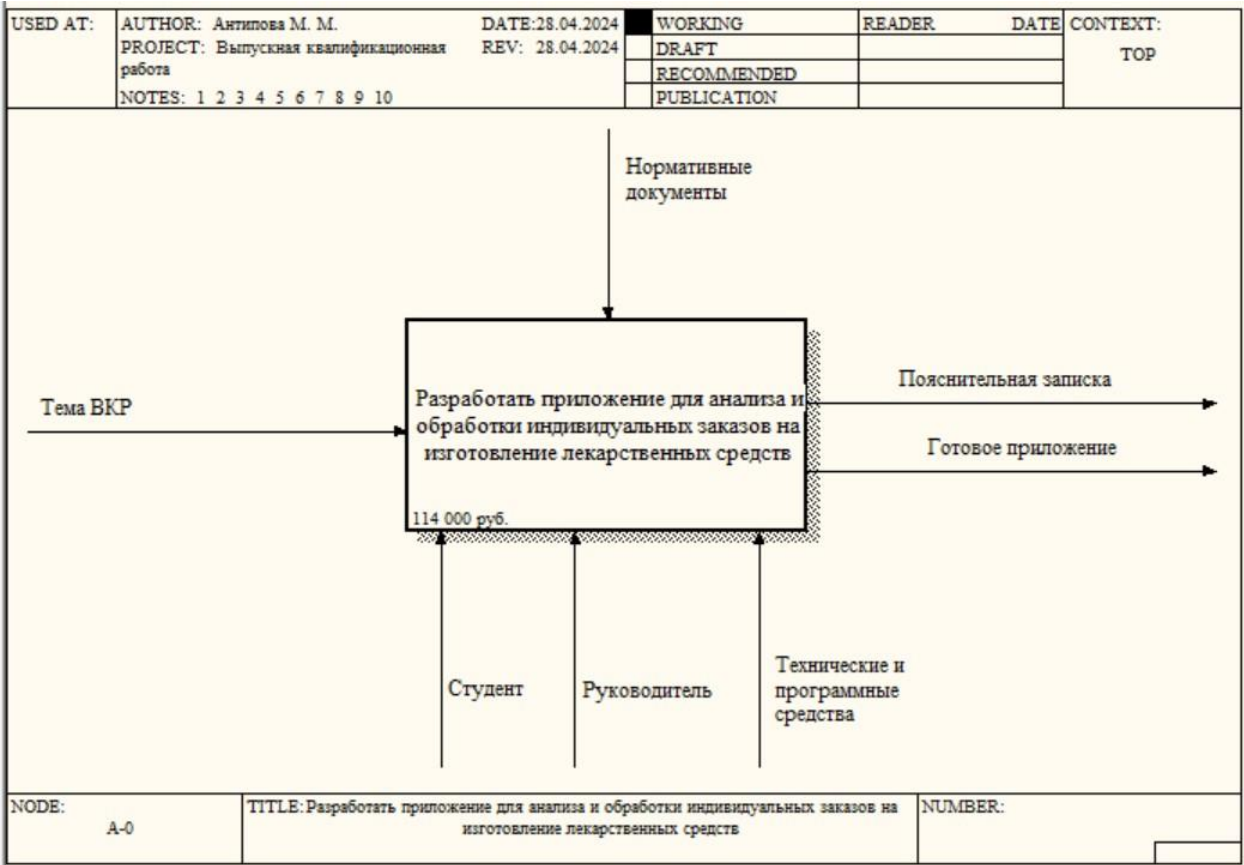


Рисунок 8 – Контекстная диаграмма процесса выполнения выпускной квалификационной работы

На контекстной диаграмме изображено планирование процесса разработки в целом. Входом является «Тема ВКР», который используется или преобразуется работой для получения результата (выхода). Стрелкой сверху на диаграмме указано управление: «Нормативные документы».

Управление – это стратегии, правила, процедуры или стандарты, которыми руководствуется работа. Управление влияет на работу, но не преобразуется работой.

Механизмами являются ресурсы: «Студент», «Руководитель», «Технические и программные средства», которые выполняют работу.

В качестве выходной информации получаем «Пояснительную записку» и «Готовое приложение».

Для более детального описания процесса разработки используется диаграмма декомпозиции первого уровня, которая иллюстрирует выполнение функций, которые необходимо осуществить для реализации основной задачи. На рисунке 9 представлена данная диаграмма.

На диаграмме декомпозиции изображены пять процессов разработки: «Проанализировать предметную область и определить задачи на разработку приложения», «Спроектировать приложение», «Разработать программное обеспечение», «Выполнить контроль качества программного обеспечения», «Оформить пояснительную записку».

На диаграмме декомпозиции можно детально увидеть в каких процессах используются конкретные механизмы и управления.

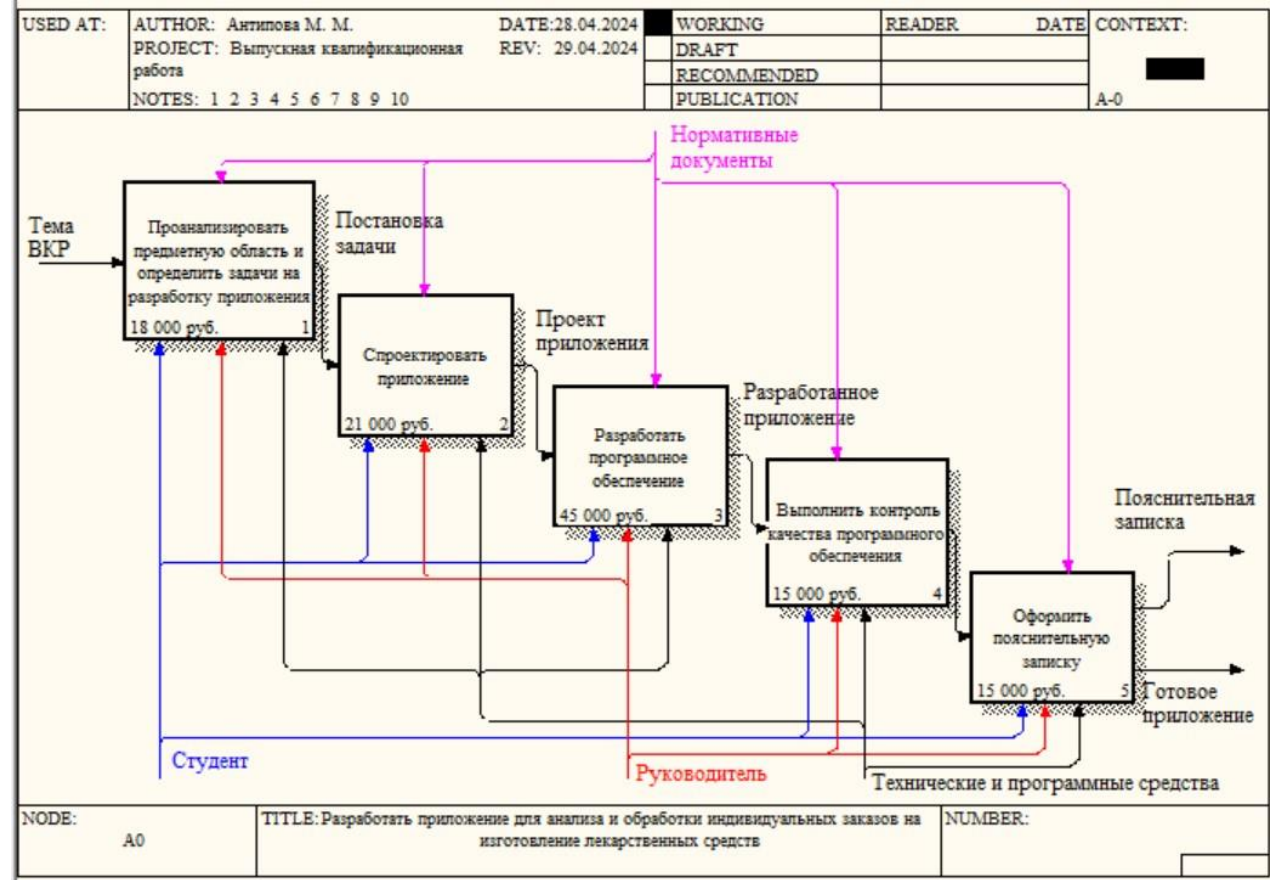


Рисунок 9 – Декомпозиция диаграммы «ВКР»

Итоги подсчетов затрат представлены в таблице 13.

Таблица 13 – Затраты на разработку проекта

Этап	Время	Цена	Затраты
Анализ предметной области и постановка задачи на разработку приложения	48 ч.	375руб./час	18000 руб.
Проектирование приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств	56 ч.	375руб./час	21000 руб.
Разработка программного обеспечения	120 ч.	375руб./час	45000 руб.
Контроль качества программного обеспечения	40 ч.	375руб./час	15000 руб.
Оформление пояснительной записки	40 ч.	375руб./час	15000 руб.

В результате анализа было выявлено, что стоимость создания приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств составляет 114000 рублей.

2 Проектирование приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств

2.1 Моделирование функциональных возможностей приложения

Для наглядного представления функциональных возможностей приложения была создана диаграмма вариантов использования в нотации UML [12], которая представлена в приложении Б на рисунках Б.1 – Б.3.

Элементами данной диаграммы является:

- система – обозначается как прямоугольник с названием системы сверху, в котором размещены все варианты использования;
- вариант использования – это бизнес-процессы, бизнес-операции и операции с данными, которые показывают, как система будет функционировать;
- актер – это человек, организация или внешняя система, которые играют роль в одном или нескольких взаимодействиях с системой.

Существует несколько типов отношений между элементами:

- связь между актером и вариантом использования;
- связь между двумя вариантами использования;
- обобщение между двумя актерами;
- обобщение между двумя вариантами использования.

Для представления отношений между различными вариантами использования, используются отношения расширения, включения и обобщения.

Отношение расширения – это отношение между вариантами использования, которое показывает, что один вариант использования является необязательным и следует за базовым вариантом использования. Он представлен пунктирной стрелкой в направлении базового варианта использования с обозначением «extend».

Отношение включения – это отношение между вариантами использования, которое показывает, что один вариант использования является обязательным и является частью базового варианта использования. Он представлен пунктирной

стрелкой в направлении включенного варианта использования с обозначением «include».

Отношение обобщения – это отношение между вариантами использования, которое показывает, что один вариант использования является частным случаем другого варианта использования. Он представлен сплошной стрелкой в направлении от частного к общему.

На диаграммах можно увидеть все бизнес-процессы, осуществляемые пользователями в программе. Для удобства на диаграммах использованы цветовые обозначения:

- бизнес-процессы показаны зеленым цветом;
- бизнес-операции – желтым цветом;
- операции с данными – серым цветом.

На диаграммах изображены три актера: клиент, фармацевт и провизор.

Помимо актеров и вариантов использования диаграмма вариантов использования может содержать примечания – элементы, которые служат для размещения на диаграмме поясняющей текстовой информации.

Клиент в программе совершает пять бизнес-процессов: регистрация, авторизация, создание заказа, просмотр истории заказов, просмотр личных данных.

Краткое описание каждого варианта использования (ВИ) функциональных возможностей клиента представлено в таблице 14.

Таблица 14 – Краткое описание вариантов использования для клиента

Наименование	Тип ВИ	Описание
Зарегистрироваться	Бизнес-процесс	Приложение дает пользователю возможность создать учетную запись, где после регистрации он может войти в систему и получить доступ к функциям и данным приложения
Авторизоваться	Бизнес-процесс	Обеспечивает вход пользователя в систему, используя свои учетные данные

Продолжение таблицы 14

Наименование	Тип ВИ	Описание
Сделать заказ	Бизнес-процесс	Дает пользователю возможность оформить заказ на приобретение необходимого лекарственного препарата по рецепту
Выбрать пункт выдачи	Бизнес-операция	При оформлении заказа пользователю нужно выбрать пункт выдачи, где он заберет заказ
Ввести данные по лекарствам	Бизнес-операция	При оформлении заказа клиенту необходимо ввести данные из рецепта для изготовления препарата
Внести предоплату	Бизнес-операция	Перед окончанием оформления заказа нужно внести предоплату
Выгрузить отчет по оплатам	Бизнес-операция	Пользователь может выгрузить отчет по оплатам в Excel
Загрузить рецепт	Бизнес-операция	Пользователь может загрузить рецепт в систему в виде документа или картинки
Выгрузить отчет по рецепту	Бизнес-операция	Пользователь может выгрузить отчет по рецепту в Excel
Выгрузить отчет по заказу	Бизнес-операция	Система дает возможность выгрузить отчет по заказам клиента в Excel
Оплатить заказ	Бизнес-операция	После получения пользователь может оплатить заказ полностью
Выгрузить чек об оплате	Бизнес-операция	Приложение обеспечивает выгрузку чека оплаты в Excel
Посмотреть историю заказов	Бизнес-процесс	Дает пользователю возможность посмотреть информацию о ранее заказанных препаратах в системе
Посмотреть статус заказа	Бизнес-операция	Приложение дает возможность посмотреть статус заказа
Отменить заказ	Операции с данными	Обеспечивает отмену заказа до его получения и полной оплаты

К созданию заказа относятся следующие варианты использования: внести данные по лекарствам, выбрать пункт выдачи и внести предоплату, загрузить рецепт, выгрузить отчет по заказу, оплатить заказ. Также есть возможность выгрузить отчет по оплатам, по рецепту и выгрузить чек об оплате. Данные варианты использования позволяют оформить заказ на изготовление лекарственных средств по рецепту. К просмотру истории заказов относится вариант использования «Посмотреть статус заказа», где клиент сможет посмотреть состояние своего заказа и отменить заказ, и сортировка заказов клиента. К варианту использования «Посмотреть личные данные» относится изменение личных данных о клиенте.

Фармацевт в программе совершает шесть бизнес-процессов: авторизация, оформление заказа, просмотр информации по заказам, просмотр отчета историй по заказам клиентов, просмотр отчета по клиентам, просмотр информации по клиентам.

Краткое описание каждого варианта использования функциональных возможностей фармацевта представлено в таблице 15.

Таблица 15 – Краткое описание вариантов использования для фармацевта

Название	Тип ВИ	Описание
Авторизироваться	Бизнес-процесс	Обеспечивает вход пользователя в систему, используя свои учетные данные
Оформить заказ	Бизнес-процесс	В приложении есть возможность принять заказ на изготовление лекарств через фармацевта, где сотрудник заполняет все необходимые данные
Произвести предоплату	Бизнес-операция	При оформлении заказа необходимо внести предоплату клиентом и указать это в приложении
Изменить статус заказа	Операции с данными	Фармацевт может изменять статус заказа в приложении
Внести данные по препаратам	Бизнес-операция	При оформлении заказа фармацевт должен внести данные по препаратам из рецепта

Продолжение таблицы 15

Название	Тип ВИ	Описание
Загрузить рецепт	Бизнес-операция	Приложение дает возможность загрузить рецепт в приложение в виде картинки, либо документа
Выгрузить отчет по рецептам	Бизнес-операция	Приложение дает возможность выгрузить отчет по рецептам в Excel
Выбрать пункт выдачи	Бизнес-операция	При оформлении заказа приложение дает возможность выбрать пункт выдачи
Произвести полную оплату	Бизнес-операция	После получения лекарства необходимо внести полную оплату за заказ и отметить это в приложении
Выгрузить отчет по оплатам	Бизнес-операция	Система обеспечивает выгрузку отчета по оплатам в Excel
Выдать чек об оплате	Бизнес-операция	После полной оплаты приложение дает возможность выгрузить чек об оплате
Посмотреть информацию по заказам	Бизнес-процесс	Приложение дает возможность посмотреть все заказы клиентов
Сортировать	Бизнес-операция	Приложение дает возможность сортировать информацию по заказам
Изменить информацию по заказам	Операции с данными	В приложении фармацевт может изменить информацию по заказам
Удалить информацию по заказам	Операции с данными	В приложении фармацевт может удалить информацию по заказам
Выгрузить отчет по всем заказам клиентов	Бизнес-операция	Система обеспечивает выгрузку отчета по всем заказам клиентов в Excel
Посмотреть отчет историй по заказам клиентов	Бизнес-процесс	Приложение дает возможность посмотреть историю по заказам клиентов
Сортировать заказы клиентов	Бизнес-операция	Приложение дает возможность отсортировать заказы клиентов
Посмотреть отчет по клиентам	Бизнес-процесс	Фармацевт может посмотреть отчет по клиентам

Окончание таблицы 15

Название	Тип ВИ	Описание
Выгрузить отчет по клиентам	Бизнес-операция	Приложение дает возможность выгрузить отчет по клиентам в Excel
Посмотреть информацию по клиентам	Бизнес-процесс	Фармацевт может посмотреть всю информацию по клиентам
Добавить информацию по клиенту	Операции с данными	Приложение дает возможность добавить информацию о клиенте
Изменить информацию о клиенте	Операции с данными	Приложение дает возможность изменить информацию о клиенте
Удалить информацию о клиенте	Операции с данными	Приложение дает возможность удалить информацию о клиенте

К оформлению заказа относятся следующие варианты использования: внести данные по препаратам, выбрать пункт выдачи, произвести предоплату, либо полную оплату, загрузить рецепт, изменить статус заказа. Также есть возможность выгрузить отчет по рецепту, отчет по оплатам и выдать чек об оплате после полной оплаты заказа. Данные варианты использования позволяют оформить заказ на изготовление лекарственных средств по рецепту. К варианту использования «Посмотреть информацию по заказам» относятся: сортировка заказов, изменение и удаление информации по заказам в БД, выгрузка отчета по всем заказам клиента в Excel. К варианту использования «Посмотреть отчет историй по заказам клиентов» относится сортировка заказов клиентов. Также фармацевт может посмотреть отчет по клиентам, где может его выгрузить в Excel. Вариант использования «Посмотреть информацию по клиентам» включает в себя добавление, удаление и изменение данных о клиентах. В результате получаем, что фармацевт может оформлять заказы у клиентов и управляет всей необходимой документацией.

Провизор в программе совершает шесть бизнес-процессов: авторизация, изменение данных в базе данных, просмотр заказов от фармацевта, создание заказа на недостающие ингредиенты, просмотр отчета по поставщикам, просмотр справочника по поставщикам.

Краткое описание каждого варианта использования функциональных возможностей провизора представлено в таблице 16.

Таблица 16 – Краткое описание вариантов использования для провизора

Название	Тип ВИ	Описание
Авторизироваться	Бизнес-процесс	Обеспечивает вход пользователя в систему, используя свои учетные данные
Изменить данные в базе данных	Бизнес-процесс	Провизор может менять данные в базе данных (добавить, удалить, редактировать)
Изменить информацию о лекарственных средствах	Операции с данными	Провизор может изменять данные о лекарственных средствах в базе данных (добавить, удалить, редактировать)
Изменить информацию о заказах	Операции с данными	Провизор может изменять данные о заказах в базе данных (добавить, удалить, редактировать)
Изменить статус заказа	Операции с данными	Провизор может изменять статус заказа в приложении
Изменить информацию о препаратах	Операции с данными	Провизор может изменять данные о препаратах
Добавить препарат	Операции с данными	Приложение дает возможность провизору добавлять препарат
Поиск по названию	Бизнес-операция	Приложение дает возможность осуществлять поиск по названию препарата
Посмотреть заказы от фармацевта	Бизнес-процесс	В приложении есть возможность посмотреть заказы, поступившие от фармацевта
Посмотреть состав заказа	Бизнес-операция	Провизор может посмотреть состав заказа в приложении
Посмотреть рецепт	Бизнес-операция	В приложении провизор может посмотреть рецепт, загруженный клиентом, либо фармацевтом
Выгрузить рецепт	Бизнес-операция	Провизор может выгрузить рецепт в Excel
Выгрузить отчет по заказам	Бизнес-операция	Провизор может выгрузить отчет по заказам в Excel
Сортировать	Бизнес-операция	Приложение дает возможность сортировать заказы от фармацевта

Продолжение таблицы 16

Название	Тип ВИ	Описание
Сделать заказ на недостающие ингредиенты	Бизнес-процесс	Если ингредиентов не хватает в наличии, провизор может сделать заказ поставщику через приложение
Внести данные по препаратам	Бизнес-операция	При оформлении заказа поставщику провизор вносит данные по препаратам в приложение

К изменению данных в базе данных относятся следующие варианты использования: изменить информацию о препаратах, изменить информацию о заявках, изменить информацию о лекарственных средствах. В свою очередь есть возможность изменить статус заказа, добавить препарат и воспользоваться поиском по названию препарата. Данные варианты использования показывают работу с базой данных. К варианту использования «Посмотреть заказы от фармацевта» относятся: посмотреть состав заказа, посмотреть рецепт, выгрузить отчет по заказам, сортировка. Также есть возможность выгрузить рецепт в Excel. К варианту использования «Сделать заказ на недостающие препараты» относятся: внести данные по рецептам, выгрузить отчет по заказам поставщику в Excel, сортировать заказы. Провизор также может посмотреть отчет по поставщикам, где есть возможность выгрузить отчет в Excel. В приложении есть возможность посмотреть справочник по поставщикам, где провизор может изменять данные в БД: добавлять, изменять и удалять информацию о поставщиках.

В таблице 17 описывается сценарий для прецедента «Сделать заказ».

Таблица 17 – Сценарий прецедента «Сделать заказ»

Название варианта использования	Сделать заказ
Актеры	Клиент
Краткое описание	Оформление заказа на изготовление лекарства по рецепту
Второстепенные действующие лица	Нет
Предусловие	Клиент авторизировался в приложении

Продолжение таблицы 17

Типичный ход событий	<ol style="list-style-type: none"> 1. Клиент вносит данные по препаратам из рецепта в состав заказа. 2. Выбирает пункт выдачи заказа. 3. Загружает рецепт. 4. Вносит предоплату, либо оплачивает заказ полностью. 5. Может выгрузить отчет по оплатам. 6. Может выгрузить отчет по рецепту. 7. Может выгрузить отчет по заказу. 8. Может выгрузить чек оплаты.
Постусловие	Приложение сохранило созданный заказ

В таблице 18 описывается сценарий для прецедента «Оформить заказ».

Таблица 18 – Сценарий прецедента «Оформить заказ»

Название варианта использования	Оформить заказ
Актеры	Фармацевт
Краткое описание	Оформление заказа на изготовление лекарства по рецепту клиента на пункте выдачи
Второстепенные действующие лица	Нет
Предусловие	Фармацевт вошел в свой аккаунт
Типичный ход событий	<ol style="list-style-type: none"> 1. Фармацевт открывает вкладку с заказами. 2. Вносит данные из рецепта клиента. 3. Выбирает пункт выдачи заказа со согласованием клиента. 4. Загружает рецепт. 5. Записывает в приложение, что предоплата была внесена, либо полная оплата. 6. Изменяет статус заказа. 7. Может выгрузить отчет по рецепту. 8. Может выгрузить отчет по оплатам. 9. Может выдать чек об оплате.
Постусловие	Приложение сохранило созданный заказ

В таблице 19 описывается сценарий для прецедента «Посмотреть заказы от фармацевта».

Таблица 19 – Сценарий прецедента «Посмотреть заказы от фармацевта»

Название варианта использования	Посмотреть заказы от фармацевта
Актеры	Провизор
Краткое описание	Просмотр заявок, которые поступили после оформления заказа от клиента
Второстепенные действующие лица	Нет
Предусловие	Провизор вошел в свой аккаунт

Продолжение таблицы 19

Типичный ход событий	1. Провизор открывает вкладку с заявками от фармацевта. 2. Открывает состав заказа. 3. Открывает рецепт. 4. Может выгрузить рецепт в Excel. 5. При необходимости может сделать сортировку заявок. 6. Может выгрузить отчет по заказам клиентов.
Постусловие	Приложение обновило статус заявки

Описание остальных сценариев бизнес-процессов приведено в таблицах В.1 – В.10 в приложении В.

2.2 Разработка концептуальной модели предметной области приложения

На основе описания бизнес-процессов приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств была построена исходная (первоначальная) концептуальная модель предметной области.

Концептуальная модель представляет информационное описание предметной области с учетом логических взаимосвязей. Сущности моделируются исходя из состава входных и выходных данных бизнес-процессов. Связи между сущностями показывают семантику взаимоотношения сущностей в предметной области приложения.

Чтобы разработать базу данных, необходимо построить концептуальную модель (entity-relationshipmodel) «сущность-связь». Для построения модели воспользуемся программой ERwin. ERwin является наиболее мощным средством для разработки структуры данных, как на логическом, так и на физическом уровне. Для построения диаграммы рассмотрим основные сущности, которые требуется использовать при создании базы данных.

Было выявлено 14 сущностей, которые представлены на рисунке 10.

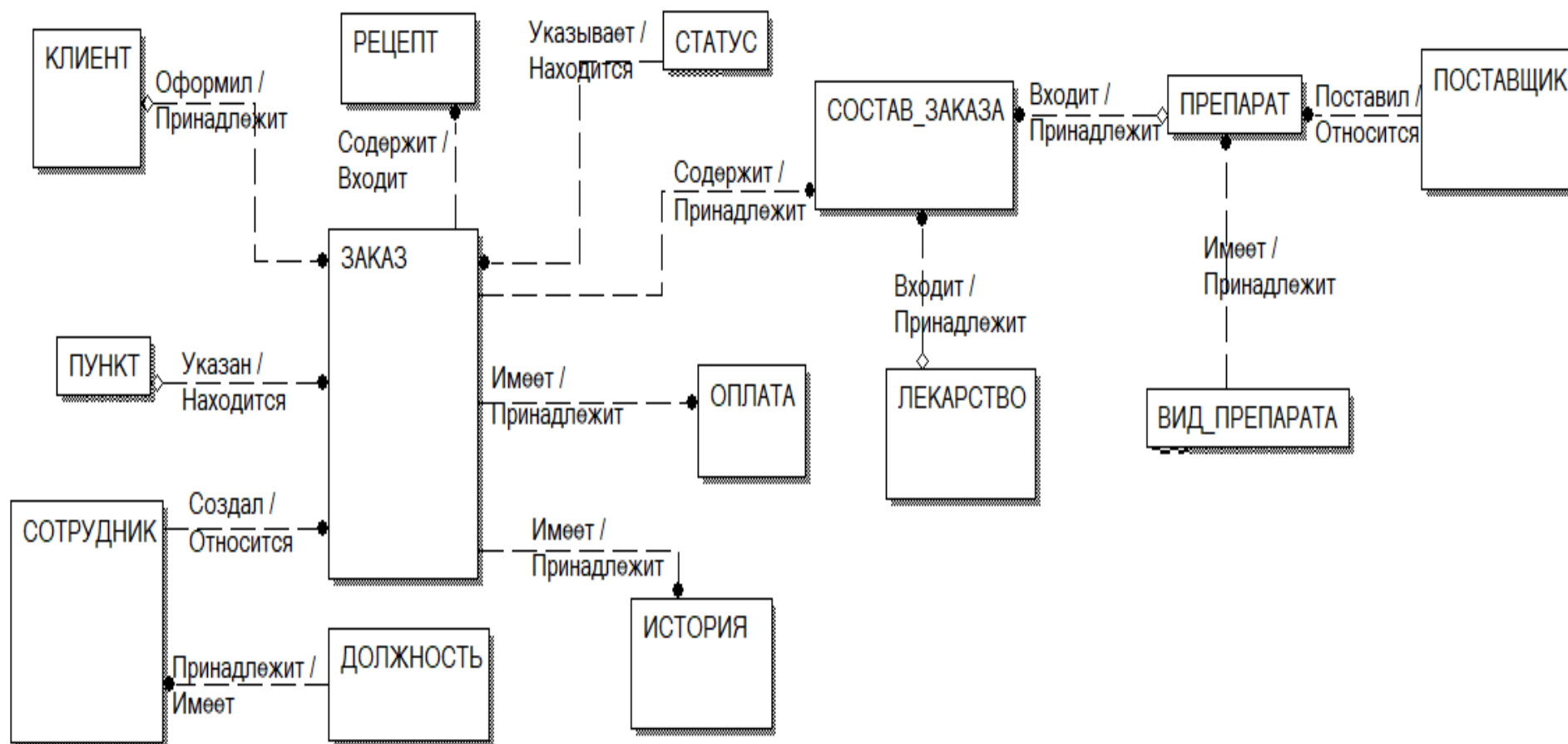


Рисунок 10 – Концептуальная модель предметной области приложения

С программой будут работать сотрудники медицинского учреждения, а также создавать и сопровождать заказы. Значит, нужна сущность, которая будет хранить эти данные. Назовем ее «СОТРУДНИК». У сотрудников имеются должности – сущность «ДОЛЖНОСТЬ».

Организация занимается изготовлением лекарственных средств, для хранения данных будем использовать сущность «ЛЕКАРСТВО». Изготовление лекарственных средств заказывают клиенты – сущность «КЛИЕНТ». Клиенты оформляют заказы на изготовление лекарства, данные по заказам определим в сущность «ЗАКАЗ». Клиенты предъявляют рецепты при оформлении заказа, для которых создадим сущность «РЕЦЕПТ». Заказы в процессе своего жизненного срока могут принимать различные статусы, значения которых будут храниться в сущности «СТАТУС». Выдача заказов происходит в пунктах выдачи – сущность «ПУНКТ». Оплаты по заказам зафиксируем в сущности «ОПЛАТА», а историю действий в сущности «ИСТОРИЯ». Заказ может содержать информацию о лекарствах для изготовления, для хранения состава заказа определим сущность «СОСТАВ_ЗАКАЗА».

Организация так же занимается оформлением заказов у поставщиков ингредиентов для изготовления лекарственных средств. Информация о поставщиках будет содержаться в сущности «ПОСТАВЩИК», а данные препаратов в сущности «ПРЕПАРАТ». Препараты подразделяются на виды, значения которых определим в сущности «ВИД_ПРЕПАРАТА».

Ниже представлены описания сущностей предметной области (таблица 20).

Таблица 20 – Описания сущностей предметной области

№	Наименование	Описание	Категория
1	Сущность «ЛЕКАРСТВО»	Для хранения информации о лекарствах, которые изготавливаются медицинской организацией	Стержневая
2	Сущность «ПОСТАВЩИК»	Для хранения справочных данных о поставщиках ингредиентов (медицинских препаратов)	Стержневая

Продолжение таблицы 20

№	Наименование	Описание	Категория
3	Сущность «СОТРУДНИК»	Для хранения информации о сотрудниках медицинской организации, которые работают фармацевтами и провизорами	Стержневая
4	Сущность «ВИД_ПРЕПАРАТА»	Для хранения справочных данных о видах препаратов (ингредиентов), поставляемых поставщиками в медицинскую организацию.	Характеристическая
5	Сущность «ПРЕПАРАТ»	Для хранения информации о препаратах	Стержневая
6	Сущность «КЛИЕНТ»	Для хранения информации о клиентах, которые оформляют заказы на изготовление лекарственных средств	Стержневая
7	Сущность «СТАТУС»	Для хранения справочных данных о статусах заказов	Стержневая
8	Сущность «ОПЛАТА»	Для хранения информации об оплатах за заказы	Стержневая
9	Сущность «ЗАКАЗ»	Для хранения информации о заказах клиентов	Стержневая
10	Сущность «СОСТАВ_ЗАКАЗА»	Для хранения информации о составе заказа	Ассоциативная
11	Сущность «ИСТОРИЯ»	Для хранения информации об истории по заказу	Стержневая
12	Сущность «ПУНКТ»	Для хранения информации о пунктах выдачи заказов	Стержневая
13	Сущность «РЕЦЕПТ»	Для хранения информации о рецептах от клиентов на изготовление лекарственных средств	Стержневая
14	Сущность «ДОЛЖНОСТЬ»	Для хранения справочной информации о названиях должностей сотрудников медицинской организации	Характеристическая

Описание связей между сущностями представлено в таблице 21.

Таблица 21 Описание связей между сущностями

№	Сущность 1	Сущность 2	Описание	Мощность	Обязательность	Ограничения
1	Клиенты	Заказы	Клиент делает заказ на изготовление препарата	Один ко многим	Да	Нет
2	Пункты	Заказы	В пункт выдачи доставляются заказы	Один ко многим	Да	Нет

Продолжение таблицы 21

№	Сущность 1	Сущность 2	Описание	Мощность	Обязательность	Ограничения
3	Сотрудники	Заказы	Сотрудник создает заказы	Один ко многим	Да	Нет
4	Должности	Сотрудники	Должность принадлежит сотрудникам	Один ко многим	Да	Нет
5	Заказы	Рецепты	По заказу имеются рецепты	Один ко многим	Да	Нет
6	Статусы	Заказы	Заказы могут быть в каком-то статусе, например, выполнен.	Один ко многим	Да	Нет
7	Заказы	Состав заказа	Заказ состоит из нескольких позиций состава заказа	Один ко многим	Да	Нет
8	Заказы	Оплаты	По заказу может быть несколько оплат	Один ко многим	Да	Нет
9	Заказы	История	По заказу имеется несколько записей из истории действий	Один ко многим	Да	Нет
10	Лекарства	Состав заказа	Лекарство может состоять в нескольких составах заказов	Один ко многим	Да	Нет
11	Препараты	Состав заказа	Препарат может состоять в нескольких составах заказов	Один ко многим	Да	Нет
12	Поставщики	Препараты	Поставщик предоставляет препараты	Один ко многим	Да	Нет
13	Виды препаратов	Препараты	По виду препарата находится несколько препаратов	Один ко многим	Да	Нет

Полученные результаты являются исходными данными для моделирования базы данных.

2.3 Разработка программной архитектуры приложения

Для реализации приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств была выбрана клиент-серверная архитектура [13].

Клиент-сервер – это архитектура, в которой участвуют две стороны: клиент – это заказчик определенных услуг и сервер – поставщик услуг. Клиент отправляет запрос на сервер с целью выполнения задач или получения информации, сервер в свою очередь обрабатывает полученный запрос и возвращает клиенту результат.

Преимущества данной архитектуры:

- разделение программного кода клиентского и серверного приложений;
- основная часть вычислительных операций выполняется на сервере;
- разделение задач разработки и, следовательно, более быстрое ее выполнение;
- гибкость и масштабируемость.

Для описания архитектуры проектируемого программного средства была разработана диаграмма компонентов, которая представлена на рисунке 11.

В системе обработки и анализа индивидуальных заказов на изготовление лекарственных средств представлено клиентское приложение и сервер.

Клиентское приложение включает в себя такие компоненты как пользовательский интерфейс, система доступа к данным, система обработки данных. Задачами пользовательского интерфейса заключается введение данных пользователем, формирование пользовательского запроса и вывод на экран пользователя информации в удобном виде. На стороне клиента содержатся все инструменты для взаимодействия с БД: добавления, изменения и удаления записей.

В качестве сервера выступает СУБД, которая предоставляет доступ метаданным БД: заказы на изготовление лекарства по рецепту, заказы на недостающие ингредиенты, данные о пользователях, медикаментах и поставщиках, электронные документы. СУБД принимает запрос от приложения пользователя и выполняет необходимые действия с данными, после чего возвращает результат.

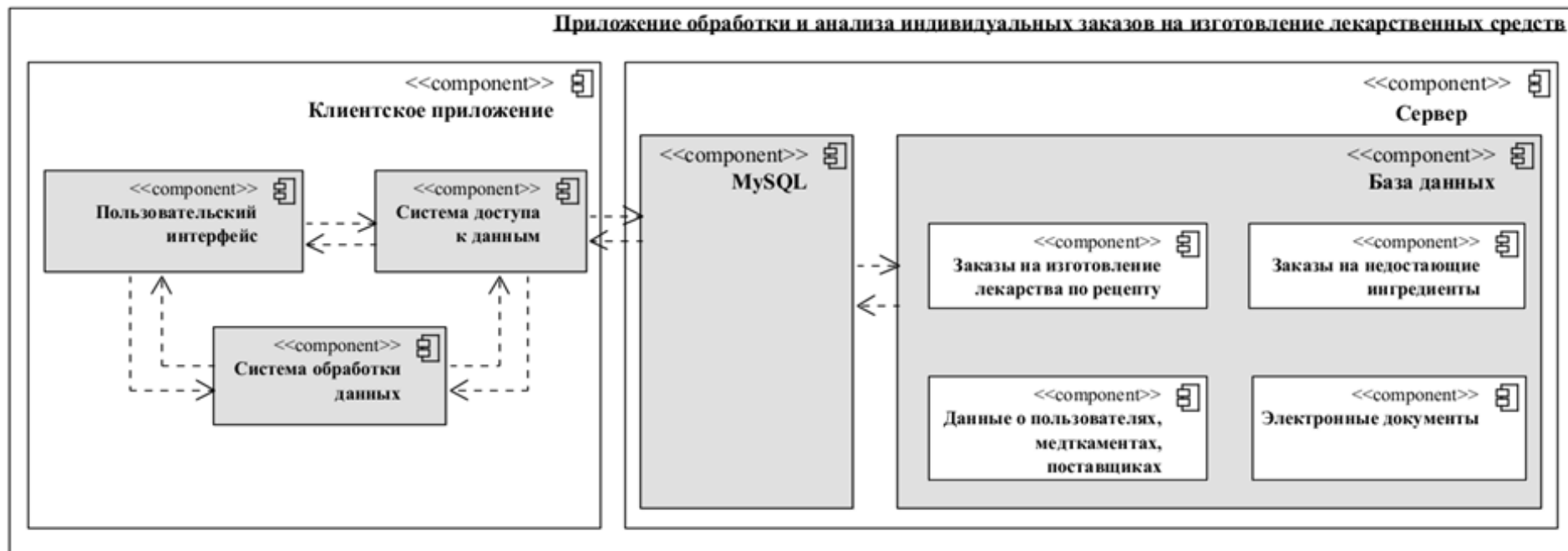


Рисунок 11 – Диаграмма компонентов приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств

Таким образом, разработанная программная архитектура клиент-серверного приложения обеспечивает высокую производительность, безопасность и готовность к масштабированию.

2.4 Разработка прототипа пользовательского интерфейса

Прототип интерфейса пользователя разработан с помощью Windows Forms на С# в Visual Studio. В данной главе представлены грубые шаблоны форм приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств в процессе разработки.

Шаблон формы авторизации приложения представлен на рисунке 12.



Рисунок 12 – Шаблон формы авторизации

Компоненты, указанные выносками на рисунке 12, описаны в таблице 22.

Таблица 22 – Описание компонентов формы авторизации

№	Имя компонента	Тип компонента	Назначение
1	label	Надпись	Компонент для названия полей ввода информации
2	textbox	Текстовое поле	Компонент для ввода информации
3	panel	Панель	Компонент для рамки логина и пароля
4	button	Кнопка	Кнопка для входа в приложения и для регистрации
5	pictureBox	Картинка	Компонент, который позволяет отображать изображения на форме

На форме авторизации расположены элементы для входа или регистрации в приложение. На отдельной панели размещены поля для ввода сотрудников логина и пароля. Клиенту нужно будет ввести фамилию и серию, и номер паспорта для входа. После того как пользователь введет свои данные и нажмет на кнопку «Вход», он попадает в главное меню приложения.

Шаблон формы главного меню приложения представлен на рисунке 13.



Рисунок 13 – Шаблон формы главного меню приложения

Компоненты, указанные выносками на рисунке 13, описаны в таблице 23.

Таблица 23 – Описание компонентов формы меню

№	Имя компонента	Тип компонента	Назначение
1	menustrip	Меню	Компонент, который позволяет создать меню программы
2	panel	Панель	Компонент для рамки с кнопками приложения
3	button	Кнопка	Кнопки, которые расположены на панели
4	pictureBox	Картинка	Компонент, который позволяет отображать изображения на форме

На главной форме расположено много различных элементов, выполняющих те или иные задачи. Главная рабочая область представлена в справа приложения. На ней будут отображаться данные о сотрудниках организации, клиентах, которые заказывали препарат на изготовления, заказы клиентов, а также в кнопке «Заказы» будут заказы у поставщиков, если

пользователь вошел как провизор, препараты и выход из приложения. Также на главной форме вверху есть меню приложения.

Рассмотрим форму «Заказы», на которую можно попасть из главной формы приложения. На рисунке 14 изображена шаблон формы «Заказы».

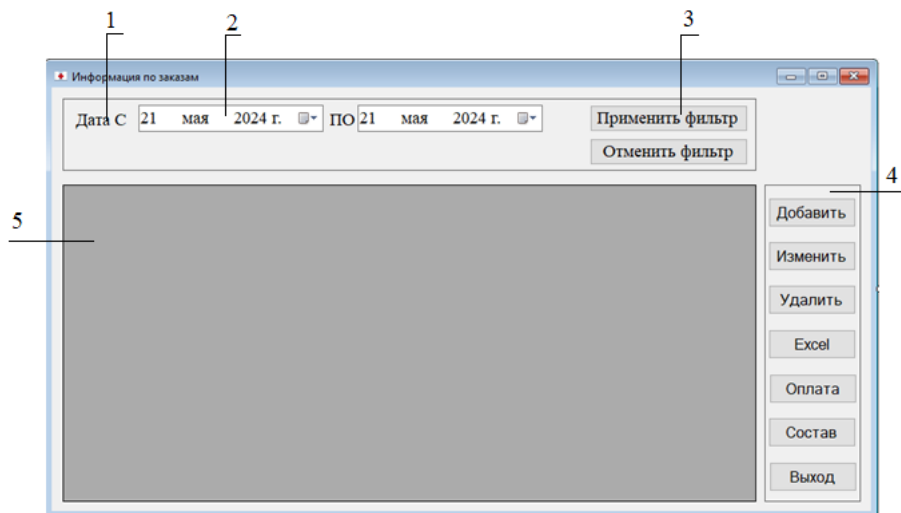


Рисунок 14 – Шаблон формы отображения заказов пользователей

Компоненты, указанные выносками на рисунке 14, описаны в таблице 24.

Таблица 24 – Описание компонентов формы заказов от клиентов

№	Имя компонента	Тип компонента	Назначение
1	label	Надпись	Компонент для названия полей ввода дат
2	datetimapicker	Даты	Компонент для ввода дат
3	button	Кнопка	Кнопки, которые расположены на панели
4	panel	Панель	Компонент для рамки с кнопками приложения
5	datagridview	Таблица	Компонент для отображения данных по заказам в табличном виде

На этой форме будут отображаться заказы клиентов. Для удобства можно применить фильтр по датам, тогда заказы клиентов будут отображаться в определенные даты. Справа изображена панель с кнопками, где можно модифицировать конкретный заказ клиента: добавить, изменить, удалить. Для отражения данных по оформленному заказу клиентом на изготовление лекарственных средств используется печатная форма заказа, если нажать на кнопку «Excel». Шаблон печатной формы можно посмотреть в разделе 1.1.3 Классы информационных объектов и документов в процессе обработки и

анализа индивидуальных заказов на изготовление лекарственных средств (рисунок 4).

Рассмотрим шаблон формы «Состав заказа», который изображен на рисунке 15.

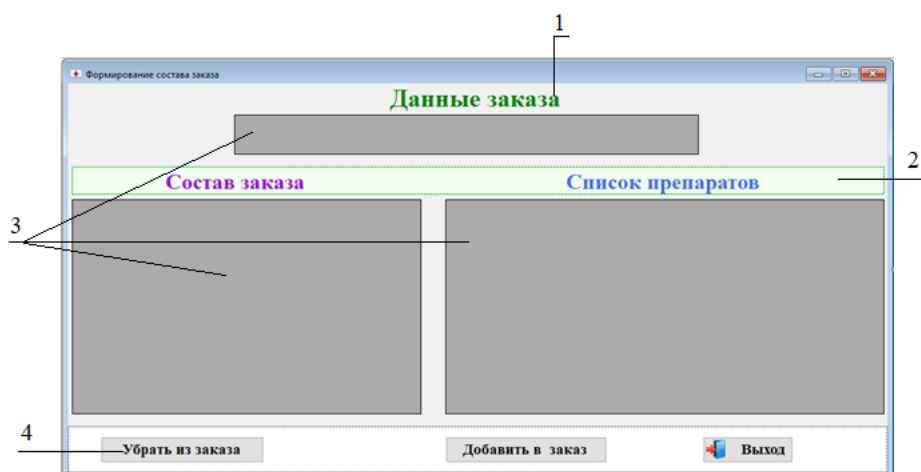


Рисунок 15 – Шаблон формы «Состав заказа»

Компоненты, указанные выносками на рисунке 15, описаны в таблице 25.

Таблица 25 – Описание компонентов формы состава заказа

№	Имя компонента	Тип компонента	Назначение
1	label	Надпись	Компонент для названия таблиц
2	panel	Панель	Компонент для рамки с названиями таблиц
3	datagridview	Таблица	Компонент для отображения данных о составе заказа
4	button	Кнопка	Кнопки, которые расположены на панели

На главной форме выбираем из списка заказов конкретный заказ клиента и переходим на форму состава заказа. На данной форме вверху будет фиксироваться выбранный заказ с данными о нем. Таблица справа показывает список препаратов, из которых будет изготавливаться лекарство из заказа. В таблицу слева можно добавить позицию из списка препаратов или удалить.

В данном разделе были описаны прототипы пользовательского интерфейса и их компоненты.

2.5 Разработка требований к аппаратно-программному окружению

Для разработки приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств необходимы следующие минимальные требования к программно-аппаратным средствам:

- операционная система: Microsoft Windows 7, 8.1, 10;
- среда разработки Microsoft Visual Studio v.19 или выше;
- фреймворк Microsoft Framework .NET v.4.8 или выше;
- введение проекта в git для возможности соединения приложения пользователя и программных средств взаимодействия с СУБД.

Техническое обеспечение системы должно максимально и наиболее эффективным образом использовать существующие в автоматизированной информационной системе технические средства, такие как сервер БД и ПК пользователей.

Требования к техническим характеристикам ПК пользователя:

- процессор: Intel Core i3 или эквивалентный;
- оперативная память: 4 ГБ;
- жесткий диск: 500 МБ свободного пространства;
- сетевой адаптер – 100 Мбит.

Требования к техническим характеристикам системы хранения данных: дисковая подсистема 2,5 Тб.

Требования к техническим характеристикам сервера БД:

- процессор: Intel Core i3 или эквивалентный;
- оперативная память: 8 ГБ;
- жесткий диск: 20 ГБ свободного пространства;
- сетевой адаптер – 100 Мбит.

Тщательное соблюдение требований к аппаратно-программному окружению обеспечит стабильную и производительную работу приложения.

2.6 Обоснование выбора средств разработки

2.6.1 Обоснование выбора средств автоматизации

При разработке информационных систем, одним из ключевых этапов является моделирование и проектирование базы данных. Для обеспечения эффективности, надежности и масштабируемости системы необходимо использовать средства автоматизации, которые позволяют создавать, управлять и оптимизировать модели данных.

ERwin Data Modeler является одним из наиболее популярных инструментов для проектирования и моделирования баз данных. Его использование обосновано рядом преимуществ, которые обеспечивают высокую производительность и качество работы с данными [14].

Преимущества ERwin Data Modeler:

- понятный интерфейс;
- поддержка множества СУБД;
- автоматическая генерация SQL-кода;
- мощные средства автоматизации и анализа данных.

Существуют различные инструменты для моделирования данных, такие как Microsoft Visio, IBM InfoSphere Data Architect, MySQL Workbench и другие. Однако, ERwin Data Modeler имеет ряд преимуществ, которые выделяют его среди конкурентов.

Visio позволяет создавать диаграммы баз данных, его основное предназначение заключается в создании диаграмм общего назначения [15]. В отличие от ERwin, Visio не предоставляет такой же уровень специализированных функций для анализа и оптимизации баз данных.

IBM InfoSphere Data Architect является мощным решением для моделирования данных, но его использование может быть связано с более высокой стоимостью и сложностью обучения [16]. ERwin Data Modeler предлагает более удобный интерфейс и лучшее соотношение цена/качество.

MySQL Workbe – это бесплатный инструмент, предназначенный для работы с MySQL, однако его функциональность ограничена по сравнению с ERwin. ERwin предлагает более широкий набор функций и поддержку множества СУБД, что делает его более универсальным инструментом [17].

Все вышеуказанные преимущества обеспечивают высокую производительность и качество разработки информационных систем, что делает ERwin Data Modeler идеальным выбором.

2.6.2 Обоснование выбора инструментальных средств разработки программного обеспечения

Наиболее популярной и удобной средой для разработки на языке C# является интегрированная среда разработки Microsoft Visual Studio 2022. Эта среда обеспечивает полную поддержку на всех этапах, от написания кода до его внедрения. На сегодняшний день Visual Studio является одним из самых мощных инструментов, созданных компанией Microsoft. Для удобной работы с языком C# Visual Studio поддерживает архитектуру .NET Framework, что делает процесс разработки быстрым и удобным. Главной особенностью платформы .NET является общезыковая среда исполнения Common Language Runtime (CLR), которая упрощает разработку на различных языках программирования [18].

Некоторые возможности Visual Studio включают:

- поддержка множества языков: C#, VB.NET, C++, F#, Delphi.NET;
- обширная библиотека классов;
- кроссплатформенность: разработка для Windows, MacOS, Linux, Android, iOS, Tizen;
- быстрый доступ к любому символу, типу, элементу или файлу;
- возможность просмотра истории изменений, авторов, дат изменений в строке;
- подключение к облачному хранению данных;
- пакет Entity Framework для удобного подключения к различным СУБД;

– множество возможностей для веб-разработки.

Этот список представляет лишь часть преимуществ разработки с использованием Visual Studio и .NET Framework. Платформа продолжает развиваться, предлагая множество инструментов для повышения эффективности разработки.

Приведем сравнение выбранной среды разработки Visual Studio с остальными средами в таблице 26, максимальная оценка за критерий равна десяти баллам.

Таблица 26 – Сравнение средств разработки программ

Критерий для сравнения	C++Builder 2010	NetBeans 8.2	Dev C++ 5	Visual Studio 2022
Надежность	5	7	6	10
Эффективность	6	6	5	8
Сопровождение	4	6	4	10
Портативность	7	7	7	9
Функциональность	5	7	6	10
Удобная работа с базами данных	8	6	5	9

Получаем из таблицы 26, что среда разработки Visual Studio более подходит для разработки нового программного продукта. Именно поэтому для создания системы была выбрана среда Microsoft Visual Studio.

2.6.3 Обоснование выбора языковых средств разработки

Для разработки новой системы была выбрана среда разработки Visual Studio и язык программирования C#. C# – молодой, но широко используемый язык программирования, относящийся к Си-семейству.

Одним из ключевых нововведений платформы .NET была технология ASP.NET (Active Server Pages), предназначенная для создания серверных веб-приложений с возможностью взаимодействия с базами данных. Специально для ASP.NET был разработан язык программирования C#, который стал основным языком разработки на этой платформе. Более того, сама технология ASP.NET была полностью написана на C# [19].

Си Шарп поддерживает множество полезных функций, таких как:

- инкапсуляция,
- наследование,
- полиморфизм,
- перегрузка операторов,
- статическая типизация.

При этом язык все еще активно развивается, и с каждой новой версией появляются новые интересные возможности, такие как лямбда-выражения, динамическое связывание, асинхронные методы и другие.

C# имеет синтаксис, похожий на C, и в этом отношении он близок к C++ и Java. В качестве объектно-ориентированного языка C# унаследовал многое от Java и C++. Как и Java, C# изначально был разработан для веб-приложений, и около 75% его синтаксических возможностей схожи с Java. C# также называют «очищенной версией Java». Еще 10% возможностей он заимствовал у C++, а 5% – у Visual Basic. Оставшиеся 10% – это уникальные идеи разработчиков. Благодаря объектно-ориентированному подходу C# позволяет создавать крупные, но гибкие, масштабируемые и расширяемые приложения.

2.6.4 Обоснование выбора средств управления данными

В качестве хранилища данных была выбрана СУБД MySQL. Приведем описание данной СУБД.

MySQL – это популярная система управления базами данных. Первая версия этой СУБД была выпущена в 1994 году компанией MySQL AB. В 2008 году эту компанию приобрела Sun Microsystems, которую, в свою очередь, приобрела Oracle в 2010 году [20].

Рассмотрим ряд преимуществ СУБД MySQL:

- производительность;
- кроссплатформенность;
- бесплатная СУБД;
- выбор движков.

Для веб-приложений важна скорость работы. Пользователь не будет ждать загрузки страницы – он просто уйдет с неё, если приложение не будет работать достаточно быстро. В плане производительности приложения «болезненным» местом является база данных, и MySQL отлично справляется с этой проблемой.

MySQL предоставляет кроссплатформенную базу данных: она работает на Linux, FreeBSD и конечно на Windows. Этот критерий нужно учитывать при выборе СУБД для проектов, нацеленных на несколько платформ, в частности веб-приложений. К слову, MySQL входит в состав популярных стеков для разработки веб-приложений – LAMP (Linux, Apache, MySQL, PHP) и WAMP (Windows, Apache, MySQL, PHP).

Oracle развивает MySQL как проект с открытым исходным кодом, что позволяет сообществу вносить в него свой вклад и, при необходимости, адаптировать его под свои задачи. СУБД предоставляется для конечных пользователей бесплатно. Многие компании выбирают в качестве СУБД MySQL из-за того, что она бесплатна.

Oracle предлагает клиентам некоторые проприетарные модули с закрытым кодом, для использования которых потребуется денежное вложение. Но эти модули нужны не всем и не всегда.

Движок базы данных (database engine) – это программный модуль, с помощью которого СУБД создает, читает и обновляет данные в базе. В MySQL есть два типа движков: транзакционные и нетранзакционные. Транзакционные движки хранения данных подразумевают, что операции в этих БД могут быть отменены, если они не завершены.

Проведем небольшое сравнение выбранной СУБД MsSQLServer с остальными системами управления базами данных в таблице 27.

Таблица 27 – Сравнительная характеристика СУБД

Критерий для сравнения	MsSQLServer 2012	MsAccess 2010	MySQL 8.0
Возможность расширения (масштабирования)	+	-	+
Возможность настраивать различной сложности SQL запросы, используемые в программе	+	-	+
Быстрая работа с большими таблицами	-	-	+

Продолжение таблицы 27

Критерий для сравнения	MsSQLServer 2012	MsAccess 2010	MySQL 8.0
Легкость в настройке параметров СУБД.	-	+	+
Поддержка транзакций.	+	-	+

В результате проведенного анализа по вышеизложенным критериям была выбрана СУБД MySQL.

3 Разработка программного обеспечения

3.1 Разработка физической схемы базы данных

При построении физической модели в данной бакалаврской работе была использована программа для проектирования баз данных ERwin Data Modeler. При отображении сущностей концептуальной модели в их физические эквиваленты была получена физическая модель, представленная в приложении Г.

На рисунке 16 приведена схема модели базы данных, созданная средствами СУБД MySQL.

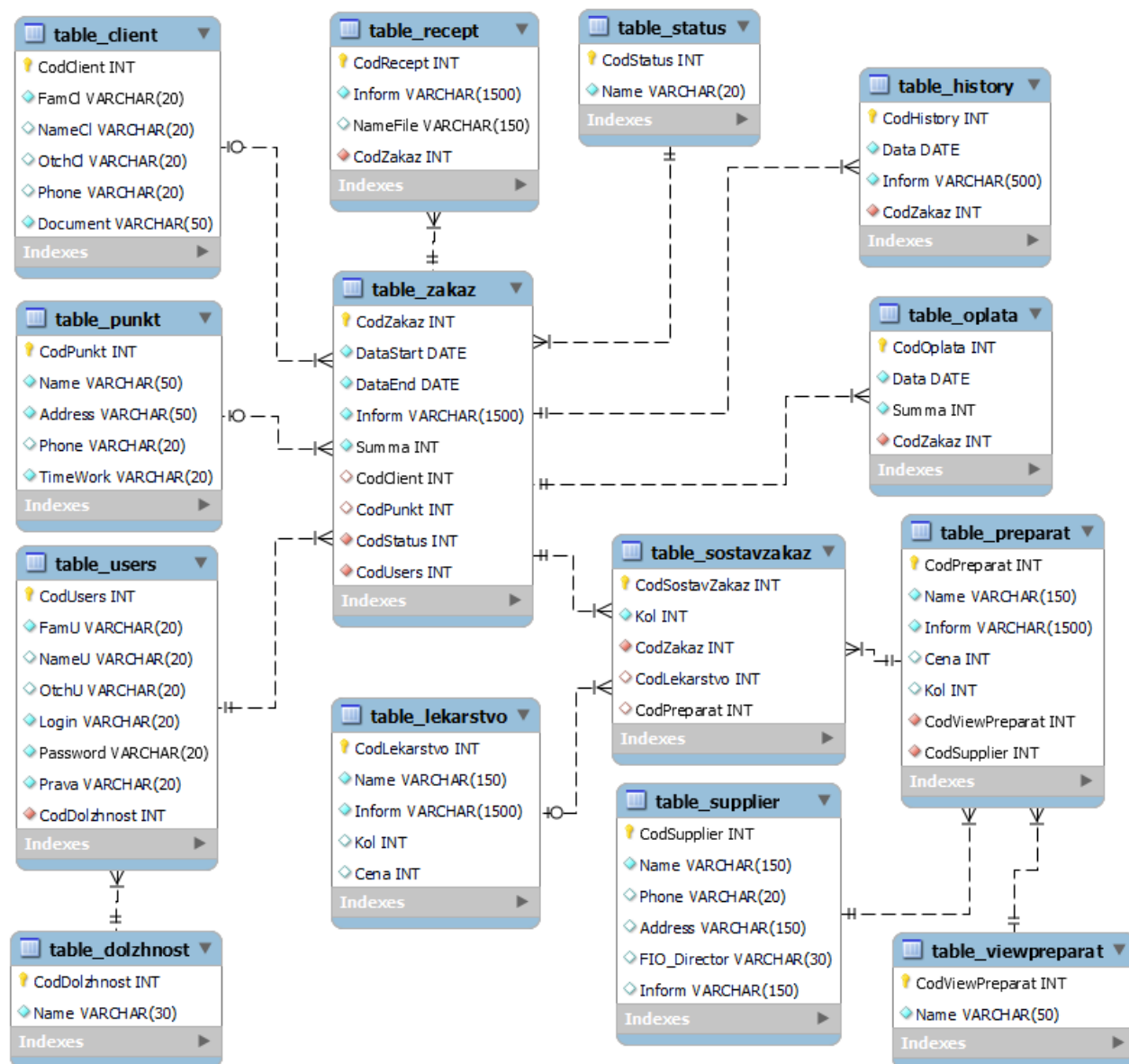


Рисунок 16 – Схема модели БД (MySQL)

Приведем полное описание атрибутов для каждой из 14 сущностей модели.

Таблица «Table_Lekarstvo» предназначена для хранения информации о лекарствах, которые изготавливаются медицинской организацией. В таблице 28 приведено описание атрибутов данной таблицы.

Таблица 28 – Атрибуты таблицы «Table_Lekarstvo»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_Lekarstvo. CodLekarstvo (Код лекарства)	integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.
2	Table_Lekarstvo. Name (Название)	varchar(150)	НЕТ	Хранится название лекарственного средства.
3	Table_Lekarstvo. Inform (Информация)	varchar(1500)	НЕТ	Хранится описание лекарственного средства.
4	Table_Lekarstvo. Kol (Количество)	integer	НЕТ	Хранится количество лекарственного средства, которое имеется в наличии.
5	Table_Lekarstvo. Cena (Цена)	integer	НЕТ	Хранится стоимость лекарственного средства.

Таблица «Table_Supplier» предназначена для хранения справочных данных о поставщиках ингредиентов (медицинских препаратов). В таблице 29 приведено описание атрибутов данной таблицы.

Таблица 29 – Атрибуты таблицы «Table_Supplier»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_Supplier. CodSupplier (Код поставщика)	integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.
2	Table_Supplier. Name (Название)	varchar(150)	НЕТ	Хранится название поставщика лекарственных препаратов.
3	Table_Supplier. Phone (Телефон)	varchar(20)	НЕТ	Хранится информация о номере телефона поставщика.
4	Table_Supplier. Address (Адрес)	varchar(150)	НЕТ	Хранится адрес нахождения организации поставщика.

Продолжение таблицы 29

№	Обозначение (название)	Тип данных	Ключ	Описание
5	Table_Supplier. FIO_Director (ФИО директора)	varchar(30)	НЕТ	Хранится информация о ФИО директора поставщика.
6	Table_Supplier. Inform (Информация)	varchar(150)	НЕТ	Хранится дополнительная информация о поставщике.

Таблица «Table_Users» предназначена для хранения информации о сотрудниках медицинской организации, которые работают фармацевтами и провизорами. В таблице 30 приведено описание атрибутов данной таблицы.

Таблица 30 – Атрибуты таблицы «Table_Users»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_Users. CodUsers (Код сотрудника)	Integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.
2	Table_Users. FamU (Фамилия)	varchar(20)	НЕТ	Хранится фамилия сотрудника.
3	Table_Users. NameU (Имя)	varchar(20)	НЕТ	Хранится имя сотрудника.
4	Table_Users. OtchU (Отчество)	varchar(20)	НЕТ	Хранится отчество сотрудника.
5	Table_Users. Login (Логин)	varchar(20)	НЕТ	Хранится логин сотрудника для авторизации в систему.
6	Table_Users. Password (Пароль)	varchar(20)	НЕТ	Хранится пароль сотрудника для авторизации в систему.
7	Table_Users. Prava (Права)	varchar(20)	НЕТ	Хранится значение прав сотрудника для работы в системе.
8	Table_Users. CodDolzhnost (Код должности)	Integer	Вторичный	Хранится связь с сущностью Table_Dolzhnost, которая указывает, по какую должность занимает данный сотрудник.

Таблица «Table_ViewPreparat» предназначена для хранения справочных данных о видах препаратов (ингредиентов), поставляемых поставщиками в медицинскую организацию. В таблице 31 приведено описание атрибутов данной таблицы.

Таблица 31 – Атрибуты таблицы «Table_ViewPreparat»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_ViewPreparat. CodViewPreparat (Код вида препарата)	Integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.
2	Table_ViewPreparat. Name (Название)	varchar(50)	НЕТ	Хранится название вида препарата.

Таблица «Table_Preparat» предназначена для хранения информации о препаратах. В таблице 32 приведено описание атрибутов данной таблицы.

Таблица 32 – Атрибуты таблицы «Table_Preparat»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_Preparat. Cod Preparat (Код препарата)	Integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.
2	Table_Preparat. Name (Название)	varchar(150)	НЕТ	Хранится название препарата.
3	Table_Preparat. Inform (Информация)	varchar(1500)	НЕТ	Хранится описание препарата.
4	Table_Preparat. Cena (Цена)	integer	НЕТ	Хранится стоимость препарата.
5	Table_Preparat. Kol (Количество)	integer	НЕТ	Хранится количество препарата (в наличие).
6	Table_Preparat. CodSupplier (Код поставщика)	Integer	Вторичный	Хранится связь с сущностью Table_Supplier, которая указывает, какой поставщик поставил данный препарат в мед.организацию.
7	Table_Preparat. CodViewPreparat (Код вида препарата)	Integer	Вторичный	Хранится связь с сущностью Table_ViewPreparat, которая указывает, к какому виду относится препарат.

Таблица «Table_Client» предназначена для хранения информации о клиентах, которые оформляют заказы на изготовление лекарственных средств. В таблице 33 приведено описание атрибутов данной таблицы.

Таблица 33 – Атрибуты таблицы «Table_Client»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_Client. CodClient (Код клиента)	Integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.
2	Table_Client. FamCl (Фамилия)	varchar(20)	НЕТ	Хранится фамилия клиента.
3	Table_Client. NameCl (Имя)	varchar(20)	НЕТ	Хранится имя клиента.
4	Table_Client. OtchCl (Отчество)	varchar(20)	НЕТ	Хранится отчество клиента.
5	Table_Client. Phone (Телефон)	varchar(20)	НЕТ	Хранится телефон клиента.
6	Table_Client. Document (Документ)	varchar(50)	НЕТ	Хранится паспортные данные клиента.

