

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

Факультет
вычислительной техники

Кафедра
МО и ПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

Мобильное приложение аптеки

Студент	_____	<u>Карпович Светлана Владимировна</u>
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	<u>Балашова И.Ю.</u>
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролер	_____	<u>Такташкин Д.В</u>
	(подпись, дата)	(фамилия, инициалы)

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

Заведующий кафедрой	_____	<u>Козлов А.Ю.</u>
	(подпись)	(фамилия, инициалы)

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

Секретарь ГЭК	_____	<u>Попова Н.А.</u>
	(подпись)	(фамилия, инициалы)

Пенза, 2023

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
«__» _____ 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Карпович Светлана Владимировна гр. 19ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Мобильное приложение аптеки

Тема утверждена приказом ПГУ № _____ от "___" _____ 2023 г.

3. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Балашова И.Ю.

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

1) *Назначение: мобильное приложение для аптеки*

2) *Среда выполнения: операционная система Windows, Android.*

Языковые средства разработки: Kotlin, Transact-SQL, UML.

*Инструментальные средства: Android Studio, SQL Server Management Studio,
Visual Paradigm.*

Технологии разработки ООП.

3) *Основные функции приложения:*

а) регистрация и авторизация;

б) поиск товаров;

в) просмотр данных о товаре;

г) добавление позиций и оформление заказа;

д) напоминание о приеме лекарств;

е) удаление учетной записи;

5. Перечень подлежащих разработке вопросов:

1) Анализ предметной области и постановка задачи;
2) Проектирование;
3) Реализация;
4) Контроль качества программного обеспечения;
5) План разработки бюджета;
6) Оформление пояснительной записки.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%	13.04.2023 – 27.04.2023	
2	Проектирование	25%	27.04.2023 – 10.05.2023	
3	Реализация	35%	10.05.2023 – 30.06.2023	
4	Контроль качества программного обеспечения	10%	31.05.2023– 02.06.2023	
5	План разработки бюджета	5%	02.06.2023– 03.06.2023	
6	Оформление пояснительной записки	15%	03.06.2023– 17.06.2023	

Дата выдачи задания «22» _____ декабря _____ 2022 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 78 листов, 41 рисунок, 22 таблицы, 10 использованных источников и 2 приложения.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, БАЗА ДАННЫХ, ИНФОРМАЦИОННАЯ СИСТЕМА, АПТЕКА, ANDROID, KOTLIN, MICROSOFT SQL SERVER.

Объект исследования – средства автоматизации рабочих процессов аптеки.

Цель выпускной квалификационной работы – разработать мобильное приложение заказов медикаментов покупателями аптеки.

Технологии разработки – разработка мобильного приложения проводилась на языке программирования Kotlin, СУБД Microsoft SQL Server. В качестве среды разработки использовался редактор программного кода Android Studio и менеджер баз данных SQL Server Management Studio.

Среда исполнения – Android.

Результаты работы – в ходе выполнения выпускной квалификационной работы было создано мобильное приложение аптеки.

					ПГУ 09.03.04 – 08BP191.09 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Карпович С.В.			Мобильное приложение аптеки. Пояснительная записка	Лит.	Лист
Пров.		Балашова И.Ю.					4
							78
Н. Контр.		Такташкин Д.В.				Группа 19ВП1	
Утв.							

Содержание

Содержание.....	5
Введение.....	7
1 Анализ предметной области и постановка задачи	9
1.1 Обзорный анализ мобильных приложений аптек	9
1.1.1 Мобильное приложение «ЕАптека»	9
1.1.2 Мобильное приложение «Здравсити»	11
1.1.3 Мобильное приложение «Apteka.ru».....	12
1.2 Сравнительный анализ продуктов-аналогов.....	13
1.3 Анализ требований	15
2 Проектирование мобильного приложения аптеки	20
2.1 Моделирование бизнес-процессов предметной области.....	20
2.2 Разработка концептуальной модели базы данных	20
2.3 Моделирование базы данных	25
2.3.1 Логическая модель базы данных	25
2.3.2 Физическая модель базы данных.....	27
3 Реализация мобильного приложения аптеки	30
3.1 Развертывание и наполнение базы данных тестовыми данными	30
3.1.1 Развертывание базы данных в среде MS SQL Server.....	30
3.1.2 Наполнение базы данных тестовыми данными.....	33
3.2 Разработка серверной части приложения.....	36
3.2.1 Разработка запросов к базе данных и представлений для реализации аналитических бизнес-процессов	36
3.2.2 Разработка функций и хранимых процедур для реализации учетных пользовательских бизнес-процессов	37
3.2.3 Разработка триггеров для реализации учетных автоматических бизнес-процессов и обновления представлений	41
3.3 Разработка интерфейса клиентского приложения.....	44
4 Тестирование приложения	48

4.1	Выбор режима тестирования приложения	48
4.2	Тестирование серверной части приложения	50
4.3	Анализ качества кода.....	54
4.3.1	Измерение устойчивости	54
5	Планирование разработки и оценка бюджета.....	58
	Заключение	60
	Список использованных источников	62
	Приложение А.. Листинг для базы данных	63
	Приложение Б. Листинг приложения.....	70

Введение

Аптека – особая специализированная организация системы здравоохранения, занимающаяся изготовлением, фасовкой, анализом и продажей лекарственных средств [1]. Зачастую аптеки не только реализуют готовую продукцию, но также производят свои медикаменты, оказывая медицинскую помощь населению.

Для облегчения условий труда аптекарей важна автоматизация производства – применение ЭВМ в управлении производственными процессами аптеки [2]. Необходимо активно повышать технологичность аптек и применять технологические и информационные решения.

Из этого следует, что существует объективная необходимость в создании, разработке и развитии информационных систем аптек, а также создания и поддержки сервисов, направленных как на внутренний аспект производства и аптечного бизнеса, так и ориентированных на клиента. Аптечные информационные системы упрощают не только работу самих аптек и ее работников, но и значительно повышают доступность для их клиентов, так как могут применяться в клиент-ориентированных приложениях. Для этого в том числе различным сетям аптечных пунктов необходимы мобильные приложения: удобное и доступное для пользователя приложение значительно повышает продажи. Ведь сегодня практически у каждого есть телефон с доступом в интернет.

Актуальность разработки мобильного приложения аптеки обусловлена объективной необходимостью для населения доступности медицинских товаров и услуг.

Объектом исследования в данной работе выступают средства автоматизации рабочих процессов аптеки.

Целью выпускной квалификационной работы является разработка мобильного приложения заказов медикаментов покупателями аптеки.

Для достижения данной цели необходимо выполнить следующие задачи:

- провести сравнительный анализ программных продуктов-аналогов и выделить основные результаты;
- провести анализ требований к разрабатываемому программному продукту;
- осуществить проектирование программных средств и структур данных разрабатываемого программного продукта, описать их с использованием методологии UML;
- реализовать программные средства в соответствии с поставленными требованиями;
- протестировать полученные при реализации результаты;
- проанализировать качество написанного кода;
- провести планирование и расчет бюджета разрабатываемого программного продукта.

1 Анализ предметной области и постановка задачи

1.1 Обзорный анализ мобильных приложений аптек

Основная задача мобильного приложения аптеки – позволить пользователю ознакомиться с каталогом лекарственных препаратов, отыскать необходимый и заказать его доставку на дом или забронировать для покупки в ближайшей аптеке.

Для проведения сравнительного анализа программных продуктов-аналогов приложений аптек определены приложения с высоким рейтингом, на которые следует обратить внимание при разработке приложения. Рассмотрим наиболее популярные и высоко оцененные приложения, и выделим возможности, предоставляемые ими пользователям.

1.1.1 Мобильное приложение «ЕАптека»

Приложение «ЕАптека» – это интернет-аптека с возможностью доставки лекарств и витаминов как на дом, так и в удобную пользователю аптеку.

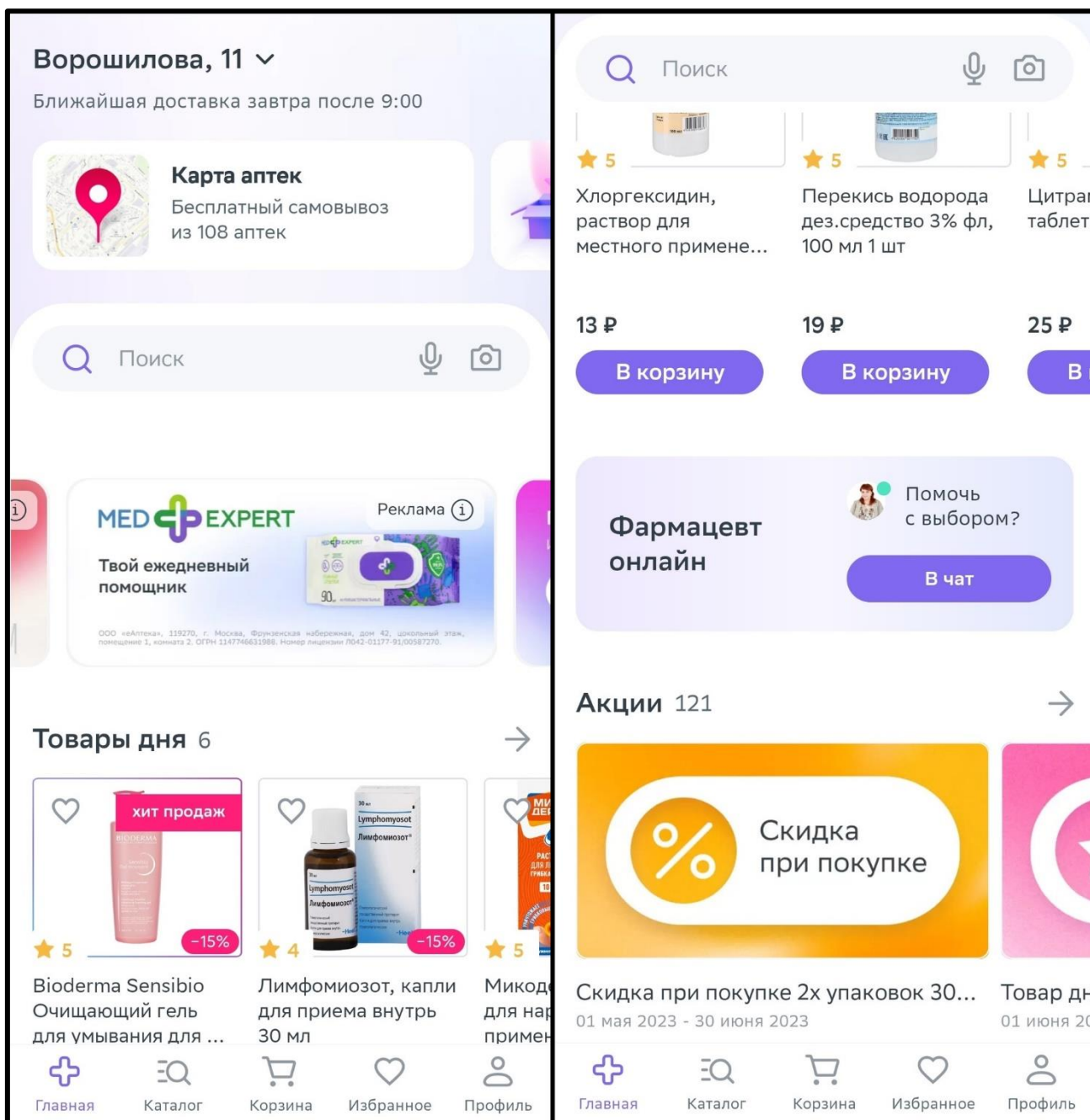
Среди плюсов данного приложения можно выделить следующие:

- оно удобно в использовании;
- в приложении имеется чат с фармацевтом;
- приложение хорошо адаптировано для лиц с ограниченными возможностями.

Среди минусов приложения можно выделить:

- невозможность просмотреть препараты-аналоги;
- невозможность установки пароля на вход;
- невозможность удалить учетную запись внутри приложения.

На рисунке ниже представлена главная страница, где есть поиск (рисунок 1, а), а также чат с фармацевтом (рисунок 1, б).



а) Строка поиска

б) Чат с фармацевтом

Рисунок 1 – Интерфейс приложения «ЕАптека»

Помимо этого, на главной странице можно увидеть множество различных подборок медикаментов, бонусные программы и акции.

На нижней панели помимо главной страницы есть разделы с каталогом, корзиной, избранными товарами, а также профилем пользователя.

1.1.2 Мобильное приложение «Здравсити»

«Здравсити» – первый крупномасштабный проект, объединяющий аптеки по всей России. Это онлайн сервис для поиска и заказа лекарств, предоставляющий услуги по доставке как в ближайшую аптеку, так и на дом.

Плюсы приложения:

- удобное в использовании;
- предоставляет на выбор множество способов оплаты;
- предоставляет информацию об аналогах.

Минусы приложения:

- невозможность установки пароля на вход;
- невозможность удалить учетную запись внутри приложения.

Из особенностей стоит отметить, что при первом запуске приложения имеется небольшая вводная секция, знакомящий нового пользователя с особенностями приложения (рисунок 2).

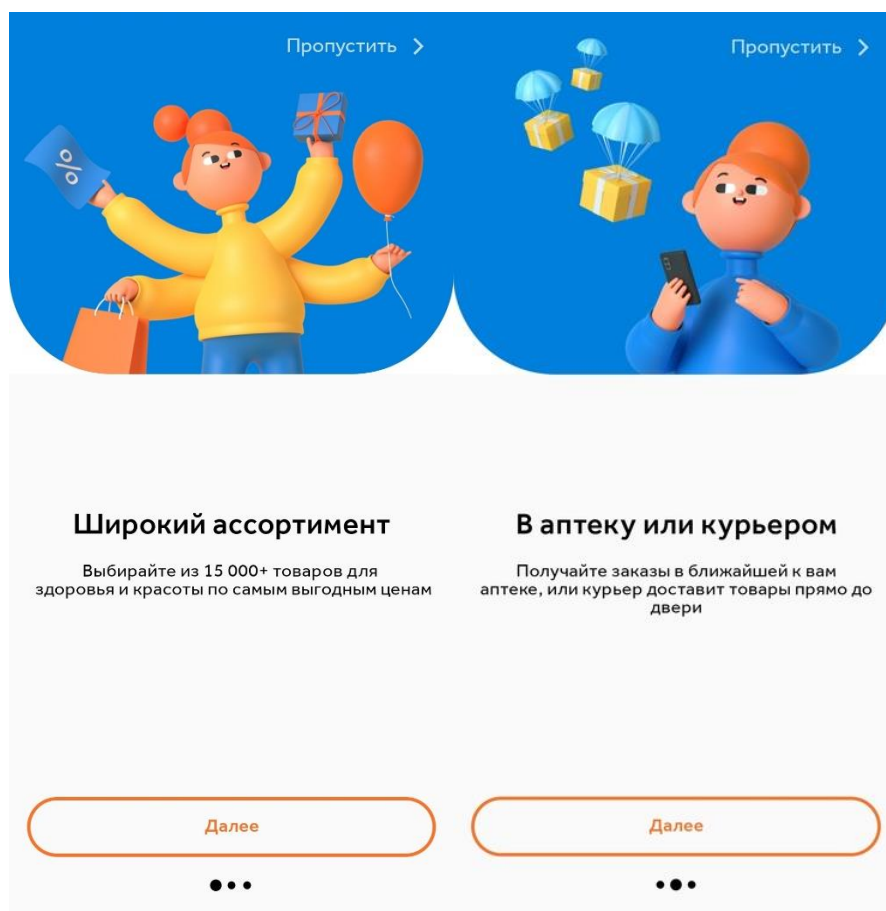


Рисунок 2 – Вводная секция приложения «Здравсити»

В остальном главная страница и нижняя панель практически идентичны по содержанию и функциональности. Здесь отсутствует чат с фармацевтом на главной странице, а также он не поддерживает текстовый формат.