Таблица «Table_Status» предназначена для хранения справочных данных о статусах заказов. В таблице 34 приведено описание атрибутов данной таблицы.

Таблица 34 – Атрибуты таблицы «Table_Status»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_Status. Cod Status (Код статуса)	Integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.
2	Table_Status. Name (Название)	varchar(20)	НЕТ	Хранится название статуса.

Таблица «Table_Oplata» предназначена для хранения информации об оплатах за заказы. В таблице 35 приведено описание атрибутов данной таблицы.

Таблица 35 – Атрибуты таблицы «Table_Oplata»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_Oplata. CodOplata (Код оплаты)	Integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.

Продолжение таблицы 35

№	Обозначение (название)	Тип данных	Ключ	Описание
2	Table_Oplata. Data (Дата)	Datetime	НЕТ	Хранится дата оплаты заказа.
3	Table_Oplata. Summa (Сумма)	Integer	НЕТ	Хранится сумма оплаты.
4	Table_Oplata. CodZakaz (Код заказа)	Integer	Вторичный	Хранится связь с сущностью Table_Zakaz, которая указывает, по какому заказу произошла оплата.

Таблица «Table_Zakaz» предназначена для хранения информации о заказах клиентов. В таблице 36 приведено описание атрибутов данной таблицы.

Таблица 36 – Атрибуты таблицы «Table_Zakaz»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_Zakaz. CodZakaz (Код заказа)	Integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.
2	Table_Zakaz. DataStart (Дата начала)	Datetime	НЕТ	Хранится дата оформления заказа.
3	Table_Zakaz. DataEnd (Дата окончания)	Datetime	НЕТ	Хранится дата окончания заказа.
4	Table_Zakaz. Summa (Сумма)	Integer	НЕТ	Хранится сумма заказа.
5	Table_Zakaz. Inform (Информация)	varchar(1500)	НЕТ	Хранится дополнительная информация по заказу.
6	Table_Zakaz. CodClient (Код клиента)	Integer	Вторичный	Хранится связь с сущностью Table_Client, которая указывает, кому из клиентов принадлежит данный заказ.
7	Table_Zakaz. Cod Punkt (Код пункта)	Integer	Вторичный	Хранится связь с сущностью Table_Punkt, которая указывает, в каком пункте будет выдан заказ.
8	Table_Zakaz. CodStatus (Код статуса)	Integer	Вторичный	Хранится связь с сущностью Table_Status, которая указывает, в каком статусе находится данный заказ.

Продолжение таблицы 36

№	Обозначение (название)	Тип данных	Ключ	Описание
9	Table_Zakaz. CodUsers (Код сотрудника)	Integer	Вторичный	Хранится связь с сущностью Table_Users, которая указывает, кто из сотрудников создал заказ.

Таблица «Table_SostavZakaz» предназначена для хранения информации о составе заказа. В таблице 37 приведено описание атрибутов данной таблицы.

Таблица 37 – Атрибуты таблицы «Table_SostavZakaz»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_SostavZakaz. CodSostavZakaz (Код состава заказа)	Integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.
2	Table_SostavZakaz. Kol (Количество)	Integer	НЕТ	Хранится количество выбранного лекарственного средства или препарата.
3	Table_SostavZakaz. CodLekarstvo (Код лекарства)	Integer	Вторичный	Хранится связь с сущностью Table_Lekarstvo, которая указывает, какое лекарственное средство входит в состав заказа.
4	Table_SostavZakaz. CodPreparat (Код препарата)	Integer	Вторичный	Хранится связь с сущностью Table_Preparat, которая указывает, какой препарат входит в состав заказа.
5	Table_SostavZakaz. CodZakaz (Код заказа)	Integer	Вторичный	Хранится связь с сущностью Table_Zakaz, которая указывает, к какому заказу принадлежат лекарства и препараты.

Таблица «Table_History» предназначена для хранения информации об истории по заказу, которая содержит вторичные ключи лекарства, препарата и заказа. В таблице 38 приведено описание атрибутов данной таблицы.

Таблица 38 – Атрибуты таблицы «Table_History»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_History. CodHistory (Код истории)	Integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.

Продолжение таблицы 38

№	Обозначение (название)	Тип данных	Ключ	Описание
2	Table_History. Data (Дата)	DateTime	НЕТ	Хранится дата выполненного действия по заказу.
3	Table_History. Inform (Информация)	varchar(500)	НЕТ	Хранится описание действия.
4	Table_History. CodZakaz (Код заказа)	Integer	Вторичный	Хранится связь с сущностью Table_Zakaz, которая указывает, по какому заказу было выполнено действие.

Таблица «Table_Punkt» предназначена для хранения информации о пунктах выдачи заказов. В таблице 39 приведено описание атрибутов данной таблицы.

Таблица 39 – Атрибуты таблицы «Table_Punkt»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_Punkt. CodPunkt (Код пункта)	Integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.
2	Table_Punkt. Name (Название)	varchar(50)	НЕТ	Хранится название пункта выдачи.
3	Table_Punkt. Address (Адрес)	varchar(50)	НЕТ	Хранится адрес пункта выдачи.
4	Table_Punkt. Phone (Телефон)	varchar(20)	НЕТ	Хранится номер телефона из пункта выдачи.
5	Table_Punkt. TimeWork (Время работы)	varchar(20)	НЕТ	Хранится время работы пункта выдачи.

Таблица «Table_Recept» предназначена для хранения информации о рецептах от клиентов на изготовление лекарственных средств. В этой таблице хранится описание рецепта и название файла с рецептом, а также первичный ключ таблицы и первичный ключ заказа от клиента. В таблице 40 приведено описание атрибутов данной таблицы.

Таблица 40 – Атрибуты таблицы «Table_Recept»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_Recept. CodRecept (Код рецепта)	Integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.
2	Table_Recept. Inform (Информация)	varchar(1500)	НЕТ	Хранится описание рецепта.
3	Table_Recept. NameFile (Имя файла)	varchar(150)	НЕТ	Хранится название файла с рецептом.
4	Table_Recept. CodZakaz (Код заказа)	Integer	Вторичный	Хранится связь с сущностью Table_Zakaz, которая указывает, по какому заказу указан данный рецепт на изготовление лекарственного средства.

Таблица «Table_Dolzhnost» предназначена для хранения справочной информации о названиях должностей сотрудников медицинской организации. В таблице 41 приведено описание атрибутов данной таблицы.

Таблица 41 – Атрибуты таблицы «Table_Dolzhnost»

№	Обозначение (название)	Тип данных	Ключ	Описание
1	Table_Dolzhnost. CodDolzhnost (Код должности)	Integer	Первичный	Принцип работы счетчика: каждой новой записи данных присваивается значение, увеличенное на единицу.
2	Table_Dolzhnost. Name (Название)	varchar(30)	НЕТ	Хранится название должности.

В таблице 42 приведена информация о связях между таблицами, изображенными в приложении Г (физическая модель предметной области). Все связи имеют отношение «один-ко-многим».

Таблица 42 – Связи между таблицами

№	Родительская таблица	Дочерняя таблица	Вид связи
R_1	Table_Client (Клиенты)	Table_Zakaz (Заказы)	1-М
R_2	Table_Punkt (Пункты)	Table_Zakaz (Заказы)	1-М

Продолжение таблицы 42

№	Родительская таблица	Дочерняя таблица	Вид связи
R_3	Table_Users (Сотрудники)	Table_Zakaz (Заказы)	1-M
R_4	Table_Dolzhnost (Должности)	Table_Users (Сотрудники)	1-M
R_5	Table_Zakaz (Заказы)	Table_Recept (Рецепты)	1-M
R_6	Table_Status (Статусы)	Table_Zakaz (Заказы)	1-M
R_7	Table_Zakaz (Заказы)	Table_SostavZakaz (Состав заказа)	1-M
R_8	Table_Zakaz (Заказы)	Table_Oplata (Оплаты)	1-M
R_9	Table_Zakaz (Заказы)	Table_History (История)	1-M
R_10	Table_Lekarstvo (Лекарства)	Table_SostavZakaz (Состав заказа)	1-M
R_11	Table_Preparat (Препараты)	Table_SostavZakaz (Состав заказа)	1-M
R_12	Table_Supplier (Поставщики)	Table_Preparat (Препараты)	1-M
R_13	Table_ViewPreparat (Виды препаратов)	Table_Preparat (Препараты)	1-M

Разработка физической схемы базы данных играет важную роль в оптимизации производительности системы, обеспечивая эффективное хранение и доступ к данным. Правильно спроектированная физическая схема позволяет улучшить производительность запросов, обеспечить надежность и безопасность данных.

Было разработано 14 таблиц в СУБД MySQL, которые представлены на рисунке 17.

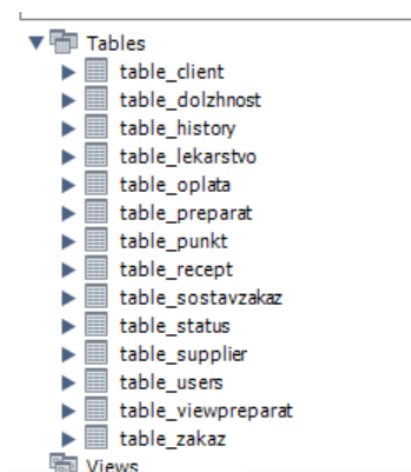


Рисунок 17 – Таблицы в БД

Все SQL-скрипты представлены на листинге Д.1 – Д.14 в приложении Д.

3.2 Разработка пользовательского интерфейса приложения

Интерфейс пользователя разработан с помощью Windows Forms на С# в Visual Studio. На рисунке 18 изображена схема иерархии окон приложения. Окна, выделенные желтым цветом, имеет доступ фармацевт. К окнам, выделенные светло-голубым цветом, имеет доступ провизор. Окна, выделенные белым цветом, имеют доступ все пользователи.

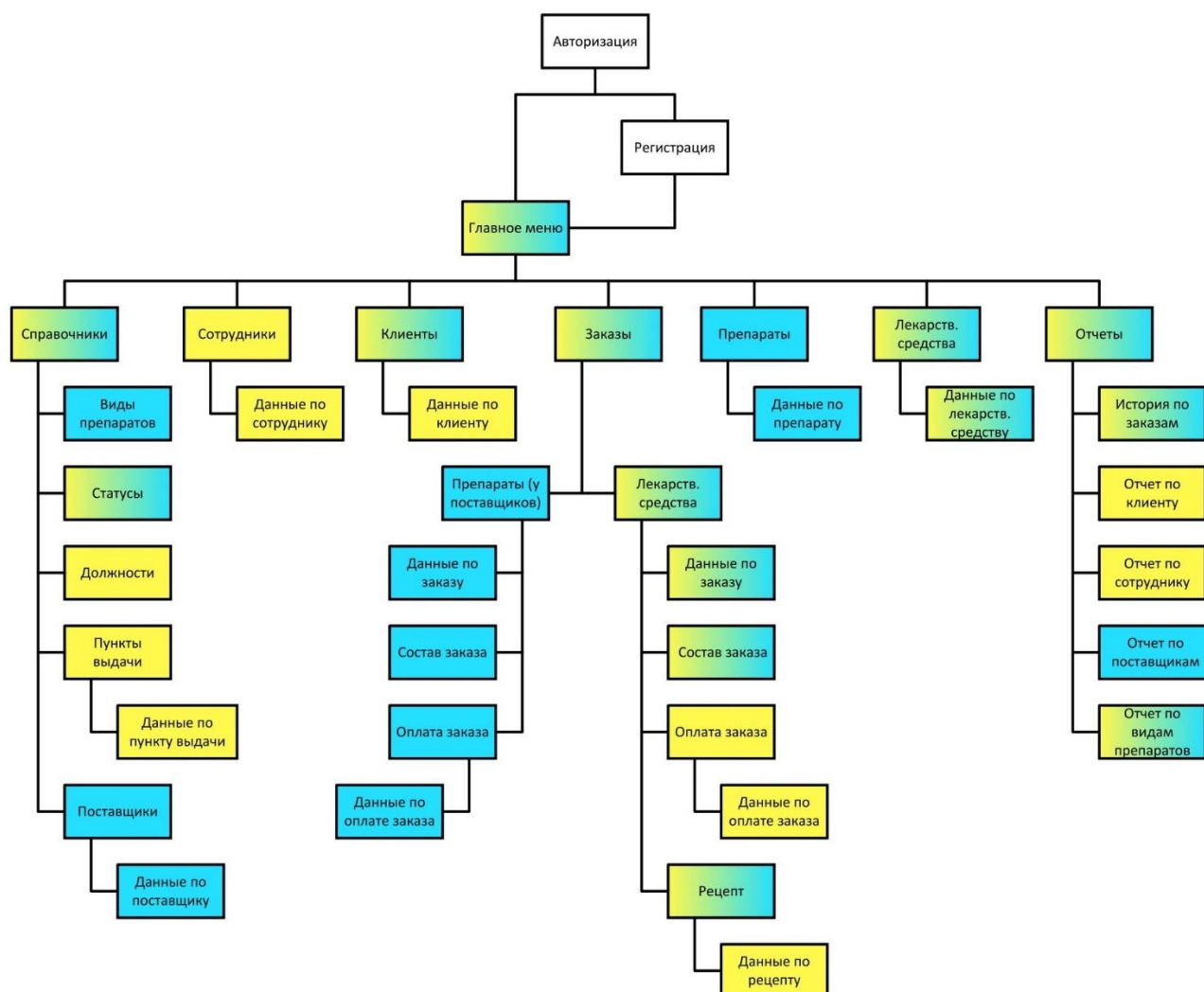


Рисунок 18 – Схема иерархии окон приложения

Разработан макет формы для авторизации в приложении, который представлен на рисунке 19.

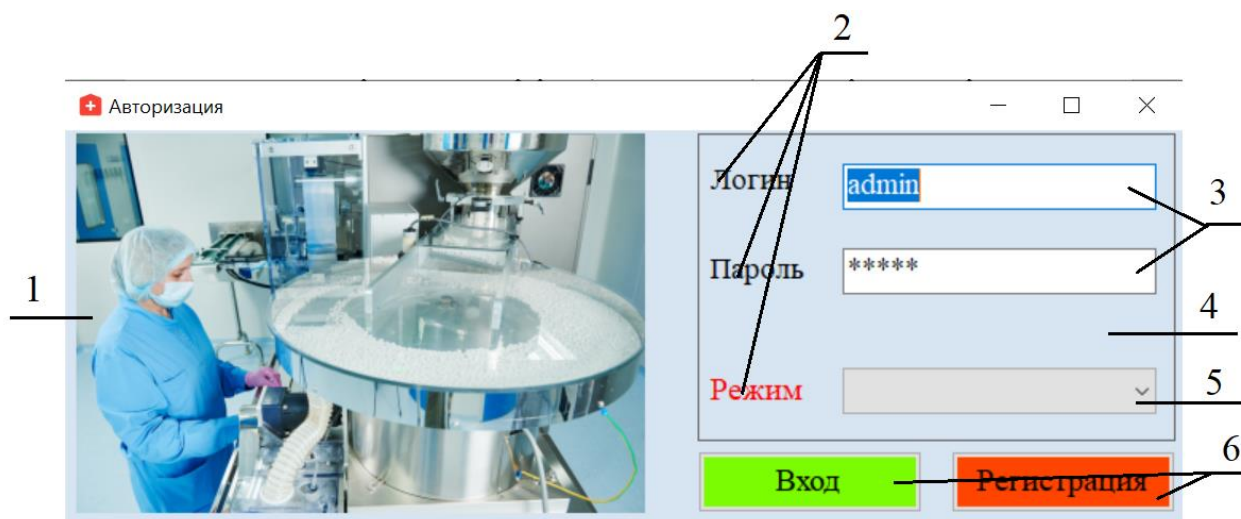


Рисунок 19 – Макет формы «Авторизация»

Рассмотрим основные элементы интерфейса формы авторизации:

- 1 – картинка, компонент pictureBox;
- 2 – надпись, компонент label;
- 3 – текстовое поле для входа в приложение, компонент textbox;
- 4 – панель для информации о входе, компонент panel;
- 5 – поле со списком пользователей, компонент combobox;
- 6 – кнопки входа и регистрации, компонент button.

С помощью компонента combobox указывается режим работы в приложении: Фармацевт, Провизор, Клиент или Администратор.

Если указан режим «Клиент», то при нажатии на кнопку «Регистрация» отобразится окно с данными для регистрации нового аккаунта клиента, иначе будет отображена форма для ввода данных сотрудника.

При установке режима «Клиент» в полях авторизации нужно указать: фамилию клиента и серию с номером паспорта (10 цифр бес пробелов).

При установке режимов «Провизор», «Фармацевт», «Администратор» в полях авторизации нужно указать: логин и пароль сотрудника.

При нажатии на кнопку «Вход» выполняются проверки на ввод значений в полях «Логин» и «Пароль» для сотрудников или «Фамилия» и «Серия, номер паспорта» для клиентов с учетом режима авторизации (компонент ComboBox).

Если авторизация в режиме сотрудника прошла успешно, то отображается главная форма программы.

Если авторизация в режиме клиента прошла успешно, то отображается форма личного кабинета клиента.

Для регистрации нового сотрудника создана специальная форма, которая вызывается с помощью обработчика кнопки «Регистрация». Обработчик кнопки представлен на листинге 1.

Листинг 1 – Обработчик кнопки «Регистрация» для сотрудника

```
FormUsersInfo newForm = new FormUsersInfo();
newForm.RegimUsers = 2; //2- режим создания
newForm.ShowDialog();
```

В переменной «RegimUsers» передается значение «2», которое означает, что будет выполняться алгоритм создания новой записи в таблице Table_Users (Сотрудники). Для редактирования данных сотрудника нужно инициализировать значением «1».

Для регистрации нового клиента форма вызывается с помощью обработчика кнопки «Регистрация». Обработчик кнопки представлен на листинге 2.

Листинг 2 – Обработчик кнопки «Регистрация» для клиента

```
FormClientInfo newForm = new FormClientInfo();
newForm.RegimClient = 2; //2- режим создания
newForm.ShowDialog();
```

В переменной «RegimClient» передается значение «2», которое означает, что будет выполняться алгоритм создания новой записи в таблице Table_Client (Клиенты). Для редактирования данных клиента нужно инициализировать значением «1».

Режим администратора для работы в приложении может быть указан любому сотруднику из фармацевтов или провизоров, которому будет предоставлен данный доступ.

Разработана форма для работы с меню программы, макет которой приведен на рисунке 20.



Рисунок 20 – Макет формы «Главное меню приложения»

Рассмотрим основные элементы интерфейса формы главного меню:

- 1 – картинка, компонент pictureBox;
- 2 – меню программы, компонент menustrip;
- 3 – панель для кнопок, компонент panel;
- 4 – кнопки для работы в приложении, компонент button.

На макете главной формы разработано меню программы (рисунок 20), в качестве пунктов меню использовался компонент «menuStrip».

С помощью обработчика событий формы и кнопок «KeyPress» отслеживается горячая клавиша ESC, по которой происходит выход из программы. Код работы кнопок «KeyPress» представлен на листинге 3.

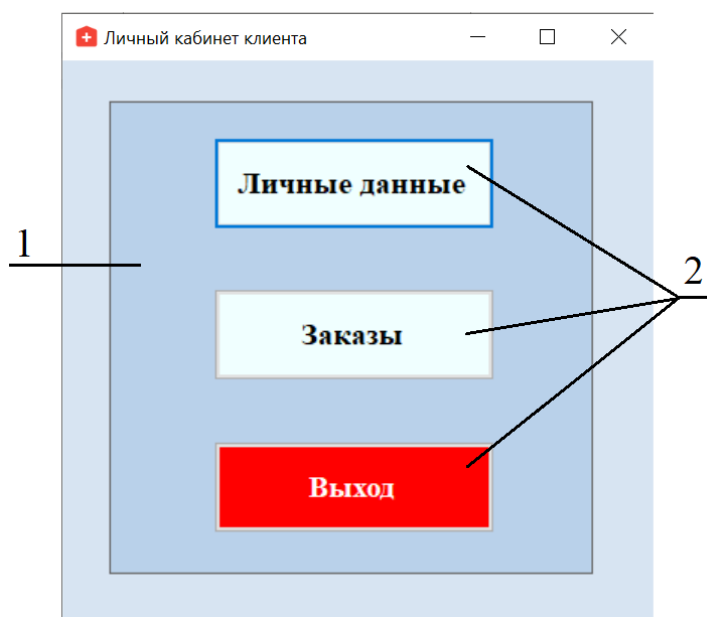
Листинг 3 – Работа кнопок «KeyPress»

```
if (e.KeyChar == (char)27)
{
    выходToolStripMenuItem_Click (sender, e);
}
```

Макет формы личного кабинета клиента приведен на рисунке 21.

Для управления разделами «Личные данные» и «Заказы» использовался компонент Button.

По кнопке «Выход» выполняется закрытие формы.



1 – панель для информации о входе, компонент panel;

2 – кнопки для работы в приложении, компонент button.

Рисунок 21 – Макет формы «Личный кабинет клиента»

Организацию работы со справочными данными рассмотрим на примере справочника «Виды препаратов». Данная форма разработана для отображения информации по видам препаратов, которые закупаются у поставщиков, макет формы представлен на рисунке 22.

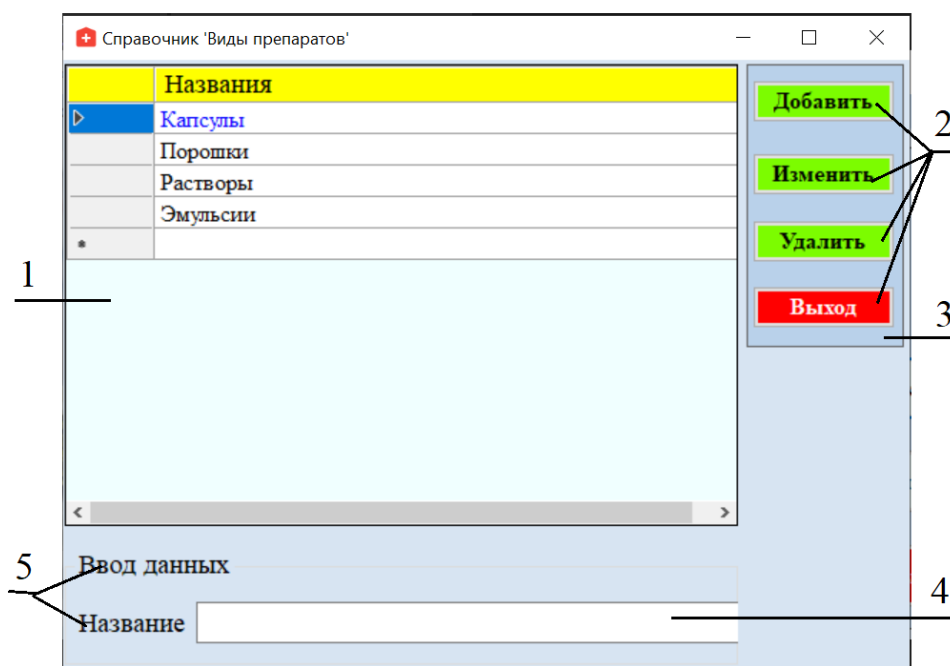


Рисунок 22 – Макет формы «Справочник»

Основные элементы интерфейса формы «Справочник»:

- 1 – таблица, где отображается информация, компонент Grid;
- 2 – кнопки для работы в приложении, компонент button;
- 3 – панель для кнопок, компонент panel;
- 4 – текстовое поле, компонент textbox;
- 5 – надпись, компонент label.

Информация по видам препаратов отображается в табличном виде с помощью компонента Grid.

Управление данными реализовано с помощью кнопок button:

- добавить – Кнопка_Add;
- изменить – Кнопка_Change;
- удалить – Кнопка_Delete.

Для удобства работы с программой используются горячие клавиши ESC и ENTER, которые реализованы на событиях «KeyPress» всех компонентов формы.

Форма для работы с заказами разработана для отображения информации по оформленным заказам на изготовление лекарственных средств и заказов препаратов у поставщиков, макет формы представлен на рисунке 23.

The screenshot shows a software form titled 'Информация по заказам'. It features a header section with filters for date, location, status, and client, followed by a table of orders and a sidebar with action buttons. Numbered annotations point to specific UI elements:

- 1: Points to the table of orders.
- 2: Points to the filter buttons (checkboxes) for 'Дата С' and 'Пункт'.
- 3: Points to the date input field for 'Дата С'.
- 4: Points to the status dropdown menu.
- 5: Points to the 'ON' and 'OFF' toggle switches.
- 6: Points to the 'Добавить', 'Изменить', and 'Удалить' buttons.
- 7: Points to the 'Состав', 'Рецепты', 'Отчет', and 'Выход' buttons.

	Код	Дата (Начало)	Дата (Конец)	Клиент	Сумма	Статус
>	10	08.06.2024	08.06.2024	Иванов Н.Е.	5000	Открыт
	8	04.06.2024	04.06.2024	Макина И.Н.	6660	Открыт
	7	01.06.2024	04.06.2024	Семенкина Е.А.	61800	Открыт
	6	01.06.2024	08.06.2024	Макина И.Н.	4760	Открыт
	1	28.05.2024	28.05.2024	Иванов Н.Е.	4500	Открыт
*						

Рисунок 23 – Макет формы «Информация по заказам»

Рассмотрим основные элементы интерфейса формы информации по заказам:

- 1 – таблица, где отображается информация, компонент Grid;
- 2 – выбор параметра фильтра для поиска, компонент CheckBox;
- 3 – поле для ввода дат, компонент DateTime;
- 4 – поле со списком, компонент combobox;
- 5 – кнопка фильтра, компонент buttonFilterOFF, buttonFilterON;
- 6 – кнопки для работы в приложении, компонент button;
- 7 – панель для кнопок, компонент panel.

Для поиска заказов можно воспользоваться данными: дата заказа, пункт выдачи, статус и клиент.

Алгоритм фильтра работает после нажатия кнопки «ON» (вкл. фильтр), для отключения фильтра используется кнопка «OFF» (выкл. фильтр).

Форма для формирования состава заказа представлена на рисунке 24.

The screenshot shows a software window titled "Формирование состава заказа". It contains several data entry sections:

- Annotation 1:** Points to a table titled "Состав заказа" (Order Composition) with columns: Название (Name), Кол (Quantity), and Цена (Price). The first row shows "Мезук ЕН" with quantity 1 and price 210.
- Annotation 2:** Points to a table titled "Данные заказа" (Order Data) with columns: Код (Code), Дата (Date), Сумма (Sum), Статус (Status), Пункт (Point), and Клиент (Client). The first row shows values: 10, 08.06.2024, 5000, Открыт, Колбасный цех, and Иванов Н.Е.
- Annotation 3:** Points to a table titled "Лекарств.средства" (Medicines) with columns: Название (Name), Кол (Quantity), and Цена (Price). The first row shows "Мезук ЕН" with quantity 0 and price 210. Other rows include "Мелука КК" (17, 99) and "Кетаферон" (2, 260).
- Annotation 4:** Points to a text input field labeled "Укажите количество:" (Specify quantity:).
- Annotation 5:** Points to a red button labeled "Выход" (Exit) in the bottom right corner.

At the bottom of the form, there are two green buttons: "Убрать из заказа" (Remove from order) and "Добавить в заказ" (Add to order).

Рисунок 24 – Макет формы «Формирование состава заказа»

Рассмотрим основные элементы интерфейса формы формирования состава заказа:

- 1 – таблица, где отображается информация, компонент Grid;
- 2 – надпись, компонент label;

3 – панель, компонент panel;

4 – текстовое поле, компонент textbox;

5 – кнопки для работы в приложении, компонент button.

В обработчиках кнопок «Убрать из заказа» и «Добавить в заказ» разработаны проверки на количество препаратов и лекарственных средств. При добавлении в заказ выполняется проверка на имеющееся в наличии количество. При удалении из заказа анализируется имеющееся количество, которое находится в составе заказа.

Для валидации ввода значений в поле с количеством реализован обработчик «KeyPress», представленный на листинге 4.

Листинг 4 – Обработчик «KeyPress» для валидации ввода значений

```
char ch = e.KeyChar;
if (!char.IsDigit(ch) && ch != 8)
{
    e.Handled = true;
}
```

Разработана форма для отображения информации по всем лекарственным средствам, макет формы представлен на рисунке 25.

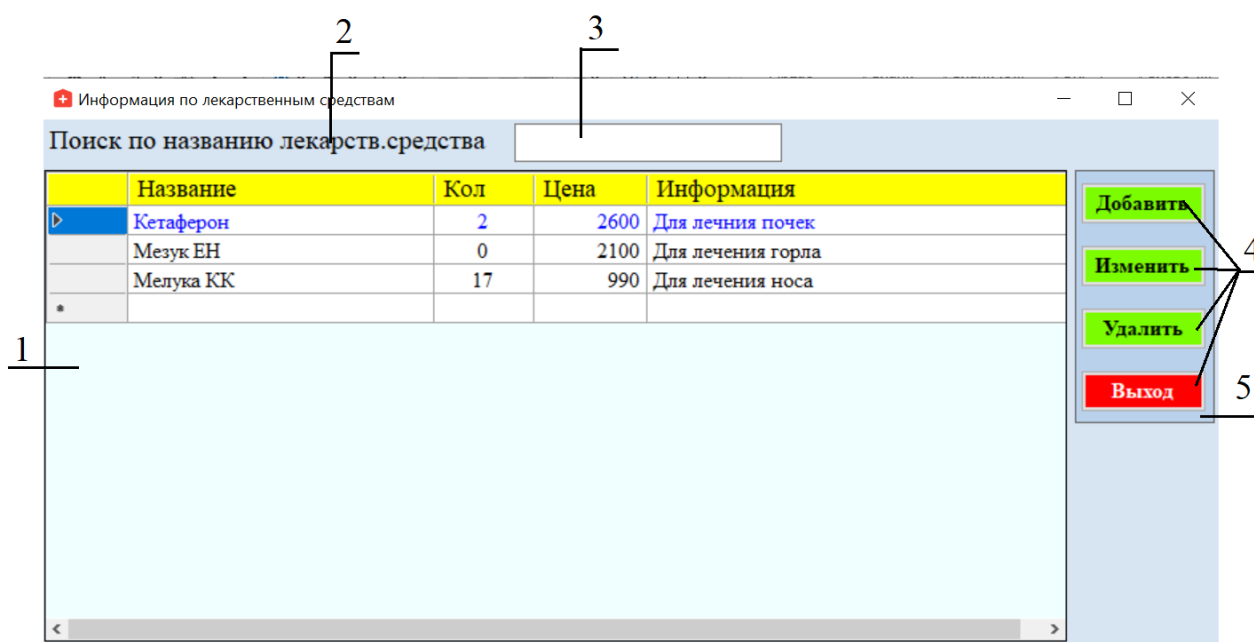


Рисунок 25 – Макет формы «Информация по лекарственным средствам»

Рассмотрим основные элементы интерфейса формы «Информация по лекарственным средствам»:

1 – таблица, где отображается информация, компонент Grid;

2 – надпись, компонент label;

3 – текстовое поле, компонент textbox;

4 – кнопки для работы в приложении, компонент button;

5 – панель, компонент panel.

Для добавления и изменения информации по лекарственному средству разработан макет формы, который представлен на рисунке 26.

Рисунок 26 – Макет формы «Данные по лекарственному средству»

Рассмотрим основные элементы интерфейса формы «Данные по лекарственному средству»:

1 – надпись, компонент label;

2 – текстовое поле, компонент textbox;

3 – кнопки для работы в приложении, компонент button.

Для проверки валидации ввода данных в полях «Количество» и «Цена» разработан обработчик «KeyPress», с помощью которого реализована проверка на ввод цифровых значений с помощью кода, представленного на листинге 15.

Листинг 5 – Обработчик «KeyPress» для проверки валидации ввода данных

```
if (e.KeyChar == (char)27){
    But_Exit_Click(sender, e);}
```

```

else if (e.KeyChar == (char)13){
    But_Save_Click(sender, e);}

char ch = e.KeyChar;
if (!char.IsDigit(ch) && ch != 8){
    e.Handled = true;}

```

Форма для работы с рецептами разработана для отображения информации по рецептам на изготовление лекарственных средств, которые выдаются лечащим врачом, макет формы представлен на рисунке 27.

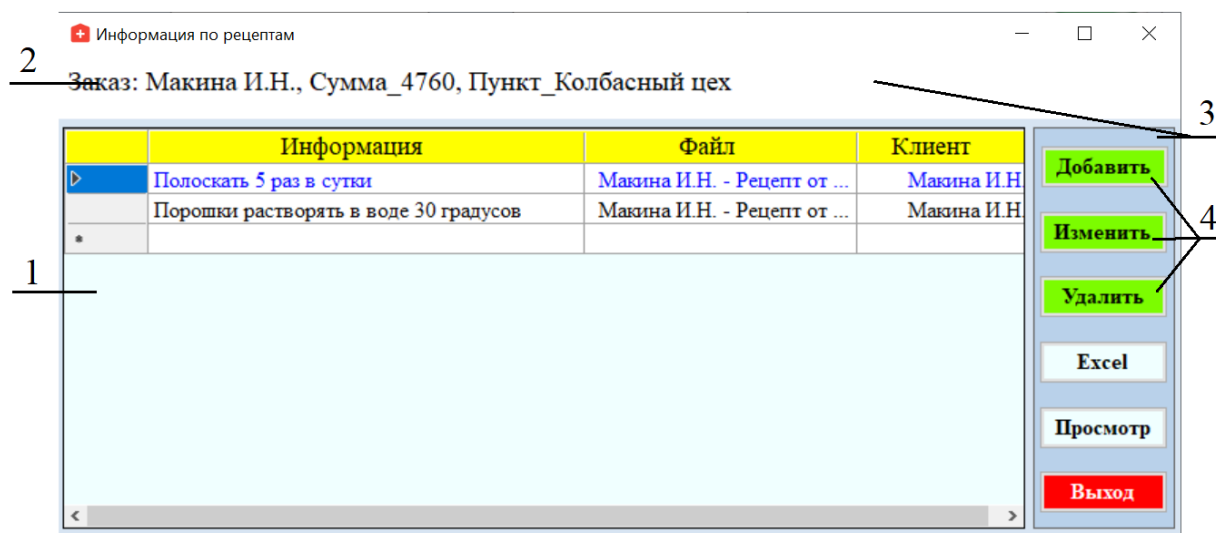


Рисунок 27 – Макет формы «Информация по рецептам»

Рассмотрим основные элементы интерфейса формы «Информация по рецептам»:

- 1 – таблица, где отображается информация, компонент Grid;
- 2 – надпись, компонент label;
- 3 – панель, компонент panel;
- 4 – кнопки для работы в приложении, компонент button.

При нажатии на кнопку «Просмотр» выполняется поиск и открытия для просмотра файла с рецептом.

Реализацию разработки отчетов рассмотрим на примере отчета «По клиентам».

Макет данной формы приведен на рисунке 28.

В первой таблице список всех клиентов, зарегистрированных в приложении, во второй список заказов клиента. В приложении можно выгрузить отчет в Excel по заказам конкретного клиента.

The screenshot shows a software window titled 'Отчет по клиентам'. It contains two tables and a search bar. Annotations point to specific UI elements:

- 1**: Points to the first table, which lists clients.
- 2**: Points to the search bar labeled 'Поиск по фамилии клиента'.
- 3**: Points to the search bar.
- 4**: Points to the second table, which lists orders for a selected client.
- 5**: Points to the 'Отчет' (Report) and 'Выход' (Exit) buttons on the right side of the window.

Фамилия	Имя	Отчество	Телефон	Паспорт
Иванов	Николай	Евгеньевич	10987656	1200123009
Макина	Ирина	Николаевна	+7-980-211-00-91	1199234098
Семенкина	Елена	Алексеевна	+7-200-100-23-09	1020768911

Код	Дата (Начало)	Дата (Конец)	Сумма	Статус	Пункт	Информация
10	08.06.2024	08.06.2024	5000	Открыт	Колбасный цех	Раствор от горла
1	28.05.2024	28.05.2024	4500	Открыт	Солнечный	222

Рисунок 28 – Макет формы отчета «По клиентам»

Рассмотрим основные элементы интерфейса формы отчета «По клиентам»:

1 – таблица, где отображается информация, компонент Grid;

2 – надпись, компонент label;

3 – текстовое поле, компонент textbox;

4 – панель, компонент panel;

5 – кнопки для работы в приложении, компонент button.

Выход из программы организован с помощью нажатия кнопки «Выход».

Листинг форм приложения будет в приложении Е листинг Е.15-Е.42.

В результате было разработано клиентское приложение обработки и анализа индивидуальных заказов на изготовление лекарственных средств.

3.3 Разработка классов приложения

В программе разработаны классы данных, представленные в таблице 43.

Таблица 43 – Классы данных

Название	Описание
ClassUsers	Рабочий персонал медицинского учреждения: фармацевты, провизоры
ClassClient	Клиенты, которые заказывают на изготовление лекарственных средства
ClassPunkt	Пункты выдачи лекарственных средств клиентам
ClassDolzhnost	Названия должностей сотрудников медицинского учреждения
ClassRecept	Рецепты на изготовление лекарственных средств для клиентов
ClassZakaz	Заказы на изготовление лекарственных средств и на закупку медицинских препаратов у поставщиков
ClassLekarstvo	Лекарственные средства
ClassStatus	Статусы заказов
ClassSostavZakaz	Содержимое заказов на изготовление лекарственных средств и на закупку медицинских препаратов у поставщиков
ClassSupplier	Поставщики медицинских препаратов в медицинское учреждение
ClassHistory	История по заказам
ClassOplata	Операции по оплатам
ClassPreparat	Медицинские препараты от поставщиков
ClassViewPreparat	Виды медицинских препаратов, поставляемых поставщиками

Класс данных ClassViewPreparat представлен на листинге 6.

Листинг 6 – Класс данных ClassViewPreparat

```
public class ClassViewPreparat
{
    public int CodViewPreparat { get; set; }
    public string Name { get; set; }

    public ClassViewPreparat() { }

    public ClassViewPreparat(int _CodViewPreparat, string _Name)
    {
        CodViewPreparat = _CodViewPreparat;
        Name = _Name;
    }
}
```

В этом классе описываются свойства для хранения первичного ключа препарата (CodViewPreparat) и для хранения имени вида препарата (Name). Также описываются два конструктора. Первый пустой конструктор, который не принимает никаких параметров. Он инициализирует объект класса с дефолтными значениями свойств. Второй конструктор принимает два

параметра: `_CodViewPreparat` (целое число) и `_Name` (строка). В теле конструктора происходит присваивание значений переданных параметров соответствующим свойствам класса.

Класс данных `ClassOplata` представлен на листинге 7.

Листинг 7 – Класс данных `ClassOplata`

```
public class ClassOplata
{
    public int CodOplata { get; set; }
    public DateTime Data { get; set; }
    public int Summa { get; set; }
    public int CodZakaz { get; set; }

    public ClassOplata() { }

    public ClassOplata(int _CodOplata, DateTime _Data, int _Summa, int
_CodZakaz)
    {
        CodOplata = _CodOplata;
        Data = _Data;
        Summa = _Summa;
        CodZakaz = _CodZakaz;
    }
}
```

В этом классе описываются свойства для хранения первичного ключа оплаты (`CodOplata`), даты оплаты (`Data`), суммы оплаты (`Summa`) и первичного ключа заказа (`CodZakaz`). Описываются два конструктора. Первый конструктор не принимает никаких параметров. Второй конструктор принимает четыре параметра: `_CodOplata` (целое число), `_Data` (дата и время), `_Summa` (целое число), `_CodZakaz` (целое число). В теле конструктора происходит присваивание значений переданных параметров соответствующим свойствам класса.

Листинг всех классов данных представлен в листинге E.1 – E.13 в приложении E.

3.4 Разработка методов доступа к базе данных

Приложение обработки и анализа индивидуальных заказов на изготовление лекарственных средств предназначено для фармацевтов, провизоров и клиентов в медицинских учреждениях и аптеках, чтобы

автоматизировать процессы оформления и ведения учета заказов на изготовление лекарственных средств.

Для работы с базой данных из приложения в проекте нужно добавить ссылку на файл «MySQL.Data.dll», а также установить пакеты NuGet: «MySQL.Data» и «MySQLConnector».

Подключение к БД организовано в классе «Query».

В проекте подключена библиотека «Newtonsoft.Json» – это высокопроизводительная JSON-библиотека для .NET. Данная библиотека используется для анализа и сериализации JSON-данных в приложениях .NET, упрощая работу с JSON-данными в C#. Данная библиотека использовалась для разработки методов с данными БД.

Для работы с данными таблиц базы данных разработаны следующие методы:

- `public DataTable GetDataFromTable(string zapros)` – метод для получения данных из таблицы БД, в качестве входного параметра метод принимает sql-запрос на выборку данных из таблицы БД;

- `public void DeleteFromTable(string NameTabl, string PoleTable, int ID)` – метод для удаления данных в таблицах БД, в качестве входных параметров принимает: название таблицы БД, наименование кода первичного ключа и значение первичного ключа;

- `public void Update_JSON(string NameTabl, string strData)` – метод для обновления данных в таблицах БД, в качестве входных параметров принимает название таблицы и строку с данными в формате JSON;

- `public int Add_JSON(string NameTabl, string strData)` – метод для обновления данных в таблицах БД, возвращаемым значением является значение первичного ключа таблицы, который был получен с помощью sql-запроса: `Select @@Identity`.

В качестве входных параметров принимает название таблицы и строку с данными в формате JSON. Все методы реализованы в классе Query.

Листинг класса Query представлен в листинге E.14 в приложении E.

При выполнении клиентом удаления записи заказа, выполняется два метода: `DeleteFromTable` и `GetDataFromTable`.

Для наглядного представления работы приложения при удалении записи была создана диаграмма последовательности. Диаграмма последовательности – диаграмма UML, которая применяется для визуализации процесса взаимодействия объектов (выделенных частей системы) [21]. Взаимодействие пользователя с приложением и ПС взаимодействия с СУБД представлено в приложении Ж.

После того, как пользователь нажимает кнопку для удаления записи, проверяется наличие заказов в таблице приложения. Если записей нет, то приложение выведет ошибку. Если проверка прошла успешно, то отправляется sql-запрос на проверку связей удаляемой записи. Если заказ связан с другими таблицами, то он не сможет удалиться. В таком случае, приложение выведет ошибку, что есть связи, из-за которых заказ не может быть удален. Если заказ не будет связан с другими таблицами, то будет вызван метод для удаления записи в базе данных `DeleteFromTable`. После чего, будет вызван метод для получения обновленной таблицы из базы данных – `GetDataFromTable`. В результате в приложении будет сообщение об успешном удалении.

3.5 Разработка запросов в базе данных

Для хранения данных была выбрана СУБД MySQL, которая относится к промышленным видам СУБД. MySQL установлена на серверной части.

В данной реализации серверная часть разработки предназначена для следующий действий:

- прием и обработка sql-запросов от клиентских приложений, которые установлены на компьютерных станциях в медицинских учреждениях;
- отправка результатов обработки sql-запросов клиентским приложениям;
- безопасность хранения данных базы данных;
- сопровождение стабильной работы многопользовательской работы

ПОДКЛЮЧЕННЫХ КЛИЕНТОВ;

– следит за целостностью информации, хранящейся в таблицах базы данных.

Спецификации sql-запросов в приложении представлены в таблице 44.

Таблица 44 – Спецификации sql-запросов в приложении

Название	Sql-запрос
SQL-запрос для поиска сотрудника	select * from Table_Users where (Login = "" + Text_Login.Text + "") and (Parol="" + Text_Parol.Text + "")
SQL-запрос для поиска клиента	select * from Table_Client "where (FamCl = "" + Text_Login.Text + "") and (Document="" + Text_Password.Text.Trim() + "")"
SQL-запрос для получения данных из таблицы с видами препаратов	select * from table_viewpreparat order by Name
SQL-запрос для проверки связей с данными препаратов при удалении	SELECT COUNT(*) FROM Table_Preparat where CodViewPreparat = @PID_Cod
SQL-запрос алгоритма фильтрации данных по лекарственным средствам	select Z.CodZakaz,Z.DataStart,Z.DataEnd, concat(CL.FamCl,'',LEFT(CL.NameCl,1),'',LEFT(CL.OtchCl,1),') as Client, Z.Summa,ST.Name as Status,PN.Name as Punkt,Z.Inform, concat(US.FamU,'',LEFT(US.NameU,1),'',LEFT(US.OtchU,1),') as Users from (((Table_Zakaz as Z inner join Table_Client as CL on Z.CodClient = CL.CodClient) inner join Table_Punkt as PN on Z.CodPunkt = PN.CodPunkt) inner join Table_Status as ST on Z.CodStatus = ST.CodStatus) inner join Table_Users as US on Z.CodUsers = US.CodUsers) where "" + "(Z.CodClient is not null)and "" + S_Find + "" order by Z.DataStart desc
SQL-запрос для фильтрации данных по препаратам	select Z.CodZakaz,Z.DataStart,Z.Summa,ST.Name as Status, Z.Inform, concat(US.FamU,'',LEFT(US.NameU,1),'', LEFT(US.OtchU,1),') as Users from ((Table_Zakaz as Z inner join Table_Status as ST on Z.CodStatus = ST.CodStatus) inner join Table_Users as US on Z.CodUsers = US.CodUsers)
SQL-запрос для отображения информации по составу заказа	select S.CodSostavZakaz,LK.Name,S.Kol,LK.Cena,S.CodLekarstvo from (Table_SostavZakaz as S inner join Table_Lekarstvo as LK on S.CodLekarstvo = LK.CodLekarstvo) where S.CodZakaz= ClassStr.TekCodZakaz.ToString()
SQL-запрос для строки поиска «Название лекарственного средства»	where Name like "%" + textBoxName.Text + "%' order by Name
SQL-запрос для поиска рецептов по указанному заказу	where R.CodZakaz = "" + int.Parse(ClassStr.TekCodZakaz.ToString())

Продолжение таблицы 44

Название	Sql-запрос
SQL-запрос для получения информации по всем клиентам	select CodClient,FamCl,NameCl,OtchCl,Phone,Document from Table_Client order by FamCl,NameCl,OtchCl
SQL-запрос для получения списка заказов по указанному клиенту	select Z.CodZakaz,Z.DataStart,Z.DataEnd, Z.Summa,ST.Name as Status,PN.Name as Punkt,Z.Inform, concat(US.FamU,'',LEFT(US.NameU,1),',', LEFT(US.OtchU,1),',') as Users from (((Table_Zakaz as Z inner join Table_Punkt as PN on Z.CodPunkt = PN.CodPunkt) inner join Table_Status as ST on Z.CodStatus = ST.CodStatus) inner join Table_Users as US on Z.CodUsers = US.CodUsers)

SQL-запросы играют ключевую роль в обеспечении взаимодействия между приложением и базой данных, позволяя выполнять основные операции: создание, обновление и удаление данных.

4 Контроль качества программного обеспечения

4.1 Планирование процесса контроля качества программного обеспечения

4.1.1 Мероприятия по контролю качества программного обеспечения

Для эффективного функционирования программных средств большое значение имеет контроль всех мероприятий и процессов, выполняемых на разных этапах жизненного цикла продукции. Поэтому, контроль качества – является одним из основных функций процесса управления качеством.

Контроль качества программного обеспечения (ПО) является завершающим этапом разработки, обеспечивающий проверку готового программного средства на соответствие заявленным целям и требованиям.

Основная задача контроля качества ПО – это проверка разработанного продукта, посредством тестирования программного средства. Из задач контроля качества также стоит отметить проверку и обеспечение адекватного уровня тестирования, чтобы ни один аспект программного продукта не оставался незамеченным или непроверенным. Иначе существует вероятность упустить потенциально критические ошибки, способные негативно повлиять на работоспособность и функциональность программного обеспечения.

Одной из важных задач контроля качества является проверка соответствия системы требованиям. Необходимо, чтобы все функциональные и нефункциональные требования были корректно реализованы [22].

Критериями качества ПО являются надежность, завершенность разработки, быстродействие, уровень требований к техническим средствам, функциональная полнота.

Мероприятия по контролю качества можно разделить на несколько групп:

- способы, связанные с выявлением реальной функциональности программного обеспечения в процессе его использования (измерение качественных показателей, различные метрики);

- методы, направленные на обнаружение нарушений разработки программного кода, проектных моделей и документации (функциональное тестирование);
- подходы к анализу исходного кода (анализ архитектуры ПО, анализ эффективности применяемых алгоритмов);
- методы определения качественных показателей на основе моделирования работы ПО с использованием различных моделей для оценки качества принимаемых решений.

В рамках данной выпускной квалификационной работы в качестве мероприятий по контролю качества было решено провести тестирование и подсчет метрик программного кода.

4.1.2 Обоснование выбора модели тестирования

На сегодняшний день существует большое количество подходов к тестированию программного обеспечения, которые позволяют проверить корректность работы системы.

Под тестированием подразумевают процесс выявления ошибок в программном продукте. Для пользовательского приложения крайне необходимо, чтобы во время тестирования было проверено не только поведение системы внутри, но интерфейс пользователя на удобство использования.

Тесты могут быть как ручные, так и автоматизированные. Ручное тестирование подразумевает проверку системы самим тестирующим, взаимодействуя с программным средством. В свою очередь автоматизированное тестирование выполняется машиной, которой предоставили алгоритмы для тестирования. Автоматизированное тестирование надежнее, чем ручное, но качество автоматических тестов зависит от качества написания алгоритмов тестирования. Тестами может проверяться как один метод в классе, так и множество выполненных действий в пользовательском интерфейсе, что в конечном итоге приведет к одному и тому же результату [23].

Различные типы тестирования и описание каждого из них приведены в

таблице 45.

Таблица 45 – Описание типов тестирования

Название	Описание
Модульное тестирование	Тестирование отдельных модулей, компонентов, функций или методов программы
Интеграционное тестирование	Тестирование на корректность работы отдельных модулей или служб системы между собой. Например, это может быть тестирование взаимодействия системы с базой данных.
Функциональное тестирование	Тестирование на основе проверки соответствия функций системы требованиям и спецификациям.
Нагрузочные тестирование	Тестирование, которое оценивает силу программного обеспечения, проверяя, какую нагрузку оно может выдержать, прежде чем достигнет критической точки.
Тестирование производительности	Тестирование, на основе которого оценивается, как система работает при определенной рабочей нагрузке. Эти тесты помогают измерить надежность, скорость, масштабируемость и скорость отклика приложения.
Регрессионные тесты	С помощью этих тестов можно определить не приводит ли добавление новых функций к снижению функциональности приложения.

Для приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств был выбран метод функционального тестирования – белый ящик и модульное тестирование, а также модульное тестирование.

Тестирование методом белого ящика – это способ проверки программного средства, который предполагает, что внутренняя структура системы известна тестирующему.

Рассмотрим достоинства данного метода тестирования:

- обеспечивает полное покрытие кода, тестируя каждый участок внутренней структуры системы;
- метод помогает оптимизировать исходный код ПО;
- помогает тестирующим найти ошибки на ранних этапах создания

ПО;

- облегчает взаимодействие между разработчиками и тестировщиками, что снижает риск недопонимания при выполнении задач [24].

Модульное тестирование (unit-тестирование) – это разновидность тестирования в программной разработке, которое заключается в проверке работоспособности отдельных функциональных модулей или частей кода приложения.

Рассмотрим достоинства модульного тестирования:

- unit-тестирование позволяет проверить код одного компонента приложения независимо от других;
- написать тест для отдельного модуля проще, чем для приложения в целом;
- хорошо составленный тест помогает разработчикам понимать функционал модуля, особенности использования;
- разработчик может вернуться к ранее написанному тесту, чтобы проверить работу компонента еще раз [25].

Таким образом, тестирование методом «белого ящика» позволяет анализировать внутреннюю структуру кода и выявлять проблемы, которые могли быть упущены при других подходах к тестированию, а unit-тестирование позволяет проверить отдельные модули на корректность данных.

4.1.3 Обоснование выбора метрик программного кода

Контроль качества программного продукта предполагает наличие метрик, с помощью которых можно оценить достижение того или иного уровня качества программного средства.

Метрикой программного обеспечения называется мера, служащая для получения численного значения некоторого свойства программного обеспечения или спецификаций. Метрики могут рассчитываться по всему коду, либо на небольших блоках кода, например, на уровне класса, метода или на более низком уровне.

Существует большое количество различных метрик программ, которые можно разделить на следующие направления:

- измерения производительности программного обеспечения;
- измерения сложности программ;
- измерения уровня языковых средств;
- оценки функциональной надежности программных систем;
- измерения восприятия и понимания программных текстов;
- измерения производительности труда программистов.

Для оценки качества программного кода были выбраны метрики программного обеспечения и сложности программ. Некоторые представители метрик сложности программ представлены в таблице 46.

Таблица 46 – Представители метрик сложности программ

Название	Описание
Метрика Холстеда	С помощью данной метрики можно получить следующие оценки: словарь, длина и объем программы, а также теоретический словарь программы (словарный запас, необходимый для написания программы). С помощью теоретического словаря программы описывается потенциальный объем программы.
Метрика МакКейба	Суть метрики заключается в определении количества базисных путей в управляющем графе. Базисные пути – это такие пути, komponуя которые можно получить все возможные пути из входа в выходы графа. МакКейб предлагает использовать цикломатическую сложность графа программы в качестве основной метрики сложности, которая отражает трудоемкость тестирования программы.
Метрика спена	Определение спена основывается на выявлении местоположения ссылок на данные в каждом разделе программы. Спен выражается количеством утверждений, содержащих конкретный идентификатор, между его первым и последним упоминанием в коде. Если спен большой, то это затрудняет процесс отладки и тестирования.
Метрика Чепина	Данная метрика направлена на оценку информационной прочности конкретного программного модуля путем анализа характера использования переменных из списка ввода-вывода.
Метрика SLOC	Это метрика программного обеспечения, предназначенная для измерения объема кода, путем подсчета количества строк в исходном коде.

Из приведенного перечня различных метрик сложности программ для оценки качества программного кода разрабатываемой системы были выбраны метрика Холстеда, метрика Чепина и метрика SLOC.

4.2 Тестирование программных средств

4.2.1 Тестирование белым ящиком

Функциональное тестирование – это процесс проверки программного обеспечения на соответствие его функциональным требованиям. Оно направлено на обеспечение корректного функционирования программы и проверку её соответствия ожиданиям пользователя.

Проведём тестирование программы обработки и анализа индивидуальных заказов на изготовление лекарственных средств с проверкой корректности добавления лекарства в состав заказа (листинг 8) методом белого ящика. При тестировании рассчитаем некоторые критерии покрытия кода, такие как покрытие операторов, покрытие решений и покрытие маршрутов.

Листинг 8 – Метод But_AddInZakaz_Click

```
private void But_AddInZakaz_Click(object sender, EventArgs e)
{
    if (GridSpisok.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
        return;
    }

    if (DBEdit_Kol.Text == "")
    {
        MessageBox.Show("Укажите кол-во ");
        DBEdit_Kol.Focus();
        return;
    }

    if (int.Parse(DBEdit_Kol.Text) <= 0)
    {
        MessageBox.Show("Укажите кол-во больше 0");
        DBEdit_Kol.Focus();
        return;
    }

    if (int.Parse(DBEdit_Kol.Text.ToString()) >
        int.Parse(GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["
Kol"].Value.ToString()))
    {
```

```

        MessageBox.Show("Указанное кол-во больше кол-ва в наличие");
        DBEdit_Kol.Focus();
        return;
    }

    var cmd = new MySqlCommand(@"");
    if (ClassStr.RegimZakaz == 1)
        cmd = new MySqlCommand(@"SELECT CodSostavZakaz FROM
Table_SostavZakaz where (CodZakaz = @PCodZakaz)and(CodLekarstvo =
@PCodLekarstvo)");
    else
        cmd = new MySqlCommand(@"SELECT CodSostavZakaz FROM
Table_SostavZakaz where (CodZakaz = @PCodZakaz)and(CodPreparat =
@PCodPreparat)");
    cmd.Connection = new MySqlConnection(Query.ConnStr);
    cmd.Connection.Open();
    cmd.Parameters.AddWithValue("PCodZakaz",
ClassStr.TekCodZakaz);
    if (ClassStr.RegimZakaz == 1)
        cmd.Parameters.AddWithValue("PCodLekarstvo",
GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["CodLekarstvo"].
Value);
    else
        cmd.Parameters.AddWithValue("PCodPreparat",
GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["CodPreparat"].V
alue);
    int CodSostavZakaz = Convert.ToInt32(cmd.ExecuteScalar());
    cmd.Connection.Close();
    if (CodSostavZakaz > 0)
    {
        cmd = new MySqlCommand("update Table_SostavZakaz set Kol=Kol+"
+ DBEdit_Kol.Text + " where CodSostavZakaz = " +
CodSostavZakaz.ToString());
        cmd.Connection = new MySqlConnection(Query.ConnStr);
        cmd.Connection.Open();
        cmd.ExecuteScalar();
        cmd.Connection.Close();
    }
    else
    {
        ClassSostavZakaz Per_2 = new ClassSostavZakaz(0,
int.Parse(DBEdit_Kol.Text),
ClassStr.TekCodZakaz,
ClassStr.RegimZakaz == 1 ?
int.Parse(GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["CodLe
karstvo"].Value.ToString()):0,
ClassStr.RegimZakaz == 1 ?
0:int.Parse(GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["Cod
Preparat"].Value.ToString())
);
        string StrJson = JsonConvert.SerializeObject(Per_2);
        Podkl.Add_JSON("Table_SostavZakaz", StrJson);
    }

    MySqlDataAdapter datadapterSostavZakaz = new
MySqlDataAdapter(ZaprosSostavZakaz, Query.ConnStr);
    DataTable bufferSostavZakaz = new DataTable();
    datadapterSostavZakaz.Fill(bufferSostavZakaz); //вызываем метод

```

```

        Fill, чтобы после запроса заполнил буферную таблицу данными
        GridSostavZakaz.DataSource = bufferSostavZakaz;

//-----изменяем кол-во товара в списке продуктов
int Kol =
int.Parse(GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["Kol"].Value
.ToString()) -
int.Parse(DBEdit_Kol.Text.ToString());

string zapros = "";
    if (ClassStr.RegimZakaz == 1)
        zapros = "update Table_Lekarstvo set Kol=" + Kol.ToString() + "
where CodLekarstvo = " +

        GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["CodLekarstvo"].
Value.ToString();
    else
        zapros = "update Table_Preparat set Kol=" + Kol.ToString() + " where
CodPreparat = " +

        GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["CodPreparat"].V
alue.ToString();
    MySqlConnection connection; //создать подключение между нашим
проектом и БД
    MySqlCommand command; //нужен для выполнения операция
вставки, редактирования и удаления данных БД
    connection = new MySqlConnection(Query.ConnStr);
    connection.Open(); //Открыть соединение
    command = new MySqlCommand(zapros, connection);
    command.ExecuteNonQuery(); //Выполняем запрос. В качестве
результата возвращает кол-во обработанных строк
    connection.Close(); //Закрываем соединение

//Запоминаем позицию в DBGrid, чтобы после редактирования в другом окне
установить курсор на эту строку
int iStr = GridSpisok.CurrentCell.RowIndex;

//обновляем данные в таблице
MySqlDataAdapter datadapterSpisok = new MySqlDataAdapter(Zapros,
Query.ConnStr);
DataTable bufferSpisok = new DataTable();
datadapterSpisok.Fill(bufferSpisok); //вызываем метод Fill, чтобы после
запроса заполнил буферную таблицу данными
    GridSpisok.DataSource = bufferSpisok;

//Установить курсор на измененную запись
GridSpisok.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
GridSpisok.CurrentCell = GridSpisok.Rows[iStr].Cells[1]; //ставим
выделение на измененную строку
//-----Конец установки курсора

    DBEdit_Kol.Text = "";
    MessageBox.Show("Операция успешно выполнена по добавлению в состав
заказа");
}

```

Для четкого понимания логики действий программы изобразим блок-схему метода But_AddInZakaz_Click. Блок-схема – это визуальное представление

программного кода для простоты восприятия программы. Для построения блок-схем применяются специальные фигуры, такие как прямоугольники, ромбы, овалы и другие фигуры, которые, в свою очередь, обозначают конкретные операции. Также используются соединительные стрелки, которые указывают последовательность шагов и направление процесса [26].

На рисунке 29 представлена блок-схема листинга 8.

На блок-схеме все операции помечены буквенными обозначениями для простоты использования в дальнейшем.

Согласно представленной блок-схеме, изобразим управляющий граф программы. Управляющий граф (Control flow graph, Flow graph) – это модель программы, которая представляет собой орграф потока управления, то есть систему управляющих связей. В ней изображаются только порядок выполнения операторов, а также информация о входных или выходных данных в моментах передачи управления между операторами [27].

Покрытие операторов – это процесс проверки всех участков кода программы хотя бы один раз. Для этого необходимо протестировать все возможные варианты выполнения операторов в коде, чтобы удостовериться, что они работают правильно и не содержат ошибок.

Покрытие решений – это выполнение проверки, которая показывает, насколько принятое решение в коде было выполнено хотя бы один раз в процессе тестирования.

Покрытие маршрутов – это проверка программного кода, благодаря которой можно оценить все возможные пути выполнения, или «маршруты», в программе. Это означает, что каждый возможный путь, начиная от входа в программу до выхода из нее, должен быть испытан хотя бы один раз в процессе тестирования.

Высокий уровень покрытия маршрутов говорит о тщательности и полноте тестирования программы, но иногда может быть трудно достигнуть 100% покрытия из-за сложности анализа всех возможных путей выполнения в коде.

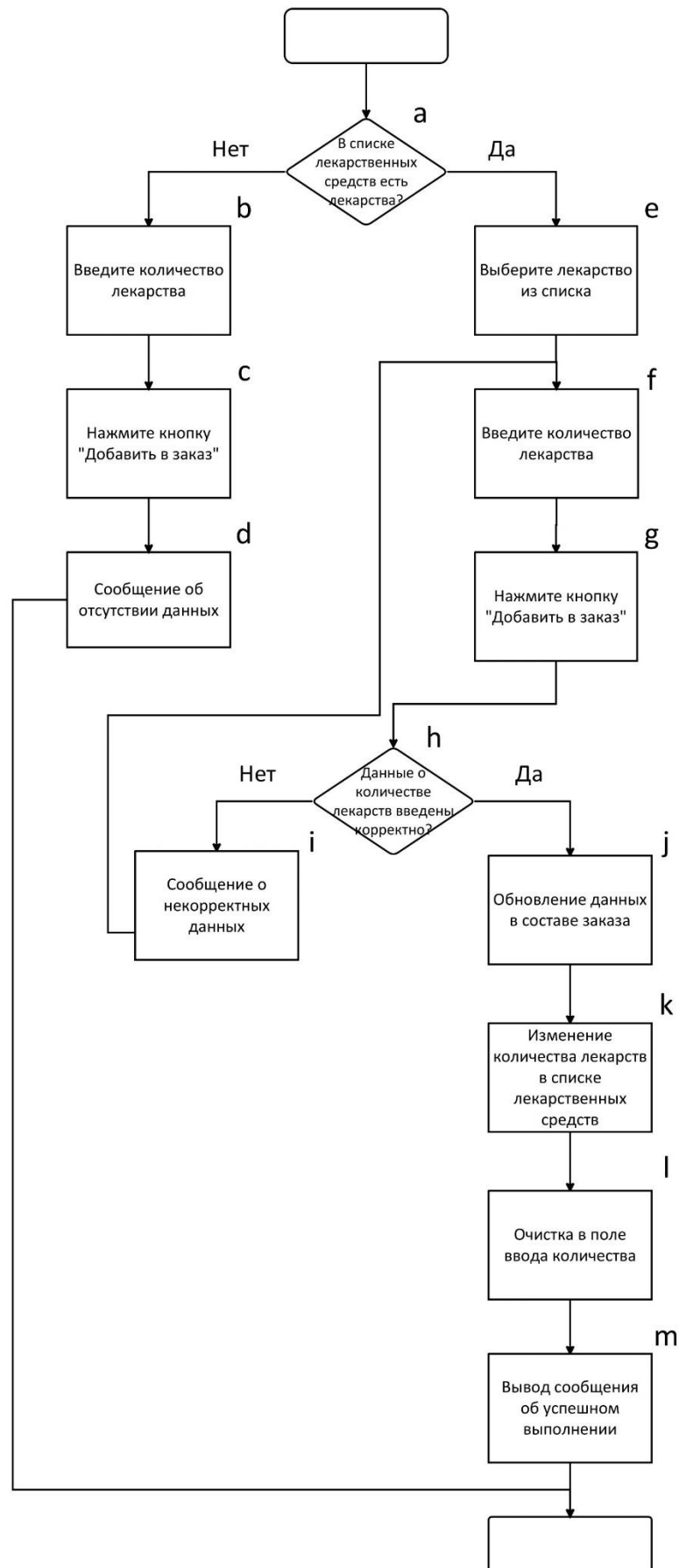


Рисунок 29 – Блок-схема метода But_AddInZakaz_Click

На рисунке 30 изображен управляющий граф метода But_AddInZakaz_Click.

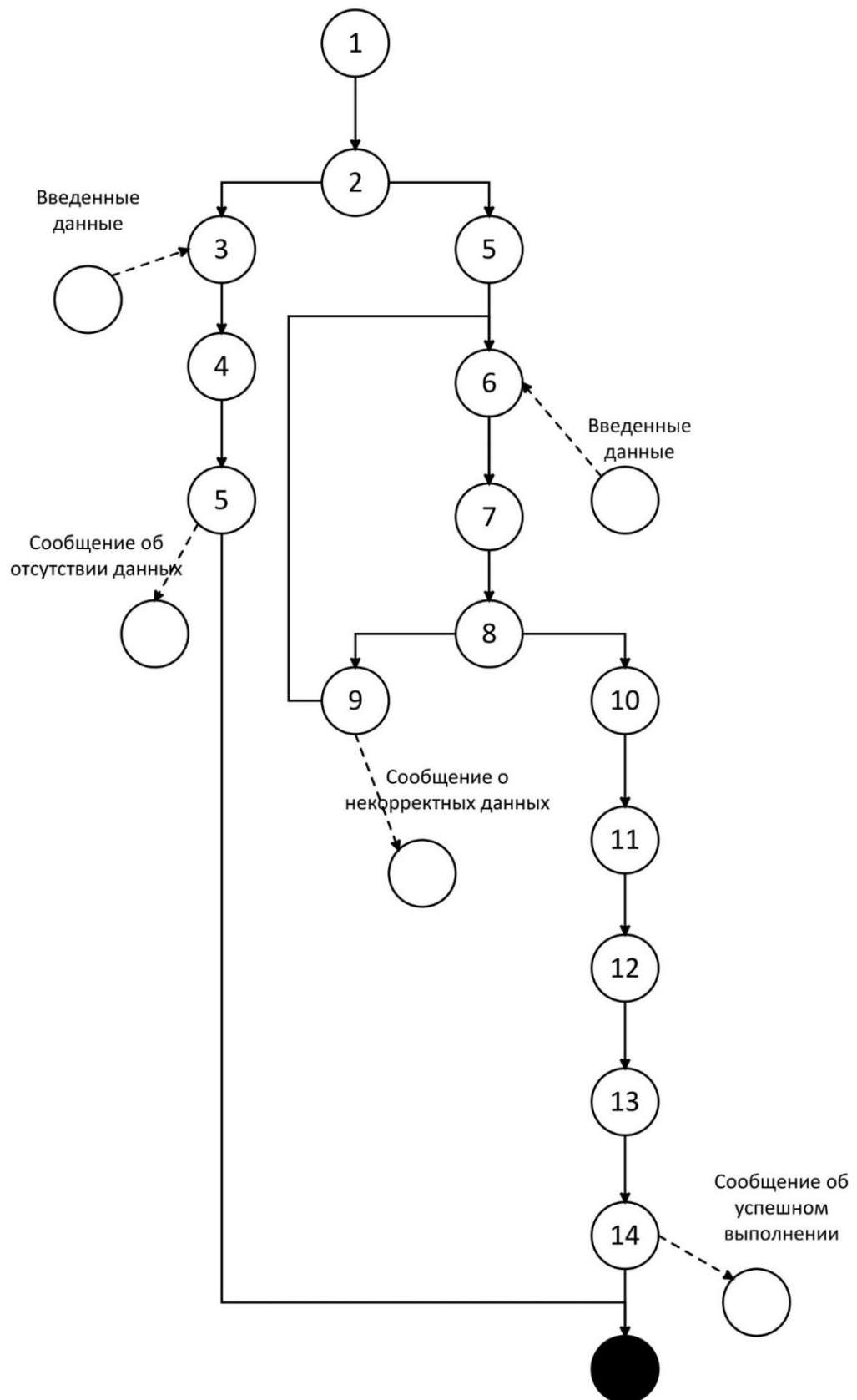


Рисунок 30 – Управляющий граф метода But_AddInZakaz_Click

Построим тесты проверки покрытия операторов (таблица 47).

Таблица 47 – Покрытие операторов

№	Операторы	Входные данные	Выходные данные
1	b	Количество препарата: 2	Сообщение «Отсутствуют данные в списке лекарственных средств»
2	e, i, j	Количество препарата: 0	Сообщение «Операция успешно выполнена по добавлению в состав заказа»

Таким образом, достаточно двух проверок для выполнения всех операторов в коде хотя бы один раз.

Использование покрытия решений в тестировании помогает обнаружить потенциальные ошибки в логике программы, такие как пропущенные условия или неправильные решения, которые могут привести к некорректному поведению при выполнении программы.

Построим тесты проверки покрытия решений (таблица 48).

Таблица 48 – Покрытие решений

Решения	Входные данные	Выходные данные
a – да, h – да	Количество препарата: 3	Сообщение «Операция успешно выполнена по добавлению в состав заказа»

Для проверки покрытия решений достаточно одного теста, чтобы протестировать выполняются ли все решения хотя бы один раз в процессе выполнения программы.

Построим тесты проверки покрытия маршрутов (таблица 49).

Таблица 49 – Покрытие маршрутов

№	Условие	Входные данные	Выходные данные
1	a – да, e, f, g, h – нет, i, f, g, h – да, j, k, l, m	Количество препарата: 0	Сообщение «Операция успешно выполнена по добавлению в состав заказа»
2	a – нет, b, c, d	Количество препарата: 5	Сообщение «Отсутствуют данные в списке лекарственных средств»
3	a – да, e, f, g, h – да, j, k, l, m	Количество препарата: 6	Сообщение «Операция успешно выполнена по добавлению в состав заказа»

Для проверки покрытия маршрутов необходимо три теста, чтобы протестировать выполняются ли все маршруты хотя бы один раз в процессе выполнения программы.

Результаты тестов представлены на рисунках И.1 – И.7 в приложении И.

Таким образом для покрытия всех операторов необходимо провести всего два тест, для покрытия всех решений один тест, а для покрытия всех маршрутов три теста. Используя эти данные, можно эффективнее создавать автоматические тесты.

4.2.2 Модульное тестирование

Unit-тесты – это специальные тесты, которые позволяют проверить отдельные компоненты программы на корректность их работы. Это могут быть отдельные блоки или функции написанного кода.

Создадим тесты для проверки работы метода `But_Save_Click`, представленный в приложении Е на листинге Е.23, на различных входных данных для обеспечения правильной обработки данных при сохранении оплаты за заказ с использованием NUnit Testing Framework. Листинг полного класса тестирования приведен в приложении К. Рассмотрим конкретно методы, которые были разработаны для тестирования формы оплаты заказа. Методы тестирования представлены на листинге 9.

Листинг 9 – Методы тестирования

```
[TestMethod]
public void But_Save_Click_WithValidData()
{
    // Arrange
    var mockPodkl = new Mock<IPodkl>();
    var form = new FormOplata(mockPodkl.Object); // Передаем макет в
    конструктор формы

    form.DBEdit_Summa = new TextBox { Text = "1000" }; // Заполняем
    поле суммы
    form.DateTimeData = new DateTimePicker
    {
        Value = DateTime.Now,
        CustomFormat = "dd-MM-yyyy HH:mm:ss"
    }; // Устанавливаем текущую дату
    form.RegimOplata = 1; // Устанавливаем режим редактирования
```

```

        form.text_ID = new Label { Text = "1" }; // Устанавливаем
идентификатор

        mockPodkl.Setup(m => m.Update_JSON("Table_Oplata",
It.IsAny<string>())); // Ожидаем вызов Update_JSON с этими параметрами

        // Act
        form.But_Save_Click(null, EventArgs.Empty);

        // Assert
        mockPodkl.Verify(m => m.Update_JSON("Table_Oplata",
It.IsAny<string>()), Times.Once); // Проверяем, что метод Update_JSON был
вызван один раз
        Assert.IsTrue(form.IsDisposed); // Проверяем, что форма была
закрыта
    }

    [TestMethod]
    public void But_Save_Click_WithInvalidData()
    {
        // Arrange
        var mockPodkl = new Mock<IPodkl>();
        var form = new FormOplata(mockPodkl.Object); // Передаем макет в
конструктор формы

        form.DBEdit_Summa = new TextBox(); // Оставляем поле суммы пустым
        form.DateTimeData = new DateTimePicker { Value = DateTime.Now }; //
Устанавливаем текущую дату
        form.RegimOplata = 1; // Устанавливаем режим редактирования
        form.text_ID = new Label { Text = "1" }; // Устанавливаем
идентификатор

        mockPodkl.Setup(m => m.Update_JSON(It.IsAny<string>(),
It.IsAny<string>())); // Заглушка для метода Update_JSON

        // Act
        form.But_Save_Click(null, EventArgs.Empty);

        // Assert
        mockPodkl.Verify(m => m.Update_JSON(It.IsAny<string>(),
It.IsAny<string>()), Times.Never); // Проверяем, что метод Update_JSON не
был вызван
        Assert.IsFalse(form.IsDisposed); // Проверяем, что форма не была
закрыта
    }
}

```

Для тестирования метода But_Save_Click было разработано два unit-теста. Первый метод But_Save_Click_WithValidData() проверяет выполнение формы при корректных введенных данных. Во входных данных для этого теста используются различные элементы управления формы FormOplata, такие как:

- поле суммы DBEdit_Summa – текстовое поле, которое имитирует ввод суммы оплаты, в данном случае, оно заполнено значением «1000»;

- элемент управления даты и времени DateTimeData – элемент управления для выбора даты и времени, в тесте он установлен на текущее время (DateTime.Now) и настроен на формат отображения «dd-MM-yyyy HH:mm»;
- режим оплаты RegimOplata – целочисленное поле, которое определяет режим работы формы, в данном тесте оно установлено на «1», что означает режим редактирования;
- идентификатор text_ID – текстовое поле, содержащее идентификатор записи, в данном тесте оно установлено на «1».

Эти данные используются для имитации ввода пользователем и тестирования поведения формы при сохранении данных. Тест проверяет, что при вводе валидных данных и нажатии кнопки «Сохранить» на форме, метод Update_JSON вызывается один раз с корректными параметрами, и форма закрывается.

Второй метод But_Save_Click_WithInvalidData() проверяет выполнение формы при некорректных данных. Во входных данных для этого теста используются различные элементы управления формы FormOplata, такие как:

- поле суммы DBEdit_Summa – текстовое поле, которое имитирует ввод суммы оплаты, в данном случае, оно не заполнено и остается пустым;
- элемент управления даты и времени DateTimeData – элемент управления для выбора даты и времени, в тесте он установлен на текущее время (DateTime.Now) и настроен на формат отображения «dd-MM-yyyy HH:mm»;
- режим оплаты RegimOplata – целочисленное поле, которое определяет режим работы формы, в данном тесте оно установлено на «1», что означает режим редактирования;
- идентификатор text_ID – текстовое поле, содержащее идентификатор записи, в данном тесте оно установлено на «1».

Этот тест проверяет сценарий, когда пользователь нажимает кнопку «Сохранить» с невалидными данными (пустое поле суммы). Он убеждается, что метод Update_JSON не вызывается и форма остается открытой, позволяя пользователю исправить ошибку.

При выполнении теста сначала открывается предупреждающее сообщение (рисунок 31), после нажатия кнопки «ОК», метод завершает проверку.

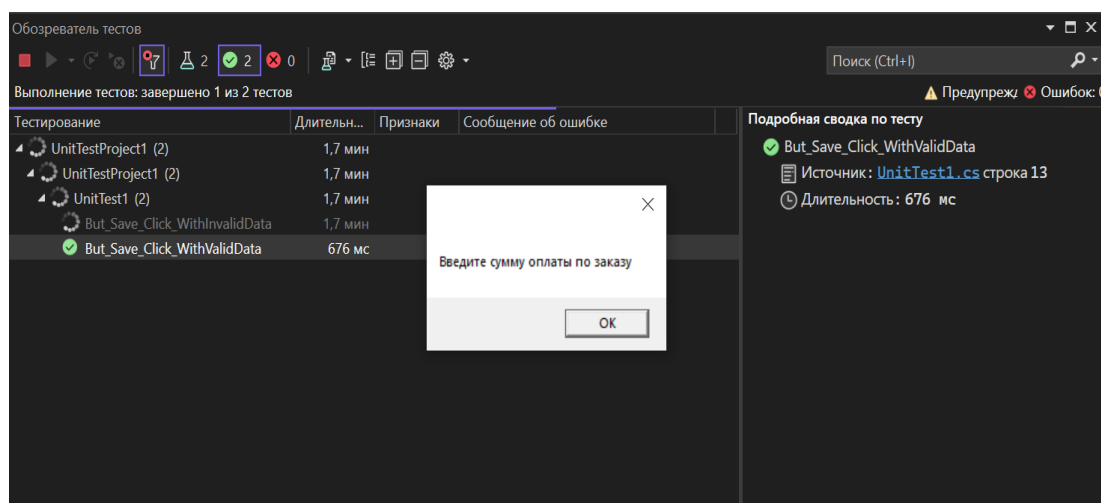


Рисунок 31 – Предупреждающее сообщение при вызове метода `But_Save_Click_WithInvalidData()`

Результат двух тестов изображен на рисунке 32.

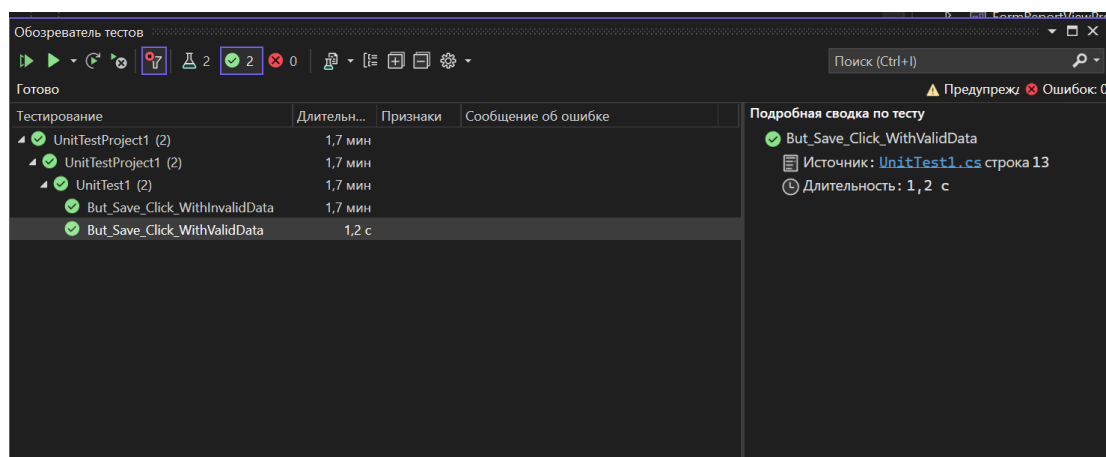


Рисунок 32 – Результат unit-тестов

В результате метод `But_Save_Click` в форме оплаты заказа был успешно протестирован.

4.3 Оценка качества программного кода на основе метрик

4.3.1 Метрика SLOC

Метрика SLOC – это метрика программного обеспечения, которая используется для измерения объема кода на основе подсчета определенных

показателей исходного кода. Исходные строки кода (SLOC) считаются программным показателем, используемым для определения размера компьютерной программы, путем подсчета количества строк кода. За основу можно брать как весь исходный код, так и фрагменты, например, классы и методы [28].

Существует три основных типа мер SLOC:

- физические строки кода;
- логические строки кода;
- строки с комментариями.

Физические строки кода определяются как общее число строк исходного кода без пустых строк. Комментарии к подсчету физических строк включаются. Пустые строки учитываются только в том случае, если в некотором фрагменте их количество не превышает 25%. Иначе пустые строки игнорируются.

Подсчет логических строк кода подразумевает количество операторов в программе. Разумеется, их количество зависит от конкретного языка программирования.

Строки с комментариями – подсчет количества строк, содержащие комментарии.

Проведем подсчет данной метрики в классе `FormSostavZakaz` в обработчике события кнопки «Добавить в заказ», который называется `But_AddInZakaz_Click`. Обработчик события описывает алгоритм проведения кнопки в форме с добавлением лекарства в состав заказа. Код метода `But_AddInZakaz_Click` приведен в листинге 8 раздела 4.2.1 Тестирование белым ящиком.

После подсчета всех параметров, объем метода `But_AddInZakaz_Click`, который осуществляет добавление лекарственного средства в состав заказа, равен 95 физическим строкам кода, 66 логическим строкам кода и 13 строкам, содержащих комментарии.

В результате, код представляет собой умеренно сложную и функционально насыщенную часть программы. Объем кода не слишком большой, но

достаточный для выполнения значимой задачи. Код также хорошо документирован, что поможет в будущем при его поддержке и доработке.

4.3.2 Метрика Холстеда

Метрика Холстеда основывается на анализе количества строк и синтаксических элементов кода программы. Она определяется через четыре основных характеристики программы:

- количество уникальных операторов, включая разделители, имена процедур и знаки операций;
- количество уникальных операндов программы;
- общее количество операторов в программе;
- общее количество операндов в программе.

В данной работе расчет метрики Холстеда приведен для метода But_AddInZakaz_Click, представленным на листинге 8.

Операторы и операнды метода But_AddInZakaz_Click приведены в таблице 50.

Таблица 50 – Операторы и операнды метода But_AddInZakaz_Click

Операторы	Число вхождений	Операнды	Число вхождений
if	8	GridSpisok.Rows.Count	9
else	4	DBEdit_Kol.Text	6
return	4	int.Parse	8
=	34	GridSpisok.CurrentRow.Index	8
<	1	GridSpisok.Rows[].Cells[«Kol»].Value	2
==	6	Query.ConnStr	4
<=	1	ClassStr.RegimZakaz	5
>	2	ClassStr.TekCodZakaz	2
?:	2	GridSpisok.Rows[].Cells[«CodLekarstvo»].Value	3
+	9	GridSpisok.Rows[].Cells[«CodPreparat»].Value	3
-	1	MySqlConnection	8
()	66	ClassSostavZakaz	2
But_AddInZakaz_Click	1	Podkl.Add_JSON	1
MessageBox.Show	5	ZaprosSostavZakaz	2
DBEdit_Kol.Focus	3	MySqlDataAdapter	3
int.Parse	8	DataTable	3

Продолжение таблицы 50

Операторы	Число вхождений	Операнды	Число вхождений
ToString	14	datadapterSostavZakaz.Fill	2
Convert.ToInt32	1		
cmd.ExecuteScalar	2		
cmd.Connection.Open	2		
cmd.Connection.Close	2		
cmd.Parameters.AddWithValue	3		
JsonConvert.SerializeObject	1		
Podkl.Add_JSON	1		
datadapterSostavZakaz.Fill	1		
datadapterSpisok.Fill	1		
command.ExecuteNonQuery	1		
{ }	7		
;	53		
" "	29		
[]	18		
.	117		
private	1		
Void	1		
var	1		
new	13		
@	7		
n1 = 37	N1 = 428	n2 = 17	N2 = 71

Длина программы – это сумма двух компонентов N1 и N2, где N1 – это общее количество операторов и операндов в программе (в данном случае 428), а N2 – это количество уникальных операторов и операндов (в данном случае 71). Определим длину программы по формуле (1):

$$N = N1 + N2 = 499. \quad (1)$$

Словарь программы (n) представляет собой количество уникальных операторов и операндов, n1 – это количество уникальных операторов (в данном случае 37), а n2 – количество уникальных операндов (в данном случае 17). Определим словарь программы по формуле (2):

$$n = n1 + n2 = 54. \quad (2)$$

При наличии этих значений можно рассчитать реальный объем программы по формуле (3) и уровень качества программирования по формуле (4):

$$V = N * \log_2 n, \quad (3)$$

$$L = \frac{(2*n2)}{(n1*N2)}. \quad (4)$$

После подстановки значений в формулы, получаем реальный объем программы $V = 2871,7$, уровень качества программирования $L = 0,013$.

Посчитаем сложность качества программирования программы, используя значения объема кода (3) и уровня качества программирования (4) по формуле (5):

$$Ec = \frac{V}{L} * 2. \quad (5)$$

После проведения расчетов, получаем сложность качества программирования $Ec = 441800$. Данный результат указывает на то, что написанный программный код не является лаконичным и простым в понимании. Тем не менее, это объясняется наличием большого количества операторов и операнд в коде.

4.3.3 Метрика Чепина

Суть метрики Чепина состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа использования переменных ввода-вывода.

Все переменные делятся на четыре функциональные группы:

- множество «Р» – вводимые переменные для расчетов и вывода;
- множество «М» – модифицируемые или создаваемые внутри программы переменные;
- множество «С» – переменные, участвующие в управлении работой программного модуля (управляющие переменные);
- множество «Т» – неиспользуемые в программе переменные [28].

В данной работе расчет метрики Чепина приведен для метода `But_AddInZakaz_Click`, представленным на листинге 8.

Все переменные представлены в таблице 51.

Таблица 51 – Все переменные метода But_AddInZakaz_Click

Множество «Р»	Множество «М»	Множество «С»	Множество «Т»
DBEdit_Kol.Text	Kol	ClassStr.RegimZakaz	Отсутствуют
GridSpisok	cmd	ClassStr.TekCodZakaz	
	CodSostavZakaz	Query.ConnStr	
	Per2	GridSpisok	
	datadapterSostavZakaz	cmd	
	bufferSostavZakaz		
	zaproz		
	connection		
	command		
	iStr		
	datadapterSpisok		
	bufferSpisok		
	StrJson		
P = 2	M = 13	C = 5	T = 0

После подсчета всех необходимых переменных выведем значение метрики

Чепина по формуле (6):

$$Q = a_1 * P + a_2 * M + a_3 * C + a_4 * T, \quad (6)$$

где a_1, a_2, a_3, a_4 весовые коэффициенты.

Весовые коэффициенты используются для того, чтобы показать различное влияние на сложность программы каждой функциональной группы. Весовые коэффициенты распределяются следующим образом: $a_1 = 1, a_2 = 2, a_3 = 3, a_4 = 0,5$. С учетом всех коэффициентов выражение имеет вид:

$$Q = P + 2M + 3C + 0,5T. \quad (6)$$

После подстановки всех значений в формулу (6) получаем $Q = 43$. В результате получаем, что код метода But_AddInZakaz_Click является достаточно сложным. Но это также говорит о том, что в коде имеется много уникальных переменных для работы с данными.

Заключение

В процессе выполнения выпускной квалификационной работы был выполнен анализ предметной области, в рамках которого рассмотрена актуальность создания приложения обработки и анализа индивидуальных заказов на изготовление лекарственных средств. Был проведен сравнительный анализ существующих программных средств, специализирующихся на автоматизации обработки и анализа индивидуальных заказов на изготовление лекарственных средств, была поставлена задача на разработку и определен бюджет разработки.

В рамках проектирования приложения пользователя были смоделированы функциональные возможности приложения, проведена разработка концептуальной модели приложения, разработка требований к создаваемой системе, разработка прототипа приложения и обоснование выбора средств разработки. Обоснование выбора средств разработки включала в себя обоснование выбора средств автоматизации, инструментальных средств разработки программного обеспечения, языковых средств разработки, средств управления данными.

В процессе разработки программного обеспечения было выполнено следующее: разработка физической схемы базы данных, пользовательского интерфейса приложения, методы доступа к базе данных, разработка запросов. Было выполнено тестирование и оценка качества программного кода на основе метрик.

В результате было разработано приложение пользователя для прототипа обработки и анализа индивидуальных заказов на изготовление лекарственных средств.

Список использованных источников

1. Методические рекомендации Минздрава РФ от 14.05.98 N 98/124 [Электронный ресурс]: сайт КонтурНорматив. URL: <https://normativ.kontur.ru/document?moduleId=1&documentId=56698> (дата обращения: 07.04.2024).
2. Изготовление лекарств по индивидуальным рецептам [Электронный ресурс]: сайт Клерк. URL: <https://www.klerk.ru/buh/articles/153506/> (дата обращения: 07.04.2024).
3. Должностные обязанности фармацевта [Электронный ресурс]: сайт «Академия профессиональных стандартов». URL: <https://dpoaps.ru/blog/dolzhnostnye-obyazannosti-farmacevta/> (дата обращения: 08.04.2024).
4. Должностные обязанности провизора в аптеке [Электронный ресурс]: сайт «Академия профессиональных стандартов». URL: <https://dpoaps.ru/blog/dolzhnostnye-obyazannosti-farmacevta/> (дата обращения: 08.04.2024).
5. Простое руководство по диаграммам активности UML [Электронный ресурс]: сайт «Creately». URL: <https://creately.com/blog/ru/uncategorized-ru/учебник-по-диаграмме-активности/> (дата обращения: 08.04.2024).
6. Описание программного продукта «Больничная аптека» [Электронный ресурс]: сайт Фирма 1С. URL: <https://med.1cfresh.com/ba.html> (дата обращения 10.04.2024).
7. Описание программного продукта «Единая система учета и контроля льготных рецептов» [Электронный ресурс]: сайт ЮНИКО. URL: <https://www.uniko.ru/software/lrecept.php> (дата обращения 10.04.2024).
8. Описание программного продукта «Mobile SMARTS: МДЛП» [Электронный ресурс]: сайт Клеверенс. URL: <https://www.cleverence.ru/solutions/track-and-trace-for-medicals-in-the-pharmacy/> (дата обращения 10.04.2024).

9. Правила оформления рецептов [Электронный ресурс]: сайт КатренСтиль. URL: https://www.katrenstyle.ru/articles/blog/vam_naznacheno (дата обращения 26.04.2024).
10. Маклаков С.В. Моделирование бизнес-процессов с BP-Win 4.0. Москва, «ДИАЛОГМИФИ», 2019 (дата обращения 03.05.2024).
11. Описание программного продукта BpWin [Электронный ресурс]: сайт Bourabai Research. URL: <http://bourabai.ru/cm/bpwin.htm> (дата обращения 10.04.2024).
12. Microsoft 365 Team Простое руководство по UML-диаграммам и моделированию БД // Microsoft [Электронный ресурс] URL: <https://www.microsoft.com/ru-ru/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling> (дата обращения: 11.05.2024).
13. Архитектура «клиент-сервер» [Электронный ресурс]. – URL: http://www.mstu.edu.ru/study/materials/zelenkov/ch_7_1.html (дата обращения 12.05.2024).
14. ERwin Data Modeler [Электронный ресурс] URL: <https://www.qbssoftware.com/erwin-data-modeler.html> (дата обращения: 14.05.2024).
15. Visio и Microsoft 365 [Электронный ресурс] URL: <https://www.microsoft.com/ru-ru/microsoft-365/visio/flowchart-software> (дата обращения: 14.05.2024).
16. IBM InfoSphere Data Architect [Электронный ресурс] URL: https://studme.org/77240/informatika/infosphere_data_architect (дата обращения: 14.05.2024).
17. MySQL Workbench [Электронный ресурс] URL: <https://www.mysql.com/products/workbench/> (дата обращения: 14.05.2024).
18. Microsoft Visual Studio 2022 [Электронный ресурс] URL: <https://visualstudio.microsoft.com/ru/> (дата обращения: 15.05.2024).
19. Язык программирования C# [Электронный ресурс] URL: <https://gb.ru/blog/yazyk-programmirovaniya-c-sharp/> (дата обращения: 16.05.2024).

20. MySQL [Электронный ресурс] URL: <https://skillbox.ru/media/code/mysql-chto-eto-kak-ustroena-chem-khorosha-i-kak-nachat-s-ney-rabotat/> (дата обращения: 17.05.2024).
21. Flexberry – Диаграмма последовательности [Электронный ресурс]. – URL: https://flexberry.github.io/ru/fd_sequence-diagram.html (дата обращения 17.05.2024).
22. Руденко И. Различия между контролем качества и обеспечением качества в IT // Wezom [Электронный ресурс]. URL: <https://wezom.com.ua/blog/razlichija-mezhdu-kontrolem-kachestva-i-obespecheniem-kachestva-v-it> (дата обращения: 18.05.2024).
23. Pittet S. The different types of software testing // atlassian [Электронный ресурс]. URL: <https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing> (дата обращения: 18.05.2024).
24. Метод белого ящика [Электронный ресурс]. URL: <https://tquality.ru/blog/testirovanie-metodom-belogo-yashchika/> (дата обращения: 20.05.2024).
25. Модульное тестирование [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/unit-testirovanie/> (дата обращения: 20.05.2024).
26. Блок-схема [Электронный ресурс]. URL: <https://www.lucidchart.com/pages/ru/%D0%B1%D0%BB%D0%BE%D0%BA-%D1%81%D1%85%D0%B5%D0%BC%D0%B0> (дата обращения: 20.05.2024).
27. Управляющий граф [Электронный ресурс]. URL: <https://studfile.net/preview/1692647/page:21/> (дата обращения: 20.05.2024).
28. Исходные строки кода [Электронный ресурс]. URL: https://translated.turbopages.org/proxy_u/en-ru.ru.cdc2b6e9-6661c422-6563db22-74722d776562/https/en.wikipedia.org/wiki/Source_lines_of_code (дата обращения: 21.05.2024).
29. Метрика Чепина [Электронный ресурс]. URL: https://studwood.net/2361022/informatika/metriki_chepina (дата обращения: 25.05.2024).

Приложение А
(справочное)
Глоссарий

Рецептурный журнал – это документ, который используется для учета индивидуального изготовления лекарственных средств в аптечных организациях.

Внутриаптечная заготовка – это процесс изготовления малых серий лекарственных препаратов в аптечных организациях.

Промышленный регламент – это технологический документ действующего серийного производства лекарственного средства.

Фармакопейная статья – это документ, в котором прописана информация об упаковках, условиях, сроках хранения, способах контроля, а также обозначены нормы качества лекарств.

Фармакологическое дело – это область медицины и фармацевтики, занимающаяся изучением лекарственных средств, их разработкой, производством, контролем качества, безопасностью, эффективностью и применением для лечения, профилактики и диагностики заболеваний.

Академия профессиональных стандартов – это организация дополнительного профессионального образования, которая организует обучение медицинских работников и врачей по дополнительным образовательным программам в соответствии с ФГОС.

Основная фармацевтическая продукция (ОФП) – это отдельные компоненты и вещества, из которых производятся лекарственные средства.

Лечебно-профилактические учреждения (ЛПУ) – это учреждения, оказывающие медицинскую помощь.

Хозрасчетные лечебно-профилактические учреждения – группа учреждений здравоохранения, которые финансируются за счет прибыли от их деятельности.

ФГИС «Мониторинг движения лекарственных препаратов» (ФГИС МДЛП) – это Федеральная государственная информационная система, созданная для отслеживания и контроля движения лекарственных препаратов на всех этапах их жизненного цикла, от производства до конечного потребителя.

Федеральный закон № 61-ФЗ «Об обращении лекарственных средств» – это нормативный акт, регулирующий производство, хранение, транспортировку, государственную регистрацию, контроль качества, реализацию и утилизацию лекарственных препаратов в Российской Федерации.

Льготный отпуск лекарственных средств – это предоставление лекарственных препаратов на льготных условиях определённым категориям граждан, которым положены льготы в соответствии с законодательством, бесплатно или по сниженной цене.

Клинико-экспертная комиссия (КЭК) – это специализированный орган, создаваемый при медицинских учреждениях и организациях здравоохранения для проведения клинико-экспертной работы.

Нозология – это раздел медицины, изучающий заболевания, их классификацию, диагностику и общие принципы лечения.

Экстемпоральная рецептура – это раздел фармации, занимающийся изготовлением лекарственных средств непосредственно в аптеках по индивидуальным рецептам врачей для конкретных пациентов.

Закон о монетизации льгот – это законодательный акт, принятый в Российской Федерации, который преобразовал систему предоставления натуральных льгот в денежные компенсации.

Обратный акцент – это термин, который используется в банковском и финансовом секторе для обозначения действия, когда получатель финансового документа, подтверждает свое согласие на его условия и обязательства.

«Честный ЗНАК» – это национальная система цифровой маркировки и прослеживания товаров в России, которая позволяет отслеживать путь товара от производителя до конечного потребителя, обеспечивая подлинность и качество продукции.

МДЛП – это организация, которая расшифровывается как Международная Доска Лекарственных Продуктов, она разрабатывает, утверждает и поддерживает общепринятые международные стандарты и рекомендации в области качества, безопасности и эффективности лекарственных препаратов.

Среда исполнения Common Language Runtime (CLR) – предоставляет среду выполнения для управляемого кода, написанного на различных языках программирования, таких как C#, Visual Basic .NET, F#, и других, является частью платформы .NET Framework, разработанной компанией Microsoft.

Кроссплатформенность – способность программного обеспечения работать на различных операционных системах или платформах без изменения исходного кода.

Компания MySQL AB – компания, специализирующейся на разработке и поддержке баз данных, включая известную реляционную систему управления базами данных MySQL.

Sun Microsystems – компания занималась разработкой и производством компьютерного оборудования, программного обеспечения и различных информационных технологий.

FreeBSD – это операционная система с открытым исходным кодом, основанная на системе UNIX.

Проприетарные модули – это программные компоненты или расширения, разработанные и распространяемые компаниями или организациями под закрытой лицензией.

Сериализация – процесс преобразования объекта в поток байтов для сохранения или передачи в память, базу данных или файл.

Приложение Б
(рекомендуемое)

Диаграмма вариантов использования приложения

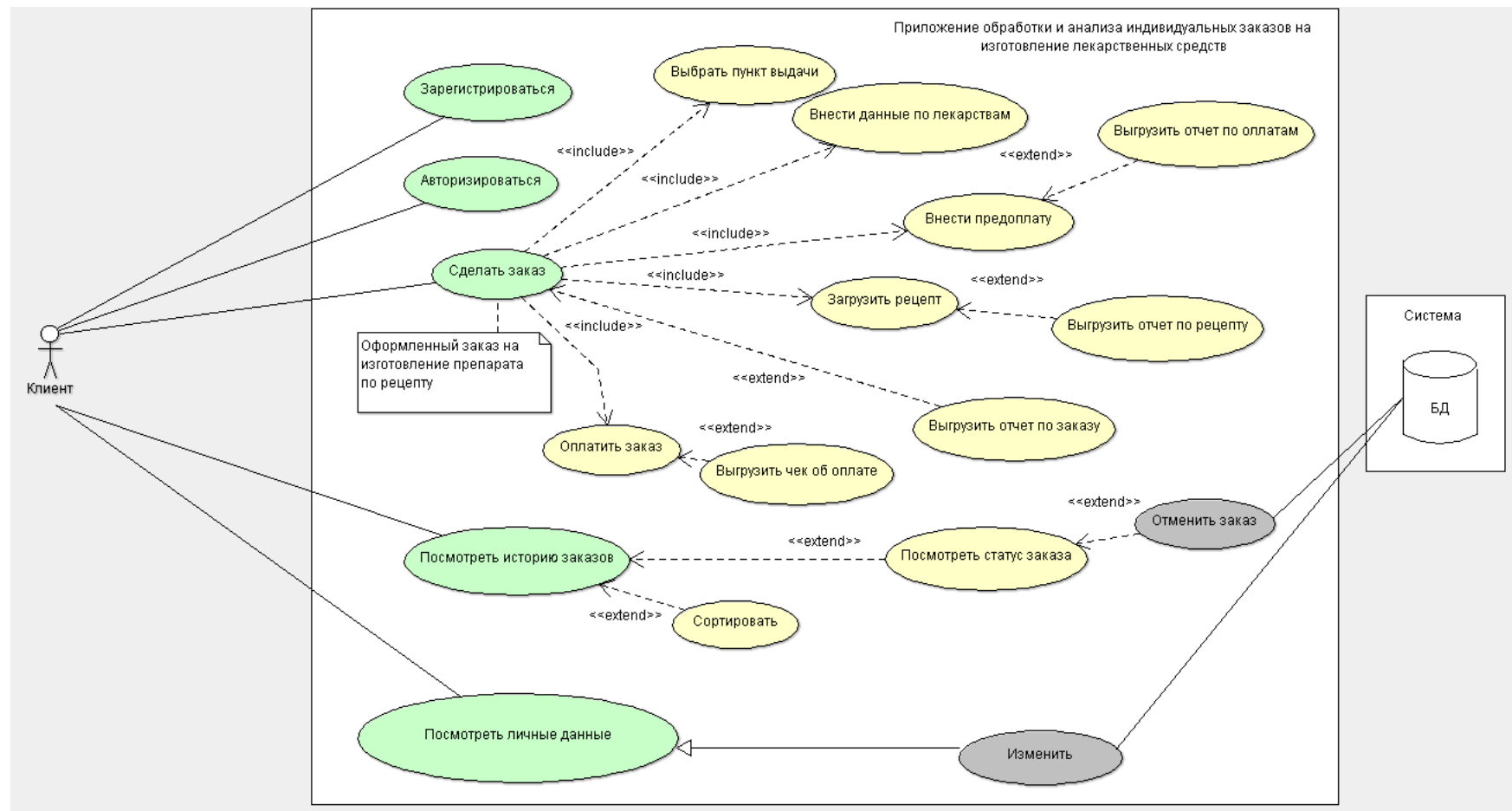


Рисунок Б.1 – Функциональные возможности клиента в приложении (диаграмма вариантов использования)

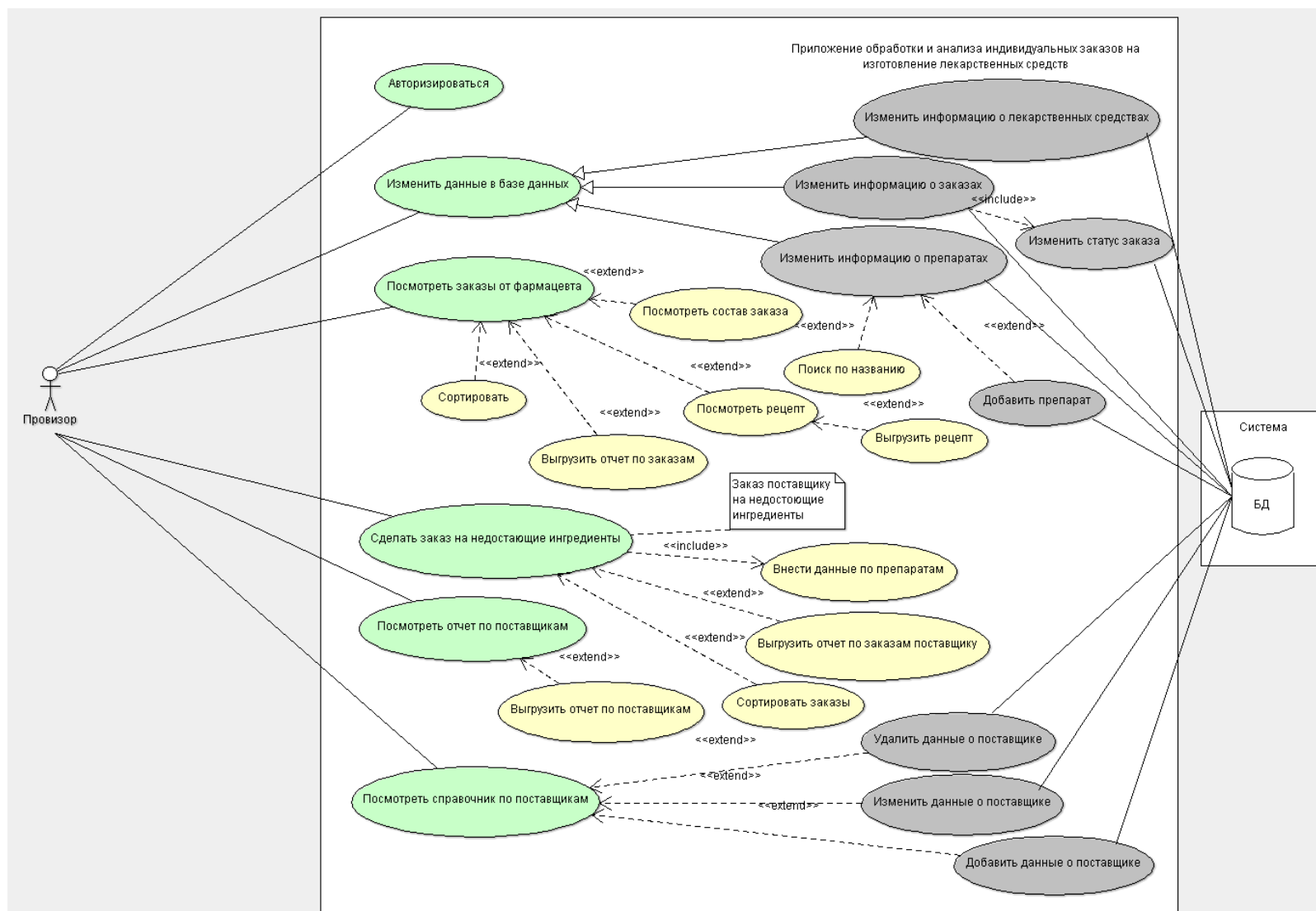


Рисунок Б.3 – Функциональные возможности провизора в приложении (диаграмма вариантов использования)

Приложение В
(рекомендуемое)
Сценарии бизнес-процессов приложения

Таблица В.1 – Сценарий прецедента «Посмотреть историю заказов»

Название варианта использования	Посмотреть историю заказов
Актеры	Клиент
Краткое описание	Просмотр истории заказов
Второстепенные действующие лица	Нет
Предусловие	Клиент зарегистрировался, вошел в свой аккаунт и сделал заказ
Типичный ход событий	<ol style="list-style-type: none"> 1. Клиент открывает вкладку с заказами. 2. Клиент отсортировал заказы. 3. Посмотрел статус заказа. 4. Отменил заказ.
Постусловие	Приложение изменило статус заказа, как отмененный

Таблица В.2 – Сценарий прецедента «Посмотреть личные данные»

Название варианта использования	Посмотреть личные данные
Актеры	Клиент
Краткое описание	Просмотр личных данных
Второстепенные действующие лица	Нет
Предусловие	Клиент зарегистрировался, вошел в свой аккаунт
Типичный ход событий	<ol style="list-style-type: none"> 1. Клиент открывает окно с личными данными. 2. Клиент изменил свои данные.
Постусловие	Приложение изменило личные данные клиента

Таблица В.3 – Сценарий прецедента «Посмотреть информацию по заказам»

Название варианта использования	Посмотреть информацию по заказам
Актеры	Фармацевт
Краткое описание	Просмотр информации по заказам клиента
Второстепенные действующие лица	Нет
Предусловие	Фармацевт авторизировался в приложении
Типичный ход событий	<ol style="list-style-type: none"> 1. Открывает вкладку с заказами. 2. Сортирует заказы. 3. Изменяет информацию по заказам. 4. Удаляет информацию по заказам. 5. Может выгрузить отчет по всем заказам клиентов и их информацией.
Постусловие	Приложение изменило информацию о заказах и выгрузило отчет

Таблица В.4 – Сценарий прецедента «Посмотреть отчет историй по всем заказам клиентов»

Название варианта использования	Посмотреть отчет историй по всем заказам клиентов
Актеры	Фармацевт
Краткое описание	Просмотр информации по заказам клиентов
Второстепенные действующие лица	Нет
Предусловие	Фармацевт авторизировался в приложении
Типичный ход событий	1. Открывает вкладку с отчетами из меню программы. 2. Сортирует заказы.
Постусловие	Приложение вывело информацию о всех заказах клиентов

Таблица В.5 – Сценарий прецедента «Посмотреть отчет по клиентам»

Название варианта использования	Посмотреть отчет по клиентам
Актеры	Фармацевт
Краткое описание	Просмотр информации о всех зарегистрированных клиентах
Второстепенные действующие лица	Нет
Предусловие	Фармацевт авторизировался в приложении
Типичный ход событий	1. Открывает вкладку с отчетами из меню программы. 2. Выгружает отчет по клиентам, которые зарегистрированы в приложении.
Постусловие	Приложение показало информацию о клиентах

Таблица В.6 – Сценарий прецедента «Посмотреть информацию по клиентам»

Название варианта использования	Посмотреть информацию по клиентам
Актеры	Фармацевт
Краткое описание	Просмотр информации и возможность изменения данных о клиентах
Второстепенные действующие лица	Нет
Предусловие	Фармацевт авторизировался в приложении
Типичный ход событий	1. Открывает окно с информацией о клиентах. 2. Добавляет информацию о новом клиенте. 3. Изменяет информацию о зарегистрированном клиенте. 4. Удаляет информацию о зарегистрированном клиенте.
Постусловие	Приложение изменило данные по клиентам

Таблица В.7 – Сценарий прецедента «Изменить данные в базе данных»

Название варианта использования	Изменить данные в базе данных
Актеры	Провизор
Краткое описание	Изменение информации о заказах, препаратах и лекарственных средствах
Второстепенные действующие лица	Нет
Предусловие	Провизор авторизировался в приложении
Типичный ход событий	<ol style="list-style-type: none"> 1. Открывает окно с заказами на изготовление лекарственных средств. 2. Изменяет статус заказа. 3. Открывает окно с информацией по препаратам. 4. По поиску ищет нужный препарат. 5. Добавляет, удаляет или изменяет препарат. 6. Открывает окно с лекарственными средствами. 7. Добавляет, удаляет или изменяет лекарственные средства.
Постусловие	Приложение изменило данные по лекарственным средствам, заказам и препаратам.

Таблица В.8 – Сценарий прецедента «Сделать заказ на недостающие ингредиенты»

Название варианта использования	Сделать заказ на недостающие ингредиенты
Актеры	Провизор
Краткое описание	Оформление заказа поставщику на недостающие препараты
Второстепенные действующие лица	Нет
Предусловие	Провизор авторизировался в приложении
Типичный ход событий	<ol style="list-style-type: none"> 1. Открывает окно с заказами поставщику. 2. Сортирует заказы. 3. Добавляет новый заказ, где вносит данные по препаратам. 4. Выгружает отчет по заказам поставщику.
Постусловие	Приложение создало заказ поставщику.

Таблица В.9 – Сценарий прецедента «Посмотреть отчет по поставщикам»

Название варианта использования	Посмотреть отчет по поставщикам
Актеры	Провизор
Краткое описание	Просмотр информации по заказам у определенного поставщика
Второстепенные действующие лица	Нет
Предусловие	Провизор авторизировался в приложении

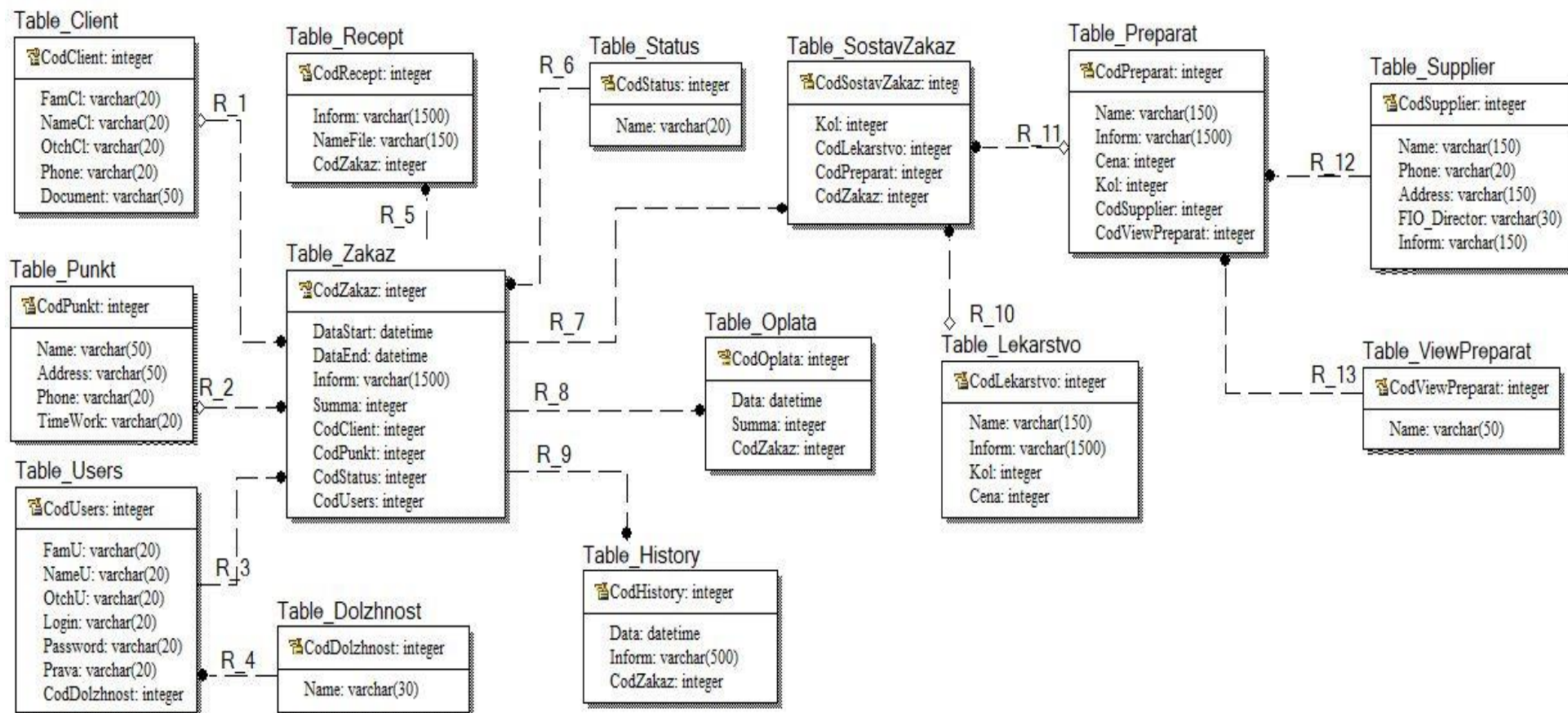
Продолжение таблицы В.9

Типичный ход событий	1. Открывает окно с отчетом по поставщику. 2. Выбирает нужного поставщика из таблицы. 3. Выгружает отчет с информацией о препаратах, заказанных у этого поставщика.
Постусловие	Приложение выгрузило отчет в Excel по поставщику.

Таблица В.10 – Сценарий прецедента «Посмотреть справочник по поставщикам»

Название варианта использования	Посмотреть справочник по поставщикам
Актеры	Провизор
Краткое описание	Просмотр информации по поставщикам в справочнике
Второстепенные действующие лица	Нет
Предусловие	Провизор авторизировался в приложении
Типичный ход событий	1. Открывает вкладку с информацией по поставщикам. 2. Добавляет нового поставщика. 3. Изменяет данные существующего поставщика. 4. Удаляет данные о поставщике.
Постусловие	Приложение изменило справочник по поставщикам.

Приложение Г
(рекомендуемое)
Физическая модель предметной области



Приложение Д
(рекомендуемое)
Листинг SQL скриптов

Листинг Д.1 – SQL-скрипт создания таблицы Lekarstvo.table_status

```

-- -----
-- Table `Lekarstvo`.`table_status`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_status` (
  `CodStatus` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NOT NULL,
  PRIMARY KEY (`CodStatus`))
ENGINE = InnoDB;

```

Листинг Д.2 – SQL-скрипт создания таблицы Lekarstvo.table_client

```

-- -----
-- Table `Lekarstvo`.`table_client`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_client` (
  `CodClient` INT NOT NULL AUTO_INCREMENT,
  `FamCl` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NOT NULL,
  `NameCl` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NULL,
  `OtchCl` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NULL,
  `Phone` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NULL,
  `Document` VARCHAR(50) CHARACTER SET 'utf8mb4' COLLATE
  'utf8mb4_0900_ai_ci' NOT NULL,
  PRIMARY KEY (`CodClient`))
ENGINE = InnoDB;

```

Листинг Д.3 – SQL-скрипт создания таблицы Lekarstvo.table_dolzhnost

```

-- -----
-- Table `Lekarstvo`.`table_dolzhnost`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_dolzhnost` (
  `CodDolzhnost` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(30) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NOT NULL,
  PRIMARY KEY (`CodDolzhnost`))
ENGINE = InnoDB;

```

Листинг Д.4 – SQL-скрипт создания таблицы Lekarstvo.table_users

```

-- -----
-- Table `Lekarstvo`.`table_users`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_users` (
  `CodUsers` INT NOT NULL AUTO_INCREMENT,
  `FamU` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NOT NULL,
  `NameU` VARCHAR(20) NULL,
  `OtchU` VARCHAR(20) NULL,
  `Login` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NOT NULL,

```

```

`Password` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE
'utf8mb4_0900_ai_ci' NOT NULL,
`Prava` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
NOT NULL,
`CodDolzhnost` INT NOT NULL,
PRIMARY KEY (`CodUsers`),
INDEX `FK_DolzhnostUsers_idx` (`CodDolzhnost` ASC) VISIBLE,
CONSTRAINT `FK_DolzhnostUsers`
    FOREIGN KEY (`CodDolzhnost`)
    REFERENCES `Lekarstvo`.`table_dolzhnost` (`CodDolzhnost`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;

```

Листинг Д.5 – SQL-скрипт создания таблицы Lekarstvo.table_punkt

```

-- -----
-- Table `Lekarstvo`.`table_punkt`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_punkt` (
  `CodPunkt` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(50) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NOT NULL,
  `Address` VARCHAR(50) CHARACTER SET 'utf8mb4' COLLATE
  'utf8mb4_0900_ai_ci' NOT NULL,
  `Phone` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NULL,
  `TimeWork` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE
  'utf8mb4_0900_ai_ci' NOT NULL,
  PRIMARY KEY (`CodPunkt`))
ENGINE = InnoDB;

```

Листинг Д.6 – SQL-скрипт создания таблицы Lekarstvo.table_zakaz

```

-- -----
-- Table `Lekarstvo`.`table_zakaz`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_zakaz` (
  `CodZakaz` INT NOT NULL AUTO_INCREMENT,
  `DataStart` DATE NOT NULL,
  `DataEnd` DATE NOT NULL,
  `Inform` VARCHAR(1500) CHARACTER SET 'utf8mb4' COLLATE
  'utf8mb4_0900_ai_ci' NOT NULL,
  `Summa` INT NOT NULL,
  `CodClient` INT NULL,
  `CodPunkt` INT NULL,
  `CodStatus` INT NOT NULL,
  `CodUsers` INT NOT NULL,
  PRIMARY KEY (`CodZakaz`),
  INDEX `FK_ClientZakaz_idx` (`CodClient` ASC) VISIBLE,
  INDEX `FK_PunktZakaz_idx` (`CodPunkt` ASC) VISIBLE,
  INDEX `FK_StatusZakaz_idx` (`CodStatus` ASC) VISIBLE,
  INDEX `FK_UsersZakaz_idx` (`CodUsers` ASC) VISIBLE,
  CONSTRAINT `FK_ClientZakaz`
    FOREIGN KEY (`CodClient`)
    REFERENCES `Lekarstvo`.`table_client` (`CodClient`)
    ON DELETE SET NULL
    ON UPDATE SET NULL,

```

```

CONSTRAINT `FK_PunktZakaz`
    FOREIGN KEY (`CodPunkt`)
    REFERENCES `Lekarstvo`.`table_punkt` (`CodPunkt`)
    ON DELETE SET NULL
ON UPDATE SET NULL,
CONSTRAINT `FK_StatusZakaz`
    FOREIGN KEY (`CodStatus`)
    REFERENCES `Lekarstvo`.`table_status` (`CodStatus`)
ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT `FK_UsersZakaz`
    FOREIGN KEY (`CodUsers`)
    REFERENCES `Lekarstvo`.`table_users` (`CodUsers`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;

```

Листинг Д.7 – SQL-скрипт создания таблицы Lekarstvo.table_lekarstvo

```

-- -----
-- Table `Lekarstvo`.`table_lekarstvo`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_lekarstvo` (
  `CodLekarstvo` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(150) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NOT NULL,
  `Inform` VARCHAR(1500) CHARACTER SET 'utf8mb4' COLLATE
  'utf8mb4_0900_ai_ci' NOT NULL,
  `Kol` INT NULL,
  `Cena` INT NULL,
  PRIMARY KEY (`CodLekarstvo`))
ENGINE = InnoDB;

```

Листинг Д.8 – SQL-скрипт создания таблицы Lekarstvo.table_viewpreparat

```

-- -----
-- Table `Lekarstvo`.`table_viewpreparat`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_viewpreparat` (
  `CodViewPreparat` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(50) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NOT NULL,
  PRIMARY KEY (`CodViewPreparat`))
ENGINE = InnoDB;

```

Листинг Д.9 – SQL-скрипт создания таблицы Lekarstvo.table_supplier

```

-- -----
-- Table `Lekarstvo`.`table_supplier`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_supplier` (
  `CodSupplier` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(150) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NOT NULL,
  `Phone` VARCHAR(20) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NULL,
  `Address` VARCHAR(150) CHARACTER SET 'utf8mb4' COLLATE
  'utf8mb4_0900_ai_ci' NULL,

```

```
`FIO_Director` VARCHAR(30) CHARACTER SET 'utf8mb4' COLLATE
'utf8mb4_0900_ai_ci' NULL,
`Inform` VARCHAR(150) CHARACTER SET 'utf8mb4' COLLATE
'utf8mb4_0900_ai_ci' NULL,
PRIMARY KEY (`CodSupplier`))
ENGINE = InnoDB;
```

Листинг Д.10 – SQL-скрипт создания таблицы Lekarstvo.table_preparat

```
-- -----
-- Table `Lekarstvo`.`table_preparat`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_preparat` (
  `CodPreparat` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(150) CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_0900_ai_ci'
  NOT NULL,
  `Inform` VARCHAR(1500) CHARACTER SET 'utf8mb4' COLLATE
  'utf8mb4_0900_ai_ci' NOT NULL,
  `Cena` INT NULL,
  `Kol` INT NULL,
  `CodViewPreparat` INT NOT NULL,
  `CodSupplier` INT NOT NULL,
  PRIMARY KEY (`CodPreparat`),
  INDEX `PreparatViewLekarstvo_idx` (`CodViewPreparat` ASC) VISIBLE,
  INDEX `PreparatSupplier_idx` (`CodSupplier` ASC) VISIBLE,
  CONSTRAINT `PreparatViewPreparat`
    FOREIGN KEY (`CodViewPreparat`)
    REFERENCES `Lekarstvo`.`table_viewpreparat` (`CodViewPreparat`)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  CONSTRAINT `PreparatSupplier`
    FOREIGN KEY (`CodSupplier`)
    REFERENCES `Lekarstvo`.`table_supplier` (`CodSupplier`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Листинг Д.11 – SQL-скрипт создания таблицы Lekarstvo.table_sostavzakaz

```
-- -----
-- Table `Lekarstvo`.`table_sostavzakaz`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_sostavzakaz` (
  `CodSostavZakaz` INT NOT NULL AUTO_INCREMENT,
  `Kol` INT NOT NULL,
  `CodZakaz` INT NOT NULL,
  `CodLekarstvo` INT NULL,
  `CodPreparat` INT NULL,
  PRIMARY KEY (`CodSostavZakaz`),
  INDEX `FK_ZakazSostavZakaz_idx` (`CodZakaz` ASC) VISIBLE,
  INDEX `FK_LekarstvoSostavZakaz_idx` (`CodLekarstvo` ASC) VISIBLE,
  INDEX `FK_PreparatSostavZakaz_idx` (`CodPreparat` ASC) VISIBLE,
  CONSTRAINT `FK_ZakazSostavZakaz`
    FOREIGN KEY (`CodZakaz`)
    REFERENCES `Lekarstvo`.`table_zakaz` (`CodZakaz`)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  CONSTRAINT `FK_LekarstvoSostavZakaz`
```

```

        FOREIGN KEY (`CodLekarstvo`)
        REFERENCES `Lekarstvo`.`table_lekarstvo` (`CodLekarstvo`)
ON DELETE SET NULL
ON UPDATE SET NULL,
CONSTRAINT `FK_PreparatSostavZakaz`
        FOREIGN KEY (`CodPreparat`)
        REFERENCES `Lekarstvo`.`table_preparat` (`CodPreparat`)
ON DELETE SET NULL
ON UPDATE SET NULL)
ENGINE = InnoDB;

```

Листинг Д.12 – SQL-скрипт создания таблицы Lekarstvo.table_oplata

```

-- -----
-- Table `Lekarstvo`.`table_oplata`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_oplata` (
  `CodOplata` INT NOT NULL AUTO_INCREMENT,
  `Data` DATE NOT NULL,
  `Summa` INT NOT NULL,
  `CodZakaz` INT NOT NULL,
  PRIMARY KEY (`CodOplata`),
  INDEX `FK_ZakazOplata_idx` (`CodZakaz` ASC) VISIBLE,
  CONSTRAINT `FK_ZakazOplata`
    FOREIGN KEY (`CodZakaz`)
    REFERENCES `Lekarstvo`.`table_zakaz` (`CodZakaz`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;

```

Листинг Д.13 – SQL-скрипт создания таблицы Lekarstvo.table_history

```

-- -----
-- Table `Lekarstvo`.`table_history`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_history` (
  `CodHistory` INT NOT NULL AUTO_INCREMENT,
  `Data` DATE NOT NULL,
  `Inform` VARCHAR(500) NOT NULL,
  `CodZakaz` INT NOT NULL,
  PRIMARY KEY (`CodHistory`),
  INDEX `FK_ZakazHistory_idx` (`CodZakaz` ASC) VISIBLE,
  CONSTRAINT `FK_ZakazHistory`
    FOREIGN KEY (`CodZakaz`)
    REFERENCES `Lekarstvo`.`table_zakaz` (`CodZakaz`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;

```

Листинг Д.14 – SQL-скрипт создания таблицы Lekarstvo.table_recept

```

-- -----
-- Table `Lekarstvo`.`table_recept`
-- -----
CREATE TABLE IF NOT EXISTS `Lekarstvo`.`table_recept` (
  `CodRecept` INT NOT NULL AUTO_INCREMENT,
  `Inform` VARCHAR(1500) CHARACTER SET 'utf8mb4' COLLATE
  'utf8mb4_0900_ai_ci' NOT NULL,

```