1.1.3 Мобильное приложение «Apteka.ru»

Мобильное приложение «Apteka.ru» дает возможность быстро находить необходимые товары и заказывать их доставку в удобную пользователю аптеку.

Плюсы:

- удобное в использовании;
- имеется чат с фармацевтом;
- имеется напоминание о приеме лекарств;
- имеется информация об аналогах препаратов.

Минусы:

- мало фильтров поиска;
- оплата только в пункте выдачи.

На рисунке 3 – интерфейс главной страницы приложения.

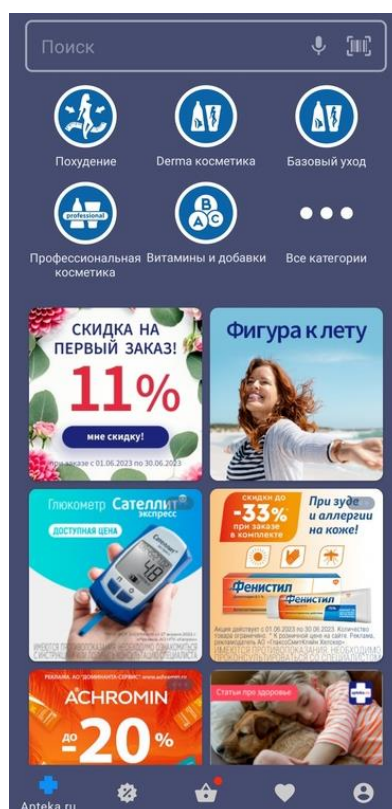


Рисунок 3 – Интерфейс приложения «Apteka.ru»

Интерфейс здесь проще – на главной странице только поиск и минималистично оформленные акции и подборки. Интересно то, что в нижней панели для акций есть отдельная вкладка. В остальном приложение оформлено также, однако уступает вышеописанным приложениям по функциональности.

1.2 Сравнительный анализ продуктов-аналогов

Опираясь на выделенные плюсы и минусы эталонных мобильных приложений в данной отрасли, в качестве критериев выбраны наиболее важные возможности, которые напрямую влияют на функциональность приложения.

При выделении и оценке значимости была использована информация, предоставленная «Роскачеством», производившим подобный рейтинговый анализ мобильных приложений аптек в 2022 году [3].

Для оценки описанных выше критериев используется шкала соответствия, где 0 – «полное несоответствие», 1 – «частичное соответствие», 2 – «полное соответствие» требованиям. В таблице 1 представлены характеристики оценки.

Таблица 1 – Характеристики оценки

Балл	Характеристика оценки
0	Критерий не реализован
1	Критерий реализован частично
2	Критерий реализован полностью

Произведем расчет аддитивной суммы интегральной оценки для каждого сравниваемого продукта по следующей формуле:

$$O_{\text{пп}} = \sum_{i=1}^n (Z_i),$$

где $O_{\text{пп}}$ – интегральная оценка программного продукта;

n – количество критериев сравнения заявленных продуктов согласно таблице критериев;

Z_i – значение выполнения критерия.

Проведем оценку приложений по выделенным критериям. Результаты отражены в таблице 2.

Таблица 2 – Результаты оценки мобильных приложений-аналогов

№	Наименование	Программный продукт, №		
		1	2	3
1	Регистрация и авторизация.	2	2	2
2	Поиск товара (поисковая строка, каталог, подборка).	2	1	1
3	Наличие карточки товара.	1	2	2
4	Наличие корзины.	2	2	2
5	Оформление заказа.	2	2	1
6	Напоминание о приеме лекарств.	0	0	2
7	Уведомление о поступлении нужного препарата в наличие.	0	0	0
8	Чат с фармацевтом.	2	1	0
9	Бесплатность продукта.	2	2	2
10	Безопасность.	1	1	1
Интегральная оценка:		14	13	13

Подводя итог сравнения, можно сделать вывод, что данные приложения заслуженно занимают первые места в рейтингах мобильных приложений аптек, так как тесты показывают примерной одинаковый результат с минимальным отрывом. Тем не менее, стоит выделить среди них приложение «ЕАптека», которое оказывает себя безусловным лидером по функциональности в сравнении со всеми мобильными приложениями аптек.

Кроме того, общей проблемой среди данных мобильных приложений была выявлена недостаточная функциональность в области безопасности. Ни одно из приложений не предоставляет пользователю механизмов для самостоятельно защиты: личных паролей, а также возможности удалить учетную запись самостоятельно.

Для наглядности сравнительного анализа, полученные результаты представлены в графической форме на рисунке 4.

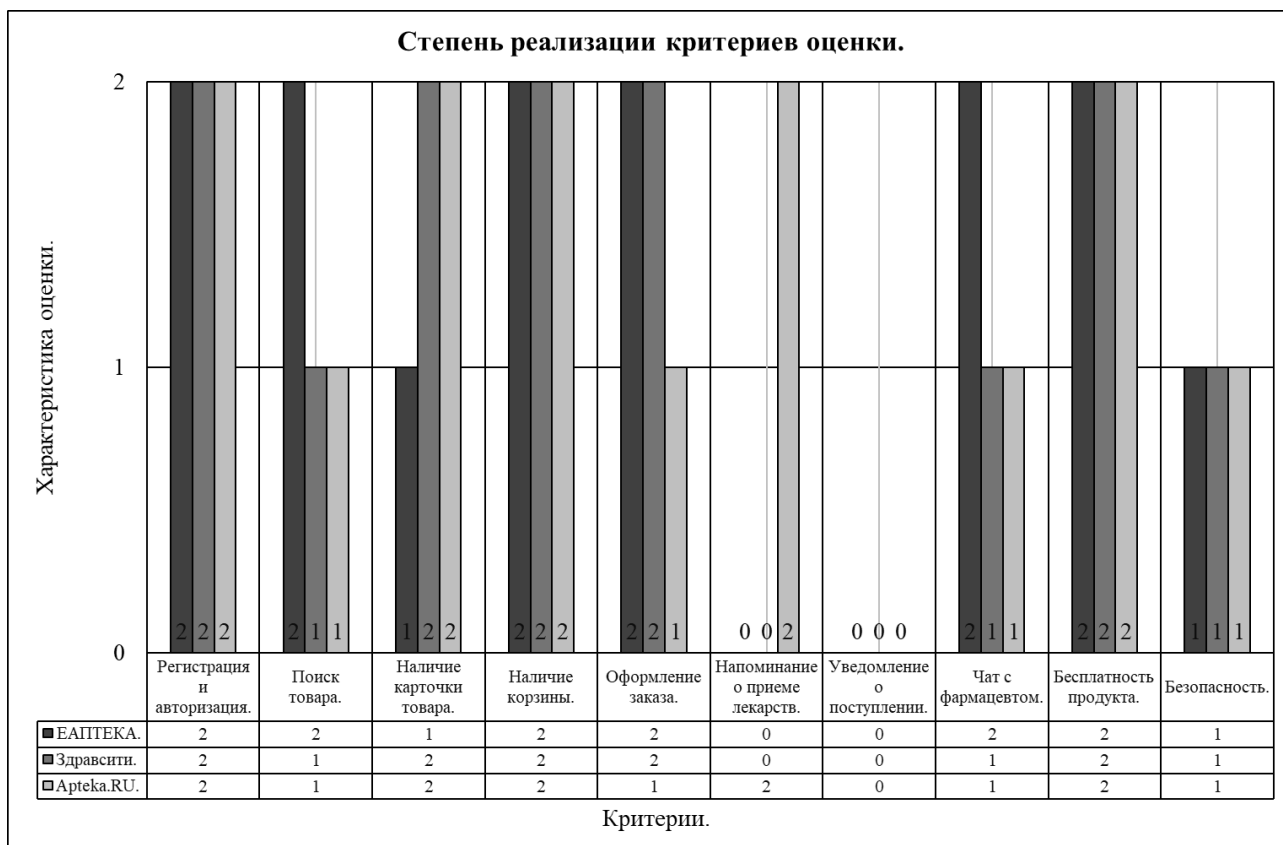


Рисунок 4 – Степень реализации критериев оценки

При разработке стоит равняться на общую функциональность данных мобильных приложений как на обязательные атрибуты таких приложений, но приложить усилия к исправлению их главных недостатков.

1.3 Анализ требований

В соответствии с выделенными критериями была разработана диаграмма вариантов использования (рисунок 5) и выделены требования:

- регистрация и авторизация;
- поиск товаров;
- карточка товара, предоставляющая всю важную информацию о лекарственном средстве;
- корзина и оформление товара;
- напоминание о приеме лекарств;
- уведомление о поступлении товара;
- удаление учетной записи.

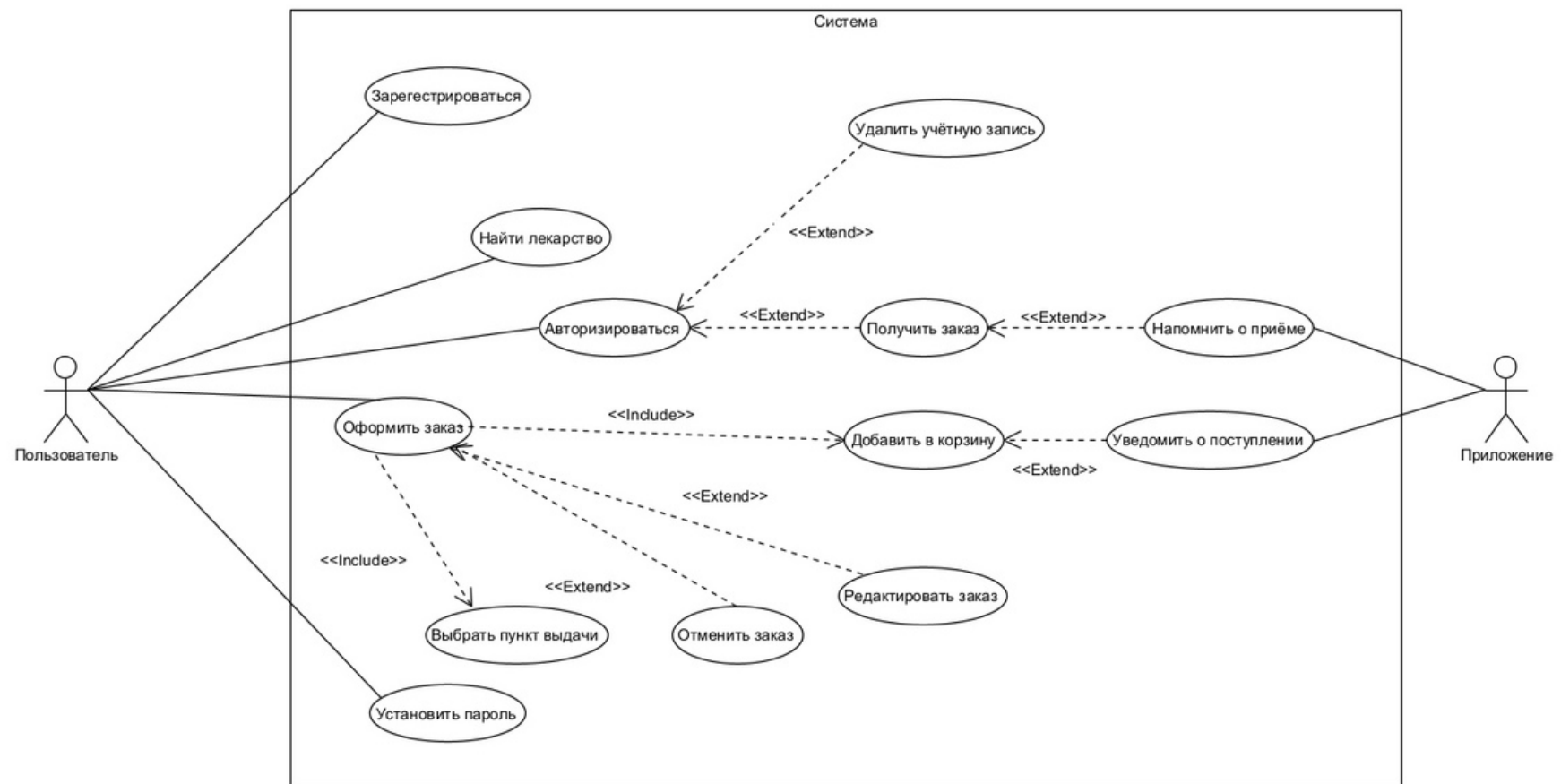


Рисунок 5 – Диаграмма вариантов использования

Для диаграммы вариантов использования были также составлены спецификации наиболее значимых: в таблице 3 приведена спецификация варианта использования «Оформить заказ».

Таблица 3 – Спецификация варианта использования «Оформить заказ»

Наименование:	оформить заказ.
ID:	1.
Краткое описание:	пользователь оформляет заказ на покупку лекарственных средств в мобильном приложении.
Действующие лица:	пользователь.
Предусловие:	пользователь авторизован и нашел необходимые лекарства с помощью поиска.
Основной поток:	а) пользователь добавляет необходимые товары в корзину: 1) если добавленного товара нет в наличии, пользователю придет уведомление, когда он вновь появится в продаже; 2) при необходимости пользователь может изменить или удалить заказ; б) пользователь выбирает пункт выдачи заказа и способ оплаты; в) проверив детали заказа, пользователь заканчивает оформление заказа.
Постусловие:	заказ оформлен и поступил в обработку.

В таблице 4 представляет спецификацию варианта использования «Напомнить о приеме».

Таблица 4 – Спецификация варианта использования «Напомнить о приеме»

Наименование:	напомнить о приеме
ID:	2
Краткое описание:	приложение согласно назначенному расписанию уведомляет пользователя о необходимости принять лекарство
Действующие лица:	Приложение
Предусловие:	пользователь приобрел и получил лекарство, для которого назначил напоминания о приеме
Основной поток:	— пользователь настраивает расписание уведомлений; — мобильное приложение выводит уведомление с напоминанием.
Постусловие:	пользователь получил уведомление с напоминанием о приеме лекарства.

Для отображения работы программы была разработана диаграмма деятельности, изображенная на рисунке 6.