```

`NameFile` VARCHAR(150) CHARACTER SET 'utf8mb4' COLLATE
'utf8mb4_0900_ai_ci' NULL,
`CodZakaz` INT NOT NULL,
PRIMARY KEY (`CodRecept`),
INDEX `FK_ZakazRecept_idx` (`CodZakaz` ASC) VISIBLE,
CONSTRAINT `FK_ZakazRecept`
    FOREIGN KEY (`CodZakaz`)
    REFERENCES `Lekarstvo`.`table_zakaz` (`CodZakaz`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

Приложение Е
(рекомендуемое)
Листинг программы

Листинг Е.1 – Класс ClassStr

```
public class ClassStr
{
    public static string TekPrava = "";
    public static int RegimZakaz = 0; //1- заказы лекарственных средств
на изготовление от клиентов
    //2- заказы препаратов у поставщиков
    public static int TekCodUsers = 0;
    public static int TekCodClient = 0;
    public static string TekFioClient = "";
    public static int TekSpravochnik = 0;

    public static int TekCodZakaz = 0;
    public static string TekNameZakaz = "";
    public static string TekFioCl = "";
}
```

Листинг Е.2 – Класс ClassUsers

```
public class ClassUsers
{
    public int CodUsers { get; set; }
    public string FamU { get; set; }
    public string NameU { get; set; }
    public string OtchU { get; set; }
    public string Login { get; set; }
    public string Password { get; set; }
    public string Prava { get; set; }
    public int CodDolzhnost { get; set; }

    public ClassUsers() { }

    public ClassUsers(int _CodUsers, string _FamU, string _NameU,
string _OtchU, string _Login, string _Password, string _Prava, int
_CodDolzhnost)
    {
        CodUsers = _CodUsers;
        FamU = _FamU;
        NameU = _NameU;
        OtchU = _OtchU;
        Login = _Login;
        Password = _Password;
        Prava = _Prava;
        CodDolzhnost = _CodDolzhnost;
    }
}
```

Листинг Е.3 – Класс ClassClient

```
public class ClassClient
{
    public int CodClient { get; set; }
    public string FamCl { get; set; }
    public string NameCl { get; set; }
    public string OtchCl { get; set; }
    public string Phone { get; set; }
    public string Document { get; set; }
}
```



```

public ClassClient() { }

public ClassClient(int _CodClient, string _FamCl, string _NameCl,
string _OtchCl, string _Phone, string _Document)
{
    CodClient = _CodClient;
    FamCl = _FamCl;
    NameCl = _NameCl;
    OtchCl = _OtchCl;
    Phone = _Phone;
    Document = _Document;
}
}

```

Листинг Е.4 – Класс ClassPunkt

```

public class ClassPunkt
{
    public int CodPunkt { get; set; }
    public string Name { get; set; }
    public string Address { get; set; }
    public string Phone { get; set; }
    public string TimeWork { get; set; }

    public ClassPunkt() { }

    public ClassPunkt(int _CodPunkt, string _Name, string _Address,
string _Phone, string _TimeWork)
    {
        CodPunkt = _CodPunkt;
        Name = _Name;
        Address = _Address;
        Phone = _Phone;
        TimeWork = _TimeWork;
    }
}

```

Листинг Е.5 – Класс ClassDolzhnost

```

public class ClassDolzhnost
{
    public int CodDolzhnost { get; set; }
    public string Name { get; set; }

    public ClassDolzhnost() { }

    public ClassDolzhnost(int _CodDolzhnost, string _Name)
    {
        CodDolzhnost = _CodDolzhnost;
        Name = _Name;
    }
}

```

Листинг Е.6 – Класс ClassRecept

```

public class ClassRecept
{
    public int CodRecept { get; set; }
}

```

```

    public string Inform { get; set; }
    public string NameFile { get; set; }
    public int CodZakaz { get; set; }

    public ClassRecept() { }

    public ClassRecept(int _CodRecept, string _Inform, string
_NameFile, int _CodZakaz)
    {
        CodRecept = _CodRecept;
        Inform = _Inform;
        NameFile = _NameFile;
        CodZakaz = _CodZakaz;
    }
}

```

Листинг Е.7 – Класс ClassZakaz

```

public class ClassZakaz
{
    public int CodZakaz { get; set; }
    public DateTime DataStart { get; set; }
    public DateTime DataEnd { get; set; }
    public string Inform { get; set; }
    public int Summa { get; set; }
    public int CodClient { get; set; }
    public int CodPunkt { get; set; }
    public int CodStatus { get; set; }
    public int CodUsers { get; set; }

    public ClassZakaz() { }

    public ClassZakaz(int _CodZakaz, DateTime _DataStart, DateTime
_DataEnd, string _Inform, int _Summa, int _CodClient, int _CodPunkt, int
_CodStatus, int _CodUsers)
    {
        CodZakaz = _CodZakaz;
        DataStart = _DataStart;
        DataEnd = _DataEnd;
        Inform = _Inform;
        Summa = _Summa;
        CodClient = _CodClient;
        CodPunkt = _CodPunkt;
        CodStatus = _CodStatus;
        CodUsers = _CodUsers;
    }
}

```

Листинг Е.8 – Класс ClassLekarstvo

```

public class ClassLekarstvo
{
    public int CodLekarstvo { get; set; }
    public string Name { get; set; }
    public string Inform { get; set; }
    public int Kol { get; set; }
    public int Cena { get; set; }
}

```

```

public ClassLekarstvo() { }

public ClassLekarstvo(int _CodLekarstvo, string _Name, string
_Inform, int _Kol, int _Cena)
{
    CodLekarstvo = _CodLekarstvo;
    Name = _Name;
    Inform = _Inform;
    Kol = _Kol;
    Cena = _Cena;
}
}

```

Листинг Е.9 – Класс ClassStatus

```

public class ClassStatus
{
    public int CodStatus { get; set; }
    public string Name { get; set; }

    public ClassStatus() { }

    public ClassStatus(int _CodStatus, string _Name)
    {
        CodStatus = _CodStatus;
        Name = _Name;
    }
}

```

Листинг Е.10 – Класс ClassSostavZakaz

```

public class ClassSostavZakaz
{
    public int CodSostavZakaz { get; set; }
    public int Kol { get; set; }
    public int CodZakaz { get; set; }
    public int CodLekarstvo { get; set; }
    public int CodPreparat { get; set; }

    public ClassSostavZakaz() { }

    public ClassSostavZakaz(int _CodSostavZakaz, int _Kol, int
_CodZakaz, int _CodLekarstvo, int _CodPreparat)
    {
        CodSostavZakaz = _CodSostavZakaz;
        Kol = _Kol;
        CodZakaz = _CodZakaz;
        CodLekarstvo = _CodLekarstvo;
        CodPreparat = _CodPreparat;
    }
}

```

Листинг Е.11 – Класс ClassSupplier

```

public class ClassSupplier
{
    public int CodSupplier { get; set; }
    public string Name { get; set; }
    public string Phone { get; set; }
}

```

```

    public string Address { get; set; }
    public string FIO_Director { get; set; }
    public string Inform { get; set; }

    public ClassSupplier() { }

    public ClassSupplier(int _CodSupplier, string _Name, string
    _Phone, string _Address, string _FIO_Director, string _Inform)
    {
        CodSupplier = _CodSupplier;
        Name = _Name;
        Phone = _Phone;
        Address = _Address;
        FIO_Director = _FIO_Director;
        Inform = _Inform;
    }
}

```

Листинг Е.12 – Класс ClassHistory

```

public class ClassHistory
{
    public int CodHistory { get; set; }
    public DateTime Data { get; set; }
    public string Inform { get; set; }
    public int CodZakaz { get; set; }

    public ClassHistory() { }

    public ClassHistory(int _CodHistory, DateTime _Data, string
    _Inform, int _CodZakaz)
    {
        CodHistory = _CodHistory;
        Data = _Data;
        Inform = _Inform;
        CodZakaz = _CodZakaz;
    }
}

```

Листинг Е.13 – Класс ClassPreparat

```

public class ClassPreparat
{
    public int CodPreparat { get; set; }
    public string Name { get; set; }
    public string Inform { get; set; }
    public int Cena { get; set; }
    public int Kol { get; set; }
    public int CodViewPreparat { get; set; }
    public int CodSupplier { get; set; }

    public ClassPreparat() { }

    public ClassPreparat(int _CodPreparat, string _Name, string
    _Inform, int _Cena, int _Kol, int _CodViewPreparat, int _CodSupplier)
    {
        CodPreparat = _CodPreparat;
        Name = _Name;
    }
}

```

```

        Inform = _Inform;
        Cena = _Cena;
        Kol = _Kol;
        CodViewPreparat = _CodViewPreparat;
        CodSupplier = _CodSupplier;
    }
}

```

Листинг Е.14 – Класс Query

```

class Query
{
    public static string ConnStr = "server=localhost; user=root;
password=root_PNZ_2023; database=lekarstvo";

    MySqlConnection connection; //создать подключение между нашим
    проектом и БД
    MySqlCommand command; //нужен для выполнения операция
    вставки, редактирования и удаления данных БД
    MySqlDataAdapter datadapter; //нужен для получения набора данных из
    БД
    DataTable bufferTable;

    //Конструктор класса
    public Query(string Conn) //Передаем строку подключения
    {
        //Инициализация
        connection = new MySqlConnection(Conn);
        bufferTable = new DataTable();
    }

    //Метод для получения данных из таблицы
    public DataTable GetDataFromTable(string zapros)
    {
        try
        {
            connection.Open(); //Открыть соединение
            //Инициализируем - передаем запрос и объект класса oledbconnection -
            connection
            datadapter = new MySqlDataAdapter(zapros, connection);
            bufferTable.Clear(); //Очищаем буфер запроса
            datadapter.Fill(bufferTable); //вызываем метод Fill, чтобы
            после запроса заполнил буферную таблицу данными
            connection.Close(); //Закрываем соединение
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        return bufferTable; //Возвращаем результат
    }

    //Метод для удаления данных во всех таблицах
    public void DeleteFromTable(string NameTabl, string PoleTable, int
ID)
    {
        string zapros = "";
    }
}

```

```

try
{
    connection.Open();
    zapros = "delete from " + NameTabl + " where " + PoleTable + "
= " + Convert.ToString(ID);
    command = new MySqlCommand(zapros, connection);
    command.ExecuteNonQuery();
    connection.Close();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}

//Метод для обновления данных в таблице
public void Update_JSON(string NameTabl, string strData)
{
    string zapros = "";

    switch (NameTabl)
    {
        case "Table_Client":
            ClassClient PerClient =
            JsonConvert.DeserializeObject<ClassClient>(strData);
            zapros = "update " + NameTabl + " set " +
                "FamCl = '" + PerClient.FamCl +
                "', NameCl = '" + PerClient.NameCl +
                "', OtchCl = '" + PerClient.OtchCl +
                "', Phone = '" + PerClient.Phone +
                "', Document = '" + PerClient.Document +
                "' where CodClient = " + PerClient.CodClient;
            break;
        case "Table_Punkt":
            ClassPunkt PerPunkt =
            JsonConvert.DeserializeObject<ClassPunkt>(strData);
            zapros = "update " + NameTabl + " set " +
                "Name = '" + PerPunkt.Name +
                "', Address = '" + PerPunkt.Address +
                "', Phone = '" + PerPunkt.Phone +
                "', TimeWork = '" + PerPunkt.TimeWork +
                "' where CodPunkt = " + PerPunkt.CodPunkt;
            break;
        case "Table_Users":
            ClassUsers PerUsers =
            JsonConvert.DeserializeObject<ClassUsers>(strData);
            zapros = "update " + NameTabl + " set " +
                "FamU = '" + PerUsers.FamU +
                "', NameU = '" + PerUsers.NameU +
                "', OtchU = '" + PerUsers.OtchU +
                "', Login = '" + PerUsers.Login +
                "', Password = '" + PerUsers.Password +
                "', Prava = '" + PerUsers.Prava +
                "', CodDolzhnost = " + PerUsers.CodDolzhnost +
                " where CodUsers = " + PerUsers.CodUsers;
            break;
        case "Table_Dolzhnost":
    
```

```

        ClassDolzhnost PerDolzhnost =
JsonConvert.DeserializeObject<ClassDolzhnost>(strData);
        zapros = "update " + NameTabl + " set " +
            "Name = '" + PerDolzhnost.Name +
            "' where CodDolzhnost = " + PerDolzhnost.CodDolzhnost;
        break;
    case "Table_Recept":
        ClassRecept PerRecept =
JsonConvert.DeserializeObject<ClassRecept>(strData);
        zapros = "update " + NameTabl + " set " +
            "Inform = '" + PerRecept.Inform +
            "', NameFile = '" + PerRecept.NameFile +
            "', CodZakaz = " + PerRecept.CodZakaz +
            " where CodRecept = " + PerRecept.CodRecept;
        break;
    case "Table_Zakaz":
        ClassZakaz PerZakaz =
JsonConvert.DeserializeObject<ClassZakaz>(strData);
        if (ClassStr.RegimZakaz == 1)
            zapros = "update " + NameTabl + " set " +
                "DataStart= '" + PerZakaz.DataStart.ToString("yyyy-MM-dd") +
                "', DataEnd= '" + PerZakaz.DataEnd.ToString("yyyy-MM-dd") +
                "', Inform = '" + PerZakaz.Inform +
                "', Summa = " + PerZakaz.Summa +
                ", CodClient= " + PerZakaz.CodClient +
                ", CodPunkt = " + PerZakaz.CodPunkt +
                ", CodStatus = " + PerZakaz.CodStatus +
                ", CodUsers = " + PerZakaz.CodUsers +
                " where CodZakaz = " + PerZakaz.CodZakaz;
        else
            zapros = "update " + NameTabl + " set " +
                "DataStart= '" + PerZakaz.DataStart.ToString("yyyy-MM-dd") +
                "', DataEnd= '" + PerZakaz.DataEnd.ToString("yyyy-MM-dd") +
                "', Inform = '" + PerZakaz.Inform +
                "', Summa = " + PerZakaz.Summa +
                ", CodClient= null" +
                ", CodPunkt = null" +
                ", CodStatus = " + PerZakaz.CodStatus +
                ", CodUsers = " + PerZakaz.CodUsers +
                " where CodZakaz = " + PerZakaz.CodZakaz;
        break;
    case "Table_SostavZakaz": ClassSostavZakaz PerSostavZakaz =
JsonConvert.DeserializeObject<ClassSostavZakaz>(strData);
        if (ClassStr.RegimZakaz == 1)
            zapros = "update " + NameTabl + " set " +
                "Kol = " + PerSostavZakaz.Kol +
                ", CodZakaz = " + PerSostavZakaz.CodZakaz +
                ", CodLekarstvo = " + PerSostavZakaz.CodLekarstvo +
                ", CodPreparat = null" +
                " where CodSostavZakaz = " + PerSostavZakaz.CodSostavZakaz;
        else
            zapros = "update " + NameTabl + " set " + "Kol = " +
                PerSostavZakaz.Kol +
                ", CodZakaz = " + PerSostavZakaz.CodZakaz +
                ", CodLekarstvo = null" + ", CodPreparat = " +
                PerSostavZakaz.CodPreparat + " where CodSostavZakaz = " +
                PerSostavZakaz.CodSostavZakaz;
        break;

```

```

        case "Table_Lekarstvo": ClassLekarstvo PerLekarstvo =
JsonConvert.DeserializeObject<ClassLekarstvo>(strData);
        zapros = "update " + NameTabl + " set " + "Name = '" +
PerLekarstvo.Name + "', Inform = '" + PerLekarstvo.Inform + "', Kol
= " + PerLekarstvo.Kol + ", Cena = " + PerLekarstvo.Cena + " where
CodLekarstvo = " + PerLekarstvo.CodLekarstvo;
        break;
        case "Table_Status": ClassStatus PerStatus =
JsonConvert.DeserializeObject<ClassStatus>(strData);
        zapros = "update " + NameTabl + " set " + "Name = '" +
PerStatus.Name + "'" where CodStatus = " + PerStatus.CodStatus;
        break;
        case "Table_Supplier": ClassSupplier PerSupplier =
JsonConvert.DeserializeObject<ClassSupplier>(strData);
        zapros = "update " + NameTabl + " set " + "Name = '" +
PerSupplier.Name + "', Phone = '" + PerSupplier.Phone + "', Address
= '" + PerSupplier.Address + "', FIO_Director = '" +
PerSupplier.FIO_Director + "', Inform = '" + PerSupplier.Inform +
"'" where CodSupplier = " + PerSupplier.CodSupplier;
        break;
        case "Table_History": ClassHistory PerHistory =
JsonConvert.DeserializeObject<ClassHistory>(strData);
        zapros = "update " + NameTabl + " set " + "Data= '" +
PerHistory.Data.ToString("yyyy-MM-dd") + "', Inform = '" +
PerHistory.Inform + "', CodZakaz= " + PerHistory.CodZakaz + " where
CodHistory = " + PerHistory.CodHistory;
        break;
        case "Table_Oplata": ClassOplata PerOplata =
JsonConvert.DeserializeObject<ClassOplata>(strData);
        zapros = "update " + NameTabl + " set " + "Data= '" +
PerOplata.Data.ToString("yyyy-MM-dd") + "', Summa= " +
PerOplata.Summa + ", CodZakaz= " + PerOplata.CodZakaz + " where
CodOplata = " + PerOplata.CodOplata;
        break;
        case "Table_Preparat": ClassPreparat PerPreparat =
JsonConvert.DeserializeObject<ClassPreparat>(strData);
        zapros = "update " + NameTabl + " set " + "Name= '" +
PerPreparat.Name + "', Inform = '" + PerPreparat.Inform + "', Cena=
" + PerPreparat.Cena + ", Kol= " + PerPreparat.Kol + ",
CodViewPreparat= " + PerPreparat.CodViewPreparat + ", CodSupplier=
" + PerPreparat.CodSupplier +" where CodPreparat = " +
PerPreparat.CodPreparat;
        break;
        case "Table_ViewPreparat":
ClassViewPreparat PerViewPreparat =
JsonConvert.DeserializeObject<ClassViewPreparat>(strData);
        zapros = "update " + NameTabl + " set " + "Name = '" +
PerViewPreparat.Name + "'" where CodViewPreparat = " +
PerViewPreparat.CodViewPreparat;
        break;
    }

    connection.Open(); //Открыть соединение

    //Инициализируем - передаем запрос и объект класса oledbconnection
- connection
        command = new MySqlCommand(zapros, connection);

```



```

        command.ExecuteNonQuery(); //Выполняем запрос. В качестве
результата возвращает кол-во обработанных строк
        connection.Close(); //Закрываем соединение
    }

    //Метод для обновления данных в таблице
    public int Add_JSON(string NameTabl, string strData)
    {
        string zapros = "";

        switch (NameTabl)
        {
            case "Table_Client": ClassClient PerClient =
JsonConvert.DeserializeObject<ClassClient>(strData);
                zapros = "insert into " + NameTabl +
"(FamCl,NameCl,OtchCl,Phone,Document) values ('" + PerClient.FamCl
+ "','" + PerClient.NameCl + "','" + PerClient.OtchCl + "','" +
PerClient.Phone + "','" + PerClient.Document + "')";
                break;
            case "Table_Punkt": ClassPunkt PerPunkt =
JsonConvert.DeserializeObject<ClassPunkt>(strData);
                zapros = "insert into " + NameTabl +
"(Name,Address,Phone,TimeWork) values ('" + PerPunkt.Name + "','" +
PerPunkt.Address + "','" + PerPunkt.Phone + "','" +
PerPunkt.TimeWork + "')";
                break;
            case "Table_Users": ClassUsers PerUsers =
JsonConvert.DeserializeObject<ClassUsers>(strData);
                zapros = "insert into " + NameTabl +
"(FamU,NameU,OtchU,Login,Password,Prava,CodDolzhnost) values ('" +
PerUsers.FamU + "','" + PerUsers.NameU + "','" + PerUsers.OtchU
+ "','" + PerUsers.Login + "','" + PerUsers.Password + "','" +
PerUsers.Prava + "','" + PerUsers.CodDolzhnost + "')";
                break;
            case "Table_Dolzhnost": ClassDolzhnost PerDolzhnost =
JsonConvert.DeserializeObject<ClassDolzhnost>(strData);
                zapros = "insert into " + NameTabl + "(Name) values ('" +
PerDolzhnost.Name + "')";
                break;
            case "Table_Recept": ClassRecept PerRecept =
JsonConvert.DeserializeObject<ClassRecept>(strData);
                zapros = "insert into " + NameTabl +
"(Inform,NameFile,CodZakaz) values ('" + PerRecept.Inform + "','" +
PerRecept.NameFile + "','" + PerRecept.CodZakaz + "')";
                break;
            case "Table_Zakaz": ClassZakaz PerZakaz =
JsonConvert.DeserializeObject<ClassZakaz>(strData);
                if (ClassStr.RegimZakaz==1)
                    zapros = "insert into " + NameTabl +
"(DataStart,DataEnd,Inform,Summa,CodClient,CodPunkt,CodStatus,CodUs
ers) values ('" + PerZakaz.DataStart.ToString("yyyy-MM-dd") + "','" +
PerZakaz.DataEnd.ToString("yyyy-MM-dd") + "','" + PerZakaz.Inform
+ "','" + PerZakaz.Summa + "','" + PerZakaz.CodClient + "','" +
PerZakaz.CodPunkt + "','" + PerZakaz.CodStatus + "','" +
PerZakaz.CodUsers + "')";
                else
                    zapros = "insert into " + NameTabl +
"(DataStart,DataEnd,Inform,Summa,CodClient,CodPunkt,CodStatus,CodUs

```

```

    ers) values ('" + PerZakaz.DataStart.ToString("yyyy-MM-dd") + "',''"
    + PerZakaz.DataEnd.ToString("yyyy-MM-dd") + "',''" +
    PerZakaz.Inform + "',''" + PerZakaz.Summa + "','null,null,'" +
    PerZakaz.CodStatus + "',''" + PerZakaz.CodUsers + "')";
    break;
    case "Table_SostavZakaz": ClassSostavZakaz PerSostavZakaz =
    JsonConvert.DeserializeObject<ClassSostavZakaz>(strData);
    if (ClassStr.RegimZakaz == 1)
        zaptros = "insert into " + NameTabl +
        "(Kol,CodZakaz,CodLekarstvo,CodPreparat) values (" +
        PerSostavZakaz.Kol + "',''" + PerSostavZakaz.CodZakaz + "',''" +
        PerSostavZakaz.CodLekarstvo + "','null)";
    else
        zaptros = "insert into " + NameTabl +
        "(Kol,CodZakaz,CodLekarstvo,CodPreparat) values (" +
        PerSostavZakaz.Kol + "',''" + PerSostavZakaz.CodZakaz + "','null,'" +
        PerSostavZakaz.CodPreparat + "')";
    break;
    case "Table_Lekarstvo": ClassLekarstvo PerLekarstvo =
    JsonConvert.DeserializeObject<ClassLekarstvo>(strData);
    zaptros = "insert into " + NameTabl + "(Name,Inform,Kol,Cena)
    values ('" + PerLekarstvo.Name + "',''" + PerLekarstvo.Inform + "',''"
    + PerLekarstvo.Kol + "',''" +
    PerLekarstvo.Cena + "')";
    break;
    case "Table_Status": ClassStatus PerStatus =
    JsonConvert.DeserializeObject<ClassStatus>(strData);
    zaptros = "insert into " + NameTabl + "(Name) values ('" +
    PerStatus.Name + "')";
    break;
    case "Table_Supplier": ClassSupplier PerSupplier =
    JsonConvert.DeserializeObject<ClassSupplier>(strData);
    zaptros = "insert into " + NameTabl +
    "(Name,Phone,Address,FIO_Director,Inform) values ('" +
    PerSupplier.Name + "',''" + PerSupplier.Phone + "',''" +
    PerSupplier.Address + "',''" + PerSupplier.FIO_Director + "',''" +
    PerSupplier.Inform + "')";
    break;
    case "Table_History": ClassHistory PerHistory =
    JsonConvert.DeserializeObject<ClassHistory>(strData);
    zaptros = "insert into " + NameTabl + "(Data,Inform,CodZakaz)
    values ('" + PerHistory.Data.ToString("yyyy-MM-dd") + "',''"
    + PerHistory.Inform + "',''" + PerHistory.CodZakaz + "')";
    break;
    case "Table_Oplata":
    ClassOplata PerOplata =
    JsonConvert.DeserializeObject<ClassOplata>(strData);
    zaptros = "insert into " + NameTabl + "(Data,Summa,CodZakaz) values
    ('" +
    PerOplata.Data.ToString("yyyy-MM-dd") + "',''" +
    PerOplata.Summa + "',''" + PerOplata.CodZakaz + "')";
    break;
    case "Table_Preparat":
    ClassPreparat PerPreparat =
    JsonConvert.DeserializeObject<ClassPreparat>(strData);
    zaptros = "insert into " + NameTabl +
    "(Name,Inform,Cena,Kol,CodViewPreparat,CodSupplier) values ('"
    + PerPreparat.Name + "',''" + PerPreparat.Inform + "',''" +

```

```

PerPreparat.Cena + "," + PerPreparat.Kol + "," +
PerPreparat.CodViewPreparat + "," + PerPreparat.CodSupplier + ")";
        break;
        case "Table_ViewPreparat":
            ClassViewPreparat PerViewPreparat
=JsonConvert.DeserializeObject<ClassViewPreparat>(strData);
            zapros = "insert into " + NameTabl + "(Name) values ('" +
            PerViewPreparat.Name + "')";
            break;
    }

    connection.Open(); //Открыть соединение
    command = new MySqlCommand(zapros, connection);
    command.ExecuteNonQuery(); //Выполняем запрос. В качестве
результата возвращает кол-во обработанных строк

    //Получаем ID добавленной записи
    command = new MySqlCommand($"Select @@Identity", connection);
    int id = Convert.ToInt32(command.ExecuteScalar());
    connection.Close(); //Закрываем соединение
    return id; //Возвращаем ID добавленной записи
}
}

```

Листинг Е.15 – Класс FormClient

```

public partial class FormClient : Form
{
    Query Podkl; //Определим наш объект класса
    public int TekID_Client = 0;
    public string Tekzapros;

    public FormClient()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
        Tekzapros = "select
CodClient,FamCl,NameCl,OtchCl,Phone,Document from Table_Client ";
        Tekzapros += " order by FamCl,NameCl,OtchCl";
    }

    //Обработчик кнопки Выход
    private void Exit_Click(object sender, EventArgs e)
    {
        Close();
    }

    //Обработчик кнопки Добавить
    private void Knopka_Add_Click(object sender, EventArgs e)
    {
        FormClientInfo newForm = new FormClientInfo();
        newForm.RegimClient = 2; //2- режим создания
        newForm.ShowDialog(); //блокируются предыдущие формы, активна
только текущая (новая открытая)
        TekID_Client = -1;
    }
}

```

```

        this.TekID_Client = newForm.CheckClient;
        if (TekID_Client < 0) //Если был выход в окне с данными, т.е.,
данные не добавлены
            return;

        GridClient.DataSource = Podkl.GetDataFromTable(Tekzapros);

        //Установить курсор на добавленную запись
        string searchValue = Convert.ToString(TekID_Client);
        GridClient.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

        try
        {
            foreach (DataGridViewRow row in GridClient.Rows)
            {
                if
(row.Cells[0].Value.ToString().Equals(searchValue))
                {
                    row.Selected = true;
                    GridClient.CurrentCell =
GridClient.Rows[row.Index].Cells[1];
                    break;
                }
            }
        }
        catch (Exception exc)
        {
            MessageBox.Show(exc.Message);
        }
        //-----Конец установки курсора
    }

    //Обработчик кнопки Изменить
    private void Knopka_Change_Click(object sender, EventArgs e)
    {
        if (GridClient.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        if
(GridClient.Rows[GridClient.CurrentRow.Index].Cells["CodClient"].Value.To
String() == "")
            return;

        //Запоминаем позицию в DBGrid, чтобы после редактирования в
другом окне установить курсор на эту строку
        int iStr = GridClient.CurrentCell.RowIndex;

        FormClientInfo newForm = new FormClientInfo();
        newForm.RegimClient = 1; //1- режим редактирования
        newForm.text_ID =
GridClient.Rows[GridClient.CurrentRow.Index].Cells["CodClient"].Value.ToS
tring();
        newForm.ShowDialog(); //блокируются предыдущие формы, активна
только текущая (новая открытая)

```

```

        GridClient.DataSource = Podkl.GetDataFromTable(Tekzapros);

        //Установить курсор на измененную запись
        GridClient.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridClient.CurrentCell = GridClient.Rows[iStr].Cells[1];
        //ставим выделение на измененную строку
    }

    //Обработчик кнопки Удалить
    private void Knopka_Delete_Click(object sender, EventArgs e)
    {
        if (GridClient.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        //Поиск связей удаляемой записи в таблице с заказами
        var cmd1 = new MySqlCommand("SELECT COUNT(*) FROM Table_Zakaz
where CodClient = @PID_Cod");
        cmd1.Connection = new MySqlConnection(Query.ConnStr);
        cmd1.Connection.Open();
        cmd1.Parameters.AddWithValue("PID_Cod",
int.Parse(GridClient.Rows[GridClient.CurrentRow.Index].Cells["CodClient"]
.Value.ToString()));
        if (Convert.ToInt32(cmd1.ExecuteScalar()) > 0)
        {
            MessageBox.Show("Данную запись по клиенту удалить
нельзя, потому что обнаружены заказы, которые связаны с клиентом!",
"ВНИМАНИЕ");
            return;
        }

        //Передаем индекс текущей записи в GridClient
        Podkl.DeleteFromTable("Table_Client", "CodClient",
int.Parse(GridClient.Rows[GridClient.CurrentRow.Index].Cells["CodClient"]
.Value.ToString()));
        GridClient.DataSource = Podkl.GetDataFromTable(Tekzapros);
        cmd1.Connection.Close();
        MessageBox.Show("Удаление выполнено успешно");
    }

    //Клавиша ESC
    private void FormClient_KeyPress(object sender, KeyPressEventArgs
e)
    {
        if (e.KeyChar == (char)27)
        {
            Exit_Click(sender, e);
        }
    }

    //Обработчик загрузки формы
    private void FormClient_Load(object sender, EventArgs e)
    {
        // задать цвет в заголовках столбцов

```

```

        GridClient.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
        // задать шрифт в заголовках столбцов
        Font F = new Font("Times New Roman", 14);
        GridClient.ColumnHeadersDefaultCellStyle.Font = F;

        Font F2 = new Font("Times New Roman", 12);
        GridClient.DefaultCellStyle.Font = F2;

        if (ClassStr.TekPrava == "Провизор")
        {
            Knopka_Add.Visible = false;
            Knopka_Change.Visible = false;
            Knopka_Delete.Visible = false;
        }
    }

    //Обработчик отображения формы
    private void FormClient_Shown(object sender, EventArgs e)
    {
        GridClient.DataSource = Podkl.GetDataFromTable(Tekzapros);
        GridClient.Columns[0].Visible = false;
        GridClient.Columns[1].Width = 140;
        GridClient.Columns[1].HeaderText = "Фамилия";
        GridClient.Columns[2].Width = 130;
        GridClient.Columns[2].HeaderText = "Имя";
        GridClient.Columns[3].Width = 130;
        GridClient.Columns[3].HeaderText = "Отчество";
        GridClient.Columns[4].Width = 150;
        GridClient.Columns[4].HeaderText = "Телефон";
        GridClient.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridClient.Columns[5].Width = 150;
        GridClient.Columns[5].HeaderText = "Паспорт";
        GridClient.Columns[5].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridClient.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridClient.DefaultCellStyle.SelectionBackColor = Color.White;
        GridClient.DefaultCellStyle.SelectionForeColor = Color.Blue;
        GridClient.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridClient.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridClient.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridClient.ColumnHeadersDefaultCellStyle.BackColor;
        GridClient.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridClient.ColumnHeadersDefaultCellStyle.ForeColor;

        GridClient.EnableHeadersVisualStyles = false;

        if (GridClient.Rows.Count > 0)
        {
            GridClient.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

```

```

        GridClient.CurrentCell =
GridClient.Rows[0].Cells[1]; //ставим выделение на самую первую строку
        GridClient.Focus();
    }
}

//Обработчик в поле поиска по фамилии
private void TextBoxFam_TextChanged(object sender, EventArgs e)
{
    string zapros = "select
CodClient,FamCl,NameCl,OtchCl,Phone,Document from Table_Client ";

    if (textBoxFam.Text != "")
    {
        zapros += " where FamCl like '%" + textBoxFam.Text
+ "%' order by FamCl,NameCl,OtchCl";
    }
    else
    {
        zapros += " order by FamCl,NameCl,OtchCl";
    }

    GridClient.DataSource = Podkl.GetDataFromTable(zapros);
    if (GridClient.Rows.Count > 0)
    {
        GridClient.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridClient.CurrentCell =
GridClient.Rows[0].Cells[1]; //ставим выделение на самую первую строку
    }
    textBoxFam.Focus();
}

//Обработчик двойного нажатия мышкой в списке сотрудников
private void GridClient_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
{
    Knopka_Change_Click(sender, e);
}

/// Преобразование текстового представления столбца таблицы Excel
в индекс столбца
/// </summary>
/// <param name="abc">текстовое представление столбца</param>
/// <returns>индекс столбца</returns>
public static int AbcToIndex(string abc)
{
    const string s = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    if (abc.Length == 1 && s.IndexOf(abc,
StringComparison.Ordinal) >= 0)
        return s.IndexOf(abc, StringComparison.Ordinal) +
1;
    if (abc.Length == 2 &&
        s.IndexOf(abc.Substring(0, 1),
StringComparison.Ordinal) >= 0 &&
        s.IndexOf(abc.Substring(1, 1),
StringComparison.Ordinal) >= 0)

```

```

        return AbcToIndex(abc.Substring(0, 1)) * 26 +
        AbcToIndex(abc.Substring(1, 1));
        return 1;
    }
}

```

Листинг E.16 – Класс FormClientInfo

```

public partial class FormClientInfo : Form
{
    Query Podkl; //Определим наш объект класса
    public string text_ID;
    public int RegimClient, CheckClient;

    public FormClientInfo()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
    }

    //Кнопка Выход
    private void But_Exit_Click(object sender, EventArgs e)
    {
        CheckClient = -1;
        Close();
    }

    //Клавиши ESC и ENTER
    private void DBEdit_Fam_KeyPress(object sender, KeyPressEventArgs
e)
    {
        if (e.KeyChar == (char)27)
        {
            But_Exit_Click(sender, e);
        }
        else if (e.KeyChar == (char)13)
        {
            But_Save_Click(sender, e);
        }
    }

    //Кнопка Сохранить
    private void But_Save_Click(object sender, EventArgs e)
    {
        //Выполняем проверки заполненных полей данных
        if (DBEdit_Fam.Text == "")
        {
            MessageBox.Show("Введите фамилию клиента");
            DBEdit_Fam.Focus();
            return;
        }

        if (DBEdit_Document.Text == "")
        {

```



```

        MessageBox.Show("Введите паспортные данные клиента
(серия, номер = 10 цифр)");
        DBEdit_Document.Focus();
        return;
    }

    string StrJson = "";

    switch (RegimClient)
    {
        case 1: //Режим редактирования
            ClassClient Per_1 = new
ClassClient(int.Parse(text_ID.ToString()),

            DBEdit_Fam.Text,

            DBEdit_Name.Text,

            DBEdit_Otch.Text,

            DBEdit_Phone.Text,

            DBEdit_Document.Text

            );

            StrJson = JsonConvert.SerializeObject(Per_1);
            Podkl.Update_JSON("Table_Client", StrJson);
            break;
        case 2: //Режим создания
            ClassClient Per_2 = new ClassClient(0,

            DBEdit_Fam.Text,

            DBEdit_Name.Text,

            DBEdit_Otch.Text,

            DBEdit_Phone.Text,

            DBEdit_Document.Text

            );

            StrJson = JsonConvert.SerializeObject(Per_2);
            CheckClient = Podkl.Add_JSON("Table_Client",
StrJson);

            break;
    }

    Close();
}

//Клавиша ESC
private void FormClientInfo_KeyPress(object sender,
KeyPressEventArgs e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
}

```

```

    }
}

//Обработчик загрузки формы
private void FormClientInfo_Load(object sender, EventArgs e)
{
    DataTable bufferClient = new DataTable();

    switch (RegimClient)
    {
        case 1: //Режим редактирования
            string zapros = "select * from Table_Client
";
            zapros += "where CodClient = ";
            zapros += text_ID;
            bufferClient =
Podkl.GetDataFromTable(zapros);

            //Заполняем поля для ввода информации
            DBEdit_Fam.Text =
bufferClient.Rows[0]["FamCl"].ToString();
            DBEdit_Name.Text =
bufferClient.Rows[0]["NameCl"].ToString();
            DBEdit_Otch.Text =
bufferClient.Rows[0]["OtchCl"].ToString();
            DBEdit_Phone.Text =
bufferClient.Rows[0]["Phone"].ToString();
            DBEdit_Document.Text =
bufferClient.Rows[0]["Document"].ToString();
            break;
        case 2: //Режим создания
            //Заполняем поля для ввода информации
            DBEdit_Fam.Text = "";
            DBEdit_Name.Text = "";
            DBEdit_Otch.Text = "";
            DBEdit_Phone.Text = "";
            DBEdit_Document.Text = "";
            break;
    }
}

private void DBEdit_Document_KeyPress(object sender,
KeyPressEventArgs e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
    else if (e.KeyChar == (char)13)
    {
        But_Save_Click(sender, e);
    }

    char ch = e.KeyChar;
    if (!char.IsDigit(ch) && ch != 8)
    {
        e.Handled = true;
    }
}

```

```

    }

    //Обработчик отображения формы
    private void FormClientInfo_Shown(object sender, EventArgs e)
    {
        DBEdit_Fam.Focus();
    }
}

```

Листинг Е.17 – Класс FormLekarstvo

```

public partial class FormLekarstvo : Form
{
    Query Podkl; //Определим наш объект класса
    public int TekID_Lekarstvo = 0;
    public string Tekzapros;

    public FormLekarstvo()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
        Tekzapros = "select CodLekarstvo,Name,Kol,Cena,Inform from
        Table_Lekarstvo order by Name";
    }

    //Обработчик кнопки Выход
    private void Exit_Click(object sender, EventArgs e)
    {
        Close();
    }

    //Обработчик кнопки Добавить
    private void Knopka_Add_Click(object sender, EventArgs e)
    {
        FormLekarstvoInfo newForm = new FormLekarstvoInfo();
        newForm.RegimLekarstvo = 2; //2- режим создания
        newForm.ShowDialog(); //блокируются предыдущие формы, активна
        только текущая (новая открытая)
        TekID_Lekarstvo = -1;
        this.TekID_Lekarstvo = newForm.CheckLekarstvo;
        if (TekID_Lekarstvo < 0) //Если был выход в окне с данными,
        т.е., данные не добавлены
            return;

        GridLekarstvo.DataSource = Podkl.GetDataFromTable(Tekzapros);

        //Установить курсор на добавленную запись
        string searchValue = Convert.ToString(TekID_Lekarstvo);
        GridLekarstvo.SelectionMode =
        DataGridViewSelectionMode.FullRowSelect;

        try
        {
            foreach (DataGridViewRow row in GridLekarstvo.Rows)
            {

```

```

        if
        (row.Cells[0].Value.ToString().Equals(searchValue))
        {
            row.Selected = true;
            GridLekarstvo.CurrentCell =
GridLekarstvo.Rows[row.Index].Cells[1];
            break;
        }
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
    //-----Конец установки курсора
}

//Обработчик кнопки Изменить
private void Knopka_Change_Click(object sender, EventArgs e)
{
    if (GridLekarstvo.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
        return;
    }

    if
    (GridLekarstvo.Rows[GridLekarstvo.CurrentRow.Index].Cells["CodLekarstvo"]
    .Value.ToString() == "")
        return;

    //Запоминаем позицию в DBGrid, чтобы после редактирования в
    другом окне установить курсор на эту строку
    int iStr = GridLekarstvo.CurrentCell.RowIndex;

    FormLekarstvoInfo newForm = new FormLekarstvoInfo();
    newForm.RegimLekarstvo = 1; //1- режим редактирования
    newForm.text_ID =
GridLekarstvo.Rows[GridLekarstvo.CurrentRow.Index].Cells["CodLekarstvo"].
Value.ToString();
    newForm.ShowDialog(); //блокируются предыдущие формы, активна
    только текущая (новая открытая)

    GridLekarstvo.DataSource = Podkl.GetDataFromTable(Tekzapros);

    //Установить курсор на измененную запись
    GridLekarstvo.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
    GridLekarstvo.CurrentCell = GridLekarstvo.Rows[iStr].Cells[1];
    //ставим выделение на измененную строку
}

//Обработчик кнопки Удалить
private void Knopka_Delete_Click(object sender, EventArgs e)
{
    if (GridLekarstvo.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
    }
}

```

```

        return;
    }

    //-----Поиск связей удаляемой записи в таблице с составом
заказа
    var cmd = new MySqlCommand("SELECT COUNT(*) FROM
Table_SostavZakaz where CodLekarstvo = @PidCod");
    cmd.Connection = new MySqlConnection(Query.ConnStr);
    cmd.Connection.Open();
    cmd.Parameters.AddWithValue("PidCod",
int.Parse(GridLekarstvo.Rows[GridLekarstvo.CurrentRow.Index].Cells["CodLe
karstvo"].Value.ToString()));
    int totalRecords = Convert.ToInt32(cmd.ExecuteScalar());
    if (totalRecords > 0)
    {
        MessageBox.Show("Данное лекарственное средство
удалить нельзя, потому что оно входит в состав заказа!", "ВНИМАНИЕ");
        return;
    }
    cmd.Connection.Close();
    //-----Конец поиска

    //Передаем индекс текущей записи в Grid
    Podkl.DeleteFromTable("Table_Lekarstvo", "CodLekarstvo",
int.Parse(GridLekarstvo.Rows[GridLekarstvo.CurrentRow.Index].Cells["CodLe
karstvo"].Value.ToString()));
    GridLekarstvo.DataSource = Podkl.GetDataFromTable(Tekzapros);
    MessageBox.Show("Удаление выполнено успешно");
}

//Клавиша ESC
private void FormLekarstvo_KeyPress(object sender,
KeyPressEventArgs e)
{
    if (e.KeyChar == (char)27)
    {
        Exit_Click(sender, e);
    }
}

//Обработчик загрузки формы
private void FormLekarstvo_Load(object sender, EventArgs e)
{
    // задать цвет в заголовках столбцов
    GridLekarstvo.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
    // задать шрифт в заголовках столбцов
    Font F = new Font("Times New Roman", 14);
    GridLekarstvo.ColumnHeadersDefaultCellStyle.Font = F;

    Font F2 = new Font("Times New Roman", 12);
    GridLekarstvo.DefaultCellStyle.Font = F2;
}

//Обработчик отображения формы
private void FormLekarstvo_Shown(object sender, EventArgs e)
{
    GridLekarstvo.DataSource = Podkl.GetDataFromTable(Tekzapros);
}

```

```

        GridLekarstvo.Columns[0].Visible = false;
        GridLekarstvo.Columns[1].Width = 230;
        GridLekarstvo.Columns[1].HeaderText = "Название";
        GridLekarstvo.Columns[2].Width = 75;
        GridLekarstvo.Columns[2].HeaderText = "Кол";
        GridLekarstvo.Columns[2].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridLekarstvo.Columns[3].Width = 85;
        GridLekarstvo.Columns[3].HeaderText = "Цена";
        GridLekarstvo.Columns[3].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridLekarstvo.Columns[4].Width = 330;
        GridLekarstvo.Columns[4].HeaderText = "Информация";
        GridLekarstvo.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;
        GridLekarstvo.DefaultCellStyle.SelectionBackColor =
Color.White;
        GridLekarstvo.DefaultCellStyle.SelectionForeColor =
Color.Blue;
        GridLekarstvo.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridLekarstvo.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовка столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridLekarstvo.ColumnHeadersDefaultCellStyle.SelectionBackColor
= GridLekarstvo.ColumnHeadersDefaultCellStyle.BackColor;
        GridLekarstvo.ColumnHeadersDefaultCellStyle.SelectionForeColor
= GridLekarstvo.ColumnHeadersDefaultCellStyle.ForeColor;

        GridLekarstvo.EnableHeadersVisualStyles = false;

        if (GridLekarstvo.Rows.Count > 0)
        {
            GridLekarstvo.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridLekarstvo.CurrentCell =
GridLekarstvo.Rows[0].Cells[1]; //ставим выделение на самую первую строку
            GridLekarstvo.Focus();
        }
    }

    //Обработчик в поле поиска по названию
    private void TextBoxName_TextChanged(object sender, EventArgs e)
    {
        string zapros = "select * from Table_Lekarstvo ";

        if (textBoxName.Text != "")
        {
            zapros += "where Name like '%" + textBoxName.Text +
            "%' order by Name";
        }
        else
        {
            zapros += "order by Name";
        }
    }

```

```

        GridLekarstvo.DataSource = Podkl.GetDataFromTable(zapros);
        if (GridLekarstvo.Rows.Count > 0)
        {
            GridLekarstvo.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridLekarstvo.CurrentCell =
GridLekarstvo.Rows[0].Cells[1]; //ставим выделение на самую первую строку
        }
        textBoxName.Focus();
    }

    //Обработчик двойного нажатия мышкой в списке данных
    private void GridLekarstvo_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        Knopka_Change_Click(sender, e);
    }
}

```

Листинг E.18 – Класс FormLekarstvoInfo

```

public partial class FormLekarstvoInfo : Form
{
    Query Podkl; //Определим наш объект класса
    public string text_ID;
    public int RegimLekarstvo, CheckLekarstvo;

    public FormLekarstvoInfo()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
    }

    //Кнопка Сохранить
    private void But_Save_Click(object sender, EventArgs e)
    {
        //Выполняем проверки заполненных полей данных
        if (DBEdit_Name.Text == "")
        {
            MessageBox.Show("Введите название");
            DBEdit_Name.Focus();
            return;
        }

        if (DBEdit_Kol.Text == "")
        {
            MessageBox.Show("Введите кол-во больше 0");
            DBEdit_Kol.Focus();
            return;
        }

        if (DBEdit_Cena.Text == "")
        {
            MessageBox.Show("Введите стоимость больше 0");
            DBEdit_Cena.Focus();
        }
    }
}

```

```

        return;
    }

    if (int.Parse(DBEdit_Kol.Text) < 1)
    {
        MessageBox.Show("Введите кол-во больше 0");
        DBEdit_Kol.Focus();
        return;
    }

    if (int.Parse(DBEdit_Cena.Text) < 1)
    {
        MessageBox.Show("Введите стоимость больше 0");
        DBEdit_Cena.Focus();
        return;
    }

    string StrJson = "";

    switch (RegimLekarstvo)
    {
        case 1: //Режим редактирования
            ClassLekarstvo Per_1 = new
ClassLekarstvo(int.Parse(text_ID.ToString()),

                DBEdit_Name.Text,

                DBEdit_Inform.Text,

                int.Parse(DBEdit_Kol.Text),

                int.Parse(DBEdit_Cena.Text)

            );

            StrJson = JsonConvert.SerializeObject(Per_1);
            Podkl.Update_JSON("Table_Lekarstvo",
StrJson);

            break;
        case 2: //Режим создания
            ClassLekarstvo Per_2 = new ClassLekarstvo(0,

                DBEdit_Name.Text,

                DBEdit_Inform.Text,

                int.Parse(DBEdit_Kol.Text),

                int.Parse(DBEdit_Cena.Text)

            );

            StrJson = JsonConvert.SerializeObject(Per_2);
            CheckLekarstvo =
Podkl.Add_JSON("Table_Lekarstvo", StrJson);
            break;
    }

    Close();
}

```



```

        //Клавиша ESC
        private void FormLekarstvoInfo_KeyPress(object sender,
        KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)27)
            {
                But_Exit_Click(sender, e);
            }
        }

        //Обработчик загрузки формы
        private void FormLekarstvoInfo_Load(object sender, EventArgs e)
        {
            DataTable bufferLekarstvo = new DataTable();

            switch (RegimLekarstvo)
            {
                case 1: //Режим редактирования
                    string zapros = "select * from
Table_Lekarstvo ";
                    zapros += "where CodLekarstvo = ";
                    zapros += text_ID;
                    bufferLekarstvo =
Podkl.GetDataFromTable(zapros);

                    //Заполняем поля для ввода информации
                    DBEdit_Name.Text =
bufferLekarstvo.Rows[0]["Name"].ToString();
                    DBEdit_Inform.Text =
bufferLekarstvo.Rows[0]["Inform"].ToString(); ;
                    DBEdit_Kol.Text =
bufferLekarstvo.Rows[0]["Kol"].ToString();
                    DBEdit_Cena.Text =
bufferLekarstvo.Rows[0]["Cena"].ToString();
                    break;
                case 2: //Режим создания
                    //Заполняем поля для ввода информации
                    DBEdit_Name.Text = "";
                    DBEdit_Inform.Text = "";
                    DBEdit_Kol.Text = "";
                    DBEdit_Cena.Text = "";
                    break;
            }
        }

        //Кнопка Выход
        private void But_Exit_Click(object sender, EventArgs e)
        {
            CheckLekarstvo = -1;
            Close();
        }

        private void DBEdit_Name_KeyPress(object sender, KeyPressEventArgs
e)
        {
            if (e.KeyChar == (char)27)
            {

```

```

        But_Exit_Click(sender, e);
    }
    else if (e.KeyChar == (char)13)
    {
        But_Save_Click(sender, e);
    }
}

private void DBEdit_Kol_KeyPress(object sender, KeyPressEventArgs
e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
        return;
    }
    else if (e.KeyChar == (char)13)
    {
        But_Save_Click(sender, e);
        return;
    }

    char ch = e.KeyChar;
    if (!char.IsDigit(ch) && ch != 8)
    {
        e.Handled = true;
    }
}

//Обработчик отображения формы
private void FormLekarstvoInfo_Shown(object sender, EventArgs e)
{
    DBEdit_Name.Focus();
}
}

```

Листинг Е.19 – Класс FormLogin

```

public partial class FormLogin : Form
{
    Query Podkl; //Определим наш объект класса

    public FormLogin()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
    }

    //Клавиша ESC
    private void FormLogin_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (e.KeyChar == (char)27)
        {
            Close();
        }
    }
}

```

```

    }

    //Кнопка - Обработчик выбора режима работы с программой
    private void comboBoxPrava_SelectedIndexChanged(object sender,
    EventArgs e)
    {
        if (comboBoxPrava.Text == "")
            return;

        switch (comboBoxPrava.Text)
        {
            case "Клиент":
                labelLogin_Fam.Text = "Фамилия";
                labelPasswordDocument.Text = "Документ";
                labelInfoDocument.Visible = true;
                break;
            case "Фармацевт":
                labelLogin_Fam.Text = "Логин";
                labelPasswordDocument.Text = "Пароль";
                labelInfoDocument.Visible = false;
                break;
            case "Провизор":
                labelLogin_Fam.Text = "Логин";
                labelPasswordDocument.Text = "Пароль";
                labelInfoDocument.Visible = false;
                break;
            case "Администратор":
                labelLogin_Fam.Text = "Логин";
                labelPasswordDocument.Text = "Пароль";
                labelInfoDocument.Visible = false;
                break;
        }
    }

    //Кнопка - Вход
    private void buttonVhod_Click(object sender, EventArgs e)
    {
        if (comboBoxPrava.Text == "")
        {
            MessageBox.Show("Укажите режим");
            comboBoxPrava.Focus();
            return;
        }

        if (Text_Login.Text == "")
        {
            MessageBox.Show(this, comboBoxPrava.Text ==
            "Клиент" ? "Укажите фамилию." : "Укажите логин.", "Режим авторизации",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
            Text_Login.Focus();
            return;
        }

        if (Text_Password.Text == "")
        {
            MessageBox.Show(this, comboBoxPrava.Text ==
            "Клиент" ? "Укажите серию и номер паспорта (10 цифр)." : "Укажите

```

```

    пароль.", "Режим авторизации", MessageBoxButtons.OK,
    MessageBoxIcon.Warning);
        Text_Password.Focus();
        return;
    }

    //Поиск сотрудника по логину и паролю
    DataTable buffer = new DataTable();

    string zapros = comboBoxPrava.Text == "Клиент" ? "select *
from Table_Client " : "select * from Table_Users ";
    zapros += comboBoxPrava.Text == "Клиент" ? "where (FamCl = '"
+ Text_Login.Text + "') and (Document='" + Text_Password.Text.Trim() +
"'" :

        "where (Login = '" + Text_Login.Text + "') and
(Password='" + Text_Password.Text + "')";
    buffer = Podkl.GetDataFromTable(zapros);
    int totalRecords = buffer.Rows.Count;
    if (totalRecords <= 0)
    {
        MessageBox.Show(this, "Проверьте правильность
введенной информации.", "Режим авторизации", MessageBoxButtons.OK,
        MessageBoxIcon.Warning);
        Text_Login.Focus();
        return;
    }
    else
    {
        ClassStr.TekPrava = comboBoxPrava.Text == "Клиент"
? "Клиент":buffer.Rows[0]["Prava"].ToString();
        if(comboBoxPrava.Text == "Клиент")
        {
            this.Hide();
            ClassStr.TekCodClient =
int.Parse(buffer.Rows[0]["CodClient"].ToString());
            FormMainClient newForm = new FormMainClient();
            newForm.ShowDialog(); //блокируются предыдущие
формы, активна только текущая (новая открытая)
        }
        else
        {
            if (comboBoxPrava.Text !=
buffer.Rows[0]["Prava"].ToString())
            {
                MessageBox.Show(this, "Выбранный режим не
совпадает с правами пользователя.", "Режим авторизации",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
                comboBoxPrava.Focus();
                return;
            }

            this.Hide();
            FormMain newForm = new FormMain();
            newForm.ShowDialog(); //блокируются предыдущие
формы, активна только текущая (новая открытая)
        }
    }
}

```

```

        Close();
    }
}

//Кнопка - Регистрация
private void ButtonRegistr_Click(object sender, EventArgs e)
{
    if (comboBoxPrava.Text == "")
    {
        MessageBox.Show("Укажите режим");
        comboBoxPrava.Focus();
        return;
    }

    if (comboBoxPrava.Text != "Клиент" && Text_Password.Text ==
"")
    {
        MessageBox.Show("Введите служебный пароль для
регистрации сотрудника");
        Text_Password.Focus();
        return;
    }

    if (comboBoxPrava.Text != "Клиент" && Text_Password.Text !=
"admin")
    {
        MessageBox.Show("Служебный пароль для регистрации
сотрудника указан неверно.");
        Text_Password.Focus();
        return;
    }

    if (comboBoxPrava.Text != "Клиент")
    {
        this.Hide();
        FormUsersInfo newForm = new FormUsersInfo();
        newForm.RegimUsers = 2; //2- режим создания
        newForm.ShowDialog(); //блокируются предыдущие
формы, активна только текущая (новая открытая)
        this.Show();

        if (newForm.CheckUsers > 0)
        {
            MessageBox.Show(this, "Новый сотрудник
успешно зарегистрирован.", "Режим регистрации", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
        }
        else
        {
            MessageBox.Show(this, "Регистрация
отклонена.", "Режим регистрации", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
        }
    }
    else
    {
        this.Hide();
        FormClientInfo newForm = new FormClientInfo();
    }
}

```

```

        newForm.RegimClient = 2; //2- режим создания
        newForm.ShowDialog(); //блокируются предыдущие
формы, активна только текущая (новая открытая)
        this.Show();

        if (newForm.CheckClient > 0)
        {
            MessageBox.Show(this, "Новый клиент успешно
зарегистрирован.", "Режим регистрации", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
        }
        else
        {
            MessageBox.Show(this, "Регистрация
отклонена.", "Режим регистрации", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
        }
    }

    Text_Login.Focus();
}

private void Text_Login_KeyPress(object sender, KeyPressEventArgs
e)
{
    if (e.KeyChar == (char)27)
    {
        Close();
    }
    else if (e.KeyChar == (char)13)
    {
        buttonVhod_Click(sender, e);
    }
}
}

```

Листинг Е.20 – Класс FormMain

```

public partial class FormMain : Form
{
    Query Podkl; //Определим наш объект класса
    ContextMenuStrip contextMenuStrip1;

    public FormMain()
    {
        InitializeComponent();

        //Пункт меню - Выход
        private void Exit_Click(object sender, EventArgs e)
        {
            Close();
        }

        //Пункт меню - Сотрудники
        private void сотрудникиToolStripMenuItem_Click(object sender,
EventArgs e)
        {

```

```

        FormUsers newForm = new FormUsers();
        newForm.ShowDialog(); //блокируются предыдущие формы,
активна только текущая (новая открытая)
    }

    //Пункт меню - Клиенты
    private void клиентыToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        FormClient newForm = new FormClient();
        newForm.ShowDialog(); //блокируются предыдущие формы,
активна только текущая (новая открытая)
    }

    private void FormMain_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (e.KeyChar == (char)27)
        {
            Exit_Click(sender, e);
        }
    }

    //Пункт меню Справочники - Виды препаратов
    private void видыПрепаратовToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        ClassStr.TekSpravochnik = 1; //Виды препаратов
        FormSpravochnik newForm = new FormSpravochnik();
        newForm.ShowDialog(); //блокируются предыдущие формы,
активна только текущая (новая открытая)
    }

    //Пункт меню Справочники - Статусы
    private void статусыToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        ClassStr.TekSpravochnik = 2; //Статусы
        FormSpravochnik newForm = new FormSpravochnik();
        newForm.ShowDialog(); //блокируются предыдущие формы,
активна только текущая (новая открытая)
    }

    //Пункт меню Справочники - Должности
    private void должностиToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        ClassStr.TekSpravochnik = 3; //Должности
        FormSpravochnik newForm = new FormSpravochnik();
        newForm.ShowDialog(); //блокируются предыдущие формы,
активна только текущая (новая открытая)
    }

    //Пункт меню Справочники - Поставщики
    private void поставщикиToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        FormSupplier newForm = new FormSupplier();

```

```

        newForm.ShowDialog(); //блокируются предыдущие формы,
активна только текущая (новая открытая)
    }

    //Пункт меню Справочники - Пункты выдачи
    private void пунктыВыдачиToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        FormPunkt newForm = new FormPunkt();
        newForm.ShowDialog(); //блокируются предыдущие формы,
активна только текущая (новая открытая)
    }

    private void заказыToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        contextMenuStrip1.Items.Clear();
        contextMenuStrip1.Items.Add("Лекарственные средства");
        if (ClassStr.TekPrava!="Фармацевт")
            contextMenuStrip1.Items.Add("Препараты (у
поставщиков)");

        contextMenuStrip1.Show(Knopka_Zakaz, new Point(0,
Knopka_Zakaz.Height));
    }

    //Пункт меню - Заказы (Лекарственные средства)
    private void лекарственныеСредстваToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        ClassStr.RegimZakaz = 1; //Заказы (Лекарственные
средства)

        FormZakaz newForm = new FormZakaz();
        newForm.ShowDialog(); //блокируются предыдущие формы,
активна только текущая (новая открытая)
    }

    //Пункт меню - Заказы (Препараты (у поставщиков))
    private void препаратыПоставщиковToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        ClassStr.RegimZakaz = 2; //Заказы (Препараты (у
поставщиков))

        FormZakaz newForm = new FormZakaz();
        newForm.ShowDialog(); //блокируются предыдущие формы,
активна только текущая (новая открытая)
    }

    private void FormMain_Load(object sender, EventArgs e)
    {
        contextMenuStrip1 = new ContextMenuStrip();
        contextMenuStrip1.ItemClicked += new
ToolStripItemClickedEventHandler(contextMenuStrip1_ItemClicked);

        if (ClassStr.TekPrava=="Фармацевт")
        {
            должностиToolStripMenuItem.Visible = false;
            поставщикиToolStripMenuItem.Visible = false;

```



```

        видыПрепаратовToolStripMenuItem.Visible = false;
        клиентыToolStripMenuItem.Visible = false;
        ПрепаратыToolStripMenuItem.Visible = false;
        препаратыПоставщиковToolStripMenuItem.Visible =
false;

        поПоставщикамToolStripMenuItem.Visible = false;
        выходToolStripMenuItem1.Visible = false;
        сотрудникиToolStripMenuItem.Visible = false;
        Кнопка_Preparat.Visible = false;
    }
    if (ClassStr.TekPrava == "Провизор")
    {
        должностиToolStripMenuItem.Visible = false;
        пунктыВыдачиToolStripMenuItem.Visible = false;
        сотрудникиToolStripMenuItem.Visible = false;
        Кнопка_Users.Visible = false;
        заказыToolStripMenuItem.Visible = false;
        ПрепаратыToolStripMenuItem.Visible = false;
        поСотрудникамToolStripMenuItem.Visible = false;
        поКлиентамToolStripMenuItem.Visible = false;
        выходToolStripMenuItem1.Visible = false;
        Кнопка_Users.Visible = false;
        Кнопка_Client.Visible = false;
    }
}

private void contextMenuStrip1_ItemClicked(object sender,
ToolStripItemClickedEventArgs e)
{
    //MessageBox.Show(e.ClickedItem.ToString());
    switch (e.ClickedItem.ToString())
    {
        case "Лекарственные средства":

лекарственныеСредстваToolStripMenuItem_Click(sender, e);
            break;
        case "Препараты (у поставщиков)":

препаратыПоставщиковToolStripMenuItem_Click(sender, e);
            break;
    }
}

//Пункт меню - Лекарственные средства
private void лекарствсредстваToolStripMenuItem_Click(object sender,
EventArgs e)
{
    FormLekarstvo newForm = new FormLekarstvo();
    newForm.ShowDialog(); //блокируются предыдущие формы,
активна только текущая (новая открытая)
}

//Пункт меню - Препараты
private void ПрепаратыToolStripMenuItem_Click(object sender,
EventArgs e)
{
    FormPreparat newForm = new FormPreparat();

```

```

        newForm.ShowDialog(); //блокируются предыдущие формы,
        активна только текущая (новая открытая)
    }

    public static void WriteInHistory(string StrInform, int
ValCodZakaz)
    {
        string StrZ = "insert into Table_History
(Data,Inform,CodZakaz) values ('" +
                                DateTime.Now.ToString("yyyy-
MM-dd") + "',''" + StrInform + "',''" + ValCodZakaz + ")";

        //Изменяем кол-во (уменьшаем) в списке состава заказа
        MySqlConnection connection; //создать подключение между
нашим проектом и БД
        MySqlCommand command; //нужен для выполнения
операция вставки, редактирования и удаления данных БД
        connection = new MySqlConnection(Query.ConnStr);
        connection.Open(); //Открыть соединение

        command = new MySqlCommand(StrZ, connection);
        command.ExecuteNonQuery(); //Выполняем запрос. В
качестве результата возвращает кол-во обработанных строк
        connection.Close(); //Закрываем соединение
    }

    //Пункт меню - Отчет (История по заказам)
    private void историяПоЗаказамToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        FormReportHistory newForm = new FormReportHistory();
        newForm.ShowDialog(); //блокируются предыдущие формы,
        активна только текущая (новая открытая)
    }

    //Пункт меню - Отчет (По клиентам)
    private void поКлиентамToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        FormReportClientZakaz newForm = new
FormReportClientZakaz();
        newForm.ShowDialog(); //блокируются предыдущие формы,
        активна только текущая (новая открытая)
    }

    //Пункт меню - Отчет (По сотрудникам)
    private void поСотрудникамToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        FormReportUsersZakaz newForm = new
FormReportUsersZakaz();
        newForm.ShowDialog(); //блокируются предыдущие формы,
        активна только текущая (новая открытая)
    }

    //Пункт меню - Отчет (По поставщикам)
    private void поПоставщикамToolStripMenuItem_Click(object sender,
EventArgs e)

```

```

    {
        FormReportSupplierPreparat newForm = new
FormReportSupplierPreparat();
        newForm.ShowDialog(); //блокируются предыдущие формы,
активна только текущая (новая открытая)
    }

    //Пункт меню - Отчет (По видам препаратов)
    private void поВидамПрепаратовToolStripMenuItem_Click(object
sender, EventArgs e)
    {
        FormReportViewPreparat newForm = new
FormReportViewPreparat();
        newForm.ShowDialog(); //блокируются предыдущие формы,
активна только текущая (новая открытая)
    }
}

```

Листинг E.21 – Класс FormMainClient

```

public partial class FormMainClient : Form
{
    public FormMainClient()
    {
        InitializeComponent();
    }

    //Обработчик кнопки Выход
    private void Exit_Click(object sender, EventArgs e)
    {
        Close();
    }

    //Клавиша ESC
    private void Exit_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (e.KeyChar == (char)27)
        {
            Exit_Click(sender, e);
        }
    }

    private void Knopka_Zakaz_Click(object sender, EventArgs e)
    {
        ClassStr.RegimZakaz = 1; //Заказы (Лекарственные средства)
        FormZakaz newForm = new FormZakaz();
        newForm.ShowDialog(); //блокируются предыдущие формы, активна
только текущая (новая открытая)
    }

    private void Knopka_DannyeClient_Click(object sender, EventArgs e)
    {
        this.Hide();
        FormClientInfo newForm = new FormClientInfo();
        newForm.RegimClient = 1; //1- режим редактирования
        newForm.text_ID = ClassStr.TekCodClient.ToString();
    }
}

```

```

        newForm.ShowDialog(); //блокируются предыдущие формы, активна
        только текущая (новая открытая)
        this.Show();
    }
}

```

Листинг E.22 – Класс FormOplataZakaz

```

public partial class FormOplataZakaz : Form
{
    Query Podkl; //Определим наш объект класса
    public int TekID_Oplata = 0;
    public string Tekzapros;
    private Microsoft.Office.Interop.Excel.Application excelapp;
    private Microsoft.Office.Interop.Excel.Window excelWindow;
    private Microsoft.Office.Interop.Excel.Sheets excelsheets;
    private Microsoft.Office.Interop.Excel.Worksheet excelworksheet;

    public FormOplataZakaz()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);

        Tekzapros = "select CodOplata,Data,Summa from Table_Oplata ";
    }

    //Обработчик кнопки Выход
    private void Exit_Click(object sender, EventArgs e)
    {
        Close();
    }

    //Обработчик кнопки Добавить
    private void Knopka_Add_Click(object sender, EventArgs e)
    {
        FormOplataZakazInfo newForm = new FormOplataZakazInfo();
        newForm.RegimOplata = 2; //2- режим создания
        newForm.ShowDialog(); //блокируются предыдущие формы, активна
        только текущая (новая открытая)
        TekID_Oplata = -1;
        this.TekID_Oplata = newForm.CheckOplata;
        if (TekID_Oplata < 0) //Если был выход в окне с данными по
        договору, т.е., данные не добавлены
            return;

        GridOplataZakaz.DataSource =
        Podkl.GetDataFromTable(Tekzapros);

        //Установить курсор на добавленную запись
        string searchValue = Convert.ToString(TekID_Oplata);
        GridOplataZakaz.SelectionMode =
        DataGridViewSelectionMode.FullRowSelect;

        try
        {

```

```

        foreach (DataGridViewRow row in GridOplataZakaz.Rows)
        {
            if
            (row.Cells[0].Value.ToString().Equals(searchValue))
            {
                row.Selected = true;
                GridOplataZakaz.CurrentCell =
GridOplataZakaz.Rows[row.Index].Cells[1];
                break;
            }
        }
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
    //-----Конец установки курсора
}

//Обработчик кнопки Изменить
private void Knopka_Change_Click(object sender, EventArgs e)
{
    if (GridOplataZakaz.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
        return;
    }

    if
    (GridOplataZakaz.Rows[GridOplataZakaz.CurrentRow.Index].Cells["CodOplata"]
    ].Value.ToString() == "")
        return;

    //Запоминаем позицию в DBGrid, чтобы после редактирования в
    другом окне установить курсор на эту строку
    int iStr = GridOplataZakaz.CurrentCell.RowIndex;

    FormOplataZakazInfo newForm = new FormOplataZakazInfo();
    newForm.RegimOplata = 1; //1- режим редактирования
    newForm.text_ID =
GridOplataZakaz.Rows[GridOplataZakaz.CurrentRow.Index].Cells["CodOplata"]
.Value.ToString();
    newForm.ShowDialog(); //блокируются предыдущие формы, активна
    только текущая (новая открытая)

    GridOplataZakaz.DataSource =
    Podkl.GetDataFromTable(Tekzapros);

    //Установить курсор на измененную запись
    GridOplataZakaz.SelectionMode =
    DataGridViewSelectionMode.FullRowSelect;
    GridOplataZakaz.CurrentCell =
    GridOplataZakaz.Rows[iStr].Cells[1]; //ставим выделение на измененную
    строку
}

//Обработчик кнопки Удалить
private void Knopka_Delete_Click(object sender, EventArgs e)

```

```

{
    if (GridOplataZakaz.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
        return;
    }

    //Передаем индекс текущей записи в Grid
    Podkl.DeleteFromTable("Table_Oplata", "CodOplata",
int.Parse(GridOplataZakaz.Rows[GridOplataZakaz.CurrentRow.Index].Cells["C
odOplata"].Value.ToString()));
    GridOplataZakaz.DataSource =
Podkl.GetDataFromTable(Tekzapros);
    MessageBox.Show("Удаление выполнено успешно");
}

//Обработчик двойного нажатия мышкой в списке данных
private void GridOplataZakaz_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
{
    if (ClassStr.TekPrava == "Клиент")
        return;

    Knopka_Change_Click(sender, e);
}

//Клавиша ESC
private void FormOplataZakaz_KeyPress(object sender,
KeyPressEventArgs e)
{
    if (e.KeyChar == (char)27)
    {
        Exit_Click(sender, e);
    }
}

//Обработчик загрузки формы
private void FormOplataZakaz_Load(object sender, EventArgs e)
{
    // задать цвет в заголовках столбцов
    GridOplataZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
    // задать шрифт в заголовках столбцов
    Font F = new Font("Times New Roman", 14);
    GridOplataZakaz.ColumnHeadersDefaultCellStyle.Font = F;

    Font F2 = new Font("Times New Roman", 12);
    GridOplataZakaz.DefaultCellStyle.Font = F2;
}

//Обработчик отображения формы
private void FormOplataZakaz_Shown(object sender, EventArgs e)
{
    Tekzapros += "where CodZakaz = " +
int.Parse(ClassStr.TekCodZakaz.ToString());
    Tekzapros += " order by Data desc";
    LabelInfo.Text = "Заказ: " + ClassStr.TekNameZakaz;
}

```

```

        if (ClassStr.TekPrava=="Клиент")
        {
            Knopka_Add.Enabled = true;
            Knopka_Change.Enabled = false;
            Knopka_Delete.Enabled = false;
        }

        GridOplataZakaz.DataSource =
Podkl.GetDataFromTable(Tekzapros);
        GridOplataZakaz.Columns[0].Visible = false;
        GridOplataZakaz.Columns[1].Width = 160;
        GridOplataZakaz.Columns[1].HeaderText = "Дата";
        GridOplataZakaz.Columns[1].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridOplataZakaz.Columns[2].Width = 200;
        GridOplataZakaz.Columns[2].HeaderText = "Сумма";
        GridOplataZakaz.Columns[2].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridOplataZakaz.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridOplataZakaz.DefaultCellStyle.SelectionBackColor =
Color.White;
        GridOplataZakaz.DefaultCellStyle.SelectionForeColor =
Color.Blue;
        GridOplataZakaz.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridOplataZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовка столбцов.
        //Будет отображаться цвет, который установлен для заголовков

        GridOplataZakaz.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridOplataZakaz.ColumnHeadersDefaultCellStyle.BackColor;

        GridOplataZakaz.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridOplataZakaz.ColumnHeadersDefaultCellStyle.ForeColor;

        GridOplataZakaz.EnableHeadersVisualStyles = false;

        if (GridOplataZakaz.Rows.Count > 0)
        {
            GridOplataZakaz.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridOplataZakaz.CurrentCell =
GridOplataZakaz.Rows[0].Cells[1]; //ставим выделение на самую первую
строку
            GridOplataZakaz.Focus();
        }
    }

    private void Knopka_Excel_Click(object sender, EventArgs e)
    {
        if (GridOplataZakaz.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }
    }

```

```

        this.Enabled = false;

        DateTime datel = DateTime.Now;
        var fileReport =
Path.GetDirectoryName(Application.ExecutablePath) +
"\Отчеты\ОплатыПоЗаказу_" + ClassStr.TekNameZakaz +
                                                    " от " +
datel.Day.ToString() + "." + datel.Month.ToString() + "." +
datel.Year.ToString() + "_" +

datel.Hour.ToString() + "_" + datel.Minute.ToString() + "_" +
datel.Second.ToString() +
                                                    ".xlsx";

        excelapp = new Microsoft.Office.Interop.Excel.Application();
        excelapp.Visible = false;

        excelapp.SheetsInNewWorkbook = 2; //кол-во листов в excel
        excelapp.Workbooks.Add(Type.Missing);
        Microsoft.Office.Interop.Excel.Range excelcells;

        //Получаем набор ссылок на объекты Workbook (на созданные
книги)
        var excelappworkbooks = excelapp.Workbooks;

        //Получаем ссылку на книгу 1 - нумерация от 1
        var excelappworkbook = excelappworkbooks[1];

        //НЕ Будем спрашивать разрешение на запись поверх
существующего документа
        excelapp.DisplayAlerts = false;

        excelsheets = excelappworkbook.Worksheets;
        //Получаем ссылку на лист 1
        excelworksheet =
(Microsoft.Office.Interop.Excel.Worksheet)excelsheets.get_Item(1);
        //Выделение группы ячеек
        excelcells = excelworksheet.get_Range("A2", "C2");
        excelcells.Merge(Type.Missing);
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 14;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
        excelcells.Value2 = @"Дата: " + datel.Day.ToString() + "." +
datel.Month.ToString() + "." + datel.Year.ToString();

        excelcells = excelworksheet.get_Range("A4", "C4");
        excelcells.Merge(Type.Missing);
        //Задаем выравнивание по центру
        excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 16;

```



```

excelcells.EntireRow.Font.Color = Color.Black;
excelcells.Value2 = @"Отчет 'Оплаты по заказу'";

excelcells = excelworksheet.get_Range("A5", "C5");
excelcells.Merge(Type.Missing);
//Задаем выравнивание по центру
excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
excelcells.EntireRow.Font.Bold = true;
excelcells.EntireRow.Font.Size = 16;
excelcells.EntireRow.Font.Color = Color.Black;
excelcells.Value2 = ClassStr.TekNameZakaz;
excelworksheet.Cells[5, FormClient.AbcToIndex("C")].WrapText
= true; //Если текст не уместается в одной ячейке, то будет перенос на
следующие строки в пределах одной ячейки

//Выводим названия столбцов
excelcells = excelworksheet.get_Range("A7", "C7");
excelcells.EntireRow.Font.Bold = true;
excelcells.EntireRow.Font.Size = 14;
excelcells.EntireRow.Font.Color = Color.Black;
excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

excelcells = excelworksheet.get_Range("A7", "A7");
excelcells.Value2 = @"№";
excelworksheet.Columns[1].ColumnWidth = 16;
excelcells = excelworksheet.get_Range("B7", "B7");
excelcells.Value2 = @"Дата";
excelworksheet.Columns[2].ColumnWidth = 26;
excelcells = excelworksheet.get_Range("C7", "C7");
excelcells.Value2 = @"Сумма";
excelworksheet.Columns[3].ColumnWidth = 26;

MySQLCommand command = new MySQLCommand();
command.CommandText = Tekzapros;
DataTable tbl = new DataTable();
using (MySQLDataAdapter dataAdapter = new
MySQLDataAdapter(command.CommandText, Query.ConnStr))
{
    dataAdapter.Fill(tbl);
}
var j = 0;
for (j = 0; j < tbl.Rows.Count; j++)
{
    {
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")] = (j + 1).ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

        DateTime dt =
DateTime.Parse(tbl.Rows[j][1].ToString());
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")] = dt.ToString("dd-MM-yyyy");
    }
}

```

```

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")] = tbl.Rows[j][2].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
    }
}

Object NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("A")];
excelcells = excelworksheet.get_Range("A7", NomCell);
excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

//Устанавливаем цвет обводки
NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("C")];
excelcells = excelworksheet.get_Range("A7", NomCell);
excelcells.Borders.ColorIndex = 1; //черный
//Устанавливаем стиль и толщину линии
excelcells.Borders.LineStyle =
Microsoft.Office.Interop.Excel.XlLineStyle.xlContinuous;
excelcells.Borders.Weight =
Microsoft.Office.Interop.Excel.XlBorderWeight.xlMedium;//.xlThick;

excelapp.Visible = true;

object misValue = System.Reflection.Missing.Value;
excelappworkbook.SaveAs(@fileReport,
Microsoft.Office.Interop.Excel.XlFileFormat.xlOpenXMLWorkbook, misValue,
        misValue, misValue, misValue,
Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive,
        misValue, misValue, misValue,
misValue, misValue);
this.Enabled = true;
}

private void Knopka_Check_Click(object sender, EventArgs e)
{
    if (GridOplataZakaz.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
        return;
    }

    this.Enabled = false;

    DateTime datel = DateTime.Now;
    var fileReport =
Path.GetDirectoryName(Application.ExecutablePath) +
"\\Чеки\\ЧекОплатыПоЗаказу_" + ClassStr.TekNameZakaz +

```

```

                                " от " +
date1.Day.ToString() + "." + date1.Month.ToString() + "." +
date1.Year.ToString() + "_" +

date1.Hour.ToString() + "_" + date1.Minute.ToString() + "_" +
date1.Second.ToString() +

                                ".xlsx";

    excelapp = new Microsoft.Office.Interop.Excel.Application();
    excelapp.Visible = false;

    excelapp.SheetsInNewWorkbook = 2; //кол-во листов в excel
    excelapp.Workbooks.Add(Type.Missing);
    Microsoft.Office.Interop.Excel.Range excelcells;

    //Получаем набор ссылок на объекты Workbook (на созданные
книги)
    var excelappworkbooks = excelapp.Workbooks;

    //Получаем ссылку на книгу 1 - нумерация от 1
    var excelappworkbook = excelappworkbooks[1];

    //НЕ Будем спрашивать разрешение на запись поверх
существующего документа
    excelapp.DisplayAlerts = false;

    excelsheets = excelappworkbook.Worksheets;
    //Получаем ссылку на лист 1
    excelworksheet =
(Microsoft.Office.Interop.Excel.Worksheet)excelsheets.get_Item(1);
    //Выделение группы ячеек
    excelcells = excelworksheet.get_Range("A2", "A2");
    excelcells.Merge(Type.Missing);
    excelcells.EntireRow.Font.Bold = true;
    excelcells.EntireRow.Font.Size = 14;
    excelcells.EntireRow.Font.Color = Color.Black;
    excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
    excelcells.Value2 =
DateTime.Parse(DateTime.Now.ToString()).ToString("dd-MM-yyyy");

    excelcells = excelworksheet.get_Range("A4", "A4");
    excelcells.Merge(Type.Missing);
    //Задаем выравнивание по центру
    excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
    excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
    excelcells.EntireRow.Font.Bold = true;
    excelcells.EntireRow.Font.Size = 16;
    excelcells.EntireRow.Font.Color = Color.Black;
    excelcells.Value2 = @"Чек оплаты по заказу";

    excelcells = excelworksheet.get_Range("A5", "A5");
    excelcells.Merge(Type.Missing);
    //Задаем выравнивание по центру

```

```

        excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 16;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.Value2 = ClassStr.TekNameZakaz;

        //Устанавливаем ширину первого столбца, куда будем все
ВЫВОДИТЬ
        excelworksheet.Columns[1].ColumnWidth = 60;
        excelcells = excelworksheet.get_Range("A7", "A12");
        excelcells.EntireRow.Font.Size = 16;

        excelcells = excelworksheet.get_Range("A7", "A7");
        excelcells.Value2 = @"Дата: " +
DateTime.Parse(GridOplataZakaz.Rows[GridOplataZakaz.CurrentRow.Index].Cells["Data"].Value.ToString()).ToString("dd-MM-yyyy");

        excelcells = excelworksheet.get_Range("A8", "A8");
        excelcells.Value2 = @"Сумма: " +
GridOplataZakaz.Rows[GridOplataZakaz.CurrentRow.Index].Cells["Summa"].Value.ToString();

        excelcells = excelworksheet.get_Range("A9", "A9");
        excelcells.Value2 = @"Оплатил: " + ClassStr.TekFioCl;

        excelcells = excelworksheet.get_Range("A11", "A11");
        excelcells.Value2 = @"Принял: ";

        excelapp.Visible = true;
        object misValue = System.Reflection.Missing.Value;
        excelappworkbook.SaveAs(@fileReport,
Microsoft.Office.Interop.Excel.XlFileFormat.xlOpenXMLWorkbook, misValue,
        misValue, misValue, misValue,
Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive,
        misValue, misValue, misValue,
misValue, misValue);
        this.Enabled = true;
    }
}

```

Листинг Е.23 – Класс FormOplataZakazInfo

```

public partial class FormOplataZakazInfo : Form
{
    Query Podkl; //Определим наш объект класса
    public string text_ID;
    public int RegimOplata, CheckOplata;

    public FormOplataZakazInfo()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
подключения
        Podkl = new Query(Query.ConnStr);
    }
}

```

```

    }

    private void DBEdit_Summa_KeyPress(object sender, KeyPressEventArgs
e)
    {
        if (e.KeyChar == (char)27)
        {
            But_Exit_Click(sender, e);
            return;
        }
        else if (e.KeyChar == (char)13)
        {
            But_Save_Click(sender, e);
            return;
        }

        char ch = e.KeyChar;
        if (!char.IsDigit(ch) && ch != 8)
        {
            e.Handled = true;
        }
    }

    //Кнопка Выход
    private void But_Exit_Click(object sender, EventArgs e)
    {
        CheckOplata = -1;
        Close();
    }

    //Кнопка Сохранить
    private void But_Save_Click(object sender, EventArgs e)
    {
        //Выполняем проверки заполненных полей данных
        if (DBEdit_Summa.Text == "")
        {
            MessageBox.Show("Введите сумму оплаты по заказу");
            DBEdit_Summa.Focus();
            return;
        }

        string StrJson = "";

        switch (RegimOplata)
        {
            case 1: //Режим редактирования
                ClassOplata Per_1 = new
ClassOplata(int.Parse(text_ID.ToString()),

                DateTime.Parse(DateTimeData.Text),

                int.Parse(DBEdit_Summa.Text),

                ClassStr.TekCodZakaz

                );
                StrJson = JsonConvert.SerializeObject(Per_1);
                Podkl.Update_JSON("Table_Oplata", StrJson);
                break;
            }
        }
    }

```

```

        case 2: //Режим создания
            ClassOplata Per_2 = new ClassOplata(0,

DateTime.Parse(DateTimeData.Text),

int.Parse(DBEdit_Summa.Text),

ClassStr.TekCodZakaz

            );
            StrJson = JsonConvert.SerializeObject(Per_2);
            CheckOplata = Podkl.Add_JSON("Table_Oplata",
StrJson);
            break;
        }

        Close();
    }

    //Обработчик загрузки формы
    private void FormOplataInfo_Load(object sender, EventArgs e)
    {
        DataTable bufferOplata = new DataTable();

        int index = 0;

        switch (RegimOplata)
        {
            case 1: //Режим редактирования
                string zapros = "select CodOplata,Data,Summa from
Table_Oplata ";

                zapros += "where CodOplata = ";
                zapros += text_ID;
                bufferOplata = Podkl.GetDataFromTable(zapros);

                //Заполняем поля для ввода информации по оплате
заказа
                DateTimeData.Text =
bufferOplata.Rows[0]["Data"].ToString();
                DBEdit_Summa.Text =
bufferOplata.Rows[0]["Summa"].ToString();
                break;
            case 2: //Режим создания
                //Заполняем поля для ввода информации по опате
заказа
                DateTimeData.Text = DateTime.Now.ToString();
                DBEdit_Summa.Text = "";
                break;
        }
    }

    //Обработчик отображения формы
    private void FormOplataInfo_Shown(object sender, EventArgs e)
    {
        DateTimeData.Focus();
    }

    //Клавиши ESC и ENTER

```

```

private void DateTimeData_KeyPress(object sender, KeyPressEventArgs
e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
    else if (e.KeyChar == (char)13)
    {
        But_Save_Click(sender, e);
    }
}

//Клавиша ESC
private void FormOplataInfo_KeyPress(object sender,
KeyPressEventArgs e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
}
}

```

Листинг Е.24 – Класс FormPreparat

```

public partial class FormPreparat : Form
{
    Query Podkl; //Определим наш объект класса
    public int TekID_Preparat = 0;
    public string Tekzapros;

    public FormPreparat()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
        Tekzapros = "select P.CodPreparat,P.Name,VP.Name as
ViewPreparat,P.Kol,";
        Tekzapros += "P.Cena,SP.Name as Supplier,P.Inform from
((Table_Preparat as P ";
        Tekzapros += "inner join Table_ViewPreparat as VP on
P.CodViewPreparat=VP.CodViewPreparat) ";
        Tekzapros += "inner join Table_Supplier as SP on
P.CodSupplier=SP.CodSupplier) ";
        Tekzapros += " order by Name";
    }

    //Обработчик кнопки Выход
    private void Exit_Click(object sender, EventArgs e)
    {
        Close();
    }

    //Обработчик кнопки Добавить
    private void Knopka_Add_Click(object sender, EventArgs e)

```

```

{
    FormPreparatInfo newForm = new FormPreparatInfo();
    newForm.RegimPreparat = 2; //2- режим создания
    newForm.ShowDialog(); //блокируются предыдущие формы, активна
    только текущая (новая открытая)
    TekID_Preparat = -1;
    this.TekID_Preparat = newForm.CheckPreparat;
    if (TekID_Preparat < 0) //Если был выход в окне с данными,
    т.е., данные не добавлены
        return;

    GridPreparat.DataSource = Podkl.GetDataFromTable(Tekzapros);

    //Установить курсор на добавленную запись
    string searchValue = Convert.ToString(TekID_Preparat);
    GridPreparat.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

    try
    {
        foreach (DataGridViewRow row in GridPreparat.Rows)
        {
            if
            (row.Cells[0].Value.ToString().Equals(searchValue))
            {
                row.Selected = true;
                GridPreparat.CurrentCell =
                GridPreparat.Rows[row.Index].Cells[1];
                break;
            }
        }
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
    //-----Конец установки курсора
}

//Обработчик кнопки Изменить
private void Knopka_Change_Click(object sender, EventArgs e)
{
    if (GridPreparat.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
        return;
    }

    if
    (GridPreparat.Rows[GridPreparat.CurrentRow.Index].Cells["CodPreparat"].Va
    lue.ToString() == "")
        return;

    //Запоминаем позицию в DBGrid, чтобы после редактирования в
    другом окне установить курсор на эту строку
    int iStr = GridPreparat.CurrentCell.RowIndex;

    FormPreparatInfo newForm = new FormPreparatInfo();

```



```

        newForm.RegimPreparat = 1; //1- режим редактирования
        newForm.text_ID =
GridPreparat.Rows[GridPreparat.CurrentRow.Index].Cells["CodPreparat"].Value.ToString();
        newForm.ShowDialog(); //блокируются предыдущие формы, активна
только текущая (новая открытая)

        GridPreparat.DataSource = Podkl.GetDataFromTable(Tekzapros);

        //Установить курсор на измененную запись
        GridPreparat.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridPreparat.CurrentCell = GridPreparat.Rows[iStr].Cells[1];
//ставим выделение на измененную строку
    }

    //Обработчик кнопки Удалить
    private void Knopka_Delete_Click(object sender, EventArgs e)
    {
        if (GridPreparat.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        //-----Поиск связей удаляемой записи в таблице с составом
заказа
        var cmd = new MySqlCommand("SELECT COUNT(*) FROM
Table_SostavZakaz where CodPreparat = @PidCod");
        cmd.Connection = new MySqlConnection(Query.ConnStr);
        cmd.Connection.Open();
        cmd.Parameters.AddWithValue("PidCod",
int.Parse(GridPreparat.Rows[GridPreparat.CurrentRow.Index].Cells["CodPrep
arat"].Value.ToString()));
        int totalRecords = Convert.ToInt32(cmd.ExecuteScalar());
        if (totalRecords > 0)
        {
            MessageBox.Show("Данное лекарственное средство удалить
нельзя, потому что оно входит в состав заказа!", "ВНИМАНИЕ");
            return;
        }
        cmd.Connection.Close();
        //-----Конец поиска

        //Передаем индекс текущей записи в Grid
        Podkl.DeleteFromTable("Table_Preparat", "CodPreparat",
int.Parse(GridPreparat.Rows[GridPreparat.CurrentRow.Index].Cells["CodPrep
arat"].Value.ToString()));
        GridPreparat.DataSource = Podkl.GetDataFromTable(Tekzapros);
        MessageBox.Show("Удаление выполнено успешно");
    }

    //Клавиша ESC
    private void FormPreparat_KeyPress(object sender, KeyPressEventArgs
e)
    {
        {
            if (e.KeyChar == (char)27)
            {

```

```

        Exit_Click(sender, e);
    }
}

//Обработчик загрузки формы
private void FormPreparat_Load(object sender, EventArgs e)
{
    // задать цвет в заголовках столбцов
    GridPreparat.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
    // задать шрифт в заголовках столбцов
    Font F = new Font("Times New Roman", 14);
    GridPreparat.ColumnHeadersDefaultCellStyle.Font = F;

    Font F2 = new Font("Times New Roman", 12);
    GridPreparat.DefaultCellStyle.Font = F2;
}

//Обработчик отображения формы
private void FormPreparat_Shown(object sender, EventArgs e)
{
    GridPreparat.DataSource = Podkl.GetDataFromTable(Tekzapros);
    GridPreparat.Columns[0].Visible = false;
    GridPreparat.Columns[1].Width = 230;
    GridPreparat.Columns[1].HeaderText = "Название";
    GridPreparat.Columns[2].Width = 130;
    GridPreparat.Columns[2].HeaderText = "Вид препарата";
    GridPreparat.Columns[2].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;
    GridPreparat.Columns[3].Width = 75;
    GridPreparat.Columns[3].HeaderText = "Кол";
    GridPreparat.Columns[3].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
    GridPreparat.Columns[4].Width = 85;
    GridPreparat.Columns[4].HeaderText = "Цена";
    GridPreparat.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
    GridPreparat.Columns[5].Width = 150;
    GridPreparat.Columns[5].HeaderText = "Поставщик";
    GridPreparat.Columns[5].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
    GridPreparat.Columns[6].Width = 330;
    GridPreparat.Columns[6].HeaderText = "Информация";
    GridPreparat.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;
    GridPreparat.DefaultCellStyle.SelectionBackColor =
Color.White;
    GridPreparat.DefaultCellStyle.SelectionForeColor = Color.Blue;
    GridPreparat.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
    GridPreparat.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

    //отменить выделение строки заголовка столбцов.
    //Будет отображаться цвет, который установлен для заголовков
    GridPreparat.ColumnHeadersDefaultCellStyle.SelectionBackColor
= GridPreparat.ColumnHeadersDefaultCellStyle.BackColor;
}

```

```

        GridPreparat.ColumnHeadersDefaultCellStyle.SelectionForeColor
= GridPreparat.ColumnHeadersDefaultCellStyle.ForeColor;

        GridPreparat.EnableHeadersVisualStyles = false;

        if (GridPreparat.Rows.Count > 0)
        {
            GridPreparat.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridPreparat.CurrentCell =
GridPreparat.Rows[0].Cells[1]; //ставим выделение на самую первую строку
            GridPreparat.Focus();
        }
    }

    //Обработчик в поле поиска по названию
    private void TextBoxName_TextChanged(object sender, EventArgs e)
    {
        string zapros = "select P.CodPreparat,P.Name,VP.Name as
ViewPreparat,P.Kol,";
        zapros += "P.Cena,SP.Name as Supplier,P.Inform from
((Table_Preparat as P ";
        zapros += "inner join Table_ViewPreparat as VP on
P.CodViewPreparat=VP.CodViewPreparat) ";
        zapros += "inner join Table_Supplier as SP on
P.CodSupplier=SP.CodSupplier) ";

        if (textBoxName.Text != "")
        {
            zapros += "where P.Name like '%" + textBoxName.Text +
"% ' order by P.Name";
        }
        else
        {
            zapros += "order by P.Name";
        }

        GridPreparat.DataSource = Podkl.GetDataFromTable(zapros);
        if (GridPreparat.Rows.Count > 0)
        {
            GridPreparat.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridPreparat.CurrentCell =
GridPreparat.Rows[0].Cells[1]; //ставим выделение на самую первую строку
        }
        textBoxName.Focus();
    }

    //Обработчик двойного нажатия мышкой в списке данных
    private void GridPreparat_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        Knopka_Change_Click(sender, e);
    }
}

```

Листинг E.25 – Класс FormPreparatInfo

```

public partial class FormPreparatInfo : Form
{
    Query Podkl; //Определим наш объект класса
    public string text_ID;
    public int RegimPreparat, CheckPreparat;

    public FormPreparatInfo()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
    }

    //Кнопка Сохранить
    private void But_Save_Click(object sender, EventArgs e)
    {
        //Выполняем проверки заполненных полей данных
        if (DBEdit_Name.Text == "")
        {
            MessageBox.Show("Введите название");
            DBEdit_Name.Focus();
            return;
        }

        if (DBEdit_Kol.Text == "")
        {
            MessageBox.Show("Введите кол-во больше 0");
            DBEdit_Kol.Focus();
            return;
        }

        if (DBEdit_Cena.Text == "")
        {
            MessageBox.Show("Введите стоимость больше 0");
            DBEdit_Cena.Focus();
            return;
        }

        if (int.Parse(DBEdit_Kol.Text) < 1)
        {
            MessageBox.Show("Введите кол-во больше 0");
            DBEdit_Kol.Focus();
            return;
        }

        if (int.Parse(DBEdit_Cena.Text) < 1)
        {
            MessageBox.Show("Введите стоимость больше 0");
            DBEdit_Cena.Focus();
            return;
        }

        if (comboBoxViewPreparat.Text == "")
        {

```

```

        MessageBox.Show("Укажите вид препарата");
        comboBoxViewPreparat.Focus();
        return;
    }

    if (comboBoxSupplier.Text == "")
    {
        MessageBox.Show("Укажите поставика");
        comboBoxSupplier.Focus();
        return;
    }

    string StrJson = "";

    switch (RegimPreparat)
    {
        case 1: //Режим редактирования
            ClassPreparat Per_1 = new
ClassPreparat(int.Parse(text_ID.ToString()),

            DBEdit_Name.Text,

            DBEdit_Inform.Text,

            int.Parse(DBEdit_Cena.Text),

            int.Parse(DBEdit_Kol.Text),

            int.Parse(comboBoxViewPreparat.SelectedValue.ToString()),

            int.Parse(comboBoxSupplier.SelectedValue.ToString())

            );
            StrJson = JsonConvert.SerializeObject(Per_1);
            Podkl.Update_JSON("Table_Preparat", StrJson);
            break;
        case 2: //Режим создания
            ClassPreparat Per_2 = new ClassPreparat(0,

            DBEdit_Name.Text,

            DBEdit_Inform.Text,

            int.Parse(DBEdit_Cena.Text),

            int.Parse(DBEdit_Kol.Text),

            int.Parse(comboBoxViewPreparat.SelectedValue.ToString()),

            int.Parse(comboBoxSupplier.SelectedValue.ToString())

            );
            StrJson = JsonConvert.SerializeObject(Per_2);
            CheckPreparat = Podkl.Add_JSON("Table_Preparat",
StrJson);
            break;
    }

    Close();
}

```

```

        //Клавиша ESC
        private void FormPreparatInfo_KeyPress(object sender,
        KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)27)
            {
                But_Exit_Click(sender, e);
            }
        }

        //Обработчик загрузки формы
        private void FormPreparatInfo_Load(object sender, EventArgs e)
        {
            DataTable bufferPreparat = new DataTable();
            DataTable bufferViewPreparat = new DataTable();
            DataTable bufferSupplier = new DataTable();

            //Заполняем combobox для видов препаратов
            //Инициализируем - передаем запрос и объект класса
            SqlConnection - connection
            string StrZ = "select * from Table_ViewPreparat order by
            Name";
            MySqlDataAdapter datadapterViewPreparat = new
            MySqlDataAdapter(StrZ, Query.ConnStr);
            datadapterViewPreparat.Fill(bufferViewPreparat); //вызываем
            метод Fill, чтобы после запроса заполнил буферную таблицу данными

            comboBoxViewPreparat.DataSource = bufferViewPreparat;
            comboBoxViewPreparat.DisplayMember = "Name";
            comboBoxViewPreparat.ValueMember = "CodViewPreparat";

            //Заполняем combobox для поставщиков
            //Инициализируем - передаем запрос и объект класса
            SqlConnection - connection
            StrZ = "select * from Table_Supplier order by Name";
            MySqlDataAdapter datadapterSupplier = new
            MySqlDataAdapter(StrZ, Query.ConnStr);
            datadapterSupplier.Fill(bufferSupplier); //вызываем метод
            Fill, чтобы после запроса заполнил буферную таблицу данными

            comboBoxSupplier.DataSource = bufferSupplier;
            comboBoxSupplier.DisplayMember = "Name";
            comboBoxSupplier.ValueMember = "CodSupplier";

            switch (RegimPreparat)
            {
                case 1: //Режим редактирования
                    string zapros = "select
                    P.CodPreparat,P.Name,VP.Name as ViewPreparat,P.Kol,";
                    zapros += "P.Cena,SP.Name as Supplier,P.Inform from
                    ((Table_Preparat as P ";
                    zapros += "inner join Table_ViewPreparat as VP on
                    P.CodViewPreparat=VP.CodViewPreparat) ";
                    zapros += "inner join Table_Supplier as SP on
                    P.CodSupplier=SP.CodSupplier) ";
                    zapros += "where P.CodPreparat = ";
                    zapros += text_ID;

```

```

        bufferPreparat = Podkl.GetDataFromTable(zapros);

        //Заполняем поля для ввода информации
        DBEdit_Name.Text =
bufferPreparat.Rows[0]["Name"].ToString();
        DBEdit_Inform.Text =
bufferPreparat.Rows[0]["Inform"].ToString(); ;
        DBEdit_Kol.Text =
bufferPreparat.Rows[0]["Kol"].ToString();
        DBEdit_Cena.Text =
bufferPreparat.Rows[0]["Cena"].ToString();

        string s_ViewPreparat =
bufferPreparat.Rows[0]["ViewPreparat"].ToString();
        int index =
comboBoxViewPreparat.FindString(s_ViewPreparat);
        comboBoxViewPreparat.SelectedIndex = index;

        string s_Supplier =
bufferPreparat.Rows[0]["Supplier"].ToString();
        index = comboBoxSupplier.FindString(s_Supplier);
        comboBoxSupplier.SelectedIndex = index;
        break;
    case 2: //Режим создания
        //Заполняем поля для ввода информации
        DBEdit_Name.Text = "";
        DBEdit_Inform.Text = "";
        DBEdit_Kol.Text = "";
        DBEdit_Cena.Text = "";
        if (comboBoxViewPreparat.Items.Count > 0)
            comboBoxViewPreparat.SelectedIndex = 0;
        if (comboBoxSupplier.Items.Count > 0)
            comboBoxSupplier.SelectedIndex = 0;
        break;
    }
}

//Кнопка Выход
private void But_Exit_Click(object sender, EventArgs e)
{
    CheckPreparat = -1;
    Close();
}

private void DBEdit_Name_KeyPress(object sender, KeyPressEventArgs
e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
    else if (e.KeyChar == (char)13)
    {
        But_Save_Click(sender, e);
    }
}

```

```

private void DBEdit_Kol_KeyPress(object sender, KeyPressEventArgs
e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
        return;
    }
    else if (e.KeyChar == (char)13)
    {
        But_Save_Click(sender, e);
        return;
    }

    char ch = e.KeyChar;
    if (!char.IsDigit(ch) && ch != 8)
    {
        e.Handled = true;
    }
}

//Обработчик отображения формы
private void FormPreparatInfo_Shown(object sender, EventArgs e)
{
    DBEdit_Name.Focus();
}
}

```

Листинг Е.26 – Класс FormPunkt

```

public partial class FormPunkt : Form
{
    Query Podkl; //Определим наш объект класса
    public int TekID_Punkt = 0;
    public string Tekzapros;

    public FormPunkt()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
        Tekzapros = "select * from Table_Punkt order by Name";
    }

    //Обработчик кнопки Выход
    private void Exit_Click(object sender, EventArgs e)
    {
        Close();
    }

    //Обработчик кнопки Добавить
    private void Knopka_Add_Click(object sender, EventArgs e)
    {
        FormPunktInfo newForm = new FormPunktInfo();
        newForm.RegimPunkt = 2; //2- режим создания
    }
}

```



```

        newForm.ShowDialog(); //блокируются предыдущие формы, активна
        только текущая (новая открытая)
        TekID_Punkt = -1;
        this.TekID_Punkt = newForm.CheckPunkt;
        if (TekID_Punkt < 0) //Если был выход в окне с данными, т.е.,
        данные не добавлены
            return;

        GridPunkt.DataSource = Podkl.GetDataFromTable(Tekzapros);

        //Установить курсор на добавленную запись
        string searchValue = Convert.ToString(TekID_Punkt);
        GridPunkt.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

        try
        {
            foreach (DataGridViewRow row in GridPunkt.Rows)
            {
                if
                (row.Cells[0].Value.ToString().Equals(searchValue))
                {
                    row.Selected = true;
                    GridPunkt.CurrentCell =
GridPunkt.Rows[row.Index].Cells[1];
                    break;
                }
            }
        }
        catch (Exception exc)
        {
            MessageBox.Show(exc.Message);
        }
        //-----Конец установки курсора
    }

    //Обработчик кнопки Изменить
    private void Knopka_Change_Click(object sender, EventArgs e)
    {
        if (GridPunkt.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        if
        (GridPunkt.Rows[GridPunkt.CurrentRow.Index].Cells["CodPunkt"].Value.ToStr
        ing() == "")
            return;

        //Запоминаем позицию в DBGrid, чтобы после редактирования в
        другом окне установить курсор на эту строку
        int iStr = GridPunkt.CurrentCell.RowIndex;

        FormPunktInfo newForm = new FormPunktInfo();
        newForm.RegimPunkt = 1; //1- режим редактирования

```

```

        newForm.text_ID =
GridPunkt.Rows[GridPunkt.CurrentRow.Index].Cells["CodPunkt"].Value.ToString();
        newForm.ShowDialog(); //блокируются предыдущие формы, активна
только текущая (новая открытая)

        GridPunkt.DataSource = Podkl.GetDataFromTable(Tekzapros);

        //Установить курсор на измененную запись
        GridPunkt.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridPunkt.CurrentCell = GridPunkt.Rows[iStr].Cells[1];
//ставим выделение на измененную строку
    }

    //Обработчик кнопки Удалить
    private void Knopka_Delete_Click(object sender, EventArgs e)
    {
        if (GridPunkt.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        //-----Поиск связей удаляемой записи в таблице с заказами
        var cmd = new MySqlCommand("SELECT COUNT(*) FROM Table_Zakaz
where CodPunkt = @PidCod");
        cmd.Connection = new MySqlConnection(Query.ConnStr);
        cmd.Connection.Open();
        cmd.Parameters.AddWithValue("PidCod",
int.Parse(GridPunkt.Rows[GridPunkt.CurrentRow.Index].Cells["CodPunkt"].Value.ToString()));
        int totalRecords = Convert.ToInt32(cmd.ExecuteScalar());
        if (totalRecords > 0)
        {
            MessageBox.Show("Данный пункт выдачи удалить нельзя,
потому что есть заказы, в которых указан данный пункт выдачи!",
"ВНИМАНИЕ");
            return;
        }
        cmd.Connection.Close();
        //-----Конец поиска

        //Передаем индекс текущей записи в Grid
        Podkl.DeleteFromTable("Table_Punkt", "CodPunkt",
int.Parse(GridPunkt.Rows[GridPunkt.CurrentRow.Index].Cells["CodPunkt"].Value.ToString()));
        GridPunkt.DataSource = Podkl.GetDataFromTable(Tekzapros);
        MessageBox.Show("Удаление выполнено успешно");
    }

    //Клавиша ESC
    private void FormPunkt_KeyPress(object sender, KeyPressEventArgs
e)
    {
        if (e.KeyChar == (char)27)
        {
            Exit_Click(sender, e);

```

```

    }
}

//Обработчик загрузки формы
private void FormPunkt_Load(object sender, EventArgs e)
{
    // задать цвет в заголовках столбцов
    GridPunkt.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
    // задать шрифт в заголовках столбцов
    Font F = new Font("Times New Roman", 14);
    GridPunkt.ColumnHeadersDefaultCellStyle.Font = F;

    Font F2 = new Font("Times New Roman", 12);
    GridPunkt.DefaultCellStyle.Font = F2;
}

//Обработчик отображения формы
private void FormPunkt_Shown(object sender, EventArgs e)
{
    GridPunkt.DataSource = Podkl.GetDataFromTable(Tekzapros);
    GridPunkt.Columns[0].Visible = false;
    GridPunkt.Columns[1].Width = 200;
    GridPunkt.Columns[1].HeaderText = "Название";
    GridPunkt.Columns[2].Width = 270;
    GridPunkt.Columns[2].HeaderText = "Адрес";
    GridPunkt.Columns[3].Width = 130;
    GridPunkt.Columns[3].HeaderText = "Телефон";
    GridPunkt.Columns[3].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
    GridPunkt.Columns[4].Width = 150;
    GridPunkt.Columns[4].HeaderText = "Время работы";
    GridPunkt.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
    GridPunkt.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
    GridPunkt.DefaultCellStyle.SelectionBackColor = Color.White;
    GridPunkt.DefaultCellStyle.SelectionForeColor = Color.Blue;
    GridPunkt.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
    GridPunkt.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

    //отменить выделение строки заголовка столбцов.
    //Будет отображаться цвет, который установлен для заголовков
    GridPunkt.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridPunkt.ColumnHeadersDefaultCellStyle.BackColor;
    GridPunkt.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridPunkt.ColumnHeadersDefaultCellStyle.ForeColor;

    GridPunkt.EnableHeadersVisualStyles = false;

    if (GridPunkt.Rows.Count > 0)
    {
        GridPunkt.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridPunkt.CurrentCell = GridPunkt.Rows[0].Cells[1];
    }
    //ставим выделение на самую первую строку

```

```

        GridPunkt.Focus();
    }
}

//Обработчик в поле поиска по названию
private void TextBoxName_TextChanged(object sender, EventArgs e)
{
    string zapros = "select * from Table_Punkt ";

    if (textBoxName.Text != "")
    {
        zapros += "where Name like '%" + textBoxName.Text + "%'
order by Name";
    }
    else
    {
        zapros += "order by Name";
    }

    GridPunkt.DataSource = Podkl.GetDataFromTable(zapros);
    if (GridPunkt.Rows.Count > 0)
    {
        GridPunkt.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridPunkt.CurrentCell = GridPunkt.Rows[0].Cells[1];
//ставим выделение на самую первую строку
    }
    textBoxName.Focus();
}

//Обработчик двойного нажатия мышкой в списке данных
private void GridPunkt_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
{
    Knopka_Change_Click(sender, e);
}
}

```

Листинг E.27 – Класс FormPunktInfo

```

public partial class FormPunktInfo : Form
{
    Query Podkl; //Определим наш объект класса
    public string text_ID;
    public int RegimPunkt, CheckPunkt;

    public FormPunktInfo()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
    }

    //Кнопка Сохранить
    private void But_Save_Click(object sender, EventArgs e)
    {

```

```

//Выполняем проверки заполненных полей данных
if (DBEdit_Name.Text == "")
{
    MessageBox.Show("Введите название");
    DBEdit_Name.Focus();
    return;
}

string StrJson = "";

switch (RegimPunkt)
{
    case 1: //Режим редактирования
        ClassPunkt Per_1 = new
ClassPunkt(int.Parse(text_ID.ToString()),

        DBEdit_Name.Text,

        DBEdit_Address.Text,

        DBEdit_Phone.Text,

        DBEdit_TimeWork.Text

        );
        StrJson = JsonConvert.SerializeObject(Per_1);
        Podkl.Update_JSON("Table_Punkt", StrJson);
        break;
    case 2: //Режим создания
        ClassPunkt Per_2 = new ClassPunkt(0,

        DBEdit_Name.Text,

        DBEdit_Address.Text,

        DBEdit_Phone.Text,

        DBEdit_TimeWork.Text

        );
        StrJson = JsonConvert.SerializeObject(Per_2);
        CheckPunkt = Podkl.Add_JSON("Table_Punkt",
StrJson);
        break;
}

Close();
}

//Клавиша ESC
private void FormPunktInfo_KeyPress(object sender,
KeyPressEventArgs e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
}

//Обработчик загрузки формы

```

```

private void FormPunktInfo_Load(object sender, EventArgs e)
{
    DataTable bufferPunkt = new DataTable();

    switch (RegimPunkt)
    {
        case 1: //Режим редактирования
            string zapros = "select * from Table_Punkt ";
            zapros += "where CodPunkt = ";
            zapros += text_ID;
            bufferPunkt = Podkl.GetDataFromTable(zapros);

            //Заполняем поля для ввода информации
            DBEdit_Name.Text =
bufferPunkt.Rows[0]["Name"].ToString();
            DBEdit_Phone.Text =
bufferPunkt.Rows[0]["Phone"].ToString();
            DBEdit_Address.Text =
bufferPunkt.Rows[0]["Address"].ToString();
            DBEdit_TimeWork.Text =
bufferPunkt.Rows[0]["TimeWork"].ToString();
            break;
        case 2: //Режим создания
            //Заполняем поля для ввода информации
            DBEdit_Name.Text = "";
            DBEdit_Phone.Text = "";
            DBEdit_Address.Text = "";
            DBEdit_TimeWork.Text = "";
            break;
    }
}

//Кнопка Выход
private void But_Exit_Click(object sender, EventArgs e)
{
    CheckPunkt = -1;
    Close();
}

private void DBEdit_Name_KeyPress(object sender, KeyPressEventArgs
e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
    else if (e.KeyChar == (char)13)
    {
        But_Save_Click(sender, e);
    }
}

//Обработчик отображения формы
private void FormPunktInfo_Shown(object sender, EventArgs e)
{
    DBEdit_Name.Focus();
}
}

```

Листинг E.28 – Класс FormRecept

```

public partial class FormRecept : Form
{
    Query Podkl; //Определим наш объект класса
    public int TekID_Recept = 0;
    public string Tekzapros;
    private Microsoft.Office.Interop.Excel.Application excelapp;
    private Microsoft.Office.Interop.Excel.Window excelWindow;
    private Microsoft.Office.Interop.Excel.Sheets excelsheets;
    private Microsoft.Office.Interop.Excel.Worksheet excelworksheet;

    public FormRecept()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);

        Tekzapros = "select R.CodRecept,R.Inform,R.NameFile, ";
        Tekzapros += " concat(CL.FamCl, '
', LEFT(CL.NameCl,1), '.', LEFT(CL.OtchCl,1), '.') as Client ";
        Tekzapros += " from (Table_Recept as R ";
        Tekzapros += " inner join (Table_Zakaz as ZK ";
        Tekzapros += " inner join Table_Client as CL on
ZK.CodClient=CL.CodClient) ";
        Tekzapros += " on R.CodZakaz=ZK.CodZakaz)";
    }

    //Обработчик кнопки Выход
    private void Exit_Click(object sender, EventArgs e)
    {
        Close();
    }

    //Обработчик кнопки Добавить
    private void Knopka_Add_Click(object sender, EventArgs e)
    {
        FormReceptInfo newForm = new FormReceptInfo();
        newForm.RegimRecept = 2; //2- режим создания
        newForm.ShowDialog(); //блокируются предыдущие формы, активна
        только текущая (новая открытая)
        TekID_Recept = -1;
        this.TekID_Recept = newForm.CheckRecept;
        if (TekID_Recept < 0) //Если был выход в окне с данными по
        договору, т.е., данные не добавлены
            return;

        GridRecept.DataSource = Podkl.GetDataFromTable(Tekzapros);

        //Установить курсор на добавленную запись
        string searchValue = Convert.ToString(TekID_Recept);
        GridRecept.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

        try
        {

```

```

        foreach (DataGridViewRow row in GridRecept.Rows)
        {
            if
            (row.Cells[0].Value.ToString().Equals(searchValue))
            {
                row.Selected = true;
                GridRecept.CurrentCell =
GridRecept.Rows[row.Index].Cells[1];
                break;
            }
        }
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
    //-----Конец установки курсора
}

//Обработчик кнопки Изменить
private void Knopka_Change_Click(object sender, EventArgs e)
{
    if (GridRecept.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
        return;
    }

    if
    (GridRecept.Rows[GridRecept.CurrentRow.Index].Cells["CodRecept"].Value.To
String() == "")
        return;

    //Запоминаем позицию в DBGrid, чтобы после редактирования в
другом окне установить курсор на эту строку
    int iStr = GridRecept.CurrentCell.RowIndex;

    FormReceptInfo newForm = new FormReceptInfo();
    newForm.RegimRecept = 1; //1- режим редактирования
    newForm.text_ID =
GridRecept.Rows[GridRecept.CurrentRow.Index].Cells["CodRecept"].Value.ToS
tring();
    newForm.ShowDialog(); //блокируются предыдущие формы, активна
только текущая (новая открытая)

    GridRecept.DataSource = Podkl.GetDataFromTable(Tekzapros);

    //Установить курсор на измененную запись
    GridRecept.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
    GridRecept.CurrentCell = GridRecept.Rows[iStr].Cells[1];
    //ставим выделение на измененную строку
}

//Обработчик кнопки Удалить
private void Knopka_Delete_Click(object sender, EventArgs e)
{
    if (GridRecept.Rows.Count < 2)

```



```

        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        //Передаем индекс текущей записи в Grid
        Podkl.DeleteFromTable("Table_Recept", "CodRecept",
int.Parse(GridRecept.Rows[GridRecept.CurrentRow.Index].Cells["CodRecept"]
.Value.ToString()));
        GridRecept.DataSource = Podkl.GetDataFromTable(Tekzapros);
        MessageBox.Show("Удаление выполнено успешно");
    }

    //Обработчик двойного нажатия мышкой в списке данных
    private void GridRecept_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        Knopka_Change_Click(sender, e);
    }

    //Клавиша ESC
    private void FormRecept_KeyPress(object sender, KeyPressEventArgs
e)
    {
        {
            if (e.KeyChar == (char)27)
            {
                Exit_Click(sender, e);
            }
        }
    }

    //Обработчик загрузки формы
    private void FormRecept_Load(object sender, EventArgs e)
    {
        // задать цвет в заголовках столбцов
        GridRecept.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
        // задать шрифт в заголовках столбцов
        Font F = new Font("Times New Roman", 14);
        GridRecept.ColumnHeadersDefaultCellStyle.Font = F;

        Font F2 = new Font("Times New Roman", 12);
        GridRecept.DefaultCellStyle.Font = F2;

        if (ClassStr.TekPrava == "Провизор")
        {
            Knopka_Add.Visible = false;
            Knopka_Change.Visible = false;
            Knopka_Delete.Visible = false;
        }
    }

    //Обработчик отображения формы
    private void FormRecept_Shown(object sender, EventArgs e)
    {
        Tekzapros += "where R.CodZakaz = " +
int.Parse(ClassStr.TekCodZakaz.ToString());
        Tekzapros += " order by R.Inform";
        LabelInfo.Text = "Заказ: " + ClassStr.TekNameZakaz;
    }

```

```

        GridRecept.DataSource = Podkl.GetDataFromTable(Tekzapros);
        GridRecept.Columns[0].Visible = false;
        GridRecept.Columns[1].Width = 320;
        GridRecept.Columns[1].HeaderText = "Информация";
        GridRecept.Columns[1].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;
        GridRecept.Columns[2].Width = 200;
        GridRecept.Columns[2].HeaderText = "Файл";
        GridRecept.Columns[2].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridRecept.Columns[3].Width = 130;
        GridRecept.Columns[3].HeaderText = "Клиент";
        GridRecept.Columns[3].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridRecept.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridRecept.DefaultCellStyle.SelectionBackColor = Color.White;
        GridRecept.DefaultCellStyle.SelectionForeColor = Color.Blue;
        GridRecept.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridRecept.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовка столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridRecept.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridRecept.ColumnHeadersDefaultCellStyle.BackColor;
        GridRecept.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridRecept.ColumnHeadersDefaultCellStyle.ForeColor;

        GridRecept.EnableHeadersVisualStyles = false;

        if (GridRecept.Rows.Count > 0)
        {
            GridRecept.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridRecept.CurrentCell = GridRecept.Rows[0].Cells[1];
            //ставим выделение на самую первую строку
            GridRecept.Focus();
        }
    }

    private void Knopka_Excel_Click(object sender, EventArgs e)
    {
        if (GridRecept.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        this.Enabled = false;

        DateTime datel = DateTime.Now;
        var fileReport =
Path.GetDirectoryName(Application.ExecutablePath) +
"\Отчеты\РецептыПоЗаказу_" + ClassStr.TekNameZakaz +

```

```

                                " от " +
date1.Day.ToString() + "." + date1.Month.ToString() + "." +
date1.Year.ToString() + "_" +

date1.Hour.ToString() + "_" + date1.Minute.ToString() + "_" +
date1.Second.ToString() +

                                ".xlsx";

    excelapp = new Microsoft.Office.Interop.Excel.Application();
    excelapp.Visible = false;

    excelapp.SheetsInNewWorkbook = 2; //кол-во листов в excel
    excelapp.Workbooks.Add(Type.Missing);
    Microsoft.Office.Interop.Excel.Range excelcells;

    //Получаем набор ссылок на объекты Workbook (на созданные
книги)
    var excelappworkbooks = excelapp.Workbooks;

    //Получаем ссылку на книгу 1 - нумерация от 1
    var excelappworkbook = excelappworkbooks[1];

    //НЕ Будем спрашивать разрешение на запись поверх
существующего документа
    excelapp.DisplayAlerts = false;

    excelsheets = excelappworkbook.Worksheets;
    //Получаем ссылку на лист 1
    excelworksheet =
(Microsoft.Office.Interop.Excel.Worksheet)excelsheets.get_Item(1);
    //Выделение группы ячеек
    excelcells = excelworksheet.get_Range("A2", "D2");
    excelcells.Merge(Type.Missing);
    excelcells.EntireRow.Font.Bold = true;
    excelcells.EntireRow.Font.Size = 14;
    excelcells.EntireRow.Font.Color = Color.Black;
    excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
    excelcells.Value2 = @"Дата: " + date1.Day.ToString() + "." +
date1.Month.ToString() + "." + date1.Year.ToString();

    excelcells = excelworksheet.get_Range("A4", "D4");
    excelcells.Merge(Type.Missing);
    //Задаем выравнивание по центру
    excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
    excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
    excelcells.EntireRow.Font.Bold = true;
    excelcells.EntireRow.Font.Size = 16;
    excelcells.EntireRow.Font.Color = Color.Black;
    excelcells.Value2 = @"Отчет 'Рецепты по заказу'";

    excelcells = excelworksheet.get_Range("A5", "D5");
    excelcells.Merge(Type.Missing);
    //Задаем выравнивание по центру
    excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

```

```

        excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 16;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.Value2 = ClassStr.TekNameZakaz;
        excelworksheet.Cells[5, FormClient.AbcToIndex("C")].WrapText
= true; //Если текст не умещается в одной ячейке, то будет перенос на
следующие строки в пределах одной ячейки

        //Выводим названия столбцов
        excelcells = excelworksheet.get_Range("A7", "D7");
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 14;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

        excelcells = excelworksheet.get_Range("A7", "A7");
        excelcells.Value2 = @"№";
        excelworksheet.Columns[1].ColumnWidth = 8;
        excelcells = excelworksheet.get_Range("B7", "B7");
        excelcells.Value2 = @"Информация";
        excelworksheet.Columns[2].ColumnWidth = 30;
        excelcells = excelworksheet.get_Range("C7", "C7");
        excelcells.Value2 = @"Файл";
        excelworksheet.Columns[4].ColumnWidth = 26;
        excelcells = excelworksheet.get_Range("D7", "D7");
        excelcells.Value2 = @"Клиент";
        excelworksheet.Columns[3].ColumnWidth = 22;

        MySqlCommand command = new MySqlCommand();
        command.CommandText = Tekzapros;
        DataTable tbl = new DataTable();
        using (MySqlDataAdapter dataAdapter = new
MySqlDataAdapter(command.CommandText, Query.ConnStr))
        {
            dataAdapter.Fill(tbl);
        }
        var j = 0;
        for (j = 0; j < tbl.Rows.Count; j++)
        {
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")] = (j + 1).ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")] = tbl.Rows[j][1].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки

```

```

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")] = tbl.Rows[j][2].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")] = tbl.Rows[j][3].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
    }
}

    Object NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("A")];
    excelcells = excelworksheet.get_Range("A7", NomCell);
    excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
    excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

    //Устанавливаем цвет обводки
    NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("D")];
    excelcells = excelworksheet.get_Range("A7", NomCell);
    excelcells.Borders.ColorIndex = 1; //черный
    //Устанавливаем стиль и толщину линии
    excelcells.Borders.LineStyle =
Microsoft.Office.Interop.Excel.XlLineStyle.xlContinuous;
    excelcells.Borders.Weight =
Microsoft.Office.Interop.Excel.XlBorderWeight.xlMedium;//.xlThick;

    excelapp.Visible = true;

    object misValue = System.Reflection.Missing.Value;
    excelappworkbook.SaveAs(@fileReport,
Microsoft.Office.Interop.Excel.XlFileFormat.xlOpenXMLWorkbook, misValue,
        misValue, misValue, misValue,
Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive,
        misValue, misValue, misValue,
misValue, misValue);
    this.Enabled = true;
}

//Кнопка - Просмотр скриншота рецепта
private void Knopka_Prosmotr_Click(object sender, EventArgs e)
{
    if (GridRecept.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
        return;
    }
}

```

```

        string path =
Path.GetDirectoryName(Application.ExecutablePath) + @"\\Рецепты\\" +

GridRecept.Rows[GridRecept.CurrentRow.Index].Cells["NameFile"].Value.ToSt
ring();

        Process.Start(path);
    }

}

```

Листинг E.29 – Класс FormReceptInfo

```

public partial class FormReceptInfo : Form
{
    Query Podkl; //Определим наш объект класса
    public string text_ID, PolnPathFile, TekNameFileRecept;
    public int RegimRecept, CheckRecept;

    public FormReceptInfo()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
подключения
        Podkl = new Query(Query.ConnStr);
    }

    //Кнопка Выход
    private void But_Exit_Click(object sender, EventArgs e)
    {
        CheckRecept = -1;
        Close();
    }

    //Кнопка Сохранить
    private void But_Save_Click(object sender, EventArgs e)
    {
        //Выполняем проверки заполненных полей данных
        if (DBEdit_Inform.Text == "")
        {
            MessageBox.Show("Введите информацию по рецепту");
            DBEdit_Inform.Focus();
            return;
        }

        if (DBEdit_File.Text == "")
        {
            MessageBox.Show("Укажите файл с рецептом");
            DBEdit_File.Focus();
            return;
        }

        //Сохраняем файл в папку с рецептами
        string path2 =
Path.GetDirectoryName(Application.ExecutablePath) + @"\\Рецепты\\" +

ClassStr.TekFioClient + " - " + DBEdit_File.Text;

```

```

File.Copy(PolnPathFile, path2, true);
//-----

string StrJson = "";

switch (RegimRecept)
{
    case 1: //Режим редактирования
        ClassRecept Per_1 = new
ClassRecept(int.Parse(text_ID.ToString()),

        DBEdit_Inform.Text,

        ClassStr.TekFioClient + " - " + DBEdit_File.Text,

        ClassStr.TekCodZakaz

        );
        StrJson = JsonConvert.SerializeObject(Per_1);
        Podkl.Update_JSON("Table_Recept", StrJson);
        break;
    case 2: //Режим создания
        ClassRecept Per_2 = new ClassRecept(0,

        DBEdit_Inform.Text,

        ClassStr.TekFioClient + " - " + DBEdit_File.Text,

        ClassStr.TekCodZakaz

        );
        StrJson = JsonConvert.SerializeObject(Per_2);
        CheckRecept = Podkl.Add_JSON("Table_Recept",
StrJson);
        break;
}

MessageBox.Show("Рецепт успешно сохранен");
Close();
}

//Обработчик загрузки формы
private void FormReceptInfo_Load(object sender, EventArgs e)
{
    DataTable bufferRecept = new DataTable();

    LabelInfo.Text = ClassStr.TekNameZakaz;

    switch (RegimRecept)
    {
        case 1: //Режим редактирования
            string zapros = "select
R.CodRecept,R.Inform,R.NameFile, ";
            zapros += " concat(CL.FamCl, '
', LEFT(CL.NameCl,1), '.', LEFT(CL.OtchCl,1), '.') as Client ";
            zapros += " from (Table_Recept as R ";
            zapros += " inner join (Table_Zakaz as ZK ";
            zapros += " inner join Table_Client as CL on
ZK.CodClient=CL.CodClient) ";
            zapros += " on R.CodZakaz=ZK.CodZakaz) ";

```

```

        zapros += " where R.CodRecept = ";
        zapros += text_ID;
        bufferRecept = Podkl.GetDataFromTable(zapros);

        //Заполняем поля для ввода информации
        DBEdit_Inform.Text =
bufferRecept.Rows[0]["Inform"].ToString();
        DBEdit_File.Text =
bufferRecept.Rows[0]["NameFile"].ToString();
        break;
    case 2: //Режим создания
        //Заполняем поля для ввода информации
        DBEdit_Inform.Text = "";
        DBEdit_File.Text = "";
        break;
    }
}

//Обработчик отображения формы
private void FormReceptInfo_Shown(object sender, EventArgs e)
{
    DBEdit_Inform.Focus();
}

private void buttonDeleteRecept_Click(object sender, EventArgs e)
{
    //OpenFile(sender, e);
    DBEdit_File.Text = "";
}

//Клавиши ESC и ENTER
private void DBEdit_Inform_KeyPress(object sender,
KeyPressEventArgs e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
    else if (e.KeyChar == (char)13)
    {
        But_Save_Click(sender, e);
    }
}

//Клавиша ESC
private void FormReceptInfo_KeyPress(object sender,
KeyPressEventArgs e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
}

//Кнопка - Выбрать файл
private void buttonRecept_Click(object sender, EventArgs e)
{
    try

```



```

        {
            OpenFileDialog dlg = new OpenFileDialog();
            dlg.InitialDirectory =
Path.GetDirectoryName(Application.ExecutablePath) + "/Рецепты/";
            //dlg.Filter = "JPG Files (*.jpg)|*.jpg";
            dlg.Filter = "Files (All Files (*.*)|*.*)";
            dlg.Title = "Выберите файл с рецептом";
            if (dlg.ShowDialog() == DialogResult.OK)
            {
                PolnPathFile = dlg.FileName.ToString();
                DBEdit_File.Text = dlg.SafeFileName;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Невозможно открыть выбранный файл",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```

Листинг Е.30 – Класс FormReportClientZakaz