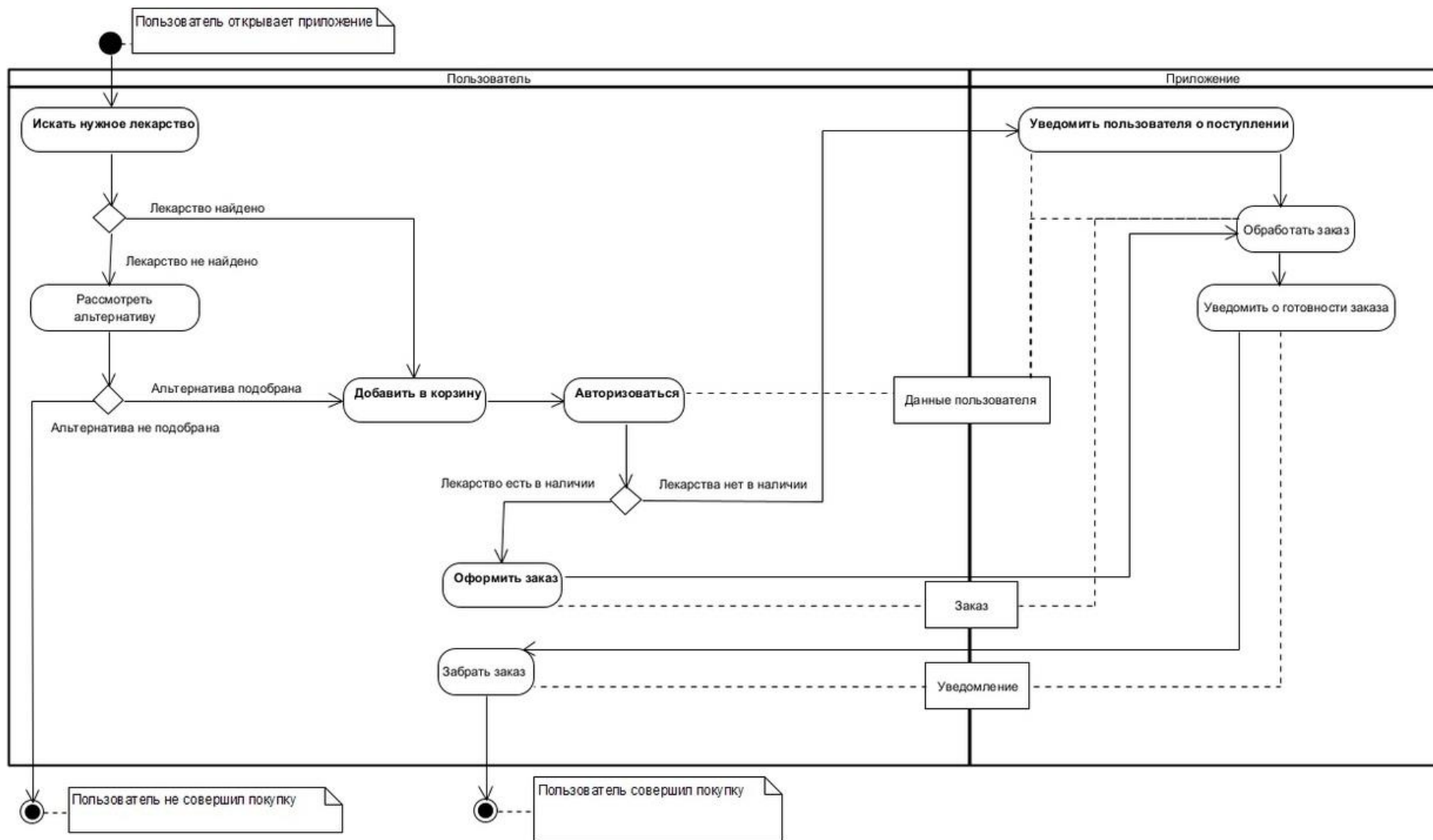


Рисунок 6 – Диаграмма деятельности

Таким образом, в результате анализа функциональных требований были выделены варианты использования программы и спецификации наиболее важных из них. Кроме того, была составлена диаграмма деятельности, что позволяет отобразить последовательность действий пользователя и системы в ходе их взаимодействия.

2 Проектирование мобильного приложения аптеки

2.1 Моделирование бизнес-процессов предметной области

На основе диаграммы вариантов использования были выделены бизнес-процессы для информационной системы аптеки (таблица 5).

Таблица 5 – Спецификация бизнес-процессов приложения

№ п/п	Бизнес-процесс	Исполнитель	Входные данные		Выходные данные	
			Поставщик	Содержание	Потребитель	Содержание
1	Оформление заказа	Покупатель	Система	Список товаров	Покупатель	Номер заказа
2	Изменение количества покупаемых товаров	Покупатель	Покупатель	Список лекарств в корзине	Покупатель	Модифицированный список лекарств в корзине
3	Отмена заказа покупателем	Покупатель	Покупатель	Заказ	Сотрудник	Удаление заказа
4	Завершение заказа	Покупатель	Покупатель	Полученный заказ	Система	Занесение информации о заказах за конкретную дату в базу

Прежде чем выполнить любой бизнес-процесс в рамках системы, пользователь должен пройти авторизацию.

2.2 Разработка концептуальной модели базы данных

Концептуальная модель представляет собой описание основных сущностей (таблиц) и связей между ними без учета принятой модели БД и синтаксиса целевой СУБД. Часто на такой модели отображаются только имена сущностей без указания их атрибутов [4]. На рисунке 7 – диаграмма «сущность-связь», выполняющая роль концептуальной модели базы данных.



Рисунок 7 – Диаграмма концептуальной модели базы данных

Была составлена таблица сущностей базы данных, приведенная в таблице 6.

Таблица 6 – Сущности базы данных

Название сущности	Идентификатор сущности	Назначение сущности
Покупатель	Buyer	Содержит сведения о покупателях
Товар	Medicine	Содержит сведения о лекарствах
Заказ покупателя	BOrder	Содержит сведения о заказах покупателей
Пункт заказа	POrder	Содержит сведения о строках заказов
Поставщик	Supplier	Содержит сведения о поставщиках
Товары поставщика	SupMeds	Содержит сведения о лекарствах, поставляемых конкретным поставщиком

Для сущностей предметной области были сформированы связи, описание связей для сущностей БД приведено в таблице 7.

Таблица 7 – Табличное описание связей

Первая сущность	Вторая сущность	Связь
Покупатель	Заказ покупателя	Один ко многим
Заказ покупателя	Пункт выдачи	Многие к одному
	Часть заказа	Один ко многим
Часть заказа	Товар	Один к одному
Товары поставщика	Товар	Один к одному
Товар	Действующее вещество	Многие к одному
	Категория товара	Многие ко многим
	Форма выпуска	Многие ко многим
	Производитель	Многие к одному

Наборы атрибутов для каждой сущности предметной области и их характеристики приведены ниже.

В таблице 8 отражены атрибуты сущности «Покупатель». Данная сущность содержит в себе информацию о клиенте и его заказах.

Таблица 8 – Покупатель

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Buyer	Номер покупателя в системе	Целый	6	‘000001-999999’	Нет	РК	000123
PNumber	Номер телефона покупателя	Символьный	11	‘0-9’	Нет		88005553535
NOrders	Количество заказов	Целый	3	‘001-999’	Нет		14

В таблице 9 отражена сущность «Товар», с информацией о товарах.

Таблица 9 – Товар

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Medicine	Номер лекарства в системе	Целый	6	‘000001-999999’	Нет	РК	000111
Name	Название лекарства	Символьный	30	‘а-я’, ‘А-Я’, ‘ ’	Нет		Анальгин
Cost	Стоимость	Число с плавающей точкой	6	‘0000,00-9999,99’	Нет		99,90
ID_ActiveSub	Действующее вещество	Целый	3	‘001-999’	Нет		1
ID_Category	Категория	Целый	2	‘01-99’	Нет		1
ID_Form	Форма выпуска	Целый	1	‘0-9’	Нет		1
ID_Manufacturer	Производитель	Целый	2	‘01-99’	Нет		1
DateManufacture	Дата производства	Дата	8	‘01.01.2022-31.12.2099’	Нет		01.03.2022

Сущность «Действующее вещество» объединяет аналоги. Атрибуты данной сущности представлены в таблице 10.

Таблица 10 – Действующее вещество

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_ActiveSub	Номер вещества в системе	Целый	1	‘0-9’	Нет	РК	1
ActiveSub	Наименование вещества	Символьный	50	‘а-я’, ‘А-Я’, ‘ , ’	Нет		Ксилометазолин

Сущность «Категория товара» объединяет товары в группы по назначению и действию. Атрибуты данной сущности представлены в таблице 11.

Таблица 11 – Категория товара

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Category	Номер категории в системе	Целый	1	'0-9'	Нет	РК	1
Category	Наименование категории	Символьный	50	'а-я', 'А-Я', ','	Нет		Жаропонижающее

Сущность «Форма выпуска» содержит различные формы выпуска лекарств (таблица 12).

Таблица 12 – Форма выпуска

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Form	Номер формы выпуска в системе	Целый	1	'0-9'	Нет	РК	1
Form	Форма выпуска	Символьный	50	'а-я', 'А-Я', ','	Нет		Капли

Сущность «Производитель» содержит перечень производителей медикаментов (таблица 13).

Таблица 13 – Производитель

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Manufacturer	Номер производителя в системе	Целый	1	'0-9'	Нет	РК	1
Manufacturer	Производитель	Символьный	50	'а-я', 'А-Я', ','	Нет		Тева

Сущность «Заказ покупателя» содержит информацию о конкретном заказе конкретного пользователя и содержит информацию, необходимую для его исполнения (таблица 14).

Таблица 14 – Заказ покупателя

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Order	Номер заказа в системе	Целый	6	'000001-999999'	Нет	РК	000111
DateOrd	Дата заказа	Дата	8	'01.01.2022-31.12.2099'	Нет		01.12.2022
NPhone	Номер телефона покупателя	Символьный	30	'0-9'	Нет		88005553535
Cost	Стоимость заказа	Число с плавающей точкой	6	'0000,00-9999,99'	Нет		199,90
ID_Place	Пункт выдачи	Целый	1	'0-9'	Нет		1
ID_Worker	Номер сотрудника в системе	Целый	6	'000001-999999'	Нет		000111

Сущность «Пункт выдачи» содержит перечень доступных аптек (таблица 15).

Таблица 15 – Пункт выдачи

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Place	Номер должности в системе	Целый	1	'0-9'	Нет	РК	1
Place	Наименование должности	Символьный	50	'а-я', 'А-Я', ' '	Нет		Менеджер по закупкам

Сущность «Часть заказа» содержит в себе информацию не по всему заказу, а по конкретным его позициям (таблица 16).

Таблица 16 – Часть заказа

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_PartOfOrder	Номер пункта заказа в системе	Целый	6	'000001-999999'	Нет	РК	000111
ID_Order	Номер заказа в системе	Целый	6	'000001-999999'	Нет		000011
ID_Medicine	ID лекарства	Целый	6	'000001-999999'	Нет		000001
Count	Количество	Целый	3	'001-999'	Нет		3
Cost	Общая стоимость	Число с плавающей точкой	6	'0000,00-9999,99'	Нет		210,90

Сущность «Поставщик» содержит перечень поставщиков и информацию о

них (таблица 17).

Таблица 17 – Поставщик

Идентификатор столбца	Содержательное описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Supplier	Номер поставщика в системе	Целый	6	‘000001-999999’	Нет	PK	000111
Adress	Адрес поставщика	Символьный	50	‘а-я’, ‘А-Я’, ‘,’	Нет		г. Пенза, ул. Мира, д. 3
PNumber	Номер телефона	Символьный	11	‘0-9’	Нет		88005553535

Таким образом была разработка концептуальная модель базы данных и приведены описания атрибутов выделенных сущностей.

2.3 Моделирование базы данных

2.3.1 Логическая модель базы данных

Логическая модель – модель базы данных, выраженная в понятиях модели данных [5]. Логическая модель расширяет концептуальную путем определения для сущностей их атрибутов, описаний и ограничений, уточняет состав сущностей и взаимосвязи между ними.

На основе анализа предметной области и результатов моделирования бизнес-процессов и концептуальной модели была составлена диаграмма логической модели базы данных в нотации «сущность-связь», представленная на рисунке 8.

На диаграмме сущности «Покупатель» и «Заказ покупателя» связаны отношением «один ко многим» – у покупателя может быть множество заказов.

Сущности «Пункт выдачи» и «Заказ покупателя» связаны отношением «один ко многим» – заказ можно доставить только в один пункт, но в пункте может выдаваться множество заказов.

Сущности «Заказ покупателя» и «Пункт заказа покупателя» связаны отношением «один ко многим» – один заказ может включать в себя несколько товаров, но один товар может входить только в один заказ.

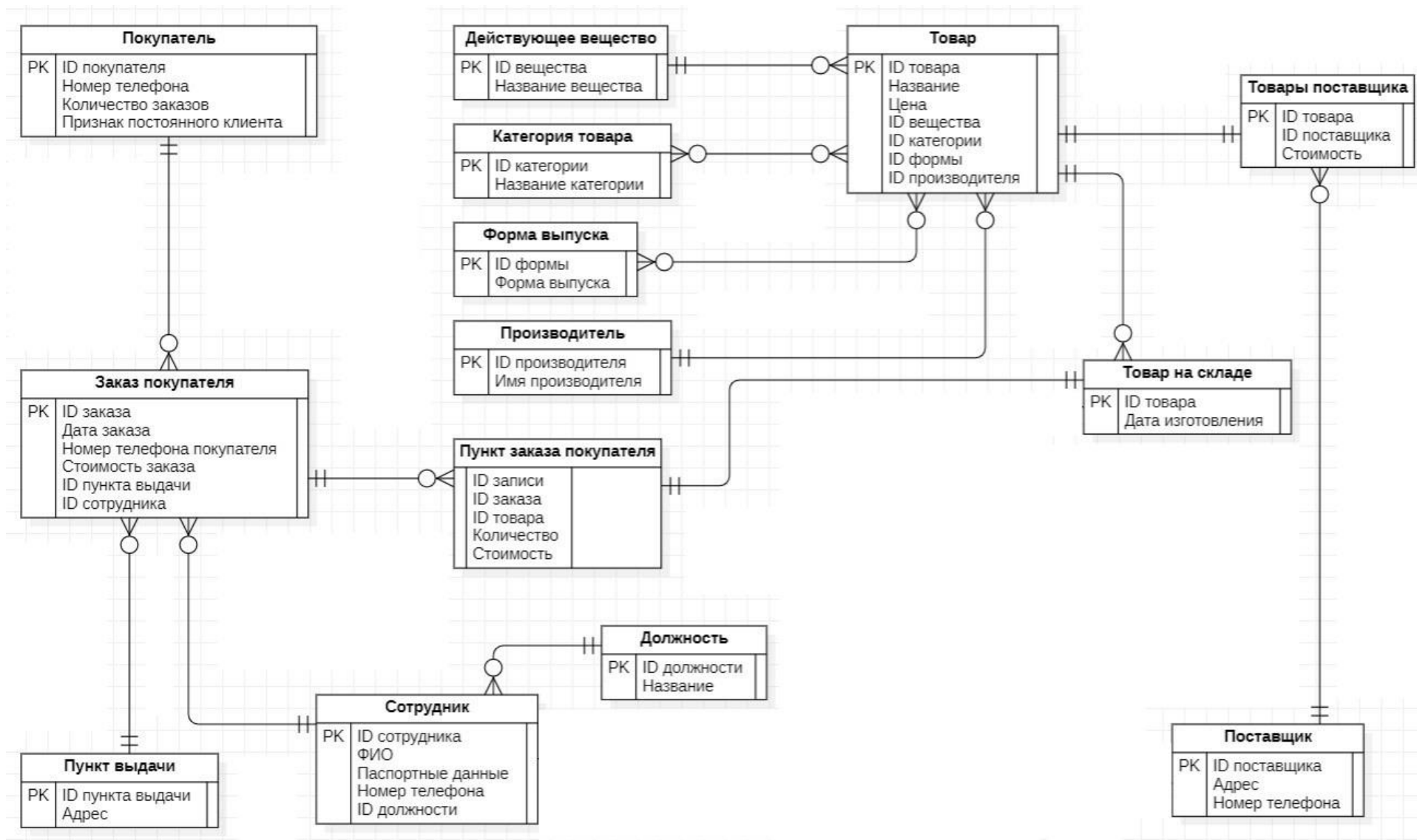


Рисунок 8 – Диаграмма логической модели базы данных

Сущности «Заказ покупателя» и «Сотрудник» связаны отношением «многие к одному», так как один сотрудник может собирать несколько заказов.