```

public partial class FormReportClientZakaz : Form
{
    Query Podkl; //Определим наш объект класса
    public string Tekzapros, ZaprosClient, ZaprosZakaz;

    private Microsoft.Office.Interop.Excel.Application excelapp;
    private Microsoft.Office.Interop.Excel.Window excelWindow;
    private Microsoft.Office.Interop.Excel.Sheets excelsheets;
    private Microsoft.Office.Interop.Excel.Worksheet excelworksheet;

    public FormReportClientZakaz()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
        ZaprosClient = "select
CodClient,FamCl,NameCl,OtchCl,Phone,Document from Table_Client ";
        ZaprosClient += " order by FamCl,NameCl,OtchCl";

        ZaprosZakaz = "select Z.CodZakaz,Z.DataStart,Z.DataEnd,";
        ZaprosZakaz += "Z.Summa,ST.Name as Status,PN.Name as
Punkt,Z.Inform,";
        ZaprosZakaz += "concat(US.FamU, '
',LEFT(US.NameU,1),'.',LEFT(US.OtchU,1),'.') as Users ";
        ZaprosZakaz += " from ((Table_Zakaz as Z ";
        ZaprosZakaz += "inner join Table_Punkt as PN on Z.CodPunkt =
PN.CodPunkt) ";
        ZaprosZakaz += "inner join Table_Status as ST on Z.CodStatus =
ST.CodStatus) ";
        ZaprosZakaz += "inner join Table_Users as US on Z.CodUsers =
US.CodUsers) ";
    }
}

```

```

    }

    //Клавиша ESC
    private void FormReportClientZakaz_KeyPress(object sender,
KeyPressEventArgs e)
    {
        if (e.KeyChar == (char)27)
        {
            Exit_Click(sender, e);
        }
    }

    //Обработчик загрузки формы
    private void FormReportClientZakaz_Load(object sender, EventArgs e)
    {
        // задать цвет в заголовках столбцов
        GridClient.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
        // задать шрифт в заголовках столбцов
        Font F = new Font("Times New Roman", 14);
        GridClient.ColumnHeadersDefaultCellStyle.Font = F;

        Font F2 = new Font("Times New Roman", 12);
        GridClient.DefaultCellStyle.Font = F2;

        // задать цвет в заголовках столбцов
        GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
        // задать шрифт в заголовках столбцов
        Font F3 = new Font("Times New Roman", 14);
        GridZakaz.ColumnHeadersDefaultCellStyle.Font = F3;

        Font F4 = new Font("Times New Roman", 12);
        GridZakaz.DefaultCellStyle.Font = F4;
    }

    //Обработчик отображения формы
    private void FormReportClientZakaz_Shown(object sender, EventArgs
e)
    {
        GridClient.DataSource = Podkl.GetDataFromTable(ZaprosClient);
        GridClient.Columns[0].Visible = false;
        GridClient.Columns[1].Width = 140;
        GridClient.Columns[1].HeaderText = "Фамилия";
        GridClient.Columns[2].Width = 130;
        GridClient.Columns[2].HeaderText = "Имя";
        GridClient.Columns[3].Width = 130;
        GridClient.Columns[3].HeaderText = "Отчество";
        GridClient.Columns[4].Width = 150;
        GridClient.Columns[4].HeaderText = "Телефон";
        GridClient.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridClient.Columns[5].Width = 150;
        GridClient.Columns[5].HeaderText = "Паспорт";
        GridClient.Columns[5].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
    }

```

```

        GridClient.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridClient.DefaultCellStyle.SelectionBackColor = Color.White;
        GridClient.DefaultCellStyle.SelectionForeColor = Color.Blue;
        GridClient.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridClient.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridClient.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridClient.ColumnHeadersDefaultCellStyle.BackColor;
        GridClient.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridClient.ColumnHeadersDefaultCellStyle.ForeColor;

        GridClient.EnableHeadersVisualStyles = false;

        if (GridClient.Rows.Count > 0)
        {
            GridClient.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridClient.CurrentCell = GridClient.Rows[0].Cells[1];
            //ставим выделение на самую первую строку
            GridClient.Focus();
        }

        //-----
        MySqlDataAdapter datadapterZakaz = new
        MySqlDataAdapter(ZaprosZakaz, Query.ConnStr);
        DataTable bufferZakaz = new DataTable();
        datadapterZakaz.Fill(bufferZakaz); //вызываем метод Fill,
        чтобы после запроса заполнил буферную таблицу данными
        GridZakaz.DataSource = bufferZakaz;

        GridZakaz.Columns[0].Width = 60;
        GridZakaz.Columns[0].HeaderText = "Код";
        GridZakaz.Columns[0].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.Columns[1].Width = 90;
        GridZakaz.Columns[1].HeaderText = "Дата (Начало)";
        GridZakaz.Columns[1].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.Columns[2].Width = 90;
        GridZakaz.Columns[2].HeaderText = "Дата (Конец)";
        GridZakaz.Columns[2].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.Columns[3].Width = 80;
        GridZakaz.Columns[3].HeaderText = "Сумма";
        GridZakaz.Columns[3].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.Columns[4].Width = 120;
        GridZakaz.Columns[4].HeaderText = "Статус";
        GridZakaz.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.Columns[5].Width = 180;
        GridZakaz.Columns[5].HeaderText = "Пункт";
        GridZakaz.Columns[6].Width = 250;

```

```

        GridZakaz.Columns[6].HeaderText = "Информация";
        GridZakaz.Columns[7].Width = 130;
        GridZakaz.Columns[7].HeaderText = "Сотрудник";
        GridZakaz.Columns[7].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.DefaultCellStyle.SelectionBackColor = Color.White;
        GridZakaz.DefaultCellStyle.SelectionForeColor = Color.Blue;
        GridZakaz.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridZakaz.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridZakaz.ColumnHeadersDefaultCellStyle.BackColor;
        GridZakaz.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor;

        GridZakaz.EnableHeadersVisualStyles = false;

        if (GridZakaz.Rows.Count > 0)
        {
            GridZakaz.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridZakaz.CurrentCell = GridZakaz.Rows[0].Cells[1];
            //ставим выделение на самую первую строку
            GridZakaz.Focus();
        }

        Grid_CellClick_2();
    }

    //Обработчик левого щелчка мыши в таблице со статусами
    private void Grid_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        Grid_CellClick_2();
    }

    //Обработчик нажатия мышкой в таблице данных
    private void Grid_CellClick_2()
    {
        DataTable bufferZakaz = new DataTable();

        if (GridClient.Rows.Count < 2)
        {
            ZaprozZakaz = "select * from Table_Zakaz where
CodZakaz=0";

            MySqlDataAdapter datadapter = new
MySqlDataAdapter(ZaprozZakaz, Query.ConnStr);
            datadapter.Fill(bufferZakaz); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными
            GridZakaz.DataSource = bufferZakaz;
            return;
        }
    }

```

```

    }

    //-----Заполняем Grid - заказы
    ZaprozZakaz = "select Z.CodZakaz,Z.DataStart,Z.DataEnd,";
    ZaprozZakaz += "Z.Summa,ST.Name as Status,PN.Name as
Punkt,Z.Inform,";
    ZaprozZakaz += "concat(US.FamU,'
',LEFT(US.NameU,1),'.',LEFT(US.OtchU,1),'.') as Users ";
    ZaprozZakaz += " from ((Table_Zakaz as Z ";
    ZaprozZakaz += "inner join Table_Punkt as PN on Z.CodPunkt =
PN.CodPunkt) ";
    ZaprozZakaz += "inner join Table_Status as ST on Z.CodStatus =
ST.CodStatus) ";
    ZaprozZakaz += "inner join Table_Users as US on Z.CodUsers =
US.CodUsers) ";
    ZaprozZakaz += " where Z.CodClient = " +
GridClient.Rows[GridClient.CurrentRow.Index].Cells["CodClient"].Value.ToS
tring();
    ZaprozZakaz += " order by Z.DataStart desc";

    MySqlDataAdapter datadapterZakaz = new
MySqlDataAdapter(ZaprozZakaz, Query.ConnStr);
    datadapterZakaz.Fill(bufferZakaz); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными
    GridZakaz.DataSource = bufferZakaz;
    //-----
}

//Кнопка - Отчет
private void Knopka_Report_Click(object sender, EventArgs e)
{
    if (GridZakaz.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
        return;
    }

    this.Enabled = false;

    DateTime datel = DateTime.Now;
    var fileReport =
Path.GetDirectoryName(Application.ExecutablePath) +
"\\Отчеты\\ЗаказыКлиента " +

    GridClient.Rows[GridClient.CurrentRow.Index].Cells["FamCl"].Value.To
oString()+"_" +
                                " от " +
    datel.Day.ToString() + "." + datel.Month.ToString() + "." +
    datel.Year.ToString() + "_" +

    datel.Hour.ToString() + "_" + datel.Minute.ToString() + "_" +
    datel.Second.ToString() +

                                ".xlsx";

    excelapp = new Microsoft.Office.Interop.Excel.Application();
    excelapp.Visible = false;

```

```

excelapp.SheetsInNewWorkbook = 2; //кол-во листов в excel
excelapp.Workbooks.Add(Type.Missing);
Microsoft.Office.Interop.Excel.Range excelcells;

//Получаем набор ссылок на объекты Workbook (на созданные
книги)
var excelappworkbooks = excelapp.Workbooks;

//Получаем ссылку на книгу 1 - нумерация от 1
var excelappworkbook = excelappworkbooks[1];

//НЕ Будем спрашивать разрешение на запись поверх
существующего документа
excelapp.DisplayAlerts = false;

excelsheets = excelappworkbook.Worksheets;
//Получаем ссылку на лист 1
excelworksheet =
(Microsoft.Office.Interop.Excel.Worksheet)excelsheets.get_Item(1);
//Выделение группы ячеек
excelcells = excelworksheet.get_Range("A2", "H2");
excelcells.Merge(Type.Missing);
excelcells.EntireRow.Font.Bold = true;
excelcells.EntireRow.Font.Size = 14;
excelcells.EntireRow.Font.Color = Color.Black;
excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
excelcells.Value2 = @"Дата: " + date1.Day.ToString() + "." +
date1.Month.ToString() + "." + date1.Year.ToString();

excelcells = excelworksheet.get_Range("A4", "H4");
excelcells.Merge(Type.Missing);
//Задаем выравнивание по центру
excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
excelcells.EntireRow.Font.Bold = true;
excelcells.EntireRow.Font.Size = 16;
excelcells.EntireRow.Font.Color = Color.Black;
excelcells.Value2 = @"Отчет 'Информация по заказам клиента "+

GridClient.Rows[GridClient.CurrentRow.Index].Cells["FamC1"].Value.T
oString() + "'";

excelcells = excelworksheet.get_Range("A5", "H5");
excelcells.Merge(Type.Missing);
//Задаем выравнивание по центру
excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
excelcells.EntireRow.Font.Bold = true;
excelcells.EntireRow.Font.Size = 16;
excelcells.EntireRow.Font.Color = Color.Black;
excelcells.Value2 = "";

```

```

//Выводим названия столбцов
excelcells = excelworksheet.get_Range("A7", "H7");
excelcells.EntireRow.Font.Bold = true;
excelcells.EntireRow.Font.Size = 14;
excelcells.EntireRow.Font.Color = Color.Black;
excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

var j = 0;
    excelcells = excelworksheet.get_Range("A7", "A7");
    excelcells.Value2 = @"Код";
    excelworksheet.Columns[1].ColumnWidth = 7;
    excelcells = excelworksheet.get_Range("B7", "B7");
    excelcells.Value2 = @"Дата(начало)";
    excelworksheet.Columns[2].ColumnWidth = 15;
    excelcells = excelworksheet.get_Range("C7", "C7");
    excelcells.Value2 = @"Дата(конец)";
    excelworksheet.Columns[3].ColumnWidth = 14;
    excelcells = excelworksheet.get_Range("D7", "D7");
    excelcells.Value2 = @"Сумма";
    excelworksheet.Columns[4].ColumnWidth = 10;
    excelcells = excelworksheet.get_Range("E7", "E7");
    excelcells.Value2 = @"Статус";
    excelworksheet.Columns[5].ColumnWidth = 13;
    excelcells = excelworksheet.get_Range("F7", "F7");
    excelcells.Value2 = @"Пункт";
    excelworksheet.Columns[6].ColumnWidth = 16;
    excelcells = excelworksheet.get_Range("G7", "G7");
    excelcells.Value2 = @"Информация";
    excelworksheet.Columns[7].ColumnWidth = 28;
    excelcells = excelworksheet.get_Range("H7", "H7");
    excelcells.Value2 = @"Сотрудник";
    excelworksheet.Columns[8].ColumnWidth = 20;

    MySqlCommand command = new MySqlCommand();
    command.CommandText = Zproszakaz;
    DataTable tbl = new DataTable();
    using (MySqlDataAdapter dataAdapter = new
MySqlDataAdapter(command.CommandText, Query.ConnStr))
    {
        dataAdapter.Fill(tbl);
    }
    for (j = 0; j < tbl.Rows.Count; j++)
    {
        //Код
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")] = tbl.Rows[j][0].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

        //дата начала
        DateTime dt =
DateTime.Parse(tbl.Rows[j][1].ToString());
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")] = dt.ToString("MM-dd-yyyy");

```

```

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        //дата конца
        dt = DateTime.Parse(tbl.Rows[j][2].ToString());
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")] = dt.ToString("MM-dd-yyyy");
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        //сумма
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")] = tbl.Rows[j][3].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        //статус
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")] = tbl.Rows[j][4].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        //пункт
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("F")] = tbl.Rows[j][5].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("F")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("F")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        //информация
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("G")] = tbl.Rows[j][6].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("G")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("G")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlLeft;

        //сотрудник

```



```

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("H")] = tbl.Rows[j][7].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("H").WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("H").HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
    }

    //Object NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("A")];
    //excelcells = excelworksheet.get_Range("A7", NomCell);
    //excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
    //excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

    //Устанавливаем цвет обводки
    Object NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("H")];
    excelcells = excelworksheet.get_Range("A7", NomCell);
    excelcells.Borders.ColorIndex = 1; //черный
    //Устанавливаем стиль и толщину линии
    excelcells.Borders.LineStyle =
Microsoft.Office.Interop.Excel.XlLineStyle.xlContinuous;
    excelcells.Borders.Weight =
Microsoft.Office.Interop.Excel.XlBorderWeight.xlMedium;//.xlThick;

    excelapp.Visible = true;

    object misValue = System.Reflection.Missing.Value;
    excelappworkbook.SaveAs(@fileReport,
Microsoft.Office.Interop.Excel.XlFileFormat.xlOpenXMLWorkbook, misValue,
        misValue, misValue, misValue,
Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive,
        misValue, misValue, misValue,
misValue, misValue);
    this.Enabled = true;
}

//Обработчик кнопки Выход
private void Exit_Click(object sender, EventArgs e)
{
    Close();
}

//Обработчик в поле поиска по фамилии
private void TextBoxFam_TextChanged(object sender, EventArgs e)
{
    string zapros = "select
CodClient,FamCl,NameCl,OtchCl,Phone,Document from Table_Client ";

    if (textBoxFam.Text != "")
    {

```

```

        zapros += " where FamCl like '%" + textBoxFam.Text + "%'
order by FamCl,NameCl,OtchCl";
    }
    else
    {
        zapros += " order by FamCl,NameCl,OtchCl";
    }

    GridClient.DataSource = Podkl.GetDataFromTable(zapros);
    if (GridClient.Rows.Count > 0)
    {
        GridClient.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridClient.CurrentCell = GridClient.Rows[0].Cells[1];
//ставим выделение на самую первую строку
    }
    textBoxFam.Focus();

    Grid_CellClick_2();
}
}

```

Листинг Е.31 – Класс FormReportHistory

```

public partial class FormReportHistory : Form
{
    Query Podkl; //Определим наш объект класса
    public string Tekzapros;

    public FormReportHistory()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);

        Tekzapros = "select
H.CodHistory,H.Data,H.Inform,concat(ZK.CodZakaz,' (' ,ZK.Inform,')') ";
        Tekzapros += "from (Table_History as H ";
        Tekzapros += "inner join Table_Zakaz as ZK on H.CodZakaz =
ZK.CodZakaz) ";
        Tekzapros += " order by H.Data desc";
    }

    //Кнопка - Включить фильтр
    private void ButtonFilterON_Click(object sender, EventArgs e)
    {
        string zapros = "select
H.CodHistory,H.Data,H.Inform,concat(ZK.CodZakaz,'-',ZK.Inform) ";
        zapros += "from (Table_History as H ";
        zapros += "inner join Table_Zakaz as ZK on H.CodZakaz =
ZK.CodZakaz) ";
        zapros += "where ((H.Data between '" +
dateFilter1.Value.ToString("yyyy-MM-dd") + "' and '" +
dateFilter2.Value.ToString("yyyy-MM-dd") + "'))";
        zapros += " order by H.Data desc";
    }
}

```

```

        GridHistory.DataSource = Podkl.GetDataFromTable(zapros);
    }

    //Кнопка - Выключить фильтр
    private void ButtonFilterOFF_Click(object sender, EventArgs e)
    {
        GridHistory.DataSource = Podkl.GetDataFromTable(Tekzapros);
    }

    //Обработчик кнопки Выход
    private void But_Exit_Click(object sender, EventArgs e)
    {
        Close();
    }

    //Клавиша ESC
    private void GridHistory_KeyPress(object sender, KeyPressEventArgs
e)
    {
        if (e.KeyChar == (char)27)
        {
            But_Exit_Click(sender, e);
        }
    }

    //Обработчик загрузки формы
    private void FormHistory_Load(object sender, EventArgs e)
    {
        // задать цвет в заголовках столбцов
        GridHistory.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
        // задать шрифт в заголовках столбцов
        Font F = new Font("Times New Roman", 14);
        GridHistory.ColumnHeadersDefaultCellStyle.Font = F;

        Font F2 = new Font("Times New Roman", 12);
        GridHistory.DefaultCellStyle.Font = F2;
    }

    //Обработчик отображения формы
    private void FormHistory_Shown(object sender, EventArgs e)
    {
        GridHistory.DataSource = Podkl.GetDataFromTable(Tekzapros);
        GridHistory.Columns[0].Visible = false;
        GridHistory.Columns[1].Width = 90;
        GridHistory.Columns[1].HeaderText = "Дата";
        GridHistory.Columns[2].Width = 550;
        GridHistory.Columns[2].HeaderText = "Информация";
        GridHistory.Columns[3].Width = 230;
        GridHistory.Columns[3].HeaderText = "Заказ";
        GridHistory.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridHistory.DefaultCellStyle.SelectionBackColor =
Color.White;
        GridHistory.DefaultCellStyle.SelectionForeColor = Color.Blue;
        GridHistory.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
    }

```

```

        GridHistory.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridHistory.ColumnHeadersDefaultCellStyle.SelectionBackColor
= GridHistory.ColumnHeadersDefaultCellStyle.BackColor;
        GridHistory.ColumnHeadersDefaultCellStyle.SelectionForeColor
= GridHistory.ColumnHeadersDefaultCellStyle.ForeColor;

        GridHistory.EnableHeadersVisualStyles = false;

        if (GridHistory.Rows.Count > 0)
        {
            GridHistory.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridHistory.CurrentCell = GridHistory.Rows[0].Cells[1];
//ставим выделение на самую первую строку
            GridHistory.Focus();
        }
    }
}

```

Листинг Е.32 – Класс FormReportSupplierPreparat

```

public partial class FormReportSupplierPreparat : Form
{
    Query Podkl; //Определим наш объект класса
    public string Tekzapros, ZaprosSupplier, ZaprosPreparat;

    private Microsoft.Office.Interop.Excel.Application excelapp;
    private Microsoft.Office.Interop.Excel.Window excelWindow;
    private Microsoft.Office.Interop.Excel.Sheets excelsheets;
    private Microsoft.Office.Interop.Excel.Worksheet excelworksheet;

    public FormReportSupplierPreparat()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
        ZaprosSupplier = "select * from Table_Supplier order by Name";

        ZaprosPreparat = "select P.CodPreparat,P.Name,VP.Name as
ViewPreparat,P.Kol,";
        ZaprosPreparat += "P.Cena,P.Inform from (Table_Preparat as P
";
        ZaprosPreparat += "inner join Table_ViewPreparat as VP on
P.CodViewPreparat=VP.CodViewPreparat) ";
    }

    //Клавиша ESC
    private void FormReportSupplierPreparat_KeyPress(object sender,
KeyPressEventArgs e)
    {
        if (e.KeyChar == (char)27)
        {

```

```

        Exit_Click(sender, e);
    }
}

//Обработчик загрузки формы
private void FormReportSupplierPreparat_Load(object sender,
EventArgs e)
{
    // задать цвет в заголовках столбцов
    GridSupplier.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
    // задать шрифт в заголовках столбцов
    Font F = new Font("Times New Roman", 14);
    GridSupplier.ColumnHeadersDefaultCellStyle.Font = F;

    Font F2 = new Font("Times New Roman", 12);
    GridSupplier.DefaultCellStyle.Font = F2;

    // задать цвет в заголовках столбцов
    GridPreparat.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
    // задать шрифт в заголовках столбцов
    Font F3 = new Font("Times New Roman", 14);
    GridPreparat.ColumnHeadersDefaultCellStyle.Font = F3;

    Font F4 = new Font("Times New Roman", 12);
    GridPreparat.DefaultCellStyle.Font = F4;
}

//Обработчик отображения формы
private void FormReportSupplierPreparat_Shown(object sender,
EventArgs e)
{
    GridSupplier.DataSource =
Podkl.GetDataFromTable(ZaprosSupplier);
    GridSupplier.Columns[0].Visible = false;
    GridSupplier.Columns[1].Width = 180;
    GridSupplier.Columns[1].HeaderText = "Название";
    GridSupplier.Columns[2].Width = 130;
    GridSupplier.Columns[2].HeaderText = "Телефон";
    GridSupplier.Columns[2].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
    GridSupplier.Columns[3].Width = 180;
    GridSupplier.Columns[3].HeaderText = "Адрес";
    GridSupplier.Columns[4].Width = 110;
    GridSupplier.Columns[4].HeaderText = "Директор";
    GridSupplier.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
    GridSupplier.Columns[5].Width = 200;
    GridSupplier.Columns[5].HeaderText = "Информация";
    GridSupplier.Columns[5].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;
    GridSupplier.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
    GridSupplier.DefaultCellStyle.SelectionBackColor =
Color.White;
    GridSupplier.DefaultCellStyle.SelectionForeColor = Color.Blue;
}

```

```

        GridSupplier.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridSupplier.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridSupplier.ColumnHeadersDefaultCellStyle.SelectionBackColor
= GridSupplier.ColumnHeadersDefaultCellStyle.BackColor;
        GridSupplier.ColumnHeadersDefaultCellStyle.SelectionForeColor
= GridSupplier.ColumnHeadersDefaultCellStyle.ForeColor;

        GridSupplier.EnableHeadersVisualStyles = false;

        if (GridSupplier.Rows.Count > 0)
        {
            GridSupplier.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridSupplier.CurrentCell =
GridSupplier.Rows[0].Cells[1]; //ставим выделение на самую первую строку
            GridSupplier.Focus();
        }

        //-----
        MySqlDataAdapter datadapterPreparat = new
MySqlDataAdapter(ZaprosPreparat, Query.ConnStr);
        DataTable bufferPreparat = new DataTable();
        datadapterPreparat.Fill(bufferPreparat); //вызываем метод
Fill, чтобы после запроса заполнил буферную таблицу данными
        GridPreparat.DataSource = bufferPreparat;

        GridPreparat.Columns[0].Visible = false;
        GridPreparat.Columns[1].Width = 230;
        GridPreparat.Columns[1].HeaderText = "Название";
        GridPreparat.Columns[2].Width = 130;
        GridPreparat.Columns[2].HeaderText = "Вид";
        GridPreparat.Columns[2].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;
        GridPreparat.Columns[3].Width = 60;
        GridPreparat.Columns[3].HeaderText = "Кол";
        GridPreparat.Columns[3].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridPreparat.Columns[4].Width = 70;
        GridPreparat.Columns[4].HeaderText = "Цена";
        GridPreparat.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridPreparat.Columns[5].Width = 330;
        GridPreparat.Columns[5].HeaderText = "Информация";
        GridPreparat.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;
        GridPreparat.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridPreparat.DefaultCellStyle.SelectionBackColor =
Color.White;
        GridPreparat.DefaultCellStyle.SelectionForeColor = Color.Blue;
        GridPreparat.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;

```

```

        GridPreparat.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridPreparat.ColumnHeadersDefaultCellStyle.SelectionBackColor
= GridPreparat.ColumnHeadersDefaultCellStyle.BackColor;
        GridPreparat.ColumnHeadersDefaultCellStyle.SelectionForeColor
= GridPreparat.ColumnHeadersDefaultCellStyle.ForeColor;

        GridPreparat.EnableHeadersVisualStyles = false;

        if (GridPreparat.Rows.Count > 0)
        {
            GridPreparat.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridPreparat.CurrentCell =
GridPreparat.Rows[0].Cells[1]; //ставим выделение на самую первую строку
            GridPreparat.Focus();
        }

        Grid_CellClick_2();
    }

    //Обработчик левого щелчка мыши в таблице со статусами
    private void Grid_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        Grid_CellClick_2();
    }

    //Обработчик нажатия мышкой в таблице данных
    private void Grid_CellClick_2()
    {
        DataTable bufferPreparat = new DataTable();

        if (GridSupplier.Rows.Count < 2)
        {
            ZaproPreparat = "select P.CodPreparat,P.Name,VP.Name as
ViewPreparat,P.Kol,";
            ZaproPreparat += "P.Cena,P.Inform from (Table_Preparat
as P ";
            ZaproPreparat += "inner join Table_ViewPreparat as VP
on P.CodViewPreparat=VP.CodViewPreparat) ";
            ZaproPreparat += " where P.CodPreparat=0";

            MySqlDataAdapter datadapter = new
MySqlDataAdapter(ZaproPreparat, Query.ConnStr);
            datadapter.Fill(bufferPreparat); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными
            GridPreparat.DataSource = bufferPreparat;
            return;
        }

        //-----Заполняем Grid - препараты
        ZaproPreparat = "select P.CodPreparat,P.Name,VP.Name as
ViewPreparat,P.Kol,";

```

```

        ZaproPreparat += "P.Cena,P.Inform from (Table_Preparat as P
";
        ZaproPreparat += "inner join Table_ViewPreparat as VP on
P.CodViewPreparat=VP.CodViewPreparat) ";
        ZaproPreparat += " where P.CodSupplier = " +
GridSupplier.Rows[GridSupplier.CurrentRow.Index].Cells["CodSupplier"].Val
ue.ToString();
        ZaproPreparat += " order by P.Name";

        MySqlDataAdapter datadapterPreparat = new
MySqlDataAdapter(ZaproPreparat, Query.ConnStr);
        datadapterPreparat.Fill(bufferPreparat); //вызываем метод
Fill, чтобы после запроса заполнил буферную таблицу данными
        GridPreparat.DataSource = bufferPreparat;
        //-----
    }

    //Кнопка - Отчет
    private void Knopka_Report_Click(object sender, EventArgs e)
    {
        if (GridPreparat.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        this.Enabled = false;

        DateTime datel = DateTime.Now;
        var fileReport =
Path.GetDirectoryName(Application.ExecutablePath) +
"\Отчеты\ПрепаратыПоставщика " +

        GridSupplier.Rows[GridSupplier.CurrentRow.Index].Cells["Name"].Valu
e.ToString() + "_" +

        " от " +
        datel.Day.ToString() + "." + datel.Month.ToString() + "." +
        datel.Year.ToString() + "_" +

        datel.Hour.ToString() + "_" + datel.Minute.ToString() + "_" +
        datel.Second.ToString() +

        ".xlsx";

        excelapp = new Microsoft.Office.Interop.Excel.Application();
        excelapp.Visible = false;

        excelapp.SheetsInNewWorkbook = 2; //кол-во листов в excel
        excelapp.Workbooks.Add(Type.Missing);
        Microsoft.Office.Interop.Excel.Range excelcells;

        //Получаем набор ссылок на объекты Workbook (на созданные
книги)
        var excelappworkbooks = excelapp.Workbooks;

        //Получаем ссылку на книгу 1 - нумерация от 1
        var excelappworkbook = excelappworkbooks[1];

```



```

        //НЕ Будем спрашивать разрешение на запись поверх
        существующего документа
        excelapp.DisplayAlerts = false;

        excelsheets = excelappworkbook.Worksheets;
        //Получаем ссылку на лист 1
        excelworksheet =
        (Microsoft.Office.Interop.Excel.Worksheet)excelsheets.get_Item(1);
        //Выделение группы ячеек
        excelcells = excelworksheet.get_Range("A2", "E2");
        excelcells.Merge(Type.Missing);
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 14;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.HorizontalAlignment =
        Microsoft.Office.Interop.Excel.Constants.xlRight;
        excelcells.Value2 = @"Дата: " + date1.Day.ToString() + "." +
        date1.Month.ToString() + "." + date1.Year.ToString();

        excelcells = excelworksheet.get_Range("A4", "E4");
        excelcells.Merge(Type.Missing);
        //Задаем выравнивание по центру
        excelcells.HorizontalAlignment =
        Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.VerticalAlignment =
        Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 16;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.Value2 = @"Отчет 'Информация по препаратам
        поставщика " +

        GridSupplier.Rows[GridSupplier.CurrentRow.Index].Cells["Name"].Value.
        ToString() + "'";

        excelcells = excelworksheet.get_Range("A5", "E5");
        excelcells.Merge(Type.Missing);
        //Задаем выравнивание по центру
        excelcells.HorizontalAlignment =
        Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.VerticalAlignment =
        Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 16;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.Value2 = "";

        //Выводим названия столбцов
        excelcells = excelworksheet.get_Range("A7", "E7");
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 14;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.HorizontalAlignment =
        Microsoft.Office.Interop.Excel.Constants.xlCenter;

        var j = 0;
        excelcells = excelworksheet.get_Range("A7", "A7");

```

```

excelcells.Value2 = @"Название";
excelworksheet.Columns[1].ColumnWidth = 25;
excelcells = excelworksheet.get_Range("B7", "B7");
excelcells.Value2 = @"Вид";
excelworksheet.Columns[2].ColumnWidth = 17;
excelcells = excelworksheet.get_Range("C7", "C7");
excelcells.Value2 = @"Кол";
excelworksheet.Columns[3].ColumnWidth = 6;
excelcells = excelworksheet.get_Range("D7", "D7");
excelcells.Value2 = @"Цена";
excelworksheet.Columns[4].ColumnWidth = 8;
excelcells = excelworksheet.get_Range("E7", "E7");
excelcells.Value2 = @"Информация";
excelworksheet.Columns[5].ColumnWidth = 35;

MySQLCommand command = new MySQLCommand();
command.CommandText = ZaprosPreparat;
DataTable tbl = new DataTable();
using (MySQLDataAdapter dataAdapter = new
MySQLDataAdapter(command.CommandText, Query.ConnStr))
{
    dataAdapter.Fill(tbl);
}
for (j = 0; j < tbl.Rows.Count; j++)
{
    //Название
    excelworksheet.Cells[j + 8, FormClient.AbcToIndex("A")]
= tbl.Rows[j][1].ToString();
    excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
    excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlLeft;

    //Вид
    excelworksheet.Cells[j + 8, FormClient.AbcToIndex("B")]
= tbl.Rows[j][2].ToString();
    excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
    excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

    //Кол
    excelworksheet.Cells[j + 8, FormClient.AbcToIndex("C")]
= tbl.Rows[j][3].ToString();
    excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

    //сумма
    excelworksheet.Cells[j + 8, FormClient.AbcToIndex("D")]
= tbl.Rows[j][4].ToString();

```

```

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        //информация
        excelworksheet.Cells[j + 8, FormClient.AbcToIndex("E")]
= tbl.Rows[j][5].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlLeft;
    }

    //Object NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("A")];
    //excelcells = excelworksheet.get_Range("A7", NomCell);

    //Устанавливаем цвет обводки
    Object NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("E")];
    excelcells = excelworksheet.get_Range("A7", NomCell);
    excelcells.Borders.ColorIndex = 1; //черный
    //Устанавливаем стиль и толщину линии
    excelcells.Borders.LineStyle =
Microsoft.Office.Interop.Excel.XlLineStyle.xlContinuous;
    excelcells.Borders.Weight =
Microsoft.Office.Interop.Excel.XlBorderWeight.xlMedium;//.xlThick;

    excelapp.Visible = true;

    object misValue = System.Reflection.Missing.Value;
    excelappworkbook.SaveAs(@fileReport,
Microsoft.Office.Interop.Excel.XlFileFormat.xlOpenXMLWorkbook, misValue,
        misValue, misValue, misValue,
Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive,
        misValue, misValue, misValue,
misValue, misValue);
    this.Enabled = true;
}

//Обработчик кнопки Выход
private void Exit_Click(object sender, EventArgs e)
{
    Close();
}

//Обработчик в поле поиска
private void TextBoxName_TextChanged(object sender, EventArgs e)
{
    string zapros = "select * from Table_Supplier ";

    if (textBoxName.Text != "")
    {

```

```

        zapros += "where Name like '%" + textBoxName.Text + "%'
order by Name";
    }
    else
    {
        zapros += "order by Name";
    }

    GridSupplier.DataSource = Podkl.GetDataFromTable(zapros);
    if (GridSupplier.Rows.Count > 0)
    {
        GridSupplier.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridSupplier.CurrentCell =
GridSupplier.Rows[0].Cells[1]; //ставим выделение на самую первую строку
    }
    textBoxName.Focus();

    Grid_CellClick_2();
}
}

```

Листинг Е.33 – Класс FormReportUsersZakaz

```

public partial class FormReportUsersZakaz : Form
{
    Query Podkl; //Определим наш объект класса
    public string Tekzapros, ZaprosUsers, ZaprosZakaz;

    private Microsoft.Office.Interop.Excel.Application excelapp;
    private Microsoft.Office.Interop.Excel.Window excelWindow;
    private Microsoft.Office.Interop.Excel.Sheets excelsheets;
    private Microsoft.Office.Interop.Excel.Worksheet excelworksheet;

    public FormReportUsersZakaz()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
        ZaprosUsers = "select
U.CodUsers,U.FamU,U.NameU,U.OtchU,DL.Name as Dolzhnost from (Table_Users
as U ";
        ZaprosUsers += " inner join Table_Dolzhnost as DL on
U.CodDolzhnost=DL.CodDolzhnost) ";

        ZaprosZakaz = "select Z.CodZakaz,Z.DataStart,Z.DataEnd,";
        ZaprosZakaz += "Z.Summa,ST.Name as Status,PN.Name as
Punkt,Z.Inform,";
        ZaprosZakaz += "concat(CL.FamCl,'
',LEFT(CL.NameCl,1),'.',LEFT(CL.OtchCl,1),'.') as Client ";
        ZaprosZakaz += " from ((Table_Zakaz as Z ";
        ZaprosZakaz += "inner join Table_Punkt as PN on Z.CodPunkt =
PN.CodPunkt) ";
        ZaprosZakaz += "inner join Table_Status as ST on Z.CodStatus =
ST.CodStatus) ";
    }
}

```

```

        ZaprozZakaz += "inner join Table_Client as CL on Z.CodClient =
        CL.CodClient) ";
    }

    //Клавиша ESC
    private void FormReportUsersZakaz_KeyPress(object sender,
    KeyPressEventArgs e)
    {
        if (e.KeyChar == (char)27)
        {
            Exit_Click(sender, e);
        }
    }

    //Обработчик загрузки формы
    private void FormReportUsersZakaz_Load(object sender, EventArgs e)
    {
        // задать цвет в заголовках столбцов
        GridUsers.ColumnHeadersDefaultCellStyle.ForeColor =
        Color.Purple;
        // задать шрифт в заголовках столбцов
        Font F = new Font("Times New Roman", 14);
        GridUsers.ColumnHeadersDefaultCellStyle.Font = F;

        Font F2 = new Font("Times New Roman", 12);
        GridUsers.DefaultCellStyle.Font = F2;

        // задать цвет в заголовках столбцов
        GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
        Color.Purple;
        // задать шрифт в заголовках столбцов
        Font F3 = new Font("Times New Roman", 14);
        GridZakaz.ColumnHeadersDefaultCellStyle.Font = F3;

        Font F4 = new Font("Times New Roman", 12);
        GridZakaz.DefaultCellStyle.Font = F4;
    }

    //Обработчик отображения формы
    private void FormReportUsersZakaz_Shown(object sender, EventArgs e)
    {
        GridUsers.DataSource = Podkl.GetDataFromTable(ZaprozUsers);
        GridUsers.Columns[0].Visible = false;
        GridUsers.Columns[1].Width = 190;
        GridUsers.Columns[1].HeaderText = "Фамилия";
        GridUsers.Columns[2].Width = 150;
        GridUsers.Columns[2].HeaderText = "Имя";
        GridUsers.Columns[3].Width = 160;
        GridUsers.Columns[3].HeaderText = "Отчество";
        GridUsers.Columns[4].Width = 200;
        GridUsers.Columns[4].HeaderText = "Должность";
        GridUsers.Columns[4].DefaultCellStyle.Alignment =
        DataGridViewContentAlignment.MiddleRight;
        GridUsers.ColumnHeadersDefaultCellStyle.Alignment =
        DataGridViewContentAlignment.MiddleCenter;
        GridUsers.DefaultCellStyle.SelectionBackColor = Color.White;
        GridUsers.DefaultCellStyle.SelectionForeColor = Color.Blue;
    }

```

```

        GridUsers.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridUsers.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridUsers.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridUsers.ColumnHeadersDefaultCellStyle.BackColor;
        GridUsers.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridUsers.ColumnHeadersDefaultCellStyle.ForeColor;

        GridUsers.EnableHeadersVisualStyles = false;

        if (GridUsers.Rows.Count > 0)
        {
            GridUsers.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridUsers.CurrentCell = GridUsers.Rows[0].Cells[1];
            //ставим выделение на самую первую строку
            GridUsers.Focus();
        }

        //-----
        MySqlDataAdapter datadapterZakaz = new
MySqlDataAdapter(ZaprosZakaz, Query.ConnStr);
        DataTable bufferZakaz = new DataTable();
        datadapterZakaz.Fill(bufferZakaz); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными
        GridZakaz.DataSource = bufferZakaz;

        GridZakaz.Columns[0].Width = 60;
        GridZakaz.Columns[0].HeaderText = "Код";
        GridZakaz.Columns[0].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.Columns[1].Width = 90;
        GridZakaz.Columns[1].HeaderText = "Дата (Начало)";
        GridZakaz.Columns[1].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.Columns[2].Width = 90;
        GridZakaz.Columns[2].HeaderText = "Дата (Конец)";
        GridZakaz.Columns[2].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.Columns[3].Width = 80;
        GridZakaz.Columns[3].HeaderText = "Сумма";
        GridZakaz.Columns[3].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.Columns[4].Width = 120;
        GridZakaz.Columns[4].HeaderText = "Статус";
        GridZakaz.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.Columns[5].Width = 180;
        GridZakaz.Columns[5].HeaderText = "Пункт";
        GridZakaz.Columns[6].Width = 250;
        GridZakaz.Columns[6].HeaderText = "Информация";
        GridZakaz.Columns[7].Width = 130;
        GridZakaz.Columns[7].HeaderText = "Клиент";

```

```

        GridZakaz.Columns[7].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.DefaultCellStyle.SelectionBackColor = Color.White;
        GridZakaz.DefaultCellStyle.SelectionForeColor = Color.Blue;
        GridZakaz.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridZakaz.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridZakaz.ColumnHeadersDefaultCellStyle.BackColor;
        GridZakaz.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor;

        GridZakaz.EnableHeadersVisualStyles = false;

        if (GridZakaz.Rows.Count > 0)
        {
            GridZakaz.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridZakaz.CurrentCell = GridZakaz.Rows[0].Cells[1];
            //ставим выделение на самую первую строку
            GridZakaz.Focus();
        }

        Grid_CellClick_2();
    }

    //Обработчик левого щелчка мыши в таблице со статусами
    private void Grid_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        Grid_CellClick_2();
    }

    //Обработчик нажатия мышкой в таблице данных
    private void Grid_CellClick_2()
    {
        DataTable bufferZakaz = new DataTable();

        if (GridUsers.Rows.Count < 2)
        {
            ZaprozZakaz = "select * from Table_Zakaz where
CodZakaz=0";

            MySqlDataAdapter datadapter = new
MySqlDataAdapter(ZaprozZakaz, Query.ConnStr);
            datadapter.Fill(bufferZakaz); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными
            GridZakaz.DataSource = bufferZakaz;
            return;
        }

        //-----Заполняем Grid - заказы

```

```

        ZaprozZakaz = "select Z.CodZakaz,Z.DataStart,Z.DataEnd,";
        ZaprozZakaz += "Z.Summa,ST.Name as Status,PN.Name as
Punkt,Z.Inform,";
        ZaprozZakaz += "concat(CL.FamCl,'
',LEFT(CL.NameCl,1),'.',LEFT(CL.OtchCl,1),'.') as Client ";
        ZaprozZakaz += " from ((Table_Zakaz as Z ";
        ZaprozZakaz += "inner join Table_Punkt as PN on Z.CodPunkt =
PN.CodPunkt) ";
        ZaprozZakaz += "inner join Table_Status as ST on Z.CodStatus =
ST.CodStatus) ";
        ZaprozZakaz += "inner join Table_Client as CL on Z.CodClient =
CL.CodClient) ";
        ZaprozZakaz += " where Z.CodUsers = " +
GridUsers.Rows[GridUsers.CurrentRow.Index].Cells["CodUsers"].Value.ToStri
ng();
        ZaprozZakaz += " order by Z.DataStart desc";

        MySqlDataAdapter datadapterZakaz = new
MySqlDataAdapter(ZaprozZakaz, Query.ConnStr);
        datadapterZakaz.Fill(bufferZakaz); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными
        GridZakaz.DataSource = bufferZakaz;
        //-----
    }

    //Кнопка - Отчет
    private void Knopka_Report_Click(object sender, EventArgs e)
    {
        if (GridZakaz.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        this.Enabled = false;

        DateTime datel = DateTime.Now;
        var fileReport =
Path.GetDirectoryName(Application.ExecutablePath) +
"\\Отчеты\\ЗаказыСотрудника " +

        GridUsers.Rows[GridUsers.CurrentRow.Index].Cells["FamU"].Value.ToSt
ring() + "_" +
                                " от " +
        datel.Day.ToString() + "." + datel.Month.ToString() + "." +
        datel.Year.ToString() + "_" +

        datel.Hour.ToString() + "_" + datel.Minute.ToString() + "_" +
        datel.Second.ToString() +
                                ".xlsx";

        excelapp = new Microsoft.Office.Interop.Excel.Application();
        excelapp.Visible = false;

        excelapp.SheetsInNewWorkbook = 2; //кол-во листов в excel
        excelapp.Workbooks.Add(Type.Missing);
        Microsoft.Office.Interop.Excel.Range excelcells;

```



```

//Получаем набор ссылок на объекты Workbook (на созданные
книги)
var excelappworkbooks = excelapp.Workbooks;

//Получаем ссылку на книгу 1 - нумерация от 1
var excelappworkbook = excelappworkbooks[1];

//НЕ Будем спрашивать разрешение на запись поверх
существующего документа
excelapp.DisplayAlerts = false;

excelsheets = excelappworkbook.Worksheets;
//Получаем ссылку на лист 1
excelworksheet =
(Microsoft.Office.Interop.Excel.Worksheet)excelsheets.get_Item(1);
//Выделение группы ячеек
excelcells = excelworksheet.get_Range("A2", "H2");
excelcells.Merge(Type.Missing);
excelcells.EntireRow.Font.Bold = true;
excelcells.EntireRow.Font.Size = 14;
excelcells.EntireRow.Font.Color = Color.Black;
excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
excelcells.Value2 = @"Дата: " + datel.Day.ToString() + "." +
datel.Month.ToString() + "." + datel.Year.ToString();

excelcells = excelworksheet.get_Range("A4", "H4");
excelcells.Merge(Type.Missing);
//Задаем выравнивание по центру
excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
excelcells.EntireRow.Font.Bold = true;
excelcells.EntireRow.Font.Size = 16;
excelcells.EntireRow.Font.Color = Color.Black;
excelcells.Value2 = @"Отчет 'Информация по заказам сотрудника
" +

GridUsers.Rows[GridUsers.CurrentRow.Index].Cells["FamU"].Value.ToSt
ring() + "'";

excelcells = excelworksheet.get_Range("A5", "H5");
excelcells.Merge(Type.Missing);
//Задаем выравнивание по центру
excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
excelcells.EntireRow.Font.Bold = true;
excelcells.EntireRow.Font.Size = 16;
excelcells.EntireRow.Font.Color = Color.Black;
excelcells.Value2 = "";

//Выводим названия столбцов
excelcells = excelworksheet.get_Range("A7", "H7");

```

```

        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 14;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

        var j = 0;
        excelcells = excelworksheet.get_Range("A7", "A7");
        excelcells.Value2 = @"Код";
        excelworksheet.Columns[1].ColumnWidth = 7;
        excelcells = excelworksheet.get_Range("B7", "B7");
        excelcells.Value2 = @"Дата(начало)";
        excelworksheet.Columns[2].ColumnWidth = 15;
        excelcells = excelworksheet.get_Range("C7", "C7");
        excelcells.Value2 = @"Дата(конец)";
        excelworksheet.Columns[3].ColumnWidth = 14;
        excelcells = excelworksheet.get_Range("D7", "D7");
        excelcells.Value2 = @"Сумма";
        excelworksheet.Columns[4].ColumnWidth = 10;
        excelcells = excelworksheet.get_Range("E7", "E7");
        excelcells.Value2 = @"Статус";
        excelworksheet.Columns[5].ColumnWidth = 13;
        excelcells = excelworksheet.get_Range("F7", "F7");
        excelcells.Value2 = @"Пункт";
        excelworksheet.Columns[6].ColumnWidth = 16;
        excelcells = excelworksheet.get_Range("G7", "G7");
        excelcells.Value2 = @"Информация";
        excelworksheet.Columns[7].ColumnWidth = 28;
        excelcells = excelworksheet.get_Range("H7", "H7");
        excelcells.Value2 = @"Клиент";
        excelworksheet.Columns[8].ColumnWidth = 20;

        MySqlCommand command = new MySqlCommand();
        command.CommandText = Zproszakaz;
        DataTable tbl = new DataTable();
        using (MySqlDataAdapter dataAdapter = new
        MySqlDataAdapter(command.CommandText, Query.ConnStr))
        {
            dataAdapter.Fill(tbl);
        }
        for (j = 0; j < tbl.Rows.Count; j++)
        {
            //Код
            excelworksheet.Cells[j + 8, FormClient.AbcToIndex("A")]
= tbl.Rows[j][0].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

            //дата начала
            DateTime dt = DateTime.Parse(tbl.Rows[j][1].ToString());
            excelworksheet.Cells[j + 8, FormClient.AbcToIndex("B")]
= dt.ToString("MM-dd-yyyy");
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

            //дата конца

```

```

        dt = DateTime.Parse(tbl.Rows[j][2].ToString());
        excelworksheet.Cells[j + 8, FormClient.AbcToIndex("C")]
= dt.ToString("MM-dd-yyyy");
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        //сумма
        excelworksheet.Cells[j + 8, FormClient.AbcToIndex("D")]
= tbl.Rows[j][3].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        //статус
        excelworksheet.Cells[j + 8, FormClient.AbcToIndex("E")]
= tbl.Rows[j][4].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        //пункт
        excelworksheet.Cells[j + 8, FormClient.AbcToIndex("F")]
= tbl.Rows[j][5].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("F")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("F")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        //информация
        excelworksheet.Cells[j + 8, FormClient.AbcToIndex("G")]
= tbl.Rows[j][6].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("G")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("G")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlLeft;

        //сотрудник
        excelworksheet.Cells[j + 8, FormClient.AbcToIndex("H")]
= tbl.Rows[j][7].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("H")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки

```

```

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("H")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
    }

    //Object NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("A")];
    //excelcells = excelworksheet.get_Range("A7", NomCell);
    //excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
    //excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

    //Устанавливаем цвет обводки
    Object NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("H")];
    excelcells = excelworksheet.get_Range("A7", NomCell);
    excelcells.Borders.ColorIndex = 1; //черный
    //Устанавливаем стиль и толщину линии
    excelcells.Borders.LineStyle =
Microsoft.Office.Interop.Excel.XlLineStyle.xlContinuous;
    excelcells.Borders.Weight =
Microsoft.Office.Interop.Excel.XlBorderWeight.xlMedium;//.xlThick;

    excelapp.Visible = true;

    object misValue = System.Reflection.Missing.Value;
    excelappworkbook.SaveAs(@fileReport,
Microsoft.Office.Interop.Excel.XlFileFormat.xlOpenXMLWorkbook, misValue,
                                                                    misValue, misValue, misValue,
Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive,
                                                                    misValue, misValue, misValue,
misValue, misValue);
    this.Enabled = true;
}

//Обработчик кнопки Выход
private void Exit_Click(object sender, EventArgs e)
{
    Close();
}

//Обработчик в поле поиска по фамилии
private void TextBoxFam_TextChanged(object sender, EventArgs e)
{
    string zapros = "select
U.CodUsers,U.FamU,U.NameU,U.OtchU,DL.Name as Dolzhnost from (Table_Users
as U ";
    zapros += " inner join Table_Dolzhnost as DL on
U.CodDolzhnost=DL.CodDolzhnost) ";

    if (textBoxFam.Text != "")
    {
        zapros += " where U.FamU like '%" + textBoxFam.Text +
"%" order by U.FamU,U.NameU,U.OtchU";
    }
    else

```

```

        {
            zapros += " order by U.FamU,U.NameU,U.OtchU";
        }

        GridUsers.DataSource = Podkl.GetDataFromTable(zapros);
        if (GridUsers.Rows.Count > 0)
        {
            GridUsers.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridUsers.CurrentCell = GridUsers.Rows[0].Cells[1];
//ставим выделение на самую первую строку
        }
        textBoxFam.Focus();

        Grid_CellClick_2();
    }
}

```

Листинг Е.34 – Класс FormReportViewPreparat

```

public partial class FormReportViewPreparat : Form
{
    Query Podkl; //Определим наш объект класса
    public string Tekzapros, ZaprosViewPreparat, ZaprosPreparat;

    private Microsoft.Office.Interop.Excel.Application excelapp;
    private Microsoft.Office.Interop.Excel.Window excelWindow;
    private Microsoft.Office.Interop.Excel.Sheets excelsheets;
    private Microsoft.Office.Interop.Excel.Worksheet excelworksheet;

    public FormReportViewPreparat()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
        ZaprosViewPreparat = "select * from Table_ViewPreparat order
        by Name";

        ZaprosPreparat = "select P.CodPreparat,P.Name,SP.Name as
        Supplier,P.Kol,";
        ZaprosPreparat += "P.Cena,P.Inform from (Table_Preparat as P
        ";
        ZaprosPreparat += "inner join Table_Supplier as SP on
        P.CodSupplier=SP.CodSupplier) ";
    }

    //Клавиша ESC
    private void FormReportViewPreparat_KeyPress(object sender,
    KeyPressEventArgs e)
    {
        if (e.KeyChar == (char)27)
        {
            Exit_Click(sender, e);
        }
    }
}

```

```

//Обработчик загрузки формы
private void FormReportViewPreparat_Load(object sender, EventArgs
e)
{
    // задать цвет в заголовках столбцов
    GridViewPreparat.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
    // задать шрифт в заголовках столбцов
    Font F = new Font("Times New Roman", 14);
    GridViewPreparat.ColumnHeadersDefaultCellStyle.Font = F;

    Font F2 = new Font("Times New Roman", 12);
    GridViewPreparat.DefaultCellStyle.Font = F2;

    // задать цвет в заголовках столбцов
    GridPreparat.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
    // задать шрифт в заголовках столбцов
    Font F3 = new Font("Times New Roman", 14);
    GridPreparat.ColumnHeadersDefaultCellStyle.Font = F3;

    Font F4 = new Font("Times New Roman", 12);
    GridPreparat.DefaultCellStyle.Font = F4;
}

//Обработчик отображения формы
private void FormReportViewPreparat_Shown(object sender, EventArgs
e)
{
    GridViewPreparat.DataSource =
Podkl.GetDataFromTable(ZaprosViewPreparat);
    GridViewPreparat.Columns[0].Visible = false;
    GridViewPreparat.Columns[1].Width = 440;
    GridViewPreparat.Columns[1].HeaderText = "Название";
    GridViewPreparat.Columns[1].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;
    GridViewPreparat.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
    GridViewPreparat.DefaultCellStyle.SelectionBackColor =
Color.White;
    GridViewPreparat.DefaultCellStyle.SelectionForeColor =
Color.Blue;
    GridViewPreparat.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
    GridViewPreparat.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

    //отменить выделение строки заголовков столбцов.
    //Будет отображаться цвет, который установлен для заголовков

    GridViewPreparat.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridViewPreparat.ColumnHeadersDefaultCellStyle.BackColor;

    GridViewPreparat.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridViewPreparat.ColumnHeadersDefaultCellStyle.ForeColor;

    GridViewPreparat.EnableHeadersVisualStyles = false;

```

```

        if (GridViewPreparat.Rows.Count > 0)
        {
            GridViewPreparat.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridViewPreparat.CurrentCell =
GridViewPreparat.Rows[0].Cells[1]; //ставим выделение на самую первую
строку
            GridViewPreparat.Focus();
        }

//-----
        MySqlDataAdapter datadapterPreparat = new
MySqlDataAdapter(ZaprosPreparat, Query.ConnStr);
        DataTable bufferPreparat = new DataTable();
        datadapterPreparat.Fill(bufferPreparat); //вызываем метод
Fill, чтобы после запроса заполнил буферную таблицу данными
        GridPreparat.DataSource = bufferPreparat;

        GridPreparat.Columns[0].Visible = false;
        GridPreparat.Columns[1].Width = 230;
        GridPreparat.Columns[1].HeaderText = "Название";
        GridPreparat.Columns[2].Width = 130;
        GridPreparat.Columns[2].HeaderText = "Поставщик";
        GridPreparat.Columns[2].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;
        GridPreparat.Columns[3].Width = 60;
        GridPreparat.Columns[3].HeaderText = "Кол";
        GridPreparat.Columns[3].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridPreparat.Columns[4].Width = 70;
        GridPreparat.Columns[4].HeaderText = "Цена";
        GridPreparat.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridPreparat.Columns[5].Width = 330;
        GridPreparat.Columns[5].HeaderText = "Информация";
        GridPreparat.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;
        GridPreparat.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridPreparat.DefaultCellStyle.SelectionBackColor =
Color.White;
        GridPreparat.DefaultCellStyle.SelectionForeColor = Color.Blue;
        GridPreparat.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridPreparat.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridPreparat.ColumnHeadersDefaultCellStyle.SelectionBackColor
= GridPreparat.ColumnHeadersDefaultCellStyle.BackColor;
        GridPreparat.ColumnHeadersDefaultCellStyle.SelectionForeColor
= GridPreparat.ColumnHeadersDefaultCellStyle.ForeColor;

        GridPreparat.EnableHeadersVisualStyles = false;

        if (GridPreparat.Rows.Count > 0)
        {

```

```

        GridPreparat.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridPreparat.CurrentCell =
GridPreparat.Rows[0].Cells[1]; //ставим выделение на самую первую строку
        GridPreparat.Focus();
    }

    Grid_CellClick_2();
}

//Обработчик левого щелчка мыши в таблице со статусами
private void Grid_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    Grid_CellClick_2();
}

//Обработчик нажатия мышкой в таблице данных
private void Grid_CellClick_2()
{
    DataTable bufferPreparat = new DataTable();

    if (GridViewPreparat.Rows.Count < 2)
    {
        ZaproPreparat = "select P.CodPreparat,P.Name,SP.Name as
Supplier,P.Kol,";
        ZaproPreparat += "P.Cena,P.Inform from (Table_Preparat
as P ";
        ZaproPreparat += "inner join Table_Supplier as SP on
P.CodSupplier=SP.CodSupplier) ";
        ZaproPreparat += " where P.CodPreparat=0";

        MySqlDataAdapter datadapter = new
MySqlDataAdapter(ZaproPreparat, Query.ConnStr);
        datadapter.Fill(bufferPreparat); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными
        GridPreparat.DataSource = bufferPreparat;
        return;
    }

    //-----Заполняем Grid - препараты
    ZaproPreparat = "select P.CodPreparat,P.Name,SP.Name as
Supplier,P.Kol,";
    ZaproPreparat += "P.Cena,P.Inform from (Table_Preparat as P
";
    ZaproPreparat += "inner join Table_Supplier as SP on
P.CodSupplier=SP.CodSupplier) ";
    ZaproPreparat += " where P.CodViewPreparat = " +
GridViewPreparat.Rows[GridViewPreparat.CurrentRow.Index].Cells["CodViewPr
eparat"].Value.ToString();
    ZaproPreparat += " order by P.Name";

    MySqlDataAdapter datadapterPreparat = new
MySqlDataAdapter(ZaproPreparat, Query.ConnStr);
    datadapterPreparat.Fill(bufferPreparat); //вызываем метод
Fill, чтобы после запроса заполнил буферную таблицу данными
    GridPreparat.DataSource = bufferPreparat;
    //-----

```



```

    }

    //Кнопка - Отчет
    private void Knopka_Report_Click(object sender, EventArgs e)
    {
        if (GridViewPreparat.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        this.Enabled = false;

        DateTime datel = DateTime.Now;
        var fileReport =
Path.GetDirectoryName(Application.ExecutablePath) +
"\Отчеты\ПрепаратыПоВиду " +

        GridViewPreparat.Rows[GridViewPreparat.CurrentRow.Index].Cells["Name"].Value.ToString() + "_" +

        " от " +
        datel.Day.ToString() + "." + datel.Month.ToString() + "." +
        datel.Year.ToString() + "_" +

        datel.Hour.ToString() + "_" + datel.Minute.ToString() + "_" +
        datel.Second.ToString() +

        ".xlsx";

        excelapp = new Microsoft.Office.Interop.Excel.Application();
        excelapp.Visible = false;

        excelapp.SheetsInNewWorkbook = 2; //кол-во листов в excel
        excelapp.Workbooks.Add(Type.Missing);
        Microsoft.Office.Interop.Excel.Range excelcells;

        //Получаем набор ссылок на объекты Workbook (на созданные
книги)
        var excelappworkbooks = excelapp.Workbooks;

        //Получаем ссылку на книгу 1 - нумерация от 1
        var excelappworkbook = excelappworkbooks[1];

        //НЕ Будем спрашивать разрешение на запись поверх
существующего документа
        excelapp.DisplayAlerts = false;

        excelsheets = excelappworkbook.Worksheets;
        //Получаем ссылку на лист 1
        excelworksheet =
(Microsoft.Office.Interop.Excel.Worksheet)excelsheets.get_Item(1);
        //Выделение группы ячеек
        excelcells = excelworksheet.get_Range("A2", "E2");
        excelcells.Merge(Type.Missing);
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 14;
        excelcells.EntireRow.Font.Color = Color.Black;
    }

```

```

        excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
        excelcells.Value2 = @"Дата: " + date1.Day.ToString() + "." +
date1.Month.ToString() + "." + date1.Year.ToString();

        excelcells = excelworksheet.get_Range("A4", "E4");
        excelcells.Merge(Type.Missing);
        //Задаем выравнивание по центру
        excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 16;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.Value2 = @"Отчет 'Информация по препаратам '" +

        GridViewPreparat.Rows[GridViewPreparat.CurrentRow.Index].Cells["Nam
e"].Value.ToString() + "'";

        excelcells = excelworksheet.get_Range("A5", "E5");
        excelcells.Merge(Type.Missing);
        //Задаем выравнивание по центру
        excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 16;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.Value2 = "";

        //Выводим названия столбцов
        excelcells = excelworksheet.get_Range("A7", "E7");
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 14;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

        var j = 0;
        excelcells = excelworksheet.get_Range("A7", "A7");
        excelcells.Value2 = @"Название";
        excelworksheet.Columns[1].ColumnWidth = 25;
        excelcells = excelworksheet.get_Range("B7", "B7");
        excelcells.Value2 = @"Поставщик";
        excelworksheet.Columns[2].ColumnWidth = 17;
        excelcells = excelworksheet.get_Range("C7", "C7");
        excelcells.Value2 = @"Кол";
        excelworksheet.Columns[3].ColumnWidth = 6;
        excelcells = excelworksheet.get_Range("D7", "D7");
        excelcells.Value2 = @"Цена";
        excelworksheet.Columns[4].ColumnWidth = 8;
        excelcells = excelworksheet.get_Range("E7", "E7");
        excelcells.Value2 = @"Информация";
        excelworksheet.Columns[5].ColumnWidth = 35;

        MySqlCommand command = new MySqlCommand();

```

```

        command.CommandText = ZaprosPreparat;
        DataTable tbl = new DataTable();
        using (MySqlDataAdapter dataAdapter = new
        MySqlDataAdapter(command.CommandText, Query.ConnStr))
        {
            dataAdapter.Fill(tbl);
        }
        for (j = 0; j < tbl.Rows.Count; j++)
        {
            //Название
            excelworksheet.Cells[j + 8, FormClient.AbcToIndex("A")]
= tbl.Rows[j][1].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlLeft;

            //Поставщик
            excelworksheet.Cells[j + 8, FormClient.AbcToIndex("B")]
= tbl.Rows[j][2].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

            //Кол
            excelworksheet.Cells[j + 8, FormClient.AbcToIndex("C")]
= tbl.Rows[j][3].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

            //сумма
            excelworksheet.Cells[j + 8, FormClient.AbcToIndex("D")]
= tbl.Rows[j][4].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

            //информация
            excelworksheet.Cells[j + 8, FormClient.AbcToIndex("E")]
= tbl.Rows[j][5].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlLeft;
        }
    }

```

```

        //Object NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("A")];
        //excelcells = excelworksheet.get_Range("A7", NomCell);

        //Устанавливаем цвет обводки
        Object NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("E")];
        excelcells = excelworksheet.get_Range("A7", NomCell);
        excelcells.Borders.ColorIndex = 1; //черный
        //Устанавливаем стиль и толщину линии
        excelcells.Borders.LineStyle =
Microsoft.Office.Interop.Excel.XlLineStyle.xlContinuous;
        excelcells.Borders.Weight =
Microsoft.Office.Interop.Excel.XlBorderWeight.xlMedium;//.xlThick;

        excelapp.Visible = true;

        object misValue = System.Reflection.Missing.Value;
        excelappworkbook.SaveAs(@fileReport,
Microsoft.Office.Interop.Excel.XlFileFormat.xlOpenXMLWorkbook, misValue,
                                                                misValue, misValue, misValue,
Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive,
                                                                misValue, misValue, misValue,
misValue, misValue);
        this.Enabled = true;
    }

    //Обработчик кнопки Выход
    private void Exit_Click(object sender, EventArgs e)
    {
        Close();
    }
}

```

Листинг Е.35 – Класс FormSostavZakaz

```

public partial class FormSostavZakaz : Form
{
    Query Podkl; //Определим наш объект класса
    public string ZaprosSostavZakaz, ZaprosSostavZakazLekarstvo,
ZaprosSostavZakazPreparat,
                                                                ZaprosZakaz, ZaprosZakazLekarstvo,
ZaprosZakazPreparat,
                                                                Zapros, ZaprosLekarstvo,
ZaprosPreparat;

    public FormSostavZakaz()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);

        ZaprosZakazLekarstvo = "select
Z.CodZakaz,Z.DataStart,Z.Summa,ST.Name as Status,PN.Name as Punkt,";

```

```

        ZaprozZakazLekarstvo += "concat(CL.FamCl, '
', LEFT(CL.NameCl,1), '.', LEFT(CL.OtchCl,1), '.') as Client";
        ZaprozZakazLekarstvo += " from (((Table_Zakaz as Z ";
        ZaprozZakazLekarstvo += "inner join Table_Client as CL on
Z.CodClient = CL.CodClient) ";
        ZaprozZakazLekarstvo += "inner join Table_Punkt as PN on
Z.CodPunkt = PN.CodPunkt) ";
        ZaprozZakazLekarstvo += "inner join Table_Status as ST on
Z.CodStatus = ST.CodStatus) ";
        ZaprozZakazLekarstvo += "where Z.CodZakaz = ";

        ZaprozZakazPreparat = "select
Z.CodZakaz,Z.DataStart,Z.Summa,ST.Name as Status,";
        ZaprozZakazPreparat += "concat(US.FamU, '
', LEFT(US.NameU,1), '.', LEFT(US.OtchU,1), '.') as Users ";
        ZaprozZakazPreparat += " from ((Table_Zakaz as Z ";
        ZaprozZakazPreparat += "inner join Table_Status as ST on
Z.CodStatus = ST.CodStatus) ";
        ZaprozZakazPreparat += "inner join Table_Users as US on
Z.CodUsers = US.CodUsers) ";
        ZaprozZakazPreparat += "where Z.CodZakaz = ";

        ZaprozLekarstvo = "select CodLekarstvo,Name,Kol,Cena from
Table_Lekarstvo ";

        ZaprozPreparat = "select P.CodPreparat,P.Name,VP.Name as
ViewPreparat,P.Kol,P.Cena,SP.Name as Supplier from ((Table_Preparat as P
";
        ZaprozPreparat += "left join Table_Supplier as SP on
P.CodSupplier = SP.CodSupplier) ";
        ZaprozPreparat += "left join Table_ViewPreparat as VP on
P.CodViewPreparat = VP.CodViewPreparat) ";

        ZaprozSostavZakazLekarstvo = "select
S.CodSostavZakaz,LK.Name,S.Kol,LK.Cena,S.CodLekarstvo from
(Table_SostavZakaz as S ";
        ZaprozSostavZakazLekarstvo += "inner join Table_Lekarstvo as
LK on S.CodLekarstvo = LK.CodLekarstvo) ";
        ZaprozSostavZakazLekarstvo += "where S.CodZakaz=";

        ZaprozSostavZakazPreparat = "select
S.CodSostavZakaz,PR.Name,VP.Name as
ViewPreparat,S.Kol,PR.Cena,S.CodPreparat from (Table_SostavZakaz as S ";
        ZaprozSostavZakazPreparat += "inner join (Table_Preparat as PR
inner join Table_ViewPreparat as VP on
PR.CodViewPreparat=VP.CodViewPreparat) on S.CodPreparat = PR.CodPreparat)
";
        ZaprozSostavZakazPreparat += "where S.CodZakaz=";
    }

    //Обработчик загрузки формы
    private void FormSostavZakaz_Load(object sender, EventArgs e)
    {

        if (ClassStr.RegimZakaz==1)
        {
            labelInfo_2.Text = "Лекарств.средства";
            ZaprozZakaz = ZaprozZakazLekarstvo;

```

```

        ZaproSostavZakaz = ZaproSostavZakazLekarstvo;
        ZaproS = ZaproLekarstvo;
    }
    else
    {
        labelInfo_2.Text = "Список препаратов";
        ZaproZakaz = ZaproZakazPreparat;
        ZaproSostavZakaz = ZaproSostavZakazPreparat;
        ZaproS = ZaproPreparat;
    }

    if (ClassStr.TekPrava == "Провизор")
    {
        But_AddInZakaz.Visible = true;
        But_DelFromZakaz.Visible = true;
        DBEdit_Kol.Visible = true;
        label4.Visible = true;
    }

    // задать цвет в заголовках столбцов
    GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
    // задать шрифт в заголовках столбцов
    Font F = new Font("Times New Roman", 14);
    GridZakaz.ColumnHeadersDefaultCellStyle.Font = F;
    Font F2 = new Font("Times New Roman", 12);
    GridZakaz.DefaultCellStyle.Font = F2;

    // задать цвет в заголовках столбцов
    GridSpisok.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Blue;
    // задать шрифт в заголовках столбцов
    GridSpisok.ColumnHeadersDefaultCellStyle.Font = F;
    GridSpisok.DefaultCellStyle.Font = F2;

    // задать цвет в заголовках столбцов
    GridSostavZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Blue;
    // задать шрифт в заголовках столбцов
    GridSostavZakaz.ColumnHeadersDefaultCellStyle.Font = F;
    GridSostavZakaz.DefaultCellStyle.Font = F2;

    ZaproZakaz += ClassStr.TekCodZakaz.ToString();
    GridZakaz.DataSource = Podkl.GetDataFromTable(ZaproZakaz);
    //GridZakaz.Columns[0].Visible = false;
    GridZakaz.Columns[0].Width = 70;
    GridZakaz.Columns[0].HeaderText = "Код";
    GridZakaz.Columns[0].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
    GridZakaz.Columns[1].Width = 110;
    GridZakaz.Columns[1].HeaderText = "Дата";
    GridZakaz.Columns[1].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
    GridZakaz.Columns[2].Width = 90;
    GridZakaz.Columns[2].HeaderText = "Сумма";
    GridZakaz.Columns[2].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;

```

```

GridZakaz.Columns[3].Width = 110;
GridZakaz.Columns[3].HeaderText = "Статус";
GridZakaz.Columns[3].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
if (ClassStr.RegimZakaz == 1)
{
    GridZakaz.Columns[4].Width = 120;
    GridZakaz.Columns[4].HeaderText = "Пункт";
    GridZakaz.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
    GridZakaz.Columns[5].Width = 170;
    GridZakaz.Columns[5].HeaderText = "Клиент";
    GridZakaz.Columns[5].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
}
else
{
    GridZakaz.Columns[4].Width = 300;
    GridZakaz.Columns[4].HeaderText = "Сотрудник";
    GridZakaz.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
}
//GridZakaz.Columns[5].Width = 150;
//GridZakaz.Columns[5].HeaderText = "Информация";
GridZakaz.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
GridZakaz.DefaultCellStyle.SelectionBackColor = Color.White;
GridZakaz.DefaultCellStyle.SelectionForeColor = Color.Blue;
GridZakaz.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

//отменить выделение строки заголовков столбцов.
//Будет отображаться цвет, который установлен для заголовков
GridZakaz.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridZakaz.ColumnHeadersDefaultCellStyle.BackColor;
GridZakaz.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor;

GridZakaz.EnableHeadersVisualStyles = false;

GridZakaz.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
GridZakaz.CurrentCell = GridZakaz.Rows[0].Cells[1]; //ставим
выделение на самую первую строку

//-----
//Формируем список продуктов (лекарства, препараты) в таблице
с продуктами (лекарства, препараты)
//Zapros += ClassStr.TekCodZakaz.ToString();
MySqlDataAdapter datadapterSpisok = new
MySqlDataAdapter(Zapros, Query.ConnStr);
DataTable bufferSpisok = new DataTable();
datadapterSpisok.Fill(bufferSpisok); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными
GridSpisok.DataSource = bufferSpisok;

```

```

GridSpisok.Columns[0].Visible = false;
if(ClassStr.RegimZakaz == 1)
{
    GridSpisok.Columns[1].Width = 270;
    GridSpisok.Columns[1].HeaderText = "Название";
    GridSpisok.Columns[2].Width = 80;
    GridSpisok.Columns[2].HeaderText = "Кол";
    GridSpisok.Columns[2].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
    GridSpisok.Columns[3].Width = 120;
    GridSpisok.Columns[3].HeaderText = "Цена";
    GridSpisok.Columns[3].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
}
else
{
    GridSpisok.Columns[1].Width = 140;
    GridSpisok.Columns[1].HeaderText = "Название";
    GridSpisok.Columns[2].Width = 100;
    GridSpisok.Columns[2].HeaderText = "Вид";
    GridSpisok.Columns[3].Width = 60;
    GridSpisok.Columns[3].HeaderText = "Кол";
    GridSpisok.Columns[3].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
    GridSpisok.Columns[4].Width = 70;
    GridSpisok.Columns[4].HeaderText = "Цена";
    GridSpisok.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
    GridSpisok.Columns[5].Width = 150;
    GridSpisok.Columns[5].HeaderText = "Поставщик";
}
GridSpisok.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
GridSpisok.DefaultCellStyle.SelectionBackColor = Color.White;
GridSpisok.DefaultCellStyle.SelectionForeColor = Color.Blue;
GridSpisok.ColumnHeadersDefaultCellStyle.BackColor =
Color.Cyan;
GridSpisok.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

//отменить выделение строки заголовков столбцов.
//Будет отображаться цвет, который установлен для заголовков
GridSpisok.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridSpisok.ColumnHeadersDefaultCellStyle.BackColor;
GridSpisok.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridSpisok.ColumnHeadersDefaultCellStyle.ForeColor;

GridSpisok.EnableHeadersVisualStyles = false;

GridSpisok.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
GridSpisok.CurrentCell = GridSpisok.Rows[0].Cells[1]; //ставим
выделение на самую первую строку

//-----
//Формируем список товаров в таблице с составом заказа

```



```

        ZaproSostavZakaz += ClassStr.TekCodZakaz.ToString();
        MySqlDataAdapter datadapterSostavZakaz = new
        MySqlDataAdapter(ZaproSostavZakaz, Query.ConnStr);
        DataTable bufferSostavZakaz = new DataTable();
        datadapterSostavZakaz.Fill(bufferSostavZakaz); //вызываем
        метод Fill, чтобы после запроса заполнил буферную таблицу данными
        GridSostavZakaz.DataSource = bufferSostavZakaz;

        GridSostavZakaz.Columns[0].Visible = false;
        if (ClassStr.RegimZakaz == 1)
        {
            GridSostavZakaz.Columns[1].Width = 260;
            GridSostavZakaz.Columns[1].HeaderText = "Название";
            GridSostavZakaz.Columns[2].Width = 60;
            GridSostavZakaz.Columns[2].HeaderText = "Кол";
            GridSostavZakaz.Columns[2].DefaultCellStyle.Alignment =
            DataGridViewContentAlignment.MiddleCenter;
            GridSostavZakaz.Columns[3].Width = 100;
            GridSostavZakaz.Columns[3].HeaderText = "Цена";
            GridSostavZakaz.Columns[3].DefaultCellStyle.Alignment =
            DataGridViewContentAlignment.MiddleRight;
            GridSostavZakaz.Columns[4].Visible = false;
        }
        else
        {
            GridSostavZakaz.Columns[1].Width = 180;
            GridSostavZakaz.Columns[1].HeaderText = "Название";
            GridSostavZakaz.Columns[2].Width = 120;
            GridSostavZakaz.Columns[2].HeaderText = "Вид";
            GridSostavZakaz.Columns[3].Width = 55;
            GridSostavZakaz.Columns[3].HeaderText = "Кол";
            GridSostavZakaz.Columns[3].DefaultCellStyle.Alignment =
            DataGridViewContentAlignment.MiddleCenter;
            GridSostavZakaz.Columns[4].Width = 70;
            GridSostavZakaz.Columns[4].HeaderText = "Цена";
            GridSostavZakaz.Columns[4].DefaultCellStyle.Alignment =
            DataGridViewContentAlignment.MiddleRight;
            //GridSostavZakaz.Columns[5].Width = 150;
            //GridSostavZakaz.Columns[5].HeaderText = "Поставщик";
            GridSostavZakaz.Columns[5].Visible = false;
        }
        GridSostavZakaz.ColumnHeadersDefaultCellStyle.Alignment =
        DataGridViewContentAlignment.MiddleCenter;
        GridSostavZakaz.DefaultCellStyle.SelectionBackColor =
        Color.White;
        GridSostavZakaz.DefaultCellStyle.SelectionForeColor =
        Color.Blue;
        GridSostavZakaz.ColumnHeadersDefaultCellStyle.BackColor =
        Color.Cyan;
        GridSostavZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
        Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для заголовков

        GridSostavZakaz.ColumnHeadersDefaultCellStyle.SelectionBackColor =
        GridSostavZakaz.ColumnHeadersDefaultCellStyle.BackColor;

```

```

GridSostavZakaz.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridSostavZakaz.ColumnHeadersDefaultCellStyle.ForeColor;

GridSostavZakaz.EnableHeadersVisualStyles = false;

GridSostavZakaz.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
GridSostavZakaz.CurrentCell =
GridSostavZakaz.Rows[0].Cells[1]; //ставим выделение на самую первую
строку
    }

//Обработчик кнопки Выход
private void But_Exit_Click(object sender, EventArgs e)
{
    Close();
}

private void DBEdit_Kol_KeyPress(object sender, KeyPressEventArg
e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
        return;
    }
    else if (e.KeyChar == (char)13)
    {
        But_AddInZakaz_Click(sender, e);
        return;
    }

    char ch = e.KeyChar;
    if (!char.IsDigit(ch) && ch != 8)
    {
        e.Handled = true;
    }
}

//Убрать из заказа
private void But_DelFromZakaz_Click(object sender, EventArgs e)
{
    if (DBEdit_Kol.Text == "")
    {
        MessageBox.Show("Укажите кол-во");
        DBEdit_Kol.Focus();
        return;
    }

    if (int.Parse(DBEdit_Kol.Text) <= 0)
    {
        MessageBox.Show("Укажите кол-во товара (цветов) больше
0");
        DBEdit_Kol.Focus();
        return;
    }
}

```

```

        if (GridSostavZakaz.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные в составе заказа");
            return;
        }

        if
        (int.Parse(GridSostavZakaz.Rows[GridSostavZakaz.CurrentRow.Index].Cells["
        Kol"].Value.ToString()) <
            int.Parse(DBEdit_Kol.Text.ToString()))
        {
            MessageBox.Show("Указанное кол-во превышает имеющее кол-
            во в составе заказа. Операция отменяется.");
            DBEdit_Kol.Focus();
            return;
        }

        if
        (int.Parse(GridSostavZakaz.Rows[GridSostavZakaz.CurrentRow.Index].Cells["
        Kol"].Value.ToString()) ==
            int.Parse(DBEdit_Kol.Text.ToString()))
        {
            Podkl.DeleteFromTable("Table_SostavZakaz",
            "CodSostavZakaz",
            int.Parse(GridSostavZakaz.Rows[GridSostavZakaz.CurrentRow.Index].Cells["C
            odSostavZakaz"].Value.ToString()));
        }
        else
        {
            string StrZ = "update Table_SostavZakaz set Kol=Kol-" +
            DBEdit_Kol.Text + " where CodSostavZakaz = " +

            GridSostavZakaz.Rows[GridSostavZakaz.CurrentRow.Index].Cells["CodSostavZa
            kaz"].Value.ToString();

            //Изменяем кол-во (уменьшаем) в списке состава заказа
            MySqlConnection connection;//создать подключение между
            нашим проектом и БД
            MySqlCommand command;        //нужен для выполнения
            операция вставки, редактирования и удаления данных БД
            connection = new MySqlConnection(Query.ConnStr);
            connection.Open();//Открыть соединение

            command = new MySqlCommand(StrZ, connection);
            command.ExecuteNonQuery(); //Выполняем запрос. В
            качестве результата возвращает кол-во обработанных строк
            connection.Close(); //Закрываем соединение
        }

        //Установить курсор на запись в списке, по которой произошло
        пополнение в связи удалением
        //из состава заказа
        int z = 0;
        if(ClassStr.RegimZakaz==1)
            z =
            int.Parse(GridSostavZakaz.Rows[GridSostavZakaz.CurrentRow.Index].Cells["C
            odLekarstvo"].Value.ToString());

```

```

else
    z =
int.Parse(GridSostavZakaz.Rows[GridSostavZakaz.CurrentRow.Index].Cells["CodPreparat"].Value.ToString());
    string searchValue = Convert.ToString(z);
    GridSpisok.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

    try
    {
        foreach (DataGridViewRow row in GridSpisok.Rows)
        {
            if
(row.Cells[0].Value.ToString().Equals(searchValue))
            {
                row.Selected = true;
                GridSpisok.CurrentCell =
GridSpisok.Rows[row.Index].Cells[1];
                break;
            }
        }
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
    //-----Конец установки курсора

    //-----изменяем кол-во в списке продуктов
    int Kol =
int.Parse(GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["Kol"].Value
.ToString()) +
        int.Parse(DBEdit_Kol.Text.ToString());

    string zapros = "";
    if (ClassStr.RegimZakaz == 1)
        zapros = "update Table_Lekarstvo set Kol=" +
Kol.ToString() +
            " where CodLekarstvo = " + z.ToString();
    else
        zapros = "update Table_Preparat set Kol=" +
Kol.ToString() +
            " where CodPreparat = " + z.ToString();
    MySqlConnection connection2; //создать подключение между нашим
проектом и БД
    MySqlCommand command2; //нужен для выполнения
операция вставки, редактирования и удаления данных БД
    connection2 = new MySqlConnection(Query.ConnStr);
    connection2.Open(); //Открыть соединение

    command2 = new MySqlCommand(zapros, connection2);
    command2.ExecuteNonQuery(); //Выполняем запрос. В качестве
результата возвращает кол-во обработанных строк
    connection2.Close(); //Закрываем соединение

    //Запоминаем позицию в DBGrid, чтобы после редактирования в
другом окне установить курсор на эту строку
    int iStr = GridSpisok.CurrentCell.RowIndex;

```

```

        //обновляем данные в таблице
        MySqlDataAdapter datadapterSpisok = new
        MySqlDataAdapter(Zapros, Query.ConnStr);
        DataTable bufferSpisok = new DataTable();
        datadapterSpisok.Fill(bufferSpisok); //вызываем метод Fill,
        чтобы после запроса заполнил буферную таблицу данными
        GridSpisok.DataSource = bufferSpisok;

        //Установить курсор на измененную запись
        GridSpisok.SelectionMode =
        DataGridViewSelectionMode.FullRowSelect;
        GridSpisok.CurrentCell = GridSpisok.Rows[iStr].Cells[1];
        //ставим выделение на измененную строку
        //-----Конец установки курсора

        MySqlDataAdapter datadapterSostavZakaz = new
        MySqlDataAdapter(ZaprosSostavZakaz, Query.ConnStr);
        DataTable bufferSostavZakaz = new DataTable();
        datadapterSostavZakaz.Fill(bufferSostavZakaz); //вызываем
        метод Fill, чтобы после запроса заполнил буферную таблицу данными
        GridSostavZakaz.DataSource = bufferSostavZakaz;

        DBEdit_Kol.Text = "";
        MessageBox.Show("Кол-во выбранной позиции успешно удалено из
        состава заказа");
    }

    //Добавить в заказ
    private void But_AddInZakaz_Click(object sender, EventArgs e)
    {
        if (GridSpisok.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        if (DBEdit_Kol.Text == "")
        {
            MessageBox.Show("Укажите кол-во ");
            DBEdit_Kol.Focus();
            return;
        }

        if (int.Parse(DBEdit_Kol.Text) <= 0)
        {
            MessageBox.Show("Укажите кол-во больше 0");
            DBEdit_Kol.Focus();
            return;
        }

        if (int.Parse(DBEdit_Kol.Text.ToString()) >

            int.Parse(GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["Kol"]
            .Value.ToString()))
        {

```

```

        MessageBox.Show("Указанное кол-во больше кол-ва в
наличие");
        DBEdit_Kol.Focus();
        return;
    }

    var cmd = new MySqlCommand(@"");
    if (ClassStr.RegimZakaz == 1)
        cmd = new MySqlCommand(@"SELECT CodSostavZakaz FROM
Table_SostavZakaz where (CodZakaz = @PCodZakaz)and(CodLekarstvo =
@PCodLekarstvo)");
    else
        cmd = new MySqlCommand(@"SELECT CodSostavZakaz FROM
Table_SostavZakaz where (CodZakaz = @PCodZakaz)and(CodPreparat =
@PCodPreparat)");
    cmd.Connection = new MySqlConnection(Query.ConnStr);
    cmd.Connection.Open();
    cmd.Parameters.AddWithValue("PCodZakaz",
ClassStr.TekCodZakaz);
    if (ClassStr.RegimZakaz == 1)
        cmd.Parameters.AddWithValue("PCodLekarstvo",
GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["CodLekarstvo"].Value)
;
    else
        cmd.Parameters.AddWithValue("PCodPreparat",
GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["CodPreparat"].Value);
    int CodSostavZakaz = Convert.ToInt32(cmd.ExecuteScalar());
    cmd.Connection.Close();
    if (CodSostavZakaz > 0)
    {
        cmd = new MySqlCommand("update Table_SostavZakaz set
Kol=Kol+" + DBEdit_Kol.Text + " where CodSostavZakaz = " +
                                CodSostavZakaz.ToString());
        cmd.Connection = new MySqlConnection(Query.ConnStr);
        cmd.Connection.Open();
        cmd.ExecuteScalar();
        cmd.Connection.Close();
    }
    else
    {
        ClassSostavZakaz Per_2 = new ClassSostavZakaz(0,

int.Parse(DBEdit_Kol.Text),

ClassStr.TekCodZakaz,

ClassStr.RegimZakaz == 1 ?
int.Parse(GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["CodLekarstv
o"].Value.ToString()):0,

ClassStr.RegimZakaz == 1 ?
0:int.Parse(GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["CodPrepar
at"].Value.ToString())

);
        string StrJson = JsonConvert.SerializeObject(Per_2);
        Podkl.Add_JSON("Table_SostavZakaz", StrJson);
    }
}

```

```

        MySqlDataAdapter datadapterSostavZakaz = new
MySqlDataAdapter(ZaprosSostavZakaz, Query.ConnStr);
        DataTable bufferSostavZakaz = new DataTable();
        datadapterSostavZakaz.Fill(bufferSostavZakaz); //вызываем
метод Fill, чтобы после запроса заполнил буферную таблицу данными
        GridSostavZakaz.DataSource = bufferSostavZakaz;

        //-----изменяем кол-во товара в списке продуктов
        int Kol =
int.Parse(GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["Kol"].Value
.ToString()) -
                int.Parse(DBEdit_Kol.Text.ToString());

        string zapros = "";
        if (ClassStr.RegimZakaz == 1)
            zapros = "update Table_Lekarstvo set Kol=" +
Kol.ToString() + " where CodLekarstvo = " +

GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["CodLekarstvo"].Value.
ToString();
        else
            zapros = "update Table_Preparat set Kol=" +
Kol.ToString() + " where CodPreparat = " +

GridSpisok.Rows[GridSpisok.CurrentRow.Index].Cells["CodPreparat"].Value.T
oString();
        MySqlConnection connection;//создать подключение между нашим
проектом и БД
        MySqlCommand command; //нужен для выполнения операция
вставки, редактирования и удаления данных БД
        connection = new MySqlConnection(Query.ConnStr);
        connection.Open();//Открыть соединение

        command = new MySqlCommand(zapros, connection);
        command.ExecuteNonQuery(); //Выполняем запрос. В качестве
результата возвращает кол-во обработанных строк
        connection.Close(); //Закрываем соединение

        //Запоминаем позицию в DBGrid, чтобы после редактирования в
другом окне установить курсор на эту строку
        int iStr = GridSpisok.CurrentCell.RowIndex;

        //обновляем данные в таблице
        MySqlDataAdapter datadapterSpisok = new
MySqlDataAdapter(Zapros, Query.ConnStr);
        DataTable bufferSpisok = new DataTable();
        datadapterSpisok.Fill(bufferSpisok); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными
        GridSpisok.DataSource = bufferSpisok;

        //Установить курсор на измененную запись
        GridSpisok.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridSpisok.CurrentCell = GridSpisok.Rows[iStr].Cells[1];
        //ставим выделение на измененную строку
        //-----Конец установки курсора

```

```

        DBEdit_Kol.Text = "";
        MessageBox.Show("Операция успешно выполнена по добавлению в
состав заказа");
    }

    private void GridSostavZakaz_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        But_DelFromZakaz_Click(sender, e);
    }

    private void GridSpisok_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        But_AddInZakaz_Click(sender, e);
    }

    //Клавиша ESC
    private void FormSostavZakaz_KeyPress(object sender,
KeyPressEventArgs e)
    {
        if (e.KeyChar == (char)27)
        {
            But_Exit_Click(sender, e);
        }
    }
}

```

Листинг Е.36 – Класс FormSpravochnik

```

public partial class FormSpravochnik : Form
{
    Query Podkl; //Определим наш объект класса
    int Regim = 0; //1- режим редактирования, 0 - режим создания
    public string Tekzapros;
    public string ZaprosDouble;
    public string TekTable;

    public FormSpravochnik()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
подключения
        Podkl = new Query(Query.ConnStr);
    }

    private void FormSpravochnik_Load(object sender, EventArgs e)
    {
        // задать цвет в заголовках столбцов
        Grid.ColumnHeadersDefaultCellStyle.ForeColor = Color.Purple;
        // задать шрифт в заголовках столбцов
        Font F = new Font("Times New Roman", 14);
        Grid.ColumnHeadersDefaultCellStyle.Font = F;

        Font F2 = new Font("Times New Roman", 12);
        Grid.DefaultCellStyle.Font = F2;
    }
}

```



```

private void FormSpravochnik_Shown(object sender, EventArgs e)
{
    switch (ClassStr.TekSpravochnik)
    {
        case 1://Виды препаратов
            Tekzapros = "select * from Table_ViewPreparat order
by Name";
            ZaproDouble = @"SELECT COUNT(*) FROM
Table_ViewPreparat where Name = @PName";
            TekTable = "Table_ViewPreparat";
            this.Text = "Справочник 'Виды препаратов'";
            break;
        case 2://Статусы
            Tekzapros = "select * from Table_Status order by
Name";
            ZaproDouble = @"SELECT COUNT(*) FROM Table_Status
where Name = @PName";
            TekTable = "Table_Status";
            this.Text = "Справочник 'Статусы'";
            break;
        case 3://Должности
            Tekzapros = "select * from Table_Dolzhnost order by
Name";
            ZaproDouble = @"SELECT COUNT(*) FROM
Table_Dolzhnost where Name = @PName";
            TekTable = "Table_Dolzhnost";
            this.Text = "Справочник 'Должности'";
            break;
    }

    Grid.DataSource = Podkl.GetDataFromTable(Tekzapros);
    Grid.Columns[0].Visible = false;
    Grid.Columns[1].Width = 420;
    Grid.Columns[1].HeaderText = "Названия";
    Grid.DefaultCellStyle.SelectionBackColor = Color.White;
    Grid.DefaultCellStyle.SelectionForeColor = Color.Blue;
    Grid.ColumnHeadersDefaultCellStyle.BackColor = Color.Yellow;
    Grid.ColumnHeadersDefaultCellStyle.ForeColor = Color.Black;

    //отменить выделение строки заголовков столбцов.
    //Будет отображаться цвет, который установлен для заголовков
    Grid.ColumnHeadersDefaultCellStyle.SelectionBackColor =
Grid.ColumnHeadersDefaultCellStyle.BackColor;
    Grid.ColumnHeadersDefaultCellStyle.SelectionForeColor =
Grid.ColumnHeadersDefaultCellStyle.ForeColor;

    Grid.EnableHeadersVisualStyles = false;

    Grid.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
    if (Grid.Rows.Count > 1)
    {
        Grid.CurrentCell = Grid.Rows[0].Cells[1]; //ставим
выделение на самую первую строку
    }

    Grid.Focus();
}

```

```

        private void FormSpravochnik_KeyPress(object sender,
        KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)27)
            {
                Exit_Click(sender, e);
            }
        }

        private void Grid_CellDoubleClick(object sender,
        DataGridViewCellEventArgs e)
        {
            if (Grid.Rows.Count < 2)
                return;
            DBEdit_Nazvanie.Text =
            Grid.Rows[Grid.CurrentRow.Index].Cells["Name"].Value.ToString();
            DBEdit_Nazvanie.Focus();
            Regim = 1; // Режим редактирования
        }

        private void DBEdit_Nazvanie_KeyPress(object sender,
        KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)27)
            {
                Exit_Click(sender, e);
            }
            else if (e.KeyChar == (char)13)
            {
                if (Regim == 0)
                    Knopka_Add_Click(sender, e);
                else
                    Knopka_Change_Click(sender, e);
            }
        }

        //Обработчик кнопки Добавить
        private void Knopka_Add_Click(object sender, EventArgs e)
        {
            if (DBEdit_Nazvanie.Text == "")
            {
                MessageBox.Show("Введите название");
                DBEdit_Nazvanie.Focus();
                return;
            }

            //Поиск дублей
            var cmd = new MySqlCommand(@ZaprosDouble);
            cmd.Connection = new MySqlConnection(Query.ConnStr);
            cmd.Connection.Open();
            cmd.Parameters.AddWithValue("PName", DBEdit_Nazvanie.Text);
            int totalRecords = Convert.ToInt32(cmd.ExecuteScalar());
            if (totalRecords > 0)
            {
                MessageBox.Show("Укажите другое название. Данное
                значение (" + DBEdit_Nazvanie.Text +

```

```

                                ") уже есть в таблице!",
    "ВНИМАНИЕ");
        DBEdit_Nazvanie.Focus();
        return;
    }
    //-----Конец поиска

    string StrJson = "";
    switch (ClassStr.TekSpravochnik)
    {
        case 1://Виды препаратов
            ClassViewPreparat Perl = new ClassViewPreparat(0,
DBEdit_Nazvanie.Text); ;
            StrJson = JsonConvert.SerializeObject(Perl);
            break;
        case 2://Статусы
            ClassStatus Per2 = new ClassStatus(0,
DBEdit_Nazvanie.Text); ;
            StrJson = JsonConvert.SerializeObject(Per2);
            break;
        case 3://Должности
            ClassDolzhnost Per3 = new ClassDolzhnost(0,
DBEdit_Nazvanie.Text); ;
            StrJson = JsonConvert.SerializeObject(Per3);
            break;
    }
    Podkl.Add_JSON(TekTable, StrJson);

    //Добавляем новую запись и обновляем данные в табличке
    Grid.DataSource = Podkl.GetDataFromTable(Tekzapros);

    //Установить курсор на добавленную запись
    string searchValue = DBEdit_Nazvanie.Text;
    Grid.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

    try
    {
        foreach (DataGridViewRow row in Grid.Rows)
        {
            if
DBEdit_Nazvanie.Text)
            {
                row.Selected = true;
                Grid.CurrentCell =
Grid.Rows[row.Index].Cells[1];
                break;
            }
        }
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
    //-----Конец установки курсора

    DBEdit_Nazvanie.Text = "";
    DBEdit_Nazvanie.Focus();
}

```

```

private void Knopka_Change_Click(object sender, EventArgs e)
{
    if (Grid.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
        return;
    }

    if (DBEdit_Nazvanie.Text == "")
    {
        MessageBox.Show("Введите название");
        DBEdit_Nazvanie.Focus();
        return;
    }

    string StrJson = "";
    switch (ClassStr.TekSpravochnik)
    {
        case 1://Виды препаратов
            ClassViewPreparat Perl = new
ClassViewPreparat(int.Parse(Grid.Rows[Grid.CurrentRow.Index].Cells["CodVi
ewPreparat"].Value.ToString()), DBEdit_Nazvanie.Text);
            StrJson = JsonConvert.SerializeObject(Perl);
            break;
        case 2://Статусы
            ClassStatus Per2 = new
ClassStatus(int.Parse(Grid.Rows[Grid.CurrentRow.Index].Cells["CodStatus"]
.Value.ToString()), DBEdit_Nazvanie.Text);
            StrJson = JsonConvert.SerializeObject(Per2);
            break;
        case 3://Должности
            ClassDolzhnost Per3 = new
ClassDolzhnost(int.Parse(Grid.Rows[Grid.CurrentRow.Index].Cells["CodDolzhh
nost"].Value.ToString()), DBEdit_Nazvanie.Text);
            StrJson = JsonConvert.SerializeObject(Per3);
            break;
    }
    Podkl.Update_JSON(TekTable, StrJson);
    Regim = 0;

    Grid.DataSource = Podkl.GetDataFromTable(Tekzapros);

    //Установить курсор на измененную запись
    string searchValue = DBEdit_Nazvanie.Text;
    Grid.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

    try
    {
        foreach (DataGridViewRow row in Grid.Rows)
        {
            if
(row.Cells[1].Value.ToString().Equals(searchValue))
            {
                row.Selected = true;
                Grid.CurrentCell =
Grid.Rows[row.Index].Cells[1];
                break;
            }
        }
    }
}

```

```

    }
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
    //-----Конец установки курсора

    DBEdit_Nazvanie.Text = "";
    DBEdit_Nazvanie.Focus();
}

private void Knopka_Delete_Click(object sender, EventArgs e)
{
    if (Grid.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
        return;
    }

    switch (ClassStr.TekSpravochnik)
    {
        case 1://Виды препаратов
            //Поиск связей удаляемой записи в таблице с
препаратами
            var cmd1 = new MySqlCommand("SELECT COUNT(*) FROM
Table_Preparat where CodViewPreparat = @PID_Cod");
            cmd1.Connection = new
MySQLConnection(Query.ConnStr);
            cmd1.Connection.Open();
            cmd1.Parameters.AddWithValue("PID_Cod",
int.Parse(Grid.Rows[Grid.CurrentRow.Index].Cells["CodViewPreparat"].Value
.ToString()));
            if (Convert.ToInt32(cmd1.ExecuteScalar()) > 0)
            {
                MessageBox.Show("Данную запись по виду
препарата удалить нельзя, потому что есть препараты с указанным видом!",
"ВНИМАНИЕ");
                return;
            }

            //Передаем индекс текущей записи в Grid
            Podkl.DeleteFromTable(TekTable, "CodViewPreparat",
int.Parse(Grid.Rows[Grid.CurrentRow.Index].Cells["CodViewPreparat"].Value
.ToString()));
            cmd1.Connection.Close();
            break;
        case 2://Статусы
            //Поиск связей удаляемой записи в таблице с
заказами
            var cmd2 = new MySqlCommand("SELECT COUNT(*) FROM
Table_Zakaz where CodStatus = @PID_Cod");
            cmd2.Connection = new
MySQLConnection(Query.ConnStr);
            cmd2.Connection.Open();

```

```

        cmd2.Parameters.AddWithValue("PID_Cod",
int.Parse(Grid.Rows[Grid.CurrentRow.Index].Cells["CodStatus"].Value.ToString()));
        if (Convert.ToInt32(cmd2.ExecuteScalar()) > 0)
        {
            MessageBox.Show("Данную запись по статусу
удалить нельзя, потому что есть данные по заказам с указанным статусом!",
"ВНИМАНИЕ");
            return;
        }

        //Передаем индекс текущей записи в Grid
        Podkl.DeleteFromTable(TekTable, "CodStatus",
int.Parse(Grid.Rows[Grid.CurrentRow.Index].Cells["CodStatus"].Value.ToString()));
        cmd2.Connection.Close();
        break;
    case 3://Должности
        //Поиск связей удаляемой записи в таблице с
сотрудниками
        var cmd3 = new MySqlCommand("SELECT COUNT(*) FROM
Table_Users where CodDolzhnost = @PID_Cod");
        cmd3.Connection = new
MySQLConnection(Query.ConnStr);
        cmd3.Connection.Open();
        cmd3.Parameters.AddWithValue("PID_Cod",
int.Parse(Grid.Rows[Grid.CurrentRow.Index].Cells["CodDolzhnost"].Value.ToString()));
        if (Convert.ToInt32(cmd3.ExecuteScalar()) > 0)
        {
            MessageBox.Show("Данную запись по должности
удалить нельзя, потому что есть данные по сотрудникам с указанной
должностью!", "ВНИМАНИЕ");
            return;
        }

        //Передаем индекс текущей записи в Grid
        Podkl.DeleteFromTable(TekTable, "CodDolzhnost",
int.Parse(Grid.Rows[Grid.CurrentRow.Index].Cells["CodDolzhnost"].Value.ToString()));
        cmd3.Connection.Close();
        break;
    }
    Grid.DataSource = Podkl.GetDataFromTable(Tekzapros);
    MessageBox.Show("Удаление выполнено успешно");
}

//Кнопка Выход
private void Exit_Click(object sender, EventArgs e)
{
    Close();
}
}

```

Листинг Е.37 – Класс FormSupplier

```

public partial class FormSupplier : Form
{

```

```

Query Podkl; //Определим наш объект класса
public int TekID_Supplier = 0;
public string Tekzapros;

public FormSupplier()
{
    InitializeComponent();

    //Инициализируем. В качестве параметра передаем строку
подключения
    Podkl = new Query(Query.ConnStr);
    Tekzapros = "select * from Table_Supplier order by Name";
}

//Обработчик кнопки Выход
private void Exit_Click(object sender, EventArgs e)
{
    Close();
}

//Обработчик кнопки Добавить
private void Knopka_Add_Click(object sender, EventArgs e)
{
    FormSupplierInfo newForm = new FormSupplierInfo();
    newForm.RegimSupplier = 2; //2- режим создания
    newForm.ShowDialog(); //блокируются предыдущие формы, активна
только текущая (новая открытая)
    TekID_Supplier = -1;
    this.TekID_Supplier = newForm.CheckSupplier;
    if (TekID_Supplier < 0) //Если был выход в окне с данными,
т.е., данные не добавлены
        return;

    GridSupplier.DataSource = Podkl.GetDataFromTable(Tekzapros);

    //Установить курсор на добавленную запись
    string searchValue = Convert.ToString(TekID_Supplier);
    GridSupplier.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

    try
    {
        foreach (DataGridViewRow row in GridSupplier.Rows)
        {
            if
(row.Cells[0].Value.ToString().Equals(searchValue))
            {
                row.Selected = true;
                GridSupplier.CurrentCell =
GridSupplier.Rows[row.Index].Cells[1];
                break;
            }
        }
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
}

```

```

        //-----Конец установки курсора
    }

    //Обработчик кнопки Изменить
    private void Knopka_Change_Click(object sender, EventArgs e)
    {
        if (GridSupplier.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        if
        (GridSupplier.Rows[GridSupplier.CurrentRow.Index].Cells["CodSupplier"].Value.ToString() == "")
            return;

        //Запоминаем позицию в DBGrid, чтобы после редактирования в
        //другом окне установить курсор на эту строку
        int iStr = GridSupplier.CurrentRow.Index;

        FormSupplierInfo newForm = new FormSupplierInfo();
        newForm.RegimSupplier = 1; //1- режим редактирования
        newForm.text_ID =
        GridSupplier.Rows[GridSupplier.CurrentRow.Index].Cells["CodSupplier"].Value.ToString();
        newForm.ShowDialog(); //блокируются предыдущие формы, активна
        //только текущая (новая открытая)

        GridSupplier.DataSource = Podkl.GetDataFromTable(Tekzapros);

        //Установить курсор на измененную запись
        GridSupplier.SelectionMode =
        DataGridViewSelectionMode.FullRowSelect;
        GridSupplier.CurrentRow = GridSupplier.Rows[iStr];
        //ставим выделение на измененную строку
    }

    //Обработчик кнопки Удалить
    private void Knopka_Delete_Click(object sender, EventArgs e)
    {
        if (GridSupplier.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        //-----Поиск связей удаляемой записи в таблице с
        //препаратами
        var cmd = new MySqlCommand("SELECT COUNT(*) FROM
        Table_Preparat where CodSupplier = @PidCod");
        cmd.Connection = new MySqlConnection(Query.ConnStr);
        cmd.Connection.Open();
        cmd.Parameters.AddWithValue("PidCod",
        int.Parse(GridSupplier.Rows[GridSupplier.CurrentRow.Index].Cells["CodSupplier"].Value.ToString()));
        int totalRecords = Convert.ToInt32(cmd.ExecuteScalar());
        if (totalRecords > 0)
    }

```



```

        {
            MessageBox.Show("Данного поставщика удалить нельзя,
потому что есть препараты, которые поставлены этим поставщиком!",
"ВНИМАНИЕ");
            return;
        }
        cmd.Connection.Close();
        //-----Конец поиска

        //Передаем индекс текущей записи в Grid
        Podkl.DeleteFromTable("Table_Supplier", "CodSupplier",
int.Parse(GridSupplier.Rows[GridSupplier.CurrentRow.Index].Cells["CodSupp
lier"].Value.ToString()));
        GridSupplier.DataSource = Podkl.GetDataFromTable(Tekzapros);
        MessageBox.Show("Удаление выполнено успешно");
    }

    //Клавиша ESC
    private void FormSupplier_KeyPress(object sender, KeyPressEventArgs
e)
    {
        if (e.KeyChar == (char)27)
        {
            Exit_Click(sender, e);
        }
    }

    //Обработчик загрузки формы
    private void FormSupplier_Load(object sender, EventArgs e)
    {
        // задать цвет в заголовках столбцов
        GridSupplier.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
        // задать шрифт в заголовках столбцов
        Font F = new Font("Times New Roman", 14);
        GridSupplier.ColumnHeadersDefaultCellStyle.Font = F;

        Font F2 = new Font("Times New Roman", 12);
        GridSupplier.DefaultCellStyle.Font = F2;
    }

    //Обработчик отображения формы
    private void FormSupplier_Shown(object sender, EventArgs e)
    {
        GridSupplier.DataSource = Podkl.GetDataFromTable(Tekzapros);
        GridSupplier.Columns[0].Visible = false;
        GridSupplier.Columns[1].Width = 180;
        GridSupplier.Columns[1].HeaderText = "Название";
        GridSupplier.Columns[2].Width = 130;
        GridSupplier.Columns[2].HeaderText = "Телефон";
        GridSupplier.Columns[3].Width = 180;
        GridSupplier.Columns[3].HeaderText = "Адрес";
        GridSupplier.Columns[4].Width = 110;
        GridSupplier.Columns[4].HeaderText = "Директор";
        GridSupplier.Columns[5].Width = 200;
        GridSupplier.Columns[5].HeaderText = "Информация";
        GridSupplier.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
    }

```

```

        GridSupplier.DefaultCellStyle.SelectionBackColor =
Color.White;
        GridSupplier.DefaultCellStyle.SelectionForeColor = Color.Blue;
        GridSupplier.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridSupplier.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовка столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridSupplier.ColumnHeadersDefaultCellStyle.SelectionBackColor
= GridSupplier.ColumnHeadersDefaultCellStyle.BackColor;
        GridSupplier.ColumnHeadersDefaultCellStyle.SelectionForeColor
= GridSupplier.ColumnHeadersDefaultCellStyle.ForeColor;

        GridSupplier.EnableHeadersVisualStyles = false;

        if (GridSupplier.Rows.Count > 0)
        {
            GridSupplier.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridSupplier.CurrentCell =
GridSupplier.Rows[0].Cells[1]; //ставим выделение на самую первую строку
            GridSupplier.Focus();
        }
    }

    //Обработчик в поле поиска по названию
    private void TextBoxName_TextChanged(object sender, EventArgs e)
    {
        string zapros = "select * from Table_Supplier ";

        if (textBoxName.Text != "")
        {
            zapros += "where Name like '%" + textBoxName.Text + "%'
order by Name";
        }
        else
        {
            zapros += "order by Name";
        }

        GridSupplier.DataSource = Podkl.GetDataFromTable(zapros);
        if (GridSupplier.Rows.Count > 0)
        {
            GridSupplier.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridSupplier.CurrentCell =
GridSupplier.Rows[0].Cells[1]; //ставим выделение на самую первую строку
        }
        textBoxName.Focus();
    }

    //Обработчик двойного нажатия мышкой в списке данных
    private void GridSupplier_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        Knopka_Change_Click(sender, e);
    }

```

```

    }
}

```

Листинг Е.38 – Класс FormSupplierInfo

```

public partial class FormSupplierInfo : Form
{
    Query Podkl; //Определим наш объект класса
    public string text_ID;
    public int RegimSupplier, CheckSupplier;

    public FormSupplierInfo()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
    }

    //Кнопка Сохранить
    private void But_Save_Click(object sender, EventArgs e)
    {
        //Выполняем проверки заполненных полей данных
        if (DBEdit_Name.Text == "")
        {
            MessageBox.Show("Введите название");
            DBEdit_Name.Focus();
            return;
        }

        string StrJson = "";

        switch (RegimSupplier)
        {
            case 1: //Режим редактирования
                ClassSupplier Per_1 = new
ClassSupplier(int.Parse(text_ID.ToString()),
                DBEdit_Name.Text,
                DBEdit_Phone.Text,
                DBEdit_Address.Text,
                DBEdit_FIO_Director.Text,
                DBEdit_Inform.Text
                );
                StrJson = JsonConvert.SerializeObject(Per_1);
                Podkl.Update_JSON("Table_Supplier", StrJson);
                break;
            case 2: //Режим создания
                ClassSupplier Per_2 = new ClassSupplier(0,
                DBEdit_Name.Text,
                DBEdit_Phone.Text,

```

```

        DBEdit_Address.Text,

        DBEdit_FIO_Director.Text,

        DBEdit_Inform.Text
    );
    StrJson = JsonConvert.SerializeObject(Per_2);
    CheckSupplier = Podkl.Add_JSON("Table_Supplier",
StrJson);
    break;
}

    Close();
}

//Клавиша ESC
private void FormSupplierInfo_KeyPress(object sender,
KeyPressEventArgs e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
}

//Обработчик загрузки формы
private void FormSupplierInfo_Load(object sender, EventArgs e)
{
    DataTable bufferSupplier = new DataTable();

    switch (RegimSupplier)
    {
        case 1: //Режим редактирования
            string zapros = "select * from Table_Supplier ";
            zapros += "where CodSupplier = ";
            zapros += text_ID;
            bufferSupplier = Podkl.GetDataFromTable(zapros);

            //Заполняем поля для ввода информации
            DBEdit_Name.Text =
bufferSupplier.Rows[0]["Name"].ToString();
            DBEdit_Phone.Text =
bufferSupplier.Rows[0]["Phone"].ToString();
            DBEdit_Address.Text =
bufferSupplier.Rows[0]["Address"].ToString();
            DBEdit_FIO_Director.Text =
bufferSupplier.Rows[0]["FIO_Director"].ToString();
            DBEdit_Inform.Text =
bufferSupplier.Rows[0]["Inform"].ToString(); ;
            break;
        case 2: //Режим создания
            //Заполняем поля для ввода информации
            DBEdit_Name.Text = "";
            DBEdit_Phone.Text = "";
            DBEdit_Address.Text = "";
            DBEdit_FIO_Director.Text = "";
            DBEdit_Inform.Text = "";
    }
}

```

```

        break;
    }
}

//Кнопка Выход
private void But_Exit_Click(object sender, EventArgs e)
{
    CheckSupplier = -1;
    Close();
}

private void DBEdit_Name_KeyPress(object sender, KeyPressEventArgs
e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
    else if (e.KeyChar == (char)13)
    {
        But_Save_Click(sender, e);
    }
}

//Обработчик отображения формы
private void FormSupplierInfo_Shown(object sender, EventArgs e)
{
    DBEdit_Name.Focus();
}
}

```

Листинг Е.39 – Класс FormUsers

```

public partial class FormUsers : Form
{
    Query Podkl; //Определим наш объект класса
    public int TekID_Users = 0;
    public string Tekzapros;

    public FormUsers()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
        Tekzapros = "select
U.CodUsers,U.FamU,U.NameU,U.OtchU,U.Login,U.Password,U.Prava,DL.Name as
Dolzhnost from (Table_Users as U ";
        Tekzapros += " inner join Table_Dolzhnost as DL on
U.CodDolzhnost=DL.CodDolzhnost) ";
        Tekzapros += " order by U.FamU,U.NameU,U.OtchU";
    }

    //Обработчик кнопки Выход
    private void Exit_Click(object sender, EventArgs e)
    {
        Close();
    }
}

```

```

    }

    //Обработчик кнопки Добавить
    private void Knopka_Add_Click(object sender, EventArgs e)
    {
        FormUsersInfo newForm = new FormUsersInfo();
        newForm.RegimUsers = 2; //2- режим создания
        newForm.ShowDialog(); //блокируются предыдущие формы, активна
        только текущая (новая открытая)
        TekID_Users = -1;
        this.TekID_Users = newForm.CheckUsers;
        if (TekID_Users < 0) //Если был выход в окне с данными, т.е.,
        данные не добавлены
            return;

        GridUsers.DataSource = Podkl.GetDataFromTable(Tekzapros);

        //Установить курсор на добавленную запись
        string searchValue = Convert.ToString(TekID_Users);
        GridUsers.SelectionMode =
        DataGridViewSelectionMode.FullRowSelect;

        try
        {
            foreach (DataGridViewRow row in GridUsers.Rows)
            {
                if
                (row.Cells[0].Value.ToString().Equals(searchValue))
                {
                    row.Selected = true;
                    GridUsers.CurrentCell =
                    GridUsers.Rows[row.Index].Cells[1];
                    break;
                }
            }
        }
        catch (Exception exc)
        {
            MessageBox.Show(exc.Message);
        }
        //-----Конец установки курсора
    }

    //Обработчик кнопки Изменить
    private void Knopka_Change_Click(object sender, EventArgs e)
    {
        if (GridUsers.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        if
        (GridUsers.Rows[GridUsers.CurrentRow.Index].Cells["CodUsers"].Value.ToStr
        ing() == "")
            return;
    }

```

```

        //Запоминаем позицию в DBGrid, чтобы после редактирования в
        другом окне установить курсор на эту строку
        int iStr = GridUsers.CurrentRow.RowIndex;

        FormUsersInfo newForm = new FormUsersInfo();
        newForm.RegimUsers = 1; //1- режим редактирования
        newForm.text_ID =
GridUsers.Rows[GridUsers.CurrentRow.Index].Cells["CodUsers"].Value.ToStri
ng();
        newForm.ShowDialog(); //блокируются предыдущие формы, активна
        только текущая (новая открытая)

        GridUsers.DataSource = Podkl.GetDataFromTable(Tekzapros);

        //Установить курсор на измененную запись
        GridUsers.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridUsers.CurrentCell = GridUsers.Rows[iStr].Cells[1];
        //ставим выделение на измененную строку
    }

    //Обработчик кнопки Удалить
    private void Knopka_Delete_Click(object sender, EventArgs e)
    {
        if (GridUsers.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        //Поиск связей удаляемой записи в таблице с заказами
        var cmd1 = new MySqlCommand("SELECT COUNT(*) FROM Table_Zakaz
where CodUsers = @PID_Cod");
        cmd1.Connection = new MySqlConnection(Query.ConnStr);
        cmd1.Connection.Open();
        cmd1.Parameters.AddWithValue("PID_Cod",
int.Parse(GridUsers.Rows[GridUsers.CurrentRow.Index].Cells["CodUsers"].Va
lue.ToString()));
        if (Convert.ToInt32(cmd1.ExecuteScalar()) > 0)
        {
            MessageBox.Show("Данную запись по сотруднику удалить
нельзя, потому что обнаружены заказы, которые связаны с сотрудником!",
"ВНИМАНИЕ");
            return;
        }

        //Передаем индекс текущей записи в GridUsers
        Podkl.DeleteFromTable("Table_Users", "CodUsers",
int.Parse(GridUsers.Rows[GridUsers.CurrentRow.Index].Cells["CodUsers"].Va
lue.ToString()));
        GridUsers.DataSource = Podkl.GetDataFromTable(Tekzapros);
        cmd1.Connection.Close();
        MessageBox.Show("Удаление выполнено успешно");
    }

    //Клавиша ESC
    private void FormUsers_KeyPress(object sender, KeyPressEventArgs e)
    {

```

```

        if (e.KeyChar == (char)27)
        {
            Exit_Click(sender, e);
        }
    }

    //Обработчик загрузки формы
    private void FormUsers_Load(object sender, EventArgs e)
    {
        // задать цвет в заголовках столбцов
        GridUsers.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
        // задать шрифт в заголовках столбцов
        Font F = new Font("Times New Roman", 14);
        GridUsers.ColumnHeadersDefaultCellStyle.Font = F;

        Font F2 = new Font("Times New Roman", 12);
        GridUsers.DefaultCellStyle.Font = F2;
    }

    //Обработчик отображения формы
    private void FormUsers_Shown(object sender, EventArgs e)
    {
        GridUsers.DataSource = Podkl.GetDataFromTable(Tekzapros);
        GridUsers.Columns[0].Visible = false;
        GridUsers.Columns[1].Width = 130;
        GridUsers.Columns[1].HeaderText = "Фамилия";
        GridUsers.Columns[2].Width = 130;
        GridUsers.Columns[2].HeaderText = "Имя";
        GridUsers.Columns[3].Width = 130;
        GridUsers.Columns[3].HeaderText = "Отчество";
        GridUsers.Columns[4].Width = 80;
        GridUsers.Columns[4].HeaderText = "Логин";
        GridUsers.Columns[5].Width = 80;
        GridUsers.Columns[5].HeaderText = "Пароль";
        GridUsers.Columns[6].Width = 140;
        GridUsers.Columns[6].HeaderText = "Права";
        GridUsers.Columns[7].Width = 150;
        GridUsers.Columns[7].HeaderText = "Должность";
        GridUsers.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridUsers.DefaultCellStyle.SelectionBackColor = Color.White;
        GridUsers.DefaultCellStyle.SelectionForeColor = Color.Blue;
        GridUsers.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridUsers.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridUsers.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridUsers.ColumnHeadersDefaultCellStyle.BackColor;
        GridUsers.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridUsers.ColumnHeadersDefaultCellStyle.ForeColor;

        GridUsers.EnableHeadersVisualStyles = false;

        if (GridUsers.Rows.Count > 0)

```



```

        {
            GridUsers.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridUsers.CurrentCell = GridUsers.Rows[0].Cells[1];
//ставим выделение на самую первую строку
            GridUsers.Focus();
        }
    }

//Обработчик в поле поиска по фамилии
private void TextBoxFam_TextChanged(object sender, EventArgs e)
{
    string zapros = "select
U.CodUsers,U.FamU,U.NameU,U.OtchU,U.Login,U.Password,U.Prava,DL.Name as
Dolzhnost from (Table_Users as U ";
    zapros += " inner join Table_Dolzhnost as DL on
U.CodDolzhnost=DL.CodDolzhnost) ";

    if (textBoxFam.Text != "")
    {
        zapros += " where U.FamU like '%" + textBoxFam.Text +
"%" order by U.FamU,U.NameU,U.OtchU";
    }
    else
    {
        zapros += " order by U.FamU,U.NameU,U.OtchU";
    }

    GridUsers.DataSource = Podkl.GetDataFromTable(zapros);
    if (GridUsers.Rows.Count > 0)
    {
        GridUsers.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridUsers.CurrentCell = GridUsers.Rows[0].Cells[1];
//ставим выделение на самую первую строку
    }
    textBoxFam.Focus();
}

//Обработчик двойного нажатия мышкой в списке сотрудников
private void GridUsers_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
{
    Knopka_Change_Click(sender, e);
}
}

```

Листинг E.40 – Класс FormUsersInfo

```

public partial class FormUsersInfo : Form
{
    Query Podkl; //Определим наш объект класса
    public string text_ID;
    public int RegimUsers, CheckUsers;

    public FormUsersInfo()
    {
        InitializeComponent();
    }
}

```

```

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
    }

    //Кнопка Выход
    private void But_Exit_Click(object sender, EventArgs e)
    {
        CheckUsers = -1;
        Close();
    }

    //Клавиши ESC и ENTER
    private void DBEdit_Fam_KeyPress(object sender, KeyPressEventArgs
e)
    {
        if (e.KeyChar == (char)27)
        {
            But_Exit_Click(sender, e);
        }
        else if (e.KeyChar == (char)13)
        {
            But_Save_Click(sender, e);
        }
    }

    //Кнопка Сохранить
    private void But_Save_Click(object sender, EventArgs e)
    {
        //Выполняем проверки заполненных полей данных
        if (DBEdit_Fam.Text == "")
        {
            MessageBox.Show("Введите фамилию сотрудника");
            DBEdit_Fam.Focus();
            return;
        }

        if (DBEdit_Login.Text == "")
        {
            MessageBox.Show("Введите логин сотрудника");
            DBEdit_Login.Focus();
            return;
        }

        if (DBEdit_Parol.Text == "")
        {
            MessageBox.Show("Введите пароль сотрудника");
            DBEdit_Parol.Focus();
            return;
        }

        if (comboBoxPrava.Text == "")
        {
            MessageBox.Show("Укажите права сотрудника");
            comboBoxPrava.Focus();
            return;
        }
    }

```

```

        if (comboBoxDolzhnost.Text == "")
        {
            MessageBox.Show("Укажите должность сотрудника");
            comboBoxDolzhnost.Focus();
            return;
        }

        //Поиск дублей логинов только при добавлении
        if (RegimUsers == 2)
        {
            var cmd = new MySqlCommand(@"SELECT COUNT(*) FROM
Table_Users where Login = @PLogin");
            cmd.Connection = new MySqlConnection(Query.ConnStr);
            cmd.Connection.Open();
            cmd.Parameters.AddWithValue("PLogin",
DBEdit_Login.Text);
            int totalRecords = Convert.ToInt32(cmd.ExecuteScalar());
            if (totalRecords > 0)
            {
                MessageBox.Show("Укажите другой логин сотруднику.
Данное значение (" + DBEdit_Login.Text +
                                ") уже принадлежит другому
работнику!", "ВНИМАНИЕ");
                DBEdit_Login.Focus();
                return;
            }
        }
        //-----Конец поиска

        string StrJson = "";

        switch (RegimUsers)
        {
            case 1: //Режим редактирования
                ClassUsers Per_1 = new
ClassUsers(int.Parse(text_ID.ToString()),

                DBEdit_Fam.Text,

                DBEdit_Name.Text,

                DBEdit_Otch.Text,

                DBEdit_Login.Text,

                DBEdit_Parol.Text,

                comboBoxPrava.Text,

                int.Parse(comboBoxDolzhnost.SelectedValue.ToString())
                );
                StrJson = JsonConvert.SerializeObject(Per_1);
                Podkl.Update_JSON("Table_Users", StrJson);
                break;
            case 2: //Режим создания
                ClassUsers Per_2 = new ClassUsers(0,

```

```

        DBEdit_Fam.Text,

        DBEdit_Name.Text,

        DBEdit_Otch.Text,

        DBEdit_Login.Text,

        DBEdit_Parol.Text,

        comboBoxPrava.Text,

        int.Parse(comboBoxDolzhnost.SelectedValue.ToString())
    );
    StrJson = JsonConvert.SerializeObject(Per_2);
    CheckUsers = Podkl.Add_JSON("Table_Users",
StrJson);
        break;
    }

    Close();
}

//Клавиша ESC
private void FormUsersInfo_KeyPress(object sender,
KeyPressEventArgs e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
}

//Обработчик загрузки формы
private void FormUsersInfo_Load(object sender, EventArgs e)
{
    DataTable bufferUsers = new DataTable();
    DataTable bufferDolzhnost = new DataTable();

    //Заполняем combobox для должностей
    //Инициализируем - передаем запрос и объект класса
SqlConnection - connection
    string StrZ = "select * from Table_Dolzhnost order by Name";
    MySqlDataAdapter datadapterDolzhnost = new
MySqlDataAdapter(StrZ, Query.ConnStr);
    datadapterDolzhnost.Fill(bufferDolzhnost); //вызываем метод
Fill, чтобы после запроса заполнил буферную таблицу данными

    comboBoxDolzhnost.DataSource = bufferDolzhnost;
    comboBoxDolzhnost.DisplayMember = "Name";
    comboBoxDolzhnost.ValueMember = "CodDolzhnost";

    switch (RegimUsers)
    {
        case 1: //Режим редактирования

```

```

        string zapros = "select
U.CodUsers,U.FamU,U.NameU,U.OtchU,U.Login,U.Password,U.Prava,DL.Name as
Dolzhnost from (Table_Users as U ";
        zapros += " inner join Table_Dolzhnost as DL on
U.CodDolzhnost=DL.CodDolzhnost) ";
        zapros += "where U.CodUsers = ";
        zapros += text_ID;
        bufferUsers = Podkl.GetDataFromTable(zapros);

        //Заполняем поля для ввода информации
        DBEdit_Fam.Text =
bufferUsers.Rows[0]["FamU"].ToString();
        DBEdit_Name.Text =
bufferUsers.Rows[0]["NameU"].ToString();
        DBEdit_Otch.Text =
bufferUsers.Rows[0]["OtchU"].ToString();
        DBEdit_Login.Text =
bufferUsers.Rows[0]["Login"].ToString();
        DBEdit_Parol.Text =
bufferUsers.Rows[0]["Password"].ToString();
        comboBoxPrava.Text =
bufferUsers.Rows[0]["Prava"].ToString();

        string s_Dolzhnost =
bufferUsers.Rows[0]["Dolzhnost"].ToString();
        int index =
comboBoxDolzhnost.FindString(s_Dolzhnost);
        comboBoxDolzhnost.SelectedIndex = index;
        break;
    case 2: //Режим создания
        //Заполняем поля для ввода информации
        DBEdit_Fam.Text = "";
        DBEdit_Name.Text = "";
        DBEdit_Otch.Text = "";
        DBEdit_Login.Text = "";
        DBEdit_Parol.Text = "";
        comboBoxPrava.Text = "";
        if (comboBoxDolzhnost.Items.Count > 0)
            comboBoxDolzhnost.SelectedIndex = 0;
        break;
    }
    if (RegimUsers == 1)
        DBEdit_Login.Enabled = false;
}

//Обработчик отображения формы
private void FormUsersInfo_Shown(object sender, EventArgs e)
{
    DBEdit_Fam.Focus();
}
}

```

Листинг Е.41 – Класс FormZakaz

```

public partial class FormZakaz : Form
{
    Query Podkl; //Определим наш объект класса
    public int TekID_Zakaz = 0;
}

```

```

public string Tekzapros,
        SqlLekarstvo, SqlPreparat,
        indexCheckPunkt = "",
        indexCheckStatus = "",
        indexCheckClient = "";

private Microsoft.Office.Interop.Excel.Application excelapp;
private Microsoft.Office.Interop.Excel.Window excelWindow;
private Microsoft.Office.Interop.Excel.Sheets excelsheets;
private Microsoft.Office.Interop.Excel.Worksheet excelworksheet;

public FormZakaz()
{
    InitializeComponent();

    //Инициализируем. В качестве параметра передаем строку
    подключения
    Podkl = new Query(Query.ConnStr);

    SqlLekarstvo = "select Z.CodZakaz,Z.DataStart,Z.DataEnd,";
    SqlLekarstvo += "concat(CL.FamCl,'"
    ',LEFT(CL.NameCl,1),'',LEFT(CL.OtchCl,1),'') as Client,";
    SqlLekarstvo += "Z.Summa,ST.Name as Status,PN.Name as
    Punkt,Z.Inform,";
    SqlLekarstvo += "concat(US.FamU,'"
    ',LEFT(US.NameU,1),'',LEFT(US.OtchU,1),'') as Users ";
    SqlLekarstvo += " from (((Table_Zakaz as Z ";
    SqlLekarstvo += "inner join Table_Client as CL on Z.CodClient
    = CL.CodClient) ";
    SqlLekarstvo += "inner join Table_Punkt as PN on Z.CodPunkt =
    PN.CodPunkt) ";
    SqlLekarstvo += "inner join Table_Status as ST on Z.CodStatus
    = ST.CodStatus) ";
    SqlLekarstvo += "inner join Table_Users as US on Z.CodUsers =
    US.CodUsers) ";