Сущности «Сотрудник» и «Должность» связаны отношением «один к одному», так как один сотрудник может занимать только одну должность.

Сущности «Пункт заказа покупателя» и «Товар» связаны отношением «один к одному», так как каждый пункт заказа представляет собой один товар.

Сущности «Товар» и «Действующее вещество» связаны отношением «один к одному», потому что, как правило, каждое лекарство имеет основное действующее вещество.

Сущности «Товар», «Категория товара» и «Форма выпуска» связаны отношением «один ко многим», так как один товар может относиться к различным категориям лекарственных средств и иметь различные формы выпуска.

Сущности «Товар» и «Производитель» связаны отношением «многие к одному», потому что у товара может быть только один производитель, но каждый производитель может производить несколько товаров.

Сущности «Поставщик» и «Товары поставщика» связаны отношением «один ко многим», так как один поставщик поставяет множество товаров.

Сущности «Товары поставщика» и «Товары» связаны отношением «один к одному», так как товар – это товар.

На логической модели базы данных видно взаимоотношение сущностей между друг другом.

2.3.2 Физическая модель базы данных

Физическая модель базы данных была разработана на основе концептуальной и логической моделей (рисунок 9).

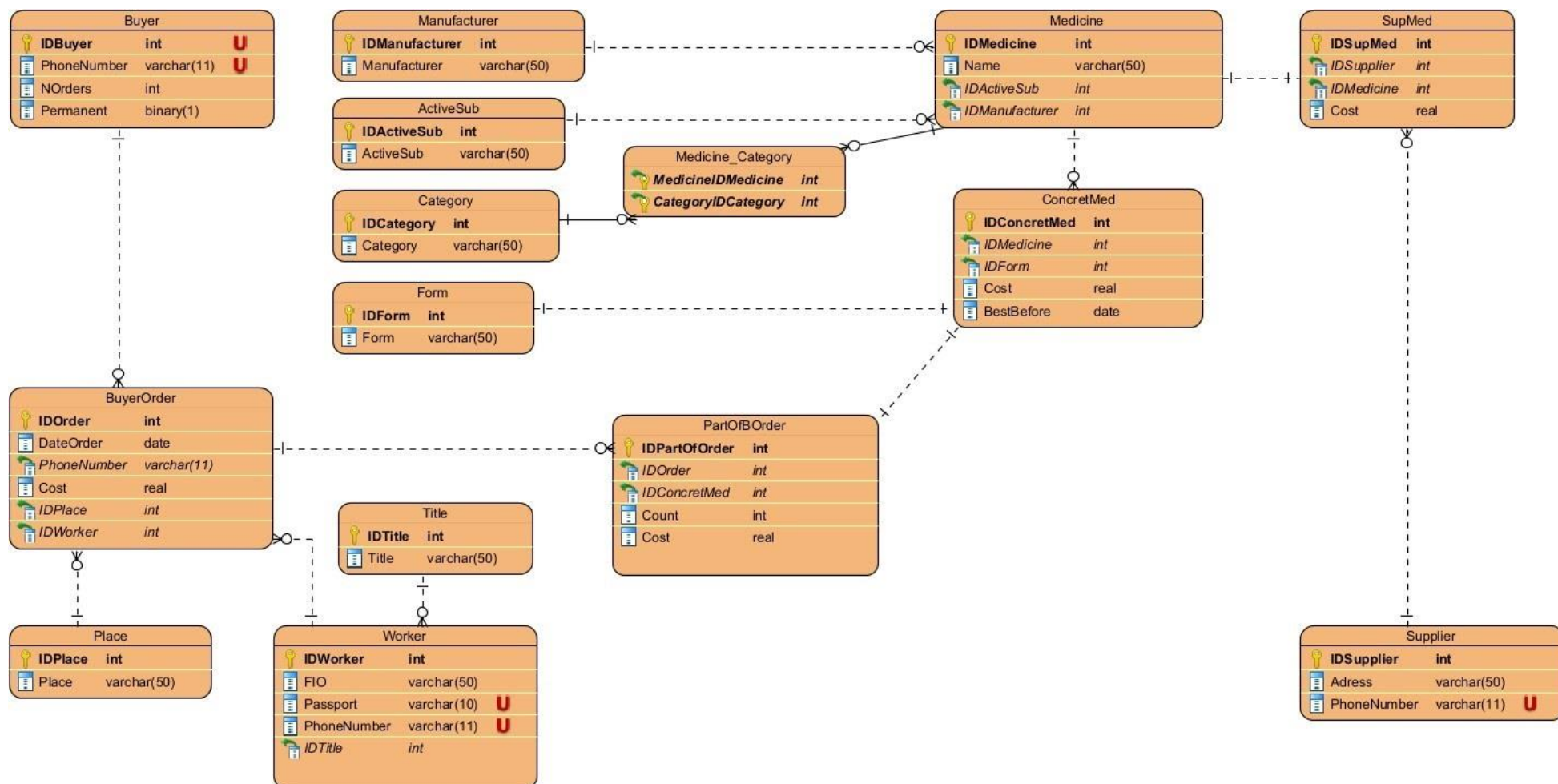


Рисунок 9 – Диаграмма физической модели базы данных

Физическая модель – логическая модель базы данных, выраженная в терминах языка описания конкретной СУБД [6]. На физической модели базы данных отражена реальная структура базы данных. Физические модели баз данных определяют способы размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне.

В данной работе в качестве СУБД используется MS SQL Server, т.к. основной во время разработки была операционная система Windows.

3 Реализация мобильного приложения аптеки

3.1 Развертывание и наполнение базы данных тестовыми данными

3.1.1 Развертывание базы данных в среде MS SQL Server

Процесс создания и развертывания базы данных приведен на рисунках 10–13. На рисунке 10 запрос создание базы данных.

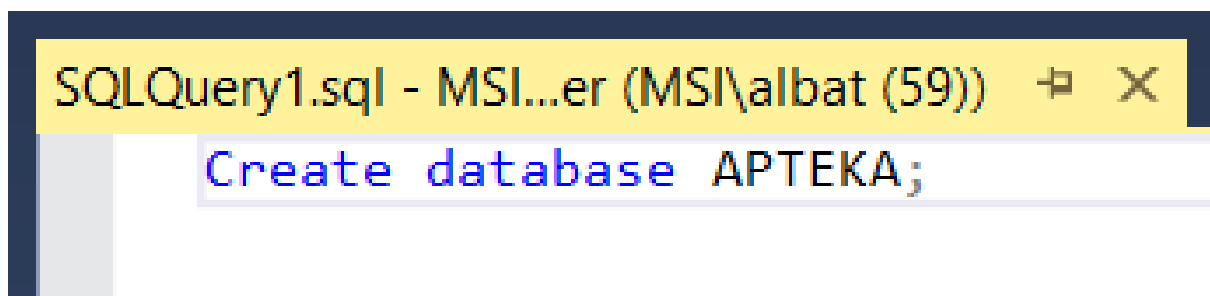


Рисунок 10 – Создание базы данных «Аптека»

На рисунке 11 отображен процесс открытия файлов базы данных в среде SQL Server Management 2018.

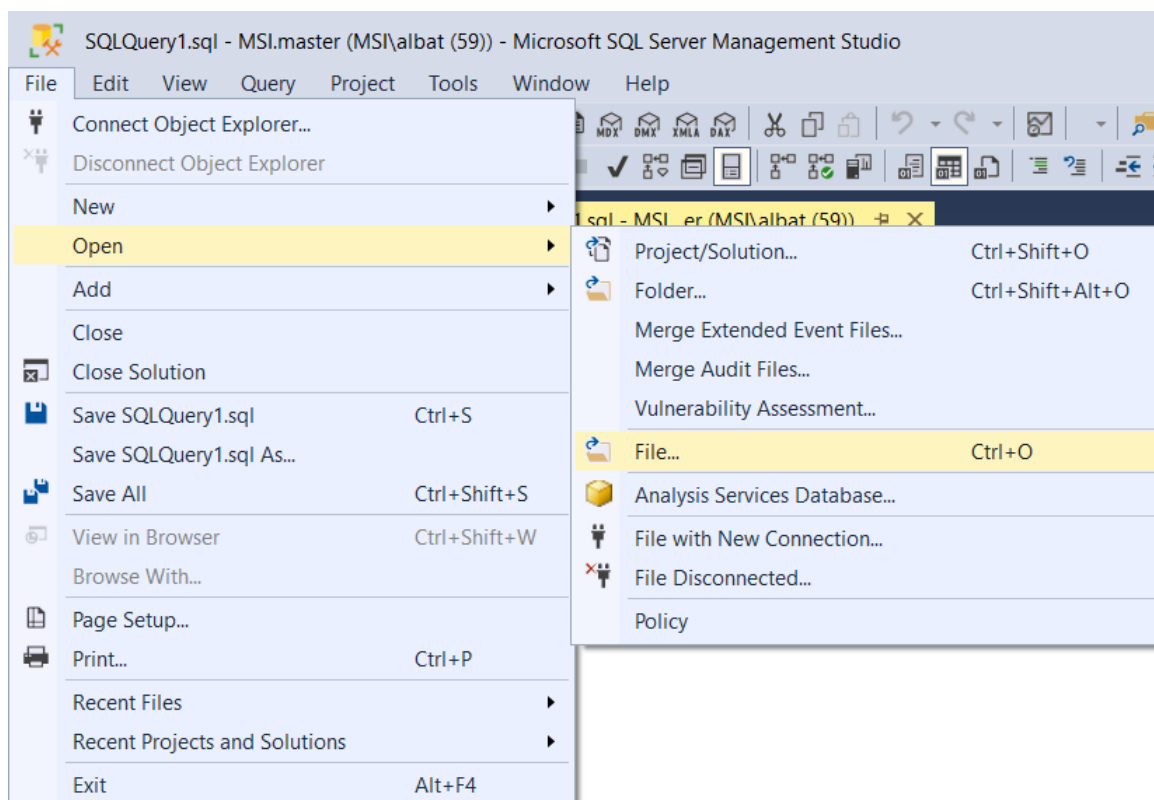


Рисунок 11 – Открытие sql-файла в MS SQL Server 2018

На рисунке 12 – запросы создания таблиц базы данных.

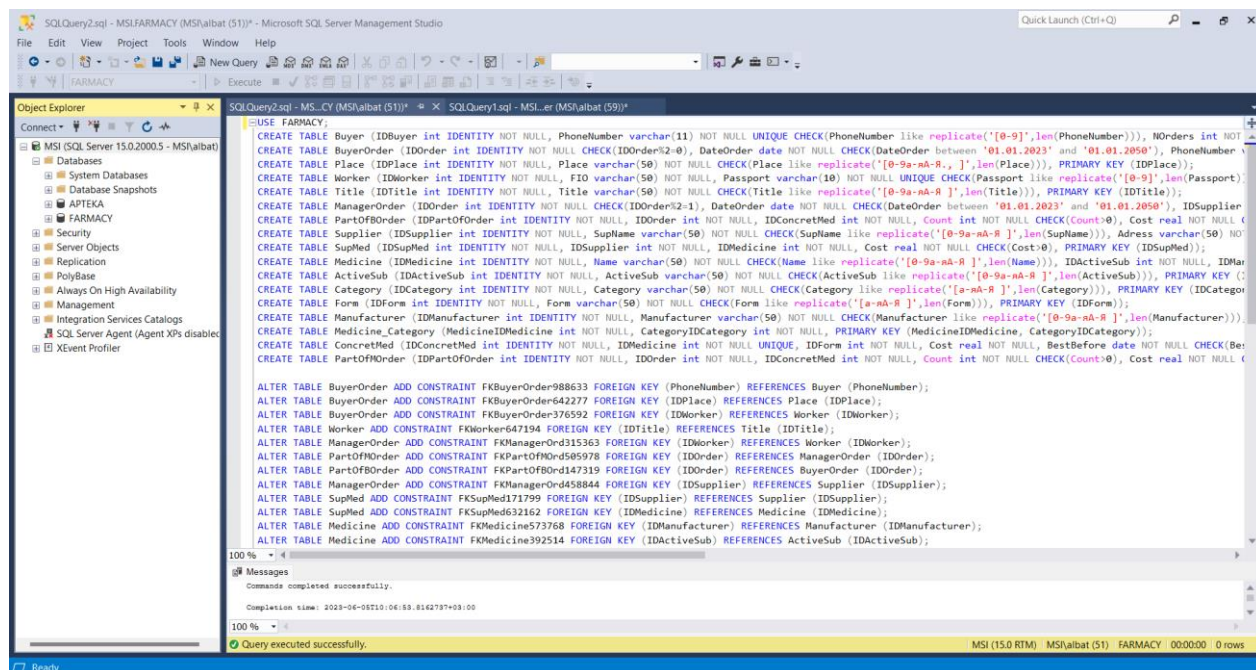


Рисунок 12 – Результат выполнения запроса создания таблиц в MS SQL Server

В результате можно убедиться, обновив правую панель (рисунок 13).

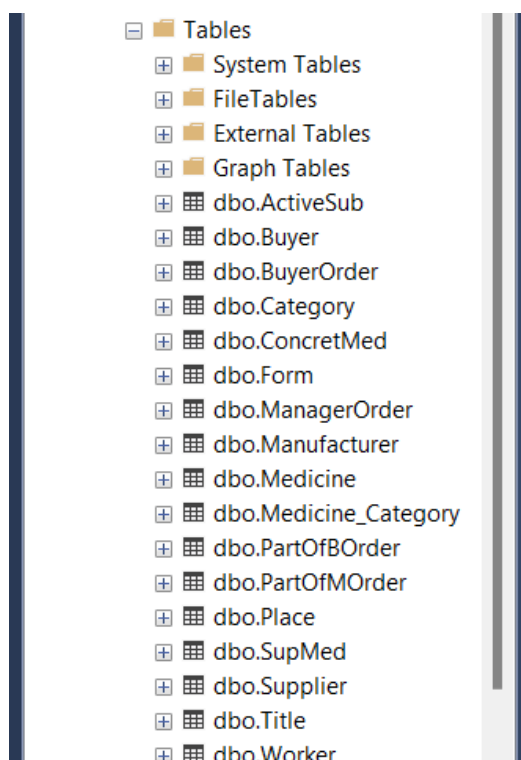


Рисунок 13 – Созданные в SQL Management Studio таблицы

Диаграмма данных, созданная в среде MS Management Studio 2018 приведена на рисунке 14.

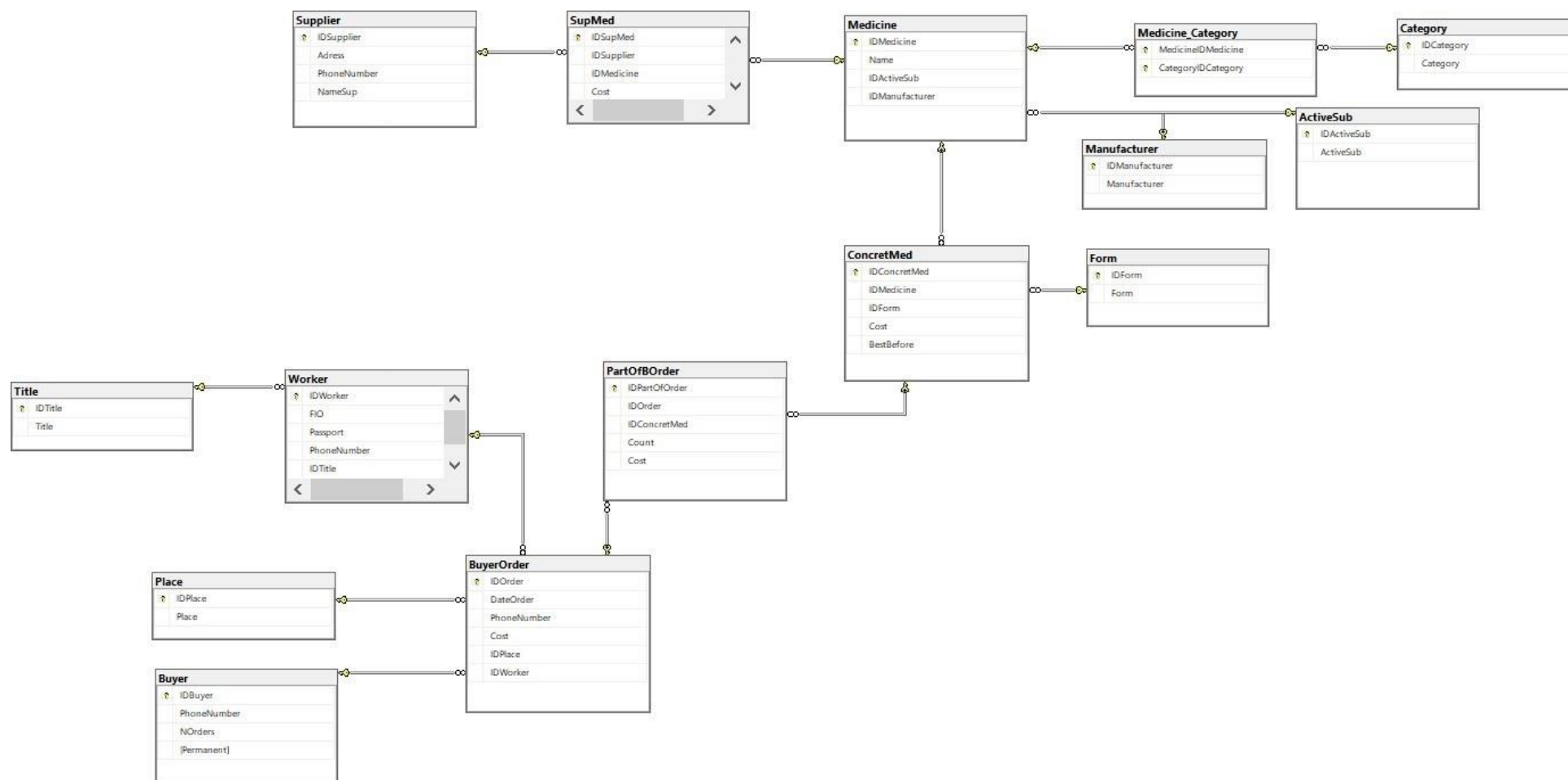


Рисунок 14 – Диаграмма, созданная в среде SQL Management Studio

Скрипт на языке Transact-SQL для создания объектов базы данных, сгенерированный в среде Visual Paradigm приведен в Приложении А.1.

3.1.2 Наполнение базы данных тестовыми данными

Заполненная база данных используется в дальнейшем для отладки и тестирования запросов и представлений, функций и хранимых процедур, триггеров и оконного приложения.

Запросы для заполнения таблиц представлены в листингах 1–4. На рисунках 15–18 результат выполнения этих запросов.

Таблица «Category» заполняется названиями групп препаратов по их назначению (листинг 1).

Листинг 1 – Заполнение таблицы Category

```
USE АРТЕКА;
INSERT INTO Category VALUES
('Обезболивающее'),
('Жаропонижающее'),
('Противовирусное'),
('Сосудосуживающее'),
('Витамины'),
('Муколитик'),
('Антибиотик');
```

Результат выполнения и внесенные в таблицу данные отображены на рисунке 15.

Results		Messages
	IDCategory	Category
1	1	Обезболивающее
2	2	Жаропонижающее
3	3	Противовирусное
4	4	Сосудосуживающее
5	5	Витамины
6	6	Муколитик
7	7	Антибиотик

Рисунок 15 – Результат заполнения таблицы Category

Таблица «Form» заполняется формами выпуска препаратов (листинг 2).

Листинг 2 – Заполнение таблицы Form

```
USE APTEKA;
INSERT INTO Form VALUES
('Таблетки'),
('Порошок'),
('Мазь'),
('Капли'),
('Суспензия'),
('Раствор');
```

Результат выполнения запроса на рисунке 16.

Results		Messages
	IDForm	Form
1	1	Таблетки
2	2	Порошок
3	3	Мазь
4	4	Капли
5	5	Суспензия
6	6	Раствор

Рисунок 16 – Результат заполнения таблицы Form

Таблица «Place» заполняется адресами пунктов выдачи (листинг 3).

Листинг 3 – Заполнение таблицы Place

```
USE APTEKA;
INSERT INTO Place VALUES
('г. Пенза, ул. Мира, д. 2'),
('г. Пенза, ул. Попова, д. 3'),
('г. Пенза, ул. 8 марта, д. 6'),
('г. Пенза, ул. Кирова, д. 8'),
('г. Пенза, ул. Володарского, д. 10');
```

Результат выполнения запроса и добавленные пункты выдачи на рисунке 17.

Results Messages		
	IDPlace	Place
1	1	г. Пенза, ул. Мира, д. 2
2	2	г. Пенза, ул. Попова, д. 3
3	3	г. Пенза, ул. 8 марта, д. 6
4	4	г. Пенза, ул. Кирова, д. 8
5	5	г. Пенза, ул. Володарского, д. 10

Рисунок 17 – Результат заполнения таблицы Place

Таблица «Title» включает в себя должности сотрудников аптеки (листинг 4).

Листинг 4 – Заполнение таблицы Title

```
USE APTEKA;
INSERT INTO Title VALUES
('Менеджер'),
('Провизор технолог рецептар'),
('Провизор аналитик'),
('Фармацевт'),
('Кассир'),
('Заведующий отделом'),
('Провизор стажер');
```

Результат выполнения запроса, приведенного выше отображен на рисунке 18.

Results Messages		
	IDTitle	Title
1	1	Менеджер
2	2	Провизор технолог рецептар
3	3	Провизор аналитик
4	4	Фармацевт
5	5	Кассир
6	6	Заведующий отделом
7	7	Провизор стажер

Рисунок 18 – Результат заполнения таблицы Title

Данные примеры являются тестовыми. На их примере можно сделать вывод о корректности работы заполнения базы данными.

3.2 Разработка серверной части приложения

3.2.1 Разработка запросов к базе данных и представлений для реализации аналитических бизнес-процессов

Спецификации запросов для реализации аналитических бизнес-процессов, спроектированных ранее, представлены в таблице 18.

Таблица 18 – Спецификации запросов к базе данных

№ п/п	Содержание	Используемые таблицы	Параметры группировки	Параметры выбора данных
1	Запрос информации о количестве заказов покупателей	Buyer BuyerOrder	По номеру телефона покупателя	order by dbo.Buyer.NOrders
2	Запрос информации о категориях товаров, заказываемых покупателями	Buyer BuyerOrder PartOfBOrder ConcretMed Medicine Medicine_Category Category	По номеру телефона покупателя	Group by dbo.Buyer.PhoneNumber, dbo.Category.Category

Спецификации представлений для выборки основных информационных объектов представлены в таблице 19.

Таблица 19 – Спецификации представлений

№ п/п	Имя представления	Назначение	Используемые таблицы	Параметры
1	AboutBOrder	Вывод информации о заказах пользователя	Buyer BuyerOrder PartOfBOrder ConcretMed Medicine	BuyerOrder.IDOrder BuyerOrder.DateOrder Buyer.PhoneNumber Medicine.Name PartOfBOrder.Count PartOfBOrder.Cost
2	AboutMed	Вывод информации о лекарствах	Medicine Manufacturer Medicine_Category Category ConcretMed Form	Medicine.Name ActiveSub.ActiveSub Manufacturer.Manufacturer Category.Category Form.Form ConcretMed.Cost ConcretMed.BestBefore

Скрипты для данных запросов и представлений представлены в

Приложении А.2.

3.2.2 Разработка функций и хранимых процедур для реализации учетных пользовательских бизнес-процессов

Спецификации функций и хранимых процедур приведены в таблице 20.

Таблица 20 – Спецификации функций и хранимых процедур учетных бизнес-процессов

№ п/п	Имя функции (процедуры)	Назначение	Описание логики работы	Параметры и возвращаемые данные
1	NewBuyer	Добавление в систему нового покупателя.	Процедура вставляет в таблицу Покупателей нового покупателя с заданным номером телефона. Количество заказов устанавливается равном 1, а флаг постоянного покупателя – 0, и выводит таблицу покупателей.	Параметры: номер телефона. Возвращаемые данные: таблица покупателей.
2	NewBuyerOrder	Добавление в систему нового заказа покупателя.	Процедура вставляет в таблицу заказов покупателей новый заказ и выводит таблицу заказов покупателей.	Параметры: номер телефона покупателя, дата заказа, стоимость заказа, номер пункта выдачи, id сотрудника. Возвращаемые данные: таблица заказов покупателей.

Продолжение таблицы 20

№ п/п	Имя функции (процедуры)	Назначение	Описание логики работы	Параметры и возвращаемые данные
3	DelBuyerOrder	Удаление заказа покупателя.	Процедура удаляет заказ покупателя.	Параметры: ID заказа. Возвращаемые данные: нет.
4	AddNumberOrders	Прибавление счетчика заказов покупателя.	Процедура увеличивает количество заказов покупателя.	Параметры: номер телефона покупателя. Возвращаемые данные: нет.
5	UpdatePartOfBorder	Изменить количество заказанных покупателем лекарств.	Процедура устанавливает новое значение в поле «количество» заказанных покупателем товаров.	Параметры: номер заказа, количество товара. Возвращаемые данные: нет.
6	UpdateBOrder	Изменить стоимость заказа покупателя.	Процедура увеличивает стоимость заказа покупателя на указанное значение.	Параметры: Номер заказа, дополнительная стоимость. Возвращаемые данные: нет.
7	NewPartOfBOrder	Добавить новые товары к заказу покупателя.	Процедура добавляет новый пункт к заказу покупателя.	Параметры: ID заказа, ID конкретного лекарства, количество, стоимость. Возвращаемые данные: нет.

Окончание таблицы 20

№ п/п	Имя функции (процедуры)	Назначение	Описание логики работы	Параметры и возвращаемые данные
8	AddNewBuyerOrder	Добавить новый заказ покупателя.	Процедура добавляет новую позицию к заказу покупателя: – если заказа с указанной датой и от данного покупателя в нет, создается новый заказ покупателя и в него добавляется поле; – если заказ в системе есть, в него добавляется новое поле, обновляется стоимость.	Параметры: Дата заказа, номер телефона покупателя, ID конкретного лекарства, количество, ID пункта выдачи, ID сотрудника. Возвращаемые данные: нет.

Были созданы следующие процедуры и функции:

- процедура для добавления в систему нового покупателя;
- процедура для добавления в систему нового заказа покупателя;
- процедура для удаления заказа покупателя;
- процедура для прибавления счетчика заказов покупателя;
- процедура для изменения количества заказанных покупателем лекарств;
- процедура для изменения стоимости заказа покупателя;
- процедура для добавления новых товаров к заказу покупателя;
- процедура для добавления нового заказа покупателя;

Диаграмма деятельности для хранимой процедуры добавления нового заказа пользователя приведена на рисунке 19.

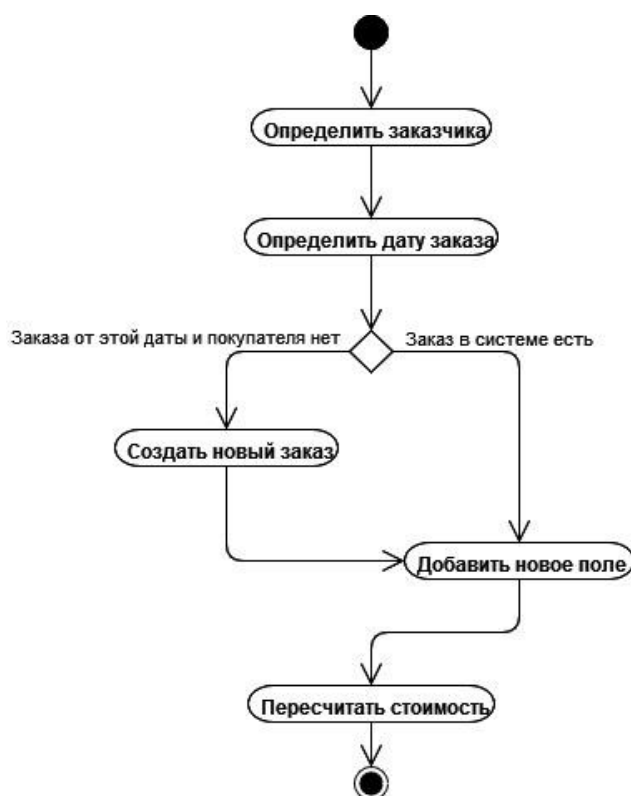


Рисунок 19 – Диаграмма деятельности для AddNewBuyerOrder

В листинге 5 представлен скрипт хранимой процедуры добавления нового заказа покупателя на языке T-SQL.

Листинг 5 – Код скрипта процедуры AddNewBuyerOrder

```

Create procedure AddNewBuyerOrder
@DO date,
@PN varchar(11),
@CM int,
@Count int,
@Pl int,
@W int as
declare @cst real
declare @i int
Set @cst = (Select ConcretMed.Cost*@Count as cst from
APTEKA.dbo.ConcretMed where ConcretMed.IDConcretMed=@CM)
IF exists (SELECT * from BuyerOrder WHERE BuyerOrder.DateOrder =
@DO and BuyerOrder.PhoneNumber = @PN and BuyerOrder.IDPlace=@Pl)
begin
    Select @i=(Select BuyerOrder.IDOrder as i from BuyerOrder
where BuyerOrder.DateOrder = @DO and BuyerOrder.PhoneNumber = @PN
and BuyerOrder.IDPlace=@Pl)
    PRINT @i
    Exec NewPartOfBOrder @i,@CM,@Count,@cst
    Exec UpdateBOrder @i,@cst
end
  
```

```

end
ELSE begin
    IF exists (SELECT * from Buyer WHERE Buyer.PhoneNumber = @PN)
        Exec AddNumberOrders @PN
    ELSE
        Exec NewBuyer @PN
        Exec NewBuyerOrder @PN,@DO,@cst,@Pl,@W
        Select @i=(Select BuyerOrder.IDOrder from BuyerOrder
where BuyerOrder.DateOrder = @DO and BuyerOrder.PhoneNumber = @PN
and BuyerOrder.IDPlace=@Pl)
        Exec NewPartOfBOrder @i,@CM,@Count,@cst
End

```

Полный текст скриптов хранимых процедур и функций приведен в Приложении А.3.

3.2.3 Разработка триггеров для реализации учетных автоматических бизнес-процессов и обновления представлений

Спецификации триггеров приведены в таблице 21.

Таблица 21 – Спецификации триггеров

№ п/п	Назначение триггера	Вид триггера	Событие триггера	Таблица триггера	Описание работы триггера
1	OnBOrderDel	Instead of	Delete	BuyerOrder	При удалении из базы заказа покупателя удаляются и все составляющие этого заказа
2	OnBOrderUpd	After	Update	PartOfBOrder	При изменении пункта заказа покупателя изменяется и общая стоимость медикамента
3	OnBOrderAdd	After	Insert	BuyerOrder	При добавлении заказа покупателя выводятся измененные таблицы

Для модификации данных при выполнении операции DELETE для не обновляемого представления был разработан следующий триггер. При удалении из базы заказа покупателя удаляются и все составляющие этого заказа. На

рисунке 20 изображена диаграмма деятельности для данного триггера.



Рисунок 20 – Диаграмма деятельности для триггера OnBOrderDel

В листинге 6 приведен скрипт аналогичный для диаграммы выше.

Листинг 6 – Скрипт триггера OnBOrderDel

```

Create trigger OnBOrderDel on BuyerOrder instead of delete as
declare @id int if @@ROWCOUNT=1
begin
    Select @id=IDOrder from Deleted
    Delete from PartOfBOrder where @id=PartOfBOrder.IDOrder
    Delete from BuyerOrder where @id = BuyerOrder.IDOrder
    print 'Заказ и его составляющие успешно удалены!'
end;
  
```

Для реализации автоматически исполняемых учетных бизнес-процессов изменения количества заказываемых медикаментов и создания нового заказа были разработаны триггеры OnBOrderUpd и OnBOrderAdd.

При изменении пункта заказа покупателя изменяется и общая стоимость медикамента. На рисунке 21 диаграмма деятельности триггера OnBOrderUpd.



Рисунок 21 – Диаграмма деятельности для триггера OnBOrderUpd

Листинг 7 содержит скрипт триггера OnBOrderUpd.

Листинг 7 – Скрипт триггера OnBOrderUpd

```

Create trigger OnBOrderUpd on PartOfBOrder after update as
declare @id int, @pr real if @@ROWCOUNT=1
begin
    Select @id-IDOrder from Deleted
    Select @pr = (select ConcretMed.Cost from ConcretMed,
PartOfBOrder where ConcretMed.IDConcretMed =
PartOfBOrder.IDConcretMed)
    update PartOfBOrder set Cost = (select PartOfBOrder.Count from
PartOfBOrder where PartOfBOrder.IDOrder = @id) * @pr where
PartOfBOrder.IDOrder = @id
    print 'Заказ успешно обновлен!'
end;
  
```

При добавлении заказа покупателя выводятся измененные таблицы. На рисунке 22 диаграмма деятельности триггера OnBOrderAdd.



Рисунок 22 – Диаграмма деятельности для триггера OnBOrderAdd

В листинге 8 – скрипт данного триггера.

Листинг 8 – Скрипт триггера OnBOrderAdd

```

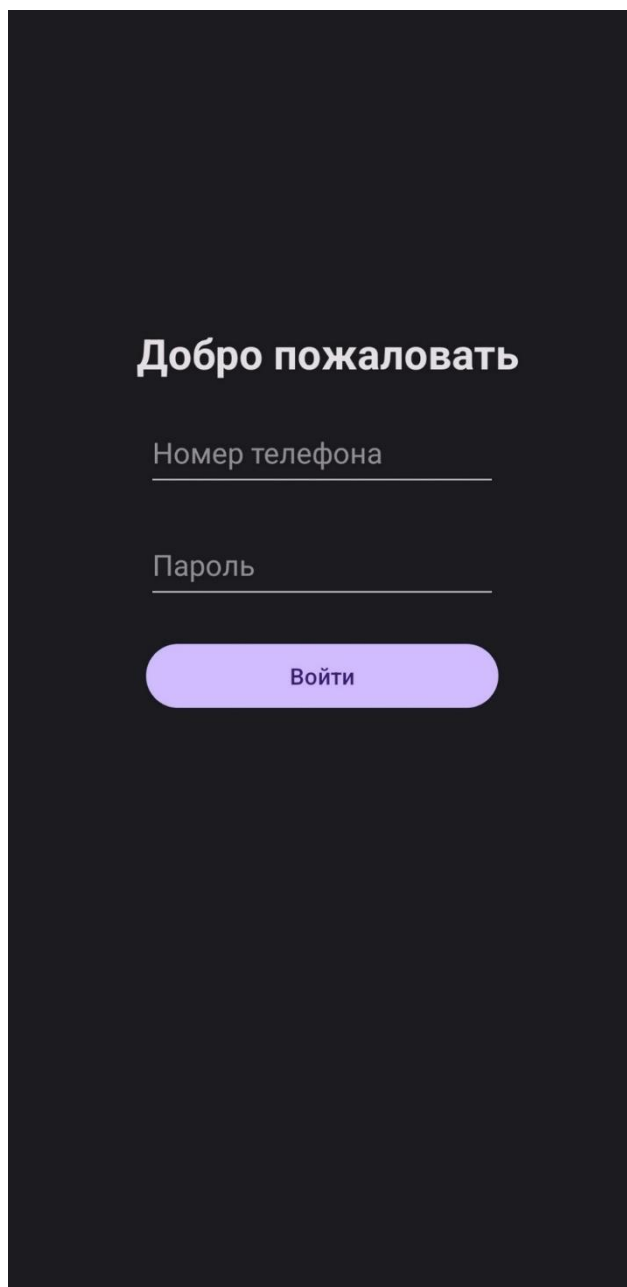
Create trigger OnBOrderAdd on BuyerOrder after insert as
declare @id int if @@ROWCOUNT=1
begin
    Select @id-IDOrder from Inserted
    print 'Заказ успешно добавлен!'
    Select * from BuyerOrder where BuyerOrder.IDOrder=@id
end;
  
```

Скрипты триггеров также приведены в Приложении А.4.

3.3 Разработка интерфейса клиентского приложения

Интерфейс пользователя каждого приложения должен быть простым и интуитивно понятным.

При запуске пользователю предлагается авторизоваться (рисунок 23).



Добро пожаловать

Номер телефона

Пароль

Войти

Рисунок 23 – Страница авторизации

Если пользователь с таким номером не зарегистрирован, ему будет предложено это сделать. В любом случае, шаг с авторизацией можно отложить до оформления заказа.

Далее, на главной странице, пользователю становятся доступны каталог товаров, а также поиск и корзина (рисунок 24).

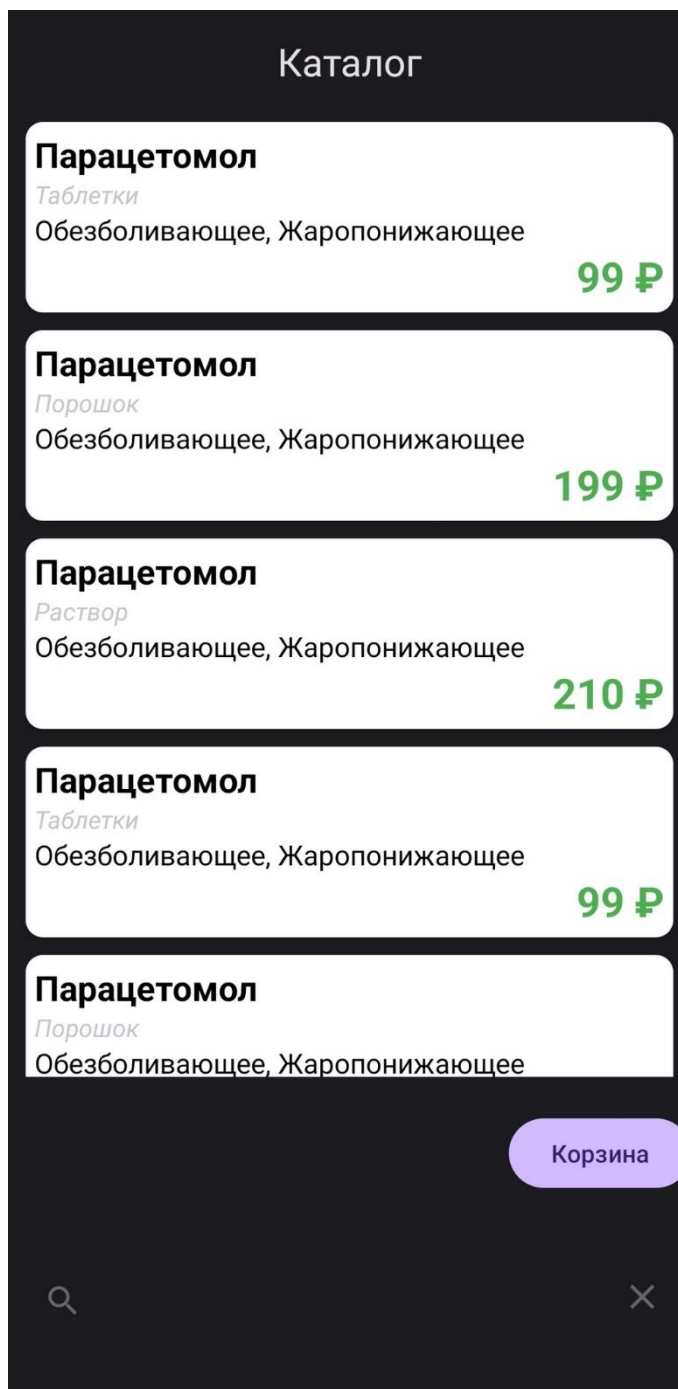


Рисунок 24 – Страница со списком лекарств

Взаимодействие с пунктом каталога открывает карточку данного товара, содержащую информацию о лекарстве (рисунок 25).

Парацетомол

Таблетки

Обезболивающее, Жаропонижающее

Активный компонент: Парацетамол

Производитель: АптекаРУ

Годеи до 2023-07-01

99 Р

Добавить в корзину

Рисунок 25 – Карточка товара

Собрав заказ, пользователь может просмотреть его в своей корзине (рисунок 26).

Корзина

Парацетомол	210 Р
Парацетомол	210 Р
Парацетомол	210 Р
Парацетомол	199 Р
Парацетомол	99 Р

Пункт доставки

г. Пенза, ул. Мира, д. 2	☒
г. Пенза, ул. Попова, д. 3	☐
г. Пенза, ул. 8 марта, д. 6	☐
г. Пенза, ул. Кирова, д. 8	☐
г. Пенза, ул. Володарского, д. 10	☐

В магазин

Продолжить

Рисунок 26 – Корзина

Здесь же пользователю доступен выбор пункта доставки и переход к оформлению заказа.

Код реализации мобильного приложения представлен в приложении Б.

4 Тестирование приложения

4.1 Выбор режима тестирования приложения

Тестирование – деятельность, выполняемая для оценки и улучшения качества программного обеспечения. Эта деятельность, в общем случае, базируется на обнаружении дефектов и проблем в программных системах.

Модель тестирования описывает, как с помощью тестов проверяются исполняемые компоненты модели реализации. Модель тестирования включает:

- набор тестовых примеров (тестов);
- классы и функции тестирования;
- тестовые компоненты (созданные для автоматизации тестирования).

Системное тестирование программного обеспечения – это тестирование программного обеспечения (ПО), выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям. Системное тестирование относится к методам тестирования черного ящика, и, тем самым, не требует знаний о внутреннем устройстве системы [7].

Наиболее известные виды системного тестирования:

- функциональное тестирование;
- нефункциональное тестирование;
- интеграционное тестирование.

Функциональное тестирование – направлено на проверку того, какие функции приложения реализованы, и что они работают верным образом. Данный метод тестирования является одним из самых применяемых, так как позволяет наиболее полно проверить программу на соответствие требованиям [8].

Нефункциональное тестирование – вид тестирования, направленный на проверку нефункциональных особенностей приложения, таких как удобство использования, совместимость, производительность, безопасность и прочее [9].

Интеграционное тестирование – направлено на проверку взаимодействия между несколькими частями приложения. Этот вид тестирования позволяет

проверить корректность взаимодействия частей ПО между собой [10].

Функциональное тестирование – это тестирование программных средств с целью проверки правильности реализации функциональных требований. Методы функционального тестирования:

- классы эквивалентности;
- метод граничных значений.

Класс эквивалентности – набор данных, обрабатываемый одинаковым образом и приводящий к одинаковому результату.

Граничное условие – значение, находящееся на границе классов эквивалентности.

Тестирование производительности в инженерии программного обеспечения – тестирование, которое проводится с целью определения, как быстро работает вычислительная система или ее часть под определенной нагрузкой. Также может служить для проверки и подтверждения других атрибутов качества системы, таких как масштабируемость, надежность и потребление ресурсов.

В тестировании производительности различают следующие направления:

- нагрузочное;
- стресс;
- тестирование стабильности;
- конфигурационное.

Возможны два подхода к тестированию производительности программного обеспечения:

- в терминах рабочей нагрузки: программное обеспечение подвергается тестированию в ситуациях, соответствующих различным сценариям использования;
- в рамках бета-тестирования, когда система испытывается реальными конечными пользователями.

Было принято решение осуществить проверку программы с помощью функционального тестирования. Такой выбор обусловлен тем, что к программе

были предъявлены в основном функциональные требования, выполнение которых нужно проверить.

4.2 Тестирование серверной части приложения

В данной работе было выполнено функциональное тестирование разработанного программного обеспечения (таблица 22).

Таблица 22 – Перечень тестов серверной части приложения

№ п/п	Назначение теста	Описание теста	Результаты теста
1	Вывод информации о категориях заказов покупателей	Проверка корректности работы представления	Информация выведена. Тест пройден. Рисунок 27.
2	Вывод информации о количестве заказов покупателей	Проверка корректности работы представления	Информация выведена. Тест пройден. Рисунок 28.
3	Вывод представления о заказах покупателей	Проверка корректности работы представления	Информация выведена. Тест пройден. Рисунок 29.
4	Вывод представления о лекарствах	Проверка корректности работы представления	Информация выведена. Тест пройден. Рисунок 30.
5	Добавление нового заказа	Проверка корректности работы процедуры	Информация выведена. Тест пройден. Рисунок 31.
6	Удаление заказа	Проверка корректности работы процедуры	Информация выведена. Тест пройден. Рисунок 32.
7	Изменение заказа	Проверка корректности работы процедуры	Информация выведена. Тест пройден. Рисунок 33.
8	Добавление нового покупателя	Проверка корректности работы процедуры	Информация выведена. Тест пройден. Рисунок 34.
9	Триггер добавления заказа	Проверка корректности работы триггера	Заказ добавлен. Тест пройден. Рисунок 35.
10	Триггер удаления заказа	Проверка корректности работы триггера	Заказ удален. Тест пройден. Рисунок 36.

Ниже, на рисунках 27–36 продемонстрированы результаты тестирования.

На рисунке 27 представлен результат для тестирования вывода информации о категориях заказов покупателей.

	Номер_телефона_покупателя	Категория	Количество
1	88001112233	Жаропонижающее	1
2	88001119900	Жаропонижающее	1
3	88005353535	Жаропонижающее	9
4	88001112233	Обезболивающее	1
5	88001119900	Обезболивающее	1
6	88005353535	Обезболивающее	9
7	88005553535	Обезболивающее	1
8	89008070	Обезболивающее	1

Рисунок 27 – Тест №1

На рисунке 28 представлен результат теста вывода информации о количестве заказов покупателей

	Номер_телефона_покупателя	Количество_заказов
1	88005353535	6
2	88005353535	6
3	88005353535	6
4	88005353535	6
5	88001112233	1
6	88001119900	1
7	88005553535	1
8	89008070	1

Рисунок 28 – Тест №2

На рисунке 29 – вывод представления с информацией о заказах покупателей.

	Номер_заказа	Дата	Номер_телефона_покупателя	Название_лекарства	Количество	Стоимость
1	9	2023-01-01	88005353535	Парацетомол	3	796
2	11	2023-01-01	88005353535	Парацетомол	2	398
3	13	2023-01-01	88001112233	Парацетомол	3	597
4	14	2023-01-01	88005353535	Парацетомол	1	99
5	14	2023-01-01	88005353535	Парацетомол	1	99
6	15	2023-01-01	88001119900	Парацетомол	2	198
7	14	2023-01-01	88005353535	Парацетомол	1	99
8	14	2023-01-01	88005353535	Парацетомол	1	99
9	14	2023-01-01	88005353535	Парацетомол	1	99
10	16	2023-06-07	88005353535	Парацетомол	1	99
11	16	2023-06-07	88005353535	Парацетомол	1	99
12	17	2023-06-07	88005553535	Анальгин	2	200
13	18	2023-06-08	89008070	Анальгин	1	100

Рисунок 29 – Тест №3

На рисунке 30– вывод представления с информацией о лекарствах.

	Название_лекарства	Действующее_вещество	Производитель	Категория	Форма_выпуска	Стоимость
1	Парацетомол	Парацетамол	АптекаРУ	Обезболивающее, Жаропонижающее	Таблетки	99
2	Парацетомол	Парацетамол	АптекаРУ	Обезболивающее, Жаропонижающее	Порошок	199
3	Парацетомол	Парацетамол	АптекаРУ	Обезболивающее, Жаропонижающее	Раствор	210
4	Парацетомол	Парацетамол	АптекаРУ	Обезболивающее, Жаропонижающее	Таблетки	99
5	Парацетомол	Парацетамол	АптекаРУ	Обезболивающее, Жаропонижающее	Порошок	199
6	Анальгин	Дротаверин	АптекаРУ	Обезболивающее	Таблетки	100

Рисунок 30 – Тест №4

На рисунке 31 – результат выполнения запроса на добавление нового заказа.

	IDOrder	DateOrder	PhoneNumber	Cost	IDPlace	IDWorker	IDOrderStatus
1	9	2023-01-01	88005353535	796	3	2	1
2	11	2023-01-01	88005353535	398	2	2	1
3	13	2023-01-01	88001112233	597	3	2	1
4	14	2023-01-01	88005353535	495	1	2	1
5	15	2023-01-01	88001119900	198	1	2	1
6	16	2023-06-07	88005353535	198	1	2	1
7	17	2023-06-07	88005553535	200	4	2	1
8	18	2023-06-08	89008070	100	3	2	1
9	20	2023-05-05	88001234545	300	2	2	1
10	21	2023-05-05	88001234545	300	3	2	1

Рисунок 31 – Тест №5

На рисунке 32 – результат удаления заказа из таблицы базы данных.

	IDOrder	DateOrder	PhoneNumber	Cost	IDPlace	IDWorker	IDOrderStatus
1	9	2023-01-01	88005353535	796	3	2	1
2	11	2023-01-01	88005353535	398	2	2	1
3	13	2023-01-01	88001112233	597	3	2	1
4	14	2023-01-01	88005353535	495	1	2	1
5	15	2023-01-01	88001119900	198	1	2	1
6	16	2023-06-07	88005353535	198	1	2	1
7	17	2023-06-07	88005553535	200	4	2	1
8	18	2023-06-08	89008070	100	3	2	1
9	20	2023-05-05	88001234545	300	2	2	1

Рисунок 32 – Тест №6

На рисунке 33 представлен результат изменения заказа.

	IDOrder	DateOrder	PhoneNumber	Cost	IDPlace	IDWorker	IDOrderStatus
1	9	2023-01-01	88005353535	796	3	2	1
2	11	2023-01-01	88005353535	398	2	2	1
3	13	2023-01-01	88001112233	597	3	2	1
4	14	2023-01-01	88005353535	495	1	2	1
5	15	2023-01-01	88001119900	198	1	2	1
6	16	2023-06-07	88005353535	198	1	2	1
7	17	2023-06-07	88005553535	200	4	2	1
8	18	2023-06-08	89008070	100	3	2	1
9	20	2023-05-05	88001234545	400	2	2	1

Рисунок 33 – Тест №7

На рисунке 34 представлено добавление нового покупателя.

	IDBuyer	PhoneNumber	NOrders
3	6	12345678901	0
4	7	88001112233	1
5	8	87776665544	0
6	9	8805353535	1
7	10	88005353566	1
8	11	88001119900	1
9	12	88005553535	1
10	13	89008070	1
11	14	88001234545	3
12	15	80008880088	1
13	17	88008008880	1

Рисунок 34 – Тест №8

На рисунке 35 результат теста №9, в котором проверяется работа триггера добавления нового заказа.

16

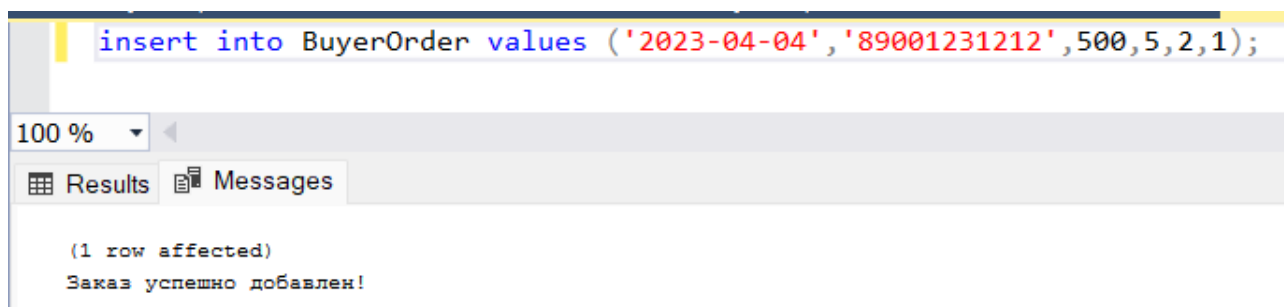


Рисунок 35 – Тест №9

Результат теста №10 для триггера удаления заказа представлен на рисунке 36.

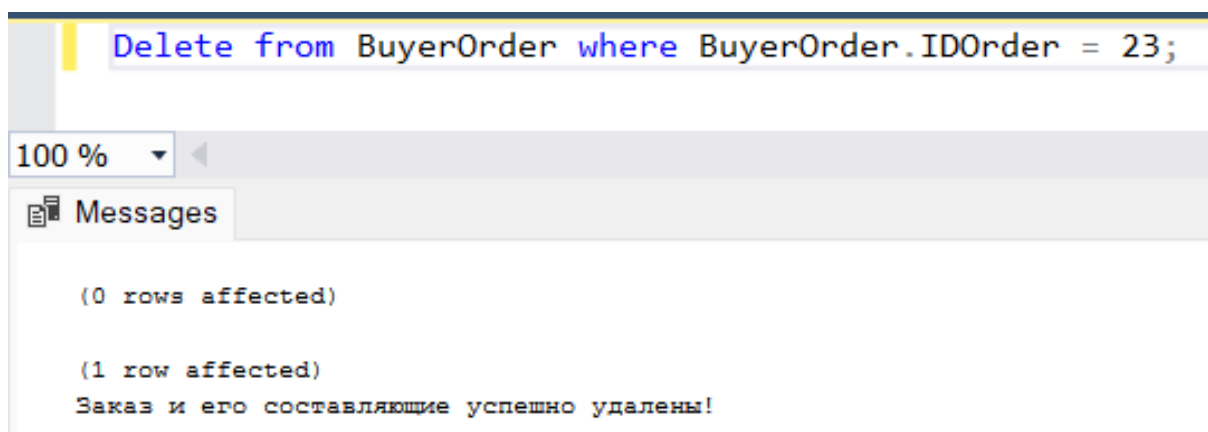


Рисунок 36 – Тест №10

В ходе выполнения тестирования несовпадения ожидаемого и наблюдаемого результата не выявлены. Следовательно, можно сделать вывод, что база данных работает корректно.

4.3 Анализ качества кода

4.3.1 Измерение устойчивости

Один из способов определения устойчивости компонент – подсчитать количество входящих и исходящих зависимостей. Вместе они позволяют вычислить позиционную устойчивость компонента:

$$I = \frac{Ce}{Ca + Ce},$$

где I – неустойчивость;

Ca – входящие связи, количество классов вне данного компонента, зависящих от классов внутри компонента;

Ce – исходящие связи, количество классов внутри данного компонента, зависящих от классов вне компонента.

Величина I изменяется в диапазоне $[0,1]$. Если $I = 0$, то компонент максимально устойчив, если $I = 1$ – максимально неустойчив. Максимальная устойчивость означает, что система неизменяема. Такая ситуация нежелательна.

Метрика A , являющаяся мерой абстрактности компонента, равна отношению количества абстрактных классов к общему количеству классов в компоненте:

$$A = \frac{Na}{Nc},$$

где: Na – количество абстрактных классов в нем же;

Nc – количество классов в компоненте.

Метрика A изменяется в диапазоне $[0,1]$, причем 0 означает, что в компоненте вообще нет абстрактных классов, а 1 – что компонент содержит только абстрактные классы. Соотношение между устойчивостью (I) и абстрактностью (A) можно представить на следующем графике, изображенном на рисунке 37.

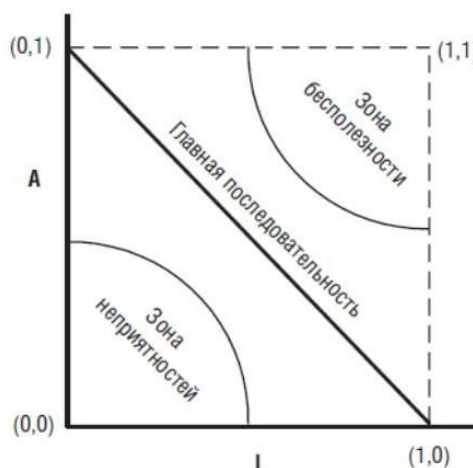


Рисунок 37 – Соотношение между устойчивостью (I) и абстрактностью (A)

На графике максимально устойчивые и абстрактные компоненты находятся в левом верхнем углу, в точке $(0,1)$, а максимально неустойчивые и конкретные – в правом нижнем углу, в точке $(1,0)$.

Вычислим устойчивость компонента `CatalogActivity`. Имеется один класс вне `CatalogActivity`, зависящий от класса внутри `CatalogActivity`. Следовательно, $Ca = 1$. Есть также один класс вне `CatalogActivity`, от которого зависит класс внутри `CatalogActivity`. Поэтому $Ce = 1$. Таким образом, $I = 1/2$. В коде программы нет абстрактных классов, поэтому метрика $A = 0$. Соотношение между устойчивостью (I) и абстрактностью (A) компонента `CatalogActivity` можно представить на следующем графике, изображенном на рисунке 38.

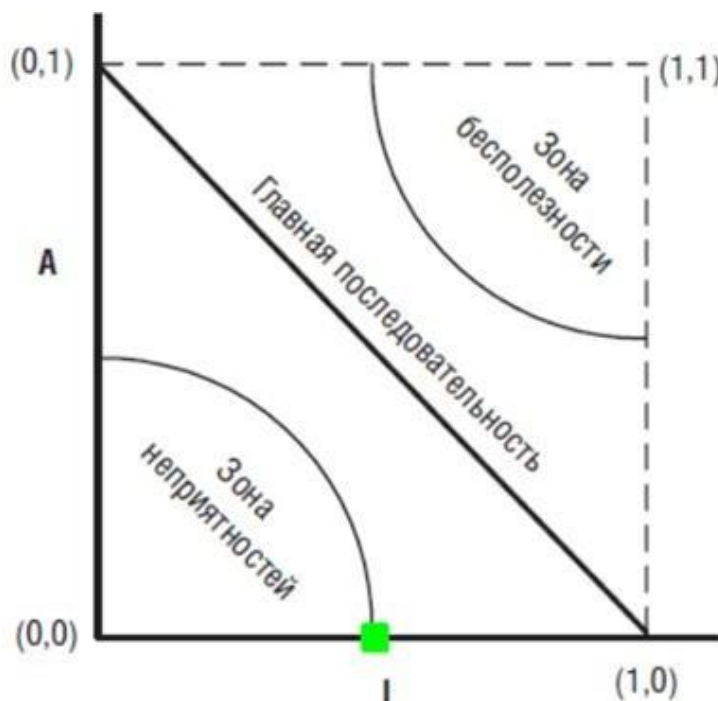


Рисунок 38 – Соотношение между устойчивостью (I) и абстрактностью (A) для компонента `CatalogActivity`

Правильно спроектированные компоненты обычно не должны находиться вблизи с точкой $(0, 0)$. Однако в действительности существуют программные сущности, попадающие в эту зону. К примеру, схема базы данных, она максимально конкретна и к ней тянется множество зависимостей. Вот одна из причин, почему изменения схемы БД обычно связано с большими неприятностями. Желательно, чтобы компонент находился на главной

последовательности или близко к ней. Для измерения, насколько далеко компонент отстоит от идеала, определяют следующую метрику:

D (расстояние от главной последовательности):

$$D = \frac{|A + I - 1|}{\sqrt{2}}$$

Величина D изменяется в диапазоне $[0; \sim 0,707]$. Для компонента CatalogActivity это приблизительно -0,35.

Для удобства анализа можно использовать нормированное расстояние:

$$D' = |A + I - 1|$$

Метрика D' изменяется в диапазоне $[0,1]$. При этом 0 означает, что компонент находится на главной последовательности, а 1 – что он максимально удален от нее. Для компонента CatalogActivity $D' = 0,5$.

5 Планирование разработки и оценка бюджета

Планирование разработки и оценка бюджета была выполнена при помощи программы для управления проектами GanttProject. Проект был начат 17 апреля 2023 года.

При планировании разработки было выделено 11 задач. Список задач и их описание представлены на рисунке 39.

Название	Дата начала	Дата окончания	Длительность	Предшественники	Исполнителей	Стоимость
Начало проекта	17.04.2023	17.04.2023	0.0 day			0
Анализ требований	17.04.2023	25.04.2023	7.0 day			8400.0
Анализ предметной области	17.04.2023	19.04.2023	3.0 day	0	Аналитик	3600.0
Анализ аналогов	20.04.2023	25.04.2023	4.0 day	10	Аналитик	4800.0
Анализ требований завершен	26.04.2023	26.04.2023	0.0 day	10,11		0
Проектирование	26.04.2023	02.05.2023	4.0 day			14360.0
Проектирование структур данных	26.04.2023	02.05.2023	4.0 day	2	Программист 1	5120.0
Проектирование интерфейса	26.04.2023	28.04.2023	3.0 day	2	Аналитик, Программист 2	3960.0
Проектирование приложения	26.04.2023	02.05.2023	4.0 day	2	Аналитик, Программист 2	5280.0
Проектирование завершено	03.05.2023	03.05.2023	0.0 day	15,16,17		0
Разработка	02.05.2023	17.05.2023	11.0 day			21200.0
Разработка структур данных	03.05.2023	10.05.2023	5.0 day	15	Программист 1	3200.0
Разработка серверной части приложения	03.05.2023	16.05.2023	9.0 day	4	Программист 1	5760.0
Разработка клиентской части приложения	02.05.2023	17.05.2023	11.0 day			12240.0
Разработка интерфейса	02.05.2023	11.05.2023	7.0 day	16	Программист 2	5040.0
Разработка приложения	03.05.2023	17.05.2023	10.0 day	17	Программист 2	7200.0
Разработка завершена	18.05.2023	18.05.2023	0.0 day	18,19,20,21,22		0
Тестирование	18.05.2023	23.05.2023	4.0 day			7840.0
Выбор режима тестирования	18.05.2023	19.05.2023	2.0 day	6,24-SF	Аналитик	2400.0
Функциональное тестирование	18.05.2023	23.05.2023	4.0 day	6	Программист 2, Программист 1	5440.0
Тестирование завершено	24.05.2023	24.05.2023	0.0 day	23,24		0
Конец проекта	24.05.2023	24.05.2023	0.0 day	1,3,5,7		0

Рисунок 39 – Список задач проекта

На рисунке 40 представлена диаграмма Ганта, построенная в среде GanttProject на основе введенных задач и ресурсов. Данная диаграмма отражает продолжительность и последовательность задач, а также распределение ресурсов, выделенных на каждую конкретную.

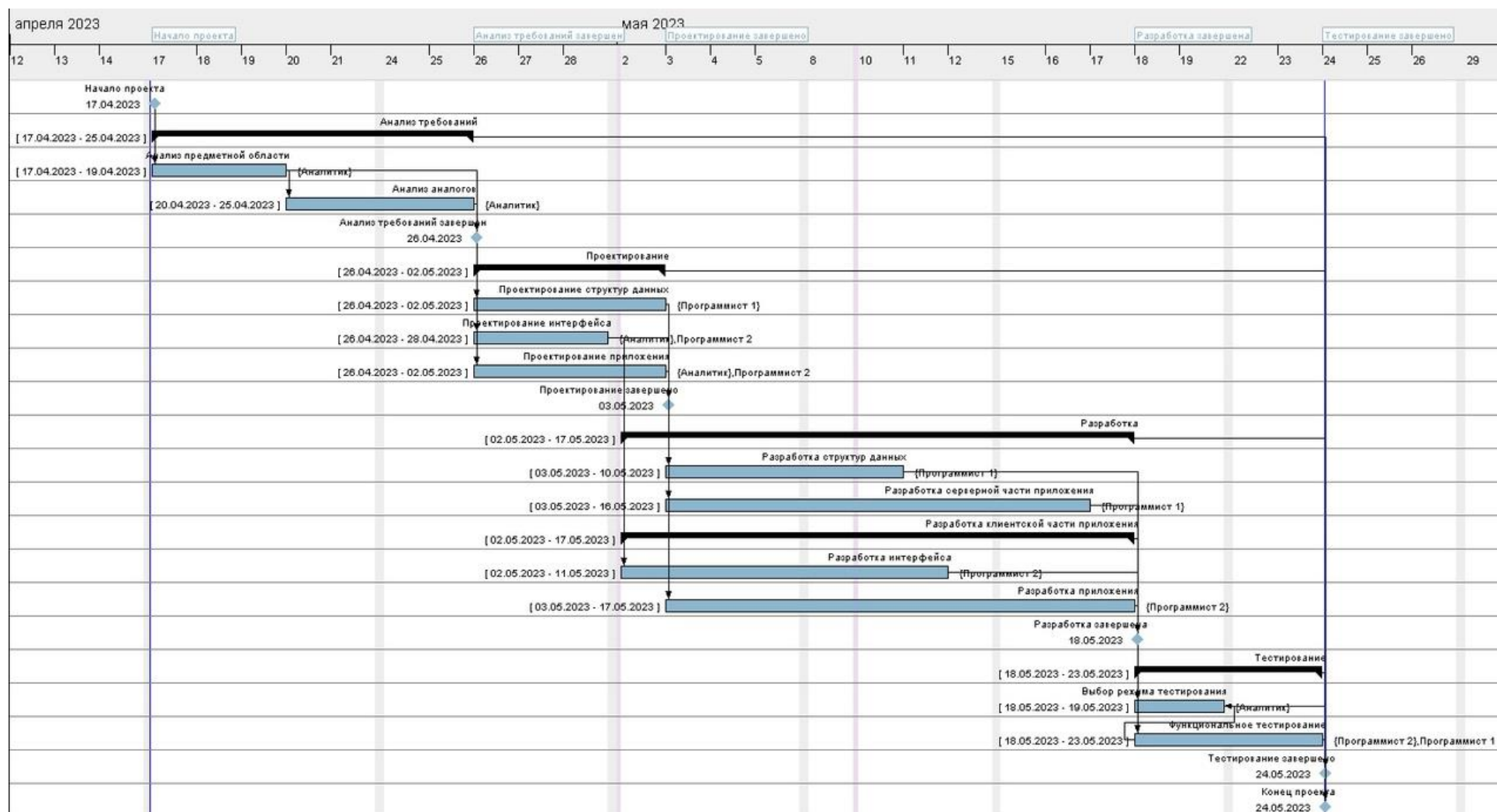


Рисунок 40 – Диаграмма Гранта

Для реализации проекта необходимо выделить основные ресурсы, их тип и стоимость. Список и описание ресурсов, используемых при выполнении задач, представлен на рисунке 41.

Имя	Стандартная роль	Стандартная ставка	Суммарная нагрузка	Суммарная стоимость
Программист 1	разработчик	1 280	13	16 640
Программист 2	разработчик	1 440	14	20 160
Аналитик	аналитик	1 200	12,5	15 000

Рисунок 41 – Список ресурсов

Таким образом из данных отчетов видно следующее. Длительность проекта – с 17 апреля по 24 мая, что составляет 24 рабочих дня при пятидневной неделе и восьмичасовом дне с учетом праздников. Трудозатраты на разработку 316 человеко-часов. Приблизительная оценка бюджета показывает стоимость в 51 800 рублей.

Заключение

В ходе выполнения выпускной квалификационной работы разработано мобильное приложение для аптеки.

Проведен анализ предметной области. На основе сравнения имеющихся аналогов и требований задания сформулирована задача на разработку программного средства, а также определены функциональные и нефункциональные требования к будущей системе.

Разработка программного средства проводилась в среде разработки Android Studio на языке программирования Kotlin в операционной системе Windows 10. Тестирование программного средства проводилось в операционных системах Windows и Android. Все тесты пройдены успешно и в результате тестирования не было выявлено ошибок.

В ходе работы выполнены планирование разработки и оценка бюджета при помощи GanttProject. Длительность проекта составила 24 рабочих дня, а суммарные затраты равны 51 800 рублей.

Таким образом в результате выполнения выпускной квалификационной работы разработано мобильное приложение аптеки.

Список использованных источников

1. Этимологический словарь русского языка: в 4 томах / Макс Фасмер; пер. с нем. и доп. О. Н. Трубачева. – 4-е изд., стер. – Москва: Астрель: АСТ, 2007.
2. ГОСТ 33707-2016 (ISO/IEC 2382:2015). Информационные технологии (ИТ). Словарь (с Поправкой). [Электронный ресурс]. URL: <https://docs.cntd.ru/document/1200139532> (дата обращения 24.05.2023).
3. Роскачество. Крупнейшие онлайн-аптеки: рейтинг // Роскачество: портал для умного покупателя. 2022. [Электронный ресурс]. URL: <https://rskrf.ru/ratings/tekhnologii/mobilnye-prilozheniya/onlayn-apteki/#details> (дата обращения 24.05.2023).
4. Верников Г., Основы методологии IDEF1X [Электронный ресурс]. URL: <https://www.cfin.ru/vernikov/idef/idef1x.shtml> (дата обращения 29.04.2022).
5. Уровни моделирования ИС [Электронный ресурс]. URL: https://dl.nure.ua/pluginfile.php/550/mod_resource/content/2/content/content4.html (дата обращения 24.05.2023).
6. Физическая модель базы данных [Электронный ресурс]. URL: https://neerc.ifmo.ru/wiki/index.php?title=Физическая_модель_базы_данных (дата обращения 24.05.2023).
7. Канер Кем, Фолк Джек, Нгуен Енг Кек. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. – Киев: ДиаСофт, 2001. – 544 с.
8. Функциональное тестирование программного обеспечения [Электронный ресурс]. URL: <https://daglab.ru/funkcionalnoe-testirovanie-programmnogo-obespechenija/> (дата обращения 24.05.2023).
9. Нефункциональное тестирование [Электронный ресурс]. URL: <https://xbsoftware.ru/testirovanie-po/polnij-tsykl/nefunctional-test/> (дата обращения 24.05.2023).
10. Для чего нужно интеграционное тестирование? [Электронный ресурс]. URL: <https://habr.com/ru/post/556002/> (дата обращения 24.05.2023).

Приложение А.
Листинг для базы данных
(рекомендуемое)

Листинг А.1 – Скрипты для создания объектов базы данных

```

USE APTEKA;
CREATE TABLE Buyer (IDBuyer int IDENTITY NOT NULL, PhoneNumber
varchar(11) NOT NULL UNIQUE CHECK(PhoneNumber like replicate('[0-
9]',len(PhoneNumber))), NOOrders int NOT NULL CHECK(NOOrders>0),
Permanent binary(1) NOT NULL, PRIMARY KEY (IDBuyer));
CREATE TABLE BuyerOrder (IDOrder int IDENTITY NOT NULL
CHECK(IDOrder%2=0), DateOrder date NOT NULL CHECK(DateOrder
between '01.01.2023' and '01.01.2050'), PhoneNumber varchar(11)
NOT NULL CHECK(PhoneNumber like replicate('[0-
9]',len(PhoneNumber))), Cost real NOT NULL CHECK(Cost>0), IDPlace
int NOT NULL, IDWorker int NOT NULL, PRIMARY KEY (IDOrder));
CREATE TABLE Place (IDPlace int IDENTITY NOT NULL, Place
varchar(50) NOT NULL CHECK(Place like replicate('[0-9a-яA-Я.,
]',len(Place))), PRIMARY KEY (IDPlace));
CREATE TABLE Worker (IDWorker int IDENTITY NOT NULL, FIO
varchar(50) NOT NULL, Passport varchar(10) NOT NULL UNIQUE
CHECK(Passport like replicate('[0-9]',len(Passport))), PhoneNumber
varchar(11) NOT NULL UNIQUE CHECK(PhoneNumber like replicate('[0-
9]',len(PhoneNumber))), IDTitle int NOT NULL, PRIMARY KEY
(IDWorker));
CREATE TABLE Title (IDTitle int IDENTITY NOT NULL, Title
varchar(50) NOT NULL CHECK(Title like replicate('[0-9a-яA-Я
]',len(Title))), PRIMARY KEY (IDTitle));
CREATE TABLE ManagerOrder (IDOrder int IDENTITY NOT NULL
CHECK(IDOrder%2=1), DateOrder date NOT NULL CHECK(DateOrder
between '01.01.2023' and '01.01.2050'), IDSupplier int NOT NULL,
Cost real NOT NULL CHECK(Cost>0), IDWorker int NOT NULL, PRIMARY
KEY (IDOrder));
CREATE TABLE PartOfBOrder (IDPartOfOrder int IDENTITY NOT NULL,
IDOrder int NOT NULL, IDConcretMed int NOT NULL, Count int NOT
NULL CHECK(Count>0), Cost real NOT NULL CHECK(Cost>0), PRIMARY KEY
(IDPartOfOrder));
CREATE TABLE Supplier (IDSupplier int IDENTITY NOT NULL, SupName
varchar(50) NOT NULL CHECK(SupName like replicate('[0-9a-яA-Я
]',len(SupName))), Adress varchar(50) NOT NULL, PhoneNumber
varchar(11) NOT NULL UNIQUE CHECK(PhoneNumber like replicate('[0-
9]',len(PhoneNumber))), PRIMARY KEY (IDSupplier));
CREATE TABLE SupMed (IDSupMed int IDENTITY NOT NULL, IDSupplier
int NOT NULL, IDMedicine int NOT NULL, Cost real NOT NULL
CHECK(Cost>0), PRIMARY KEY (IDSupMed));
CREATE TABLE Medicine (IDMedicine int IDENTITY NOT NULL, Name
varchar(50) NOT NULL CHECK(Name like replicate('[0-9a-яA-Я
]',len(Name))), IDActiveSub int NOT NULL, IDManufacturer int NOT
NULL, PRIMARY KEY (IDMedicine));
CREATE TABLE ActiveSub (IDActiveSub int IDENTITY NOT NULL,
ActiveSub varchar(50) NOT NULL CHECK(ActiveSub like replicate('[0-
9a-яA-Я ]',len(ActiveSub))), PRIMARY KEY (IDActiveSub));
CREATE TABLE Category (IDCategory int IDENTITY NOT NULL, Category
varchar(50) NOT NULL CHECK(Category like replicate('[a-яA-Я
]',len(Category))), PRIMARY KEY (IDCategory));

```

```

CREATE TABLE Form (IDForm int IDENTITY NOT NULL, Form varchar(50)
NOT NULL CHECK(Form like replicate('[a-яA-Я ]',len(Form))),
PRIMARY KEY (IDForm));
CREATE TABLE Manufacturer (IDManufacturer int IDENTITY NOT NULL,
Manufacturer varchar(50) NOT NULL CHECK(Manufacturer like
replicate('[0-9a-яA-Я ]',len(Manufacturer))), PRIMARY KEY
(IDManufacturer));
CREATE TABLE Medicine_Category (MedicineIDMedicine int NOT NULL,
CategoryIDCategory int NOT NULL, PRIMARY KEY (MedicineIDMedicine,
CategoryIDCategory));
CREATE TABLE ConcretMed (IDConcretMed int IDENTITY NOT NULL,
IDMedicine int NOT NULL UNIQUE, IDForm int NOT NULL, Cost real NOT
NULL, BestBefore date NOT NULL CHECK(BestBefore between
'01.01.2023' and '01.01.2050'), PRIMARY KEY (IDConcretMed));
CREATE TABLE PartOfMOrder (IDPartOfOrder int IDENTITY NOT NULL,
IDOrder int NOT NULL, IDConcretMed int NOT NULL, Count int NOT
NULL CHECK(Count>0), Cost real NOT NULL CHECK(Cost>0), PRIMARY KEY
(IDPartOfOrder));

ALTER TABLE BuyerOrder ADD CONSTRAINT FKBuyerOrder988633 FOREIGN
KEY (PhoneNumber) REFERENCES Buyer (PhoneNumber);
ALTER TABLE BuyerOrder ADD CONSTRAINT FKBuyerOrder642277 FOREIGN
KEY (IDPlace) REFERENCES Place (IDPlace);
ALTER TABLE BuyerOrder ADD CONSTRAINT FKBuyerOrder376592 FOREIGN
KEY (IDWorker) REFERENCES Worker (IDWorker);
ALTER TABLE Worker ADD CONSTRAINT FKWorker647194 FOREIGN KEY
(IDTitle) REFERENCES Title (IDTitle);
ALTER TABLE ManagerOrder ADD CONSTRAINT FKManagerOrd315363 FOREIGN
KEY (IDWorker) REFERENCES Worker (IDWorker);
ALTER TABLE PartOfMOrder ADD CONSTRAINT FKPartOfMOrd505978 FOREIGN
KEY (IDOrder) REFERENCES ManagerOrder (IDOrder);
ALTER TABLE PartOfBOrder ADD CONSTRAINT FKPartOfBOrd147319 FOREIGN
KEY (IDOrder) REFERENCES BuyerOrder (IDOrder);
ALTER TABLE ManagerOrder ADD CONSTRAINT FKManagerOrd458844 FOREIGN
KEY (IDSupplier) REFERENCES Supplier (IDSupplier);
ALTER TABLE SupMed ADD CONSTRAINT FKSupMed171799 FOREIGN KEY
(IDSupplier) REFERENCES Supplier (IDSupplier);
ALTER TABLE SupMed ADD CONSTRAINT FKSupMed632162 FOREIGN KEY
(IDMedicine) REFERENCES Medicine (IDMedicine);
ALTER TABLE Medicine ADD CONSTRAINT FKMedicine573768 FOREIGN KEY
(IDManufacturer