    SqlPreparat = "select Z.CodZakaz,Z.DataStart,";
    SqlPreparat += "Z.Summa,ST.Name as Status,Z.Inform,";
    SqlPreparat += "concat(US.FamU,'"
    ',LEFT(US.NameU,1),'',LEFT(US.OtchU,1),'') as Users";
    SqlPreparat += " from ((Table_Zakaz as Z ";
    SqlPreparat += "inner join Table_Status as ST on Z.CodStatus =
    ST.CodStatus) ";
    SqlPreparat += "inner join Table_Users as US on Z.CodUsers =
    US.CodUsers) ";
}

//Обработчик кнопки Выход
private void Exit_Click(object sender, EventArgs e)
{
    Close();
}

//Обработчик кнопки Добавить
private void Knopka_Add_Click(object sender, EventArgs e)
{
    FormZakazInfo newForm = new FormZakazInfo();
    newForm.RegimZakaz = 2; //2- режим создания

```

```

        newForm.ShowDialog(); //блокируются предыдущие формы, активна
        только текущая (новая открытая)
        TekID_Zakaz = -1;
        this.TekID_Zakaz = newForm.CheckZakaz;
        if (TekID_Zakaz < 0) //Если был выход в окне с данными, т.е.,
        данные не добавлены
            return;

        GridZakaz.DataSource = Podkl.GetDataFromTable(Tekzapros); //
        ClassStr.RegimZakaz ==1 ? SqlLekarstvo: SqlPreparat);

        //Установить курсор на добавленную запись
        string searchValue = Convert.ToString(TekID_Zakaz);
        GridZakaz.SelectionMode =
        DataGridViewSelectionMode.FullRowSelect;

        try
        {
            foreach (DataGridViewRow row in GridZakaz.Rows)
            {
                if
                (row.Cells[0].Value.ToString().Equals(searchValue))
                {
                    row.Selected = true;
                    GridZakaz.CurrentCell =
                    GridZakaz.Rows[row.Index].Cells[1];
                    break;
                }
            }
        }
        catch (Exception exc)
        {
            MessageBox.Show(exc.Message);
        }
        //-----Конец установки курсора
    }

    //Обработчик кнопки Изменить
    private void Knopka_Change_Click(object sender, EventArgs e)
    {
        if (GridZakaz.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        if
        (GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Value.ToStr
        ing() == "")
            return;

        //Запоминаем позицию в DBGrid, чтобы после редактирования в
        другом окне установить курсор на эту строку
        int iStr = GridZakaz.CurrentCell.RowIndex;

        FormZakazInfo newForm = new FormZakazInfo();
        newForm.RegimZakaz = 1; //1- режим редактирования

```

```

        newForm.text_ID =
GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Value.ToString();
        newForm.ShowDialog(); //блокируются предыдущие формы, активна
только текущая (новая открытая)

        GridZakaz.DataSource = Podkl.GetDataFromTable(Tekzapros); //
ClassStr.RegimZakaz == 1 ? SqlLekarstvo : SqlPreparat);

        //Установить курсор на измененную запись
        GridZakaz.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
        GridZakaz.CurrentCell = GridZakaz.Rows[iStr].Cells[1];
//ставим выделение на измененную строку
    }

    //Обработчик кнопки Удалить
    private void Knopka_Delete_Click(object sender, EventArgs e)
    {
        if (GridZakaz.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        //-----Поиск связей удаляемой записи в таблице с составом
заказа
        var cmd = new MySqlCommand("SELECT COUNT(*) FROM
Table_SostavZakaz where CodZakaz = @PidCod");
        cmd.Connection = new MySqlConnection(Query.ConnStr);
        cmd.Connection.Open();
        cmd.Parameters.AddWithValue("PidCod",
int.Parse(GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Va
lue.ToString()));
        int totalRecords = Convert.ToInt32(cmd.ExecuteScalar());
        if (totalRecords > 0)
        {
            MessageBox.Show("Данный заказ удалить нельзя, потому что
есть данные в составе заказа, которые связаны с этим заказом!",
"ВНИМАНИЕ");
            return;
        }
        cmd.Connection.Close();

        //-----Поиск связей удаляемой записи в таблице с оплатами
        var cmd2 = new MySqlCommand("SELECT COUNT(*) FROM Table_Oplata
where CodZakaz = @PidCod");
        cmd2.Connection = new MySqlConnection(Query.ConnStr);
        cmd2.Connection.Open();
        cmd2.Parameters.AddWithValue("PidCod",
int.Parse(GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Va
lue.ToString()));
        int totalRecords2 = Convert.ToInt32(cmd2.ExecuteScalar());
        if (totalRecords2 > 0)
        {
            MessageBox.Show("Данный заказ удалить нельзя, потому что
есть оплаты, которые связаны с этим заказом!", "ВНИМАНИЕ");
            return;
        }
    }

```



```

    }
    cmd2.Connection.Close();

    //-----Поиск связей удаляемой записи в таблице с историей
    var cmd3 = new MySqlCommand("SELECT COUNT(*) FROM
Table_History where CodZakaz = @PidCod");
    cmd3.Connection = new MySqlConnection(Query.ConnStr);
    cmd3.Connection.Open();
    cmd3.Parameters.AddWithValue("PidCod",
int.Parse(GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Va
lue.ToString()));
    int totalRecords3 = Convert.ToInt32(cmd3.ExecuteScalar());
    if (totalRecords3 > 0)
    {
        MessageBox.Show("Данный заказ удалить нельзя, потому что
есть записи в истории, которые связаны с этим заказом!", "ВНИМАНИЕ");
        return;
    }
    cmd3.Connection.Close();
    //-----Конец поиска

    //Передаем индекс текущей записи в GridZakaz
    Podkl.DeleteFromTable("Table_Zakaz", "CodZakaz",
int.Parse(GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Va
lue.ToString()));
    GridZakaz.DataSource = Podkl.GetDataFromTable(SqlLekarstvo);
    MessageBox.Show("Удаление выполнено успешно");
}

//Клавиша ESC
private void FormZakaz_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == (char)27)
    {
        Exit_Click(sender, e);
    }
}

//Обработчик загрузки формы
private void FormZakaz_Load(object sender, EventArgs e)
{
    // задать цвет в заголовках столбцов
    GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Purple;
    // задать шрифт в заголовках столбцов
    Font F = new Font("Times New Roman", 14);
    GridZakaz.ColumnHeadersDefaultCellStyle.Font = F;

    Font F2 = new Font("Times New Roman", 12);
    GridZakaz.DefaultCellStyle.Font = F2;

    if (ClassStr.TekPrava == "Провизор")
    {
        Knopka_Add.Visible = true;
        Knopka_Delete.Visible = true;
        Knopka_Oplata.Visible = false;
    }
}

```

```

//Обработчик отображения формы
private void FormZakaz_Shown(object sender, EventArgs e)
{
    DataTable bufferZakaz = new DataTable();
    DataTable bufferStatus = new DataTable();
    DataTable bufferPunkt = new DataTable();
    DataTable bufferClient = new DataTable();

    if (ClassStr.TekPrava=="Клиент")
    {
        checkBoxClient.Enabled = false;
        comboBoxFilterClient.Enabled = false;
        Knopka_Delete.Enabled = false;

        SqlLekarstvo = "select
Z.CodZakaz,Z.DataStart,Z.DataEnd,";
        SqlLekarstvo += "concat(CL.FamCl,'"
        SqlLekarstvo += "',LEFT(CL.NameCl,1),'.',LEFT(CL.OtchCl,1),'.') as Client,";
        SqlLekarstvo += "Z.Summa,ST.Name as Status,PN.Name as
Punkt,Z.Inform,";
        SqlLekarstvo += "concat(US.FamU,'"
        SqlLekarstvo += "',LEFT(US.NameU,1),'.',LEFT(US.OtchU,1),'.') as Users";
        SqlLekarstvo += " from (((Table_Zakaz as Z ";
        SqlLekarstvo += "inner join Table_Client as CL on
Z.CodClient = CL.CodClient) ";
        SqlLekarstvo += "inner join Table_Punkt as PN on
Z.CodPunkt = PN.CodPunkt) ";
        SqlLekarstvo += "inner join Table_Status as ST on
Z.CodStatus = ST.CodStatus) ";
        SqlLekarstvo += "inner join Table_Users as US on
Z.CodUsers = US.CodUsers) ";
        SqlLekarstvo += " where Z.CodClient=" +
ClassStr.TekCodClient.ToString();
    }

    //Заполняем combobox для статусов
    //Инициализируем - передаем запрос и объект класса
SqlConnection - connection
    string StrZ = "select * from Table_Status order by Name";
    MySqlDataAdapter datadapterStatus = new MySqlDataAdapter(StrZ,
Query.ConnStr);
    datadapterStatus.Fill(bufferStatus); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными

    comboBoxFilterStatus.DataSource = bufferStatus;
    comboBoxFilterStatus.DisplayMember = "Name";
    comboBoxFilterStatus.ValueMember = "CodStatus";

    buttonFilterOFF_Click(sender, e);
    //Tekzapros = ClassStr.RegimZakaz == 1 ?
SqlLekarstvo:SqlPreparat;

    if (ClassStr.RegimZakaz==2) //препараты от поставщиков
    {
        //comboBoxFilterPunkt.Location = new Point(130, 69);
    }
}

```

```

        Height = 29 };

        //comboBoxFilterPunkt.Size = new Size { Width = 190,
        checkBoxStatus.Location = new Point(13, 73);
        comboBoxFilterStatus.Location = new Point(95, 69);

        checkBoxClient.Visible = false;
        comboBoxFilterClient.Visible = false;
        checkBoxPunkt.Visible = false;
        comboBoxFilterPunkt.Visible = false;
        Knopka_Recept.Visible = false;
        this.Text = "Информация по заказам (поставщики)";
        checkBoxPunkt.Text = "Поставщики";

        //Заполняем combobox для поставщиков
        //Инициализируем - передаем запрос и объект класса
SqlConnection - connection
        StrZ = "select * from Table_Supplier order by Name";
        DataTable bufferSupplier = new DataTable();
        MySqlDataAdapter datadapterSupplier = new
MySqlDataAdapter(StrZ, Query.ConnStr);
        datadapterSupplier.Fill(bufferSupplier); //вызываем
метод Fill, чтобы после запроса заполнил буферную таблицу данными

        comboBoxFilterPunkt.DataSource = bufferSupplier;
        comboBoxFilterPunkt.DisplayMember = "Name";
        comboBoxFilterPunkt.ValueMember = "CodSupplier";

        //GridZakaz.DataSource =
Podkl.GetDataFromTable(Tekzapros); //SqlPreparat);
        //GridZakaz.Columns[0].Visible = false;
        GridZakaz.Columns[0].Width = 60;
        GridZakaz.Columns[0].HeaderText = "Код";
        GridZakaz.Columns[0].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.Columns[1].Width = 90;
        GridZakaz.Columns[1].HeaderText = "Дата";
        GridZakaz.Columns[1].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.Columns[2].Width = 90;
        GridZakaz.Columns[2].HeaderText = "Сумма";
        GridZakaz.Columns[2].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.Columns[3].Width = 120;
        GridZakaz.Columns[3].HeaderText = "Статус";
        GridZakaz.Columns[3].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.Columns[4].Width = 250;
        GridZakaz.Columns[4].HeaderText = "Информация";
        GridZakaz.Columns[4].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleLeft;
        GridZakaz.Columns[5].Width = 130;
        GridZakaz.Columns[5].HeaderText = "Сотрудник";
        GridZakaz.Columns[5].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        GridZakaz.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.DefaultCellStyle.SelectionBackColor =
Color.White;

```

```

        GridZakaz.DefaultCellStyle.SelectionForeColor =
Color.Blue;
        GridZakaz.ColumnHeadersDefaultCellStyle.BackColor =
Color.Yellow;
        GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для
заголовков

        GridZakaz.ColumnHeadersDefaultCellStyle.SelectionBackColor =
GridZakaz.ColumnHeadersDefaultCellStyle.BackColor;

        GridZakaz.ColumnHeadersDefaultCellStyle.SelectionForeColor =
GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor;

        GridZakaz.EnableHeadersVisualStyles = false;

        if (GridZakaz.Rows.Count > 0)
        {
            GridZakaz.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridZakaz.CurrentCell = GridZakaz.Rows[0].Cells[1];
//ставим выделение на самую первую строку
            GridZakaz.Focus();
        }

        comboBoxFilterStatus_SelectedIndexChanged(sender, e);
        comboBoxFilterPunkt_SelectedIndexChanged(sender, e);

        return;
    }

    //-----
    //Заполняем комбобокх для пунктов
    //Инициализируем - передаем запрос и объект класса
SqlConnection - connection
    StrZ = "select * from Table_Punkt order by Name";
    MySqlDataAdapter datadapterPunkt = new MySqlDataAdapter(StrZ,
Query.ConnStr);
    datadapterPunkt.Fill(bufferPunkt); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными

    comboBoxFilterPunkt.DataSource = bufferPunkt;
    comboBoxFilterPunkt.DisplayMember = "Name";
    comboBoxFilterPunkt.ValueMember = "CodPunkt";

    //Заполняем комбобокх для клиентов
    //Инициализируем - передаем запрос и объект класса
SqlConnection - connection
    StrZ = "select concat(FamCl, '
', LEFT(NameCl,1), '.', LEFT(ОтchCl,1), '.') as FIOClient, CodClient from
Table_Client order by FamCl, NameCl, ОтchCl";
    MySqlDataAdapter datadapterClient = new MySqlDataAdapter(StrZ,
Query.ConnStr);

```

```

        datadapterClient.Fill(bufferClient); //вызываем метод Fill,
        чтобы после запроса заполнил буферную таблицу данными

        comboBoxFilterClient.DataSource = bufferClient;
        comboBoxFilterClient.DisplayMember = "FIOClient";
        comboBoxFilterClient.ValueMember = "CodClient";

        //GridZakaz.DataSource = Podkl.GetDataFromTable(Tekzapros); //
        SqlLekarstvo);
        //GridZakaz.Columns[0].Visible = false;
        GridZakaz.Columns[0].Width = 60;
        GridZakaz.Columns[0].HeaderText = "Код";
        GridZakaz.Columns[0].DefaultCellStyle.Alignment =
        DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.Columns[1].Width = 90;
        GridZakaz.Columns[1].HeaderText = "Дата (Начало)";
        GridZakaz.Columns[1].DefaultCellStyle.Alignment =
        DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.Columns[2].Width = 90;
        GridZakaz.Columns[2].HeaderText = "Дата (Конец)";
        GridZakaz.Columns[2].DefaultCellStyle.Alignment =
        DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.Columns[3].Width = 130;
        GridZakaz.Columns[3].HeaderText = "Клиент";
        GridZakaz.Columns[3].DefaultCellStyle.Alignment =
        DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.Columns[4].Width = 80;
        GridZakaz.Columns[4].HeaderText = "Сумма";
        GridZakaz.Columns[4].DefaultCellStyle.Alignment =
        DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.Columns[5].Width = 120;
        GridZakaz.Columns[5].HeaderText = "Статус";
        GridZakaz.Columns[5].DefaultCellStyle.Alignment =
        DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.Columns[6].Width = 180;
        GridZakaz.Columns[6].HeaderText = "Пункт";
        GridZakaz.Columns[7].Width = 250;
        GridZakaz.Columns[7].HeaderText = "Информация";
        GridZakaz.Columns[8].Width = 130;
        GridZakaz.Columns[8].HeaderText = "Сотрудник";
        GridZakaz.Columns[8].DefaultCellStyle.Alignment =
        DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.ColumnHeadersDefaultCellStyle.Alignment =
        DataGridViewContentAlignment.MiddleCenter;
        GridZakaz.DefaultCellStyle.SelectionBackColor = Color.White;
        GridZakaz.DefaultCellStyle.SelectionForeColor = Color.Blue;
        GridZakaz.ColumnHeadersDefaultCellStyle.BackColor =
        Color.Yellow;
        GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor =
        Color.Black;

        //отменить выделение строки заголовков столбцов.
        //Будет отображаться цвет, который установлен для заголовков
        GridZakaz.ColumnHeadersDefaultCellStyle.SelectionBackColor =
        GridZakaz.ColumnHeadersDefaultCellStyle.BackColor;
        GridZakaz.ColumnHeadersDefaultCellStyle.SelectionForeColor =
        GridZakaz.ColumnHeadersDefaultCellStyle.ForeColor;

```

```

        GridZakaz.EnableHeadersVisualStyles = false;

        if (GridZakaz.Rows.Count > 0)
        {
            GridZakaz.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            GridZakaz.CurrentCell = GridZakaz.Rows[0].Cells[1];
//ставим выделение на самую первую строку
            GridZakaz.Focus();
        }

        comboBoxFilterStatus_SelectedIndexChanged(sender, e);
        comboBoxFilterClient_SelectedIndexChanged(sender, e);
        comboBoxFilterPunkt_SelectedIndexChanged(sender, e);
    }

    //Обработчик двойного нажатия мышкой в списке данных
    private void GridZakaz_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        Knopka_Change_Click(sender, e);
    }

    //Кнопка - Состав
    private void Knopka_Sostav_Click(object sender, EventArgs e)
    {
        if (GridZakaz.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        if
(GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Value.ToStr
ing() == "")
            return;

        FormSostavZakaz newForm = new FormSostavZakaz();
        ClassStr.TekCodZakaz =
int.Parse(GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Va
lue.ToString());
        ClassStr.TekNameZakaz = ClassStr.RegimZakaz==1 ?

        GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Client"].Value.To
String() + ", Сумма:" +

        GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Summa"].Value.ToS
tring()+ ", Пункт:" +

        GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Punkt"].Value.ToS
tring():

        "Код:" +
GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Value.ToStri
ng() + ", Сумма:" +

        GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Summa"].Value.ToS
tring() + ", Статус:" +

```

```

        GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Status"].Value.ToString()

        ;

        newForm.ShowDialog(); //блокируются предыдущие формы, активна
        только текущая (новая открытая)
    }

    //Кнопка - Оплаты
    private void Knopka_Oplata_Click(object sender, EventArgs e)
    {
        if (GridZakaz.Rows.Count < 2)
        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        if
        (GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Value.ToString() == "")
            return;

        FormOplataZakaz newForm = new FormOplataZakaz();
        ClassStr.TekCodZakaz =
        int.Parse(GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Value.ToString());
        ClassStr.TekNameZakaz = ClassStr.RegimZakaz == 1 ?

        GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Client"].Value.ToString() + ", Сумма_" +

        GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Summa"].Value.ToString() + ", Пункт_" +

        GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Punkt"].Value.ToString() :

        "Код_" + GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Value.ToString() + ", Сумма_" +

        GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Summa"].Value.ToString() + ", Статус_" +

        GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Status"].Value.ToString()

        ;

        if (ClassStr.RegimZakaz == 1)
            ClassStr.TekFioCl =
        GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Client"].Value.ToString();

        newForm.ShowDialog(); //блокируются предыдущие формы, активна
        только текущая (новая открытая)
    }

    //Кнопка - Рецепты
    private void Knopka_Recept_Click(object sender, EventArgs e)
    {
        if (GridZakaz.Rows.Count < 2)

```

```

        {
            MessageBox.Show("Отсутствуют данные");
            return;
        }

        if
(GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Value.ToString() == "")
            return;

            FormRecept newForm = new FormRecept();
            ClassStr.TekCodZakaz =
int.Parse(GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Value.ToString());
            ClassStr.TekFioClient =
GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Client"].Value.ToString();
            ClassStr.TekNameZakaz = ClassStr.RegimZakaz == 1 ?

            GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Client"].Value.ToString() + ", Сумма_" +

            GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Summa"].Value.ToString() + ", Пункт_" +

            GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Punkt"].Value.ToString() :

                                                "Код_" +
GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["CodZakaz"].Value.ToString() + ", Сумма_" +

            GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Summa"].Value.ToString() + ", Статус_" +

            GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Status"].Value.ToString() + " (" +

            GridZakaz.Rows[GridZakaz.CurrentRow.Index].Cells["Client"].Value.ToString() + ")"

                                                ;
            newForm.ShowDialog(); //блокируются предыдущие формы, активна
только текущая (новая открытая)
        }

//Кнопка - Отчет
private void Knopka_Excel_Click(object sender, EventArgs e)
{
    if (GridZakaz.Rows.Count < 2)
    {
        MessageBox.Show("Отсутствуют данные");
        return;
    }

    this.Enabled = false;

    DateTime datel = DateTime.Now;
    var fileReport =
Path.GetDirectoryName(Application.ExecutablePath) + "\\Отчеты\\Заказы " +

```



```

(ClassStr.RegimZakaz == 1 ? "Лекарства":"Препараты") +
                                " от " +
date1.Day.ToString() + "." + date1.Month.ToString() + "." +
date1.Year.ToString() + "_" +

    date1.Hour.ToString() + "_" + date1.Minute.ToString() + "_" +
date1.Second.ToString() +

                                ".xlsx";

    excelapp = new Microsoft.Office.Interop.Excel.Application();
    excelapp.Visible = false;

    excelapp.SheetsInNewWorkbook = 2; //кол-во листов в excel
    excelapp.Workbooks.Add(Type.Missing);
    Microsoft.Office.Interop.Excel.Range excelcells;

    //Получаем набор ссылок на объекты Workbook (на созданные
книги)
    var excelappworkbooks = excelapp.Workbooks;

    //Получаем ссылку на книгу 1 - нумерация от 1
    var excelappworkbook = excelappworkbooks[1];

    //НЕ Будем спрашивать разрешение на запись поверх
существующего документа
    excelapp.DisplayAlerts = false;

    excelsheets = excelappworkbook.Worksheets;
    //Получаем ссылку на лист 1
    excelworksheet =
(Microsoft.Office.Interop.Excel.Worksheet)excelsheets.get_Item(1);
    //Выделение группы ячеек
    excelcells = excelworksheet.get_Range("A2",
ClassStr.RegimZakaz == 1 ? "I2": "F2");
    excelcells.Merge(Type.Missing);
    excelcells.EntireRow.Font.Bold = true;
    excelcells.EntireRow.Font.Size = 14;
    excelcells.EntireRow.Font.Color = Color.Black;
    excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
    excelcells.Value2 = @"Дата: " + date1.Day.ToString() + "." +
date1.Month.ToString() + "." + date1.Year.ToString();

    excelcells = excelworksheet.get_Range("A4",
ClassStr.RegimZakaz == 1 ? "I4" : "F4");
    excelcells.Merge(Type.Missing);
    //Задаем выравнивание по центру
    excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
    excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
    excelcells.EntireRow.Font.Bold = true;
    excelcells.EntireRow.Font.Size = 16;
    excelcells.EntireRow.Font.Color = Color.Black;

```

```

        excelcells.Value2 = ClassStr.RegimZakaz == 1 ? @"Отчет
'Информация по заказам (лекарства) '": @"Отчет 'Информация по заказам
(препараты) '";

        excelcells = excelworksheet.get_Range("A5",
ClassStr.RegimZakaz == 1 ? "I5": "F5");
        excelcells.Merge(Type.Missing);
        //Задаем выравнивание по центру
        excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 16;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.Value2 = "";

        //Выводим названия столбцов
        excelcells = excelworksheet.get_Range("A7",
ClassStr.RegimZakaz == 1 ? "I7" : "F7");
        excelcells.EntireRow.Font.Bold = true;
        excelcells.EntireRow.Font.Size = 14;
        excelcells.EntireRow.Font.Color = Color.Black;
        excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

        var j = 0;
        if (ClassStr.RegimZakaz == 1)
        {
            excelcells = excelworksheet.get_Range("A7", "A7");
            excelcells.Value2 = @"Код";
            excelworksheet.Columns[1].ColumnWidth = 7;
            excelcells = excelworksheet.get_Range("B7", "B7");
            excelcells.Value2 = @"Дата (начало)";
            excelworksheet.Columns[2].ColumnWidth = 15;
            excelcells = excelworksheet.get_Range("C7", "C7");
            excelcells.Value2 = @"Дата (конец)";
            excelworksheet.Columns[3].ColumnWidth = 14;
            excelcells = excelworksheet.get_Range("D7", "D7");
            excelcells.Value2 = @"Клиент";
            excelworksheet.Columns[4].ColumnWidth = 20;
            excelcells = excelworksheet.get_Range("E7", "E7");
            excelcells.Value2 = @"Сумма";
            excelworksheet.Columns[5].ColumnWidth = 10;
            excelcells = excelworksheet.get_Range("F7", "F7");
            excelcells.Value2 = @"Статус";
            excelworksheet.Columns[6].ColumnWidth = 13;
            excelcells = excelworksheet.get_Range("G7", "G7");
            excelcells.Value2 = @"Пункт";
            excelworksheet.Columns[7].ColumnWidth = 16;
            excelcells = excelworksheet.get_Range("H7", "H7");
            excelcells.Value2 = @"Информация";
            excelworksheet.Columns[8].ColumnWidth = 28;
            excelcells = excelworksheet.get_Range("I7", "I7");
            excelcells.Value2 = @"Сотрудник";
            excelworksheet.Columns[9].ColumnWidth = 20;

            MySqlCommand command = new MySqlCommand();

```

```

        command.CommandText = Tekzapros;//SqlLekarstvo;
        DataTable tbl = new DataTable();
        using (MySqlDataAdapter dataAdapter = new
MySqlDataAdapter(command.CommandText, Query.ConnStr))
        {
            dataAdapter.Fill(tbl);
        }
        for (j = 0; j < tbl.Rows.Count; j++)
        {
            //Код
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")] = tbl.Rows[j][0].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

            //дата начала
            DateTime dt =
DateTime.Parse(tbl.Rows[j][1].ToString());
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")] = dt.ToString("MM-dd-yyyy");
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

            //дата конца
            dt = DateTime.Parse(tbl.Rows[j][2].ToString());
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")] = dt.ToString("MM-dd-yyyy");
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

            //клиент
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")] = tbl.Rows[j][3].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

            //сумма
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")] = tbl.Rows[j][4].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

            //статус
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("F")] = tbl.Rows[j][5].ToString();

```

```

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("F")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("F")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        //пункт
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("G")] = tbl.Rows[j][6].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("G")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("G")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

        //информация
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("H")] = tbl.Rows[j][7].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("H")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("H")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlLeft;

        //сотрудник
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("I")] = tbl.Rows[j][8].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("I")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("I")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
    }
} //конец regim=1
else
{
    excelcells = excelworksheet.get_Range("A7", "A7");
    excelcells.Value2 = @"Код";
    excelworksheet.Columns[1].ColumnWidth = 7;
    excelcells = excelworksheet.get_Range("B7", "B7");
    excelcells.Value2 = @"Дата";
    excelworksheet.Columns[2].ColumnWidth = 10;
    excelcells = excelworksheet.get_Range("C7", "C7");
    excelcells.Value2 = @"Сумма";
    excelworksheet.Columns[3].ColumnWidth = 10;
    excelcells = excelworksheet.get_Range("D7", "D7");
    excelcells.Value2 = @"Статус";
    excelworksheet.Columns[4].ColumnWidth = 13;
    excelcells = excelworksheet.get_Range("E7", "E7");
    excelcells.Value2 = @"Информация";
}

```

```

        excelworksheet.Columns[5].ColumnWidth = 28;
        excelcells = excelworksheet.get_Range("F7", "F7");
        excelcells.Value2 = @"Сотрудник";
        excelworksheet.Columns[6].ColumnWidth = 20;

        MySqlCommand command = new MySqlCommand();
        command.CommandText = Tekzapros; //SqlLekarstvo;
        DataTable tbl = new DataTable();
        using (MySqlDataAdapter dataAdapter = new
MySqlDataAdapter(command.CommandText, Query.ConnStr))
        {
            dataAdapter.Fill(tbl);
        }
        for (j = 0; j < tbl.Rows.Count; j++)
        {

            //Код
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")] = tbl.Rows[j][0].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("A")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

            //дата
            DateTime dt =
DateTime.Parse(tbl.Rows[j][1].ToString());
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")] = dt.ToString("MM-dd-yyyy");
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("B")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

            //сумма
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")] = tbl.Rows[j][2].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("C")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

            //статус
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")] = tbl.Rows[j][3].ToString();
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("D")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;

            //информация
            excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")] = tbl.Rows[j][4].ToString();

```

```

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("E")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlLeft;

        //сотрудник
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("F")] = tbl.Rows[j][5].ToString();
        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("F")].WrapText = true; //Если текст не умещается в
одной ячейке, то будет перенос на следующие строки в пределах одной
ячейки

        excelworksheet.Cells[j + 8,
FormClient.AbcToIndex("F")].HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlRight;
    }
} //конец regim=2

    Object NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex("A")];
    excelcells = excelworksheet.get_Range("A7", NomCell);
    excelcells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;
    excelcells.VerticalAlignment =
Microsoft.Office.Interop.Excel.Constants.xlCenter;

    //Устанавливаем цвет обводки
    NomCell = excelworksheet.Cells[j + 7,
FormClient.AbcToIndex(ClassStr.RegimZakaz == 1 ? "I": "F")];
    excelcells = excelworksheet.get_Range("A7", NomCell);
    excelcells.Borders.ColorIndex = 1; //черный
    //Устанавливаем стиль и толщину линии
    excelcells.Borders.LineStyle =
Microsoft.Office.Interop.Excel.XlLineStyle.xlContinuous;
    excelcells.Borders.Weight =
Microsoft.Office.Interop.Excel.XlBorderWeight.xlMedium;//.xlThick;

    excelapp.Visible = true;

    object misValue = System.Reflection.Missing.Value;
    excelappworkbook.SaveAs(@fileReport,
Microsoft.Office.Interop.Excel.XlFileFormat.xlOpenXMLWorkbook, misValue,
        misValue, misValue, misValue,
Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive,
        misValue, misValue, misValue,
misValue, misValue);
    this.Enabled = true;
}

    private void comboBoxFilterPunkt_SelectedIndexChanged(object
sender, EventArgs e)
    {
        indexCheckPunkt =
comboBoxFilterPunkt.SelectedValue.ToString();

```

```

    }

    private void comboBoxFilterStatus_SelectedIndexChanged(object
sender, EventArgs e)
    {
        indexCheckStatus =
comboBoxFilterStatus.SelectedValue.ToString();
    }

    private void comboBoxFilterClient_SelectedIndexChanged(object
sender, EventArgs e)
    {
        indexCheckClient =
comboBoxFilterClient.SelectedValue.ToString();
    }

    //Кнопка - ВКЛ фильтр
    private void buttonFilterON_Click(object sender, EventArgs e)
    {
        string S_Find = "";

        if (checkBoxPunkt.Checked== false && checkBoxStatus.Checked ==
false &&
            checkBoxClient.Checked == false && checkBoxData.Checked
== false)
        {
            MessageBox.Show("Не установлен ни один из критериев
фильтра.");
            return;
        }

        if (checkBoxPunkt.Checked == true && comboBoxFilterPunkt.Text
== "")
        {
            MessageBox.Show("Установлен критерий поиска по пункту,
но сам пункт не выбран.");
            comboBoxFilterPunkt.Focus();
            return;
        }

        if (checkBoxStatus.Checked == true &&
comboBoxFilterStatus.Text == "")
        {
            MessageBox.Show("Установлен критерий поиска по статусу,
но сам статус не выбран.");
            comboBoxFilterStatus.Focus();
            return;
        }

        if (checkBoxClient.Checked == true &&
comboBoxFilterClient.Text == "")
        {
            MessageBox.Show("Установлен критерий поиска по клиенту,
но сам клиент не выбран.");
            comboBoxFilterClient.Focus();
            return;
        }
    }

```

```

        if (checkBoxData.Checked == true &&
(DateTime.Parse(dateFilter1.Value.ToString("yyyy-MM-dd")) >
DateTime.Parse(dateFilter2.Value.ToString("yyyy-MM-dd"))))
        {
            MessageBox.Show("Неверно указаны даты.");
            dateFilter1.Focus();
            return;
        }

        //Собираем строку фильтра
        int z = 0;
        if (checkBoxPunkt.Checked == true)
        {
            S_Find += "(Z.CodPunkt=" + indexCheckPunkt + ")";
            z = 1;
        }

        if (checkBoxStatus.Checked == true)
        {
            if (z == 1) S_Find += " and ";
            S_Find += "(Z.CodStatus=" + indexCheckStatus + ")";
            z = 1;
        }

        if (checkBoxClient.Checked == true)
        {
            if (z == 1) S_Find += " and ";
            S_Find += "(Z.CodClient=" + indexCheckClient + ")";
            z = 1;
        }

        if (checkBoxData.Checked == true)
        {
            if (z == 1) S_Find += " and ";
            S_Find += "(Z.DataStart between '" +
dateFilter1.Value.ToString("yyyy-MM-dd") + "' and '" +
dateFilter2.Value.ToString("yyyy-MM-dd") + "')";
            z = 1;
        }

        if (ClassStr.TekPrava == "Клиент")
            Tekzapros = SqlLekarstvo + " and " + S_Find + " order by
Z.DataStart desc";
        else
            Tekzapros = ClassStr.RegimZakaz == 1 ?
                SqlLekarstvo + " where " + "(Z.CodClient
is not null)and " + S_Find + " order by Z.DataStart desc":
                SqlPreparat+ " where " + "(Z.CodClient
is null)and " + S_Find + " order by Z.DataStart desc"
                ;

        GridZakaz.DataSource = Podkl.GetDataFromTable(Tekzapros);
    }

    private void buttonFilterOFF_Click(object sender, EventArgs e)
    {
        if (ClassStr.TekPrava == "Клиент")

```



```

        Tekzapros = SqlLekarstvo + " order by Z.DataStart desc";
    else
        Tekzapros = ClassStr.RegimZakaz == 1 ?
            SqlLekarstvo + " where " +
            "(Z.CodClient is not null) order by Z.DataStart desc" :
            SqlPreparat + " where " +
            "(Z.CodClient is null) order by Z.DataStart desc";
        GridZakaz.DataSource = Podkl.GetDataFromTable(Tekzapros);
    }
}

```

Листинг Е.42 – Класс FormZakazInfo

```

public partial class FormZakazInfo : Form
{
    Query Podkl; //Определим наш объект класса
    public string text_ID;
    public int RegimZakaz, CheckZakaz;

    public FormZakazInfo()
    {
        InitializeComponent();

        //Инициализируем. В качестве параметра передаем строку
        подключения
        Podkl = new Query(Query.ConnStr);
    }

    //Кнопка Выход
    private void But_Exit_Click(object sender, EventArgs e)
    {
        CheckZakaz = -1;
        Close();
    }

    //Клавиши ESC и ENTER
    private void DBEdit_Inform_KeyPress(object sender,
    KeyPressEventEventArgs e)
    {
        if (e.KeyChar == (char)27)
        {
            But_Exit_Click(sender, e);
        }
        else if (e.KeyChar == (char)13)
        {
            But_Save_Click(sender, e);
        }
    }

    //Кнопка Сохранить
    private void But_Save_Click(object sender, EventArgs e)
    {
        //Выполняем проверки заполненных полей данных
        if (DBEdit_Summa.Text == "")
        {
            MessageBox.Show("Введите сумму заказа");
            DBEdit_Summa.Focus();
            return;
        }
    }
}

```

```

    }

    if (ClassStr.RegimZakaz == 1)
    {
        if (DateTime.Parse(dateDataStart.Value.ToString("yyyy-MM-dd")) > DateTime.Parse(dateDataEnd.Value.ToString("yyyy-MM-dd")))
        {
            MessageBox.Show("Неверно указаны даты.");
            dateDataStart.Focus();
            return;
        }
        if (comboBoxClient.Text == "")
        {
            MessageBox.Show("Укажите клиента");
            comboBoxClient.Focus();
            return;
        }

        if (comboBoxPunkt.Text == "")
        {
            MessageBox.Show("Укажите пункт");
            comboBoxPunkt.Focus();
            return;
        }
    }

    if (comboBoxStatus.Text == "")
    {
        MessageBox.Show("Укажите статус заказа");
        comboBoxStatus.Focus();
        return;
    }

    if (comboBoxUsers.Text == "")
    {
        MessageBox.Show("Укажите сотрудника");
        comboBoxUsers.Focus();
        return;
    }

    string StrInform = "";
    string StrJson = "";
    switch (RegimZakaz)
    {
        case 1: //Режим редактирования
            ClassZakaz Per_1 = new
ClassZakaz(int.Parse(text_ID.ToString()),

            DateTime.Parse(dateDataStart.Text),

            ClassStr.RegimZakaz == 1 ? DateTime.Parse(dateDataEnd.Text) :
DateTime.Parse(dateDataStart.Text),

            DBEdit_Inform.Text,

            int.Parse(DBEdit_Summa.Text),

```

```

        ClassStr.RegimZakaz == 1 ?
int.Parse(comboBoxClient.SelectedValue.ToString()):0,

        ClassStr.RegimZakaz == 1 ?
int.Parse(comboBoxPunkt.SelectedValue.ToString()):0,

        int.Parse(comboBoxStatus.SelectedValue.ToString()),

        int.Parse(comboBoxUsers.SelectedValue.ToString())
);
        StrJson = JsonConvert.SerializeObject(Per_1);
        Podkl.Update_JSON("Table_Zakaz", StrJson);
        StrInform = "Изменение. Сумма=" + DBEdit_Summa.Text
+ ", Клиент=" +
        (ClassStr.RegimZakaz == 1 ?
comboBoxClient.Text : "") + ", Пункт=" +
        (ClassStr.RegimZakaz == 1 ?
comboBoxPunkt.Text : "") + ", Статус=" +
        comboBoxStatus.Text + ",
Сотрудник=" + comboBoxUsers.Text;
        FormMain.WriteInHistory(StrInform,
int.Parse(text_ID.ToString()));
        break;
        case 2: //Режим создания
            ClassZakaz Per_2 = new ClassZakaz(0,

            DateTime.Parse(dateDataStart.Text),

            ClassStr.RegimZakaz == 1 ? DateTime.Parse(dateDataEnd.Text) :
            DateTime.Parse(dateDataStart.Text),

            DBEdit_Inform.Text,

            int.Parse(DBEdit_Summa.Text),

            ClassStr.RegimZakaz == 1 ?
int.Parse(comboBoxClient.SelectedValue.ToString()) : 0,

            ClassStr.RegimZakaz == 1 ?
int.Parse(comboBoxPunkt.SelectedValue.ToString()) : 0,

            int.Parse(comboBoxStatus.SelectedValue.ToString()),

            int.Parse(comboBoxUsers.SelectedValue.ToString())
);
            StrJson = JsonConvert.SerializeObject(Per_2);
            CheckZakaz = Podkl.Add_JSON("Table_Zakaz",
StrJson);
            StrInform = "Создание. Сумма=" + DBEdit_Summa.Text
+ ", Клиент=" +
            (ClassStr.RegimZakaz == 1 ?
comboBoxClient.Text : "") + ", Пункт=" +
            (ClassStr.RegimZakaz == 1 ?
comboBoxPunkt.Text : "") + ", Статус=" +
            comboBoxStatus.Text + ",
Сотрудник=" + comboBoxUsers.Text;
            FormMain.WriteInHistory(StrInform, CheckZakaz);

```

```

        break;
    }

    Close();
}

//Клавиша ESC
private void FormZakazInfo_KeyPress(object sender,
KeyPressEventArgs e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
    }
}

//Обработчик загрузки формы
private void FormZakazInfo_Load(object sender, EventArgs e)
{
    DataTable bufferZakaz = new DataTable();
    DataTable bufferStatus = new DataTable();
    DataTable bufferClient = new DataTable();
    DataTable bufferPunkt = new DataTable();
    DataTable bufferUsers = new DataTable();

    //Заполняем combobox для статусов
    //Инициализируем - передаем запрос и объект класса
    SqlConnection - connection
    string StrZ = "select * from Table_Status order by Name";
    MySqlDataAdapter datadapterStatus = new MySqlDataAdapter(StrZ,
    Query.ConnStr);
    datadapterStatus.Fill(bufferStatus); //вызываем метод Fill,
    чтобы после запроса заполнил буферную таблицу данными

    comboBoxStatus.DataSource = bufferStatus;
    comboBoxStatus.DisplayMember = "Name";
    comboBoxStatus.ValueMember = "CodStatus";

    //Заполняем combobox для сотрудников
    //Инициализируем - передаем запрос и объект класса
    SqlConnection - connection
    StrZ = "select concat(FamU, '
    ', LEFT(NameU,1), '.', LEFT(OtchU,1), '.') as FIOUsers, CodUsers from
    Table_Users order by FamU, NameU, OtchU";
    MySqlDataAdapter datadapterUsers = new MySqlDataAdapter(StrZ,
    Query.ConnStr);
    datadapterUsers.Fill(bufferUsers); //вызываем метод Fill,
    чтобы после запроса заполнил буферную таблицу данными

    comboBoxUsers.DataSource = bufferUsers;
    comboBoxUsers.DisplayMember = "FIOUsers";
    comboBoxUsers.ValueMember = "CodUsers";

    int index = 0;
    if (ClassStr.RegimZakaz == 2)
    {
        labelData1.Text = "Дата";
        dateDataEnd.Visible = false;
    }
}

```

```

        labelData2.Visible = false;
        PanelData.Size = new Size { Width = 767, Height = 300 };
        this.Size = new Size { Width = 793, Height = 400 };
        switch (RegimZakaz)
        {
            case 1: //Режим редактирования
                string zapros = "select
Z.CodZakaz,Z.DataStart,";
                zapros += "Z.Summa,ST.Name as
Status,Z.Inform,";
                zapros += "concat(US.FamU, '
',LEFT(US.NameU,1),'.',LEFT(US.OtchU,1),'.') as Users";
                zapros += " from ((Table_Zakaz as Z ";
                zapros += "inner join Table_Status as ST on
Z.CodStatus = ST.CodStatus) ";
                zapros += "inner join Table_Users as US on
Z.CodUsers = US.CodUsers) ";
                zapros += "where Z.CodZakaz = ";
                zapros += text_ID;
                bufferZakaz = Podkl.GetDataFromTable(zapros);

                //Заполняем поля для ввода информации
                dateDataStart.Text =
bufferZakaz.Rows[0]["DataStart"].ToString();
                DBEdit_Summa.Text =
bufferZakaz.Rows[0]["Summa"].ToString();
                DBEdit_Inform.Text =
bufferZakaz.Rows[0]["Inform"].ToString();

                string s_Status =
bufferZakaz.Rows[0]["Status"].ToString();
                index = comboBoxStatus.FindString(s_Status);
                comboBoxStatus.SelectedIndex = index;

                string s_Users =
bufferZakaz.Rows[0]["Users"].ToString();
                index = comboBoxUsers.FindString(s_Users);
                comboBoxUsers.SelectedIndex = index;
                break;
            case 2: //Режим создания
                //Заполняем поля для ввода информации
                dateDataStart.Text = DateTime.Now.ToString();
                DBEdit_Summa.Text = "";
                DBEdit_Inform.Text = "";
                if (comboBoxStatus.Items.Count > 0)
                    comboBoxStatus.SelectedIndex = 0;
                if (comboBoxUsers.Items.Count > 0)
                    comboBoxUsers.SelectedIndex = 0;
                break;
        }
        return;
    }

    //Заполняем combobox для клиентов
    //Инициализируем - передаем запрос и объект класса
    SqlConnection - connection

```

```

        StrZ = "select concat(FamCl, '
', LEFT(NameCl,1), '.', LEFT(OtchCl,1), '.') as FIOClient, CodClient from
Table_Client order by FamCl, NameCl, OtchCl";
        MySqlDataAdapter datadapterClient = new MySqlDataAdapter(StrZ,
Query.ConnStr);
        datadapterClient.Fill(bufferClient); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными

        comboBoxClient.DataSource = bufferClient;
        comboBoxClient.DisplayMember = "FIOClient";
        comboBoxClient.ValueMember = "CodClient";

        //Заполняем combobox для пунктов
        //Инициализируем - передаем запрос и объект класса
SqlConnection - connection
        StrZ = "select concat(Name, ' (' ,Address,')') as Punkt, CodPunkt
from Table_Punkt order by Name";
        MySqlDataAdapter datadapterPunkt = new MySqlDataAdapter(StrZ,
Query.ConnStr);
        datadapterPunkt.Fill(bufferPunkt); //вызываем метод Fill,
чтобы после запроса заполнил буферную таблицу данными

        comboBoxPunkt.DataSource = bufferPunkt;
        comboBoxPunkt.DisplayMember = "Punkt";
        comboBoxPunkt.ValueMember = "CodPunkt";

        switch (RegimZakaz)
        {
            case 1: //Режим редактирования
                string zapros = "select
Z.CodZakaz, Z.DataStart, Z.DataEnd, ";
                zapros += "concat(CL.FamCl, '
', LEFT(CL.NameCl,1), '.', LEFT(CL.OtchCl,1), '.') as Client, ";
                zapros += "Z.Summa, ST.Name as
Status, concat(PN.Name, ' (' ,PN.Address,')') as Punkt, Z.Inform, ";
                zapros += "concat(US.FamU, '
', LEFT(US.NameU,1), '.', LEFT(US.OtchU,1), '.') as Users";
                zapros += " from (((Table_Zakaz as Z ";
                zapros += "inner join Table_Client as CL on
Z.CodClient = CL.CodClient) ";
                zapros += "inner join Table_Punkt as PN on
Z.CodPunkt = PN.CodPunkt) ";
                zapros += "inner join Table_Status as ST on
Z.CodStatus = ST.CodStatus) ";
                zapros += "inner join Table_Users as US on
Z.CodUsers = US.CodUsers) ";
                zapros += "where Z.CodZakaz = ";
                zapros += text_ID;
                bufferZakaz = Podkl.GetDataFromTable(zapros);

                //Заполняем поля для ввода информации
                dateDataStart.Text =
bufferZakaz.Rows[0]["DataStart"].ToString();
                dateDataEnd.Text =
bufferZakaz.Rows[0]["DataEnd"].ToString();
                DBEdit_Summa.Text =
bufferZakaz.Rows[0]["Summa"].ToString();

```

```

        DBEdit_Inform.Text =
bufferZakaz.Rows[0]["Inform"].ToString();

        string s_Status =
bufferZakaz.Rows[0]["Status"].ToString();
        index = comboBoxStatus.FindString(s_Status);
        comboBoxStatus.SelectedIndex = index;

        string s_Client =
bufferZakaz.Rows[0]["Client"].ToString();
        index = comboBoxClient.FindString(s_Client);
        comboBoxClient.SelectedIndex = index;

        string s_Punkt =
bufferZakaz.Rows[0]["Punkt"].ToString();
        index = comboBoxPunkt.FindString(s_Punkt);
        comboBoxPunkt.SelectedIndex = index;

        string s_Users =
bufferZakaz.Rows[0]["Users"].ToString();
        index = comboBoxUsers.FindString(s_Users);
        comboBoxUsers.SelectedIndex = index;
        break;
    case 2: //Режим создания
        //Заполняем поля для ввода информации
        dateDataStart.Text = DateTime.Now.ToString();
        dateDataEnd.Text = DateTime.Now.ToString();
        DBEdit_Summa.Text = "";
        DBEdit_Inform.Text = "";
        if (comboBoxStatus.Items.Count > 0)
            comboBoxStatus.SelectedIndex = 0;
        if (comboBoxClient.Items.Count > 0)
            comboBoxClient.SelectedIndex = 0;
        if (comboBoxPunkt.Items.Count > 0)
            comboBoxPunkt.SelectedIndex = 0;
        if (comboBoxUsers.Items.Count > 0)
            comboBoxUsers.SelectedIndex = 0;
        break;
    }
}

private void DBEdit_Summa_KeyPress(object sender, KeyPressEventArgs
e)
{
    if (e.KeyChar == (char)27)
    {
        But_Exit_Click(sender, e);
        return;
    }
    else if (e.KeyChar == (char)13)
    {
        But_Save_Click(sender, e);
        return;
    }

    char ch = e.KeyChar;
    if (!char.IsDigit(ch) && ch != 8)
    {

```

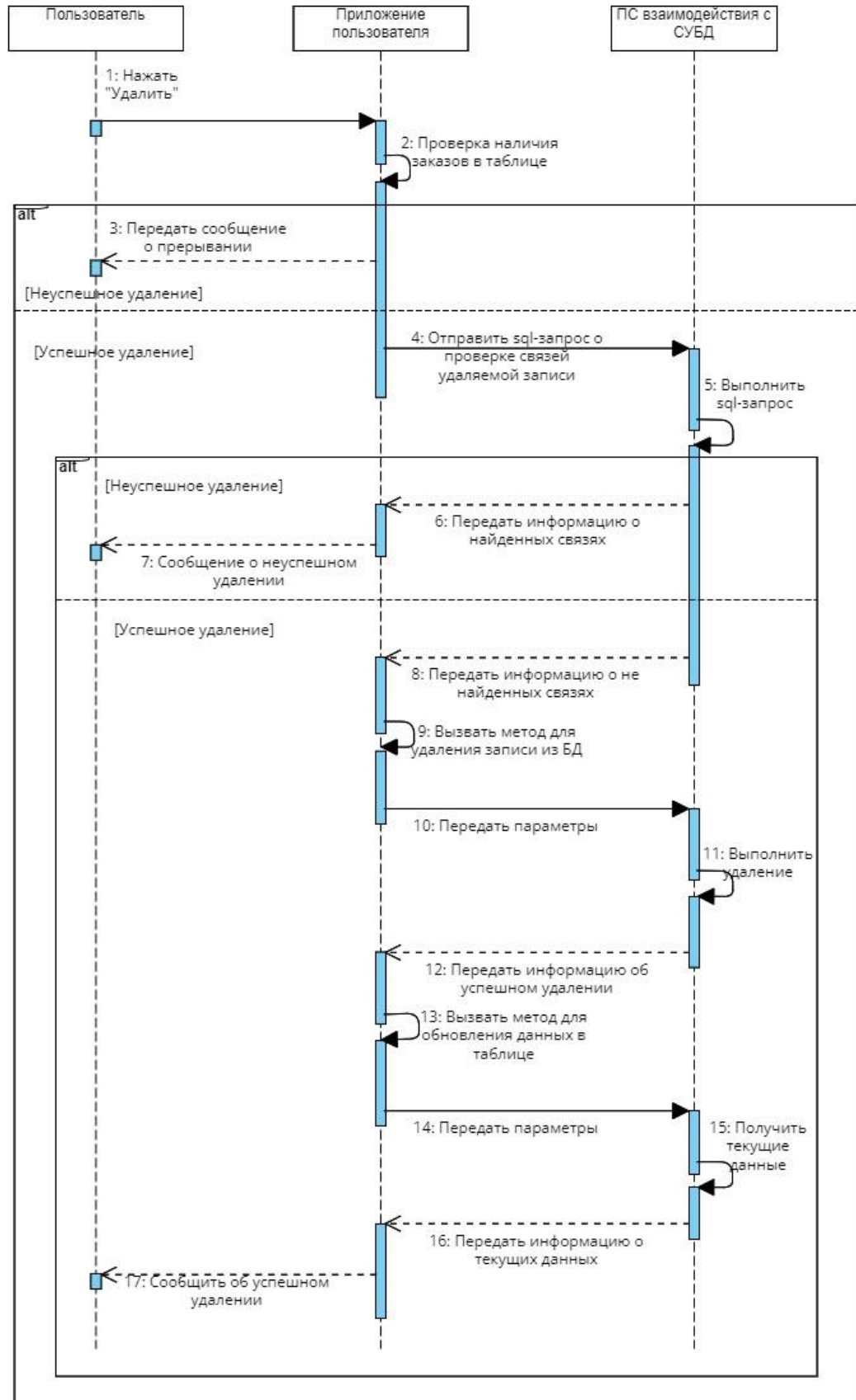
```
        e.Handled = true;
    }
}

//Обработчик отображения формы
private void FormZakazInfo_Shown(object sender, EventArgs e)
{
    dateDataStart.Focus();
}
}
```


Приложение Ж
(рекомендуемое)

Диаграмма последовательности предметной области

sd Приложение обработки и анализа индивидуальных заказов на изготовление лекарственных средств



Приложение И
(рекомендуемое)
Результаты тестирования приложения

Формирование состава заказа

Данные заказа

	Код	Дата	Сумма	Статус	Пункт	Клиент
▷	12	14.06.2024	500	Открыт	Кировский	Иванов Н.Е.

Состав заказа

	Название	Кол	Цена
▷			

Лекарств. средства

	Название	Кол	Цена

Отсутствуют данные в списке лекарственных средств

OK

Укажите количество: 2

Убрать из заказа

Добавить в заказ

Выход

Рисунок И.1 – Результат тестирования покрытия операторов первого теста

Формирование состава заказа

Данные заказа

	Код	Дата	Сумма	Статус	Пункт	Клиент
▶	12	14.06.2024	500	Открыт	Кировский	Иванов Н.Е.

Состав заказа

	Название	Кол	Цена
*			

Лекарств. средства

	Название	Кол	Цена
▶	АК	15	700

Укажите количество больше 0

OK

Укажите количество: 0

Убрать из заказа

Добавить в заказ

Выход

Рисунок И.2 – Промежуточный результат тестирования покрытия операторов первого теста

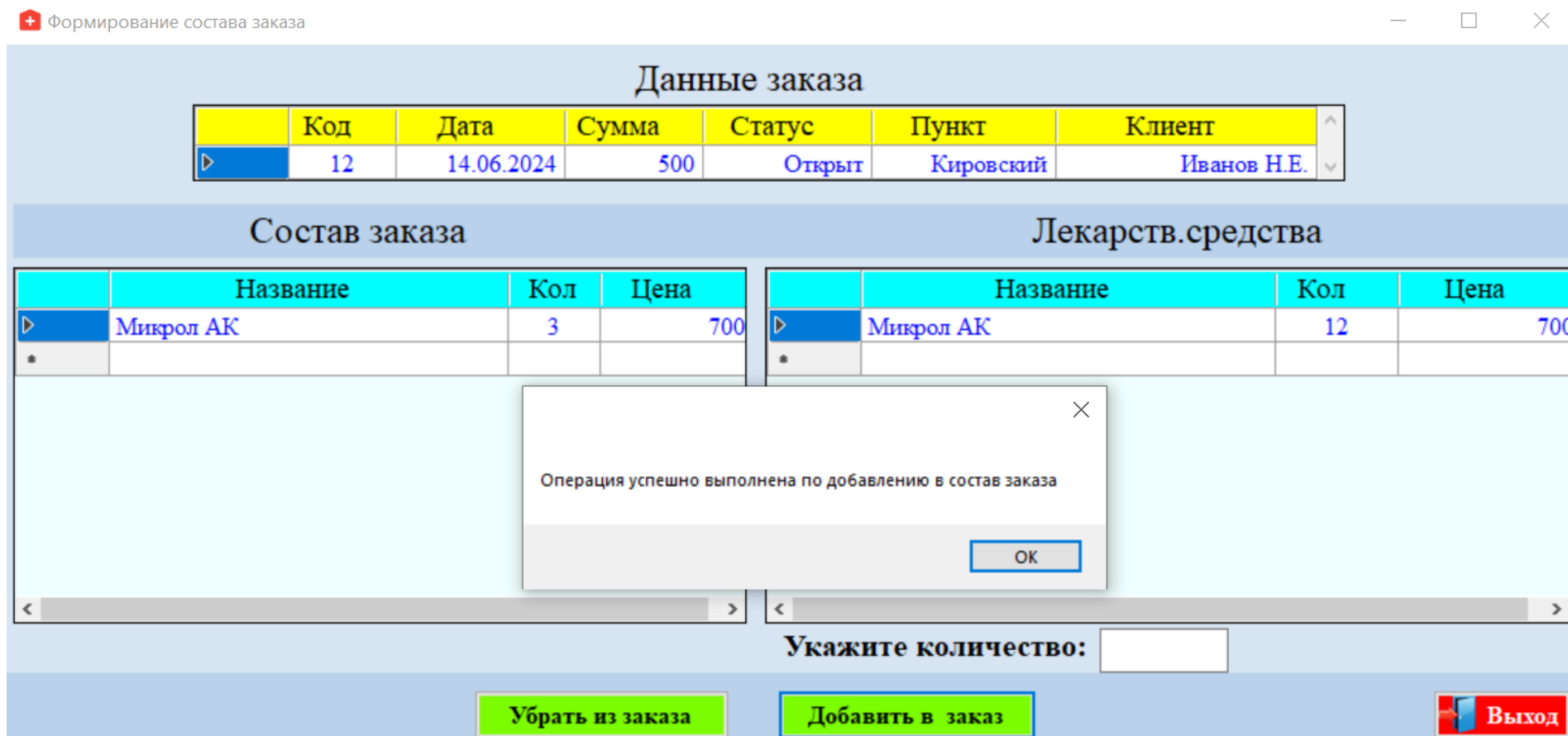


Рисунок И.3 – Результат тестирования покрытия операторов второго теста

+

Формирование состава заказа

—

□

×

Данные заказа

	Код	Дата	Сумма	Статус	Пункт	Клиент
▶	12	14.06.2024	500	Открыт	Кировский	Иванов Н.Е.

Состав заказа

Лекарств.средства

	Название	Кол	Цена
▶	Микрол АК	3	700
	Кетарофил	3	2000
*			

	Название	Кол	Цена
	Микрол АК	12	700
▶	Кетарофил	14	2000
*			

×

Операция успешно выполнена по добавлению в состав заказа

ОК

Укажите количество:

Убрать из заказа

Добавить в заказ

Выход

Рисунок И.4 – Результат тестирования покрытия решений

Формирование состава заказа

— □ ×

Данные заказа

	Код	Дата	Сумма	Статус	Пункт	Клиент	^
▷	12	14.06.2024	500	Открыт	Кировский	Иванов Н.Е.	▽

Состав заказа

Лекарств. средства

	Название	Кол	Цена
▷	Микрол АК	3	700
	Кетарофил	3	2000
*			

	Название	Кол	Цена
	Микрол АК	12	700
▷	Кетарофил	14	2000
*			

Укажите количество больше 0

OK

Укажите количество: 0

Убрать из заказа

Добавить в заказ

Выход

Формирование состава заказа

Данные заказа

	Код	Дата	Сумма	Статус	Пункт	Клиент
▶	12	14.06.2024	500	Открыт	Кировский	Иванов Н.Е.

Состав заказа

	Название	Кол	Цена
▶			

Лекарств.средства

	Название	Кол	Цена
▶			

Отсутствуют данные в списке лекарственных средств

OK

Укажите количество: 6

Убрать из заказа

Добавить в заказ

Выход

Рисунок И.6 – Результат тестирования покрытия маршрутов второго теста

+

Формирование состава заказа

— □ ×

Данные заказа

	Код	Дата	Сумма	Статус	Пункт	Клиент
▷	12	14.06.2024	500	Открыт	Кировский	Иванов Н.Е.

Состав заказа

	Название	Кол	Цена
▷	Микрол АК	6	700
*			

Лекарств. средства

	Название	Кол	Цена
▷	Микрол АК	9	700
	Кетарофил	17	2000
*			

Операция успешно выполнена по добавлению в состав заказа

OK

Укажите количество:

Убрать из заказа

Добавить в заказ

Выход

Рисунок И.7 – Результат тестирования покрытия маршрутов третьего теста

Приложение К
(рекомендуемое)
Листинг класса тестирования

```

[TestClass]
public class UnitTest1
{
    [TestMethod]
    public void But_Save_Click_WithValidData()
    {
        // Arrange
        var mockPodkl = new Mock<IPodkl>();
        var form = new FormOplata(mockPodkl.Object); // Передаем
макет в конструктор формы

        form.DBEdit_Summa = new TextBox { Text = "1000" }; //
Заполняем поле суммы
        form.DateTimeData = new DateTimePicker
        {
            Value = DateTime.Now,
            CustomFormat = "dd-MM-yyyy HH:mm:ss"
        }; // Устанавливаем текущую дату
        form.RegimOplata = 1; // Устанавливаем режим редактирования
        form.text_ID = new Label { Text = "1" }; // Устанавливаем
идентификатор

        mockPodkl.Setup(m => m.Update_JSON("Table_Oplata",
It.IsAny<string>())); // Ожидаем вызов Update_JSON с этими параметрами

        // Act
        form.But_Save_Click(null, EventArgs.Empty);

        // Assert
        mockPodkl.Verify(m => m.Update_JSON("Table_Oplata",
It.IsAny<string>()), Times.Once); // Проверяем, что метод Update_JSON был
вызван один раз
        Assert.IsTrue(form.IsDisposed); // Проверяем, что форма была
закрыта
    }

    [TestMethod]
    public void But_Save_Click_WithInvalidData()
    {
        // Arrange
        var mockPodkl = new Mock<IPodkl>();
        var form = new FormOplata(mockPodkl.Object); // Передаем
макет в конструктор формы

        form.DBEdit_Summa = new TextBox(); // Оставляем поле суммы
пустым
        form.DateTimeData = new DateTimePicker { Value = DateTime.Now
}; // Устанавливаем текущую дату
        form.RegimOplata = 1; // Устанавливаем режим редактирования
        form.text_ID = new Label { Text = "1" }; // Устанавливаем
идентификатор

        mockPodkl.Setup(m => m.Update_JSON(It.IsAny<string>(),
It.IsAny<string>())); // Заглушка для метода Update_JSON

        // Act
        form.But_Save_Click(null, EventArgs.Empty);
    }
}

```

```

        // Assert
        mockPodkl.Verify(m => m.Update_JSON(It.IsAny<string>()),
            It.IsAny<string>()), Times.Never); // Проверяем, что метод Update_JSON не
        был вызван
        Assert.IsFalse(form.IsDisposed); // Проверяем, что форма не
        была закрыта
    }
}

// Интерфейс для заглушки Podkl
public interface IPodkl
{
    void Update_JSON(string table, string json);
}

// Заглушка для класса Podkl
public static class Podkl
{
    public static void Update_JSON(string table, string json)
    {
        // Реализация метода
    }

    public static int Add_JSON(string table, string json)
    {
        // Реализация метода
        return 1; // Возвращаем значение для теста
    }
}

// Заглушка для класса ClassStr
public static class ClassStr
{
    public static int TekCodZakaz = 1; // Пример значения для теста
}

// Заглушка для класса ClassOplata
public class ClassOplata
{
    public ClassOplata(int id, DateTime date, int summa, int codZakaz)
    {
        // Инициализация полей
    }
}

// Заглушка для формы
public class FormOplata : Form
{
    public TextBox DBEdit_Summa { get; set; }
    public DateTimePicker DateTimeData { get; set; }
    public int RegimOplata { get; set; }
    public Label text_ID { get; set; }
    private readonly IPodkl _podkl;

    public FormOplata(IPodkl podkl)
    {
        _podkl = podkl;
    }
}

```

```

public void But_Save_Click(object sender, EventArgs e)
{
    if (DBEdit_Summa.Text == "")
    {
        MessageBox.Show("Введите сумму оплаты по заказу");
        DBEdit_Summa.Focus();
        return;
    }

    string StrJson = "";
    int CheckOplata = 0;

    switch (RegimOplata)
    {
        case 1: //Режим редактирования
            ClassOplata Per_1 = new
ClassOplata(int.Parse(text_ID.Text),
DateTime.Parse(DateTimeData.Value.ToString("dd-MM-yyyy HH:mm:ss")),
int.Parse(DBEdit_Summa.Text),
ClassStr.TekCodZakaz
);
            StrJson = JsonConvert.SerializeObject(Per_1);
            _podkl.Update_JSON("Table_Oplata", StrJson); //
Вызываем метод через макет
            break;
        case 2: //Режим создания
            ClassOplata Per_2 = new ClassOplata(0,
DateTime.Parse(DateTimeData.Value.ToString("dd-MM-yyyy HH:mm:ss")),
int.Parse(DBEdit_Summa.Text),
ClassStr.TekCodZakaz
);
            StrJson = JsonConvert.SerializeObject(Per_2);
            Podkl.Add_JSON("Table_Oplata", StrJson);
            break;
    }

    Close();
}
}

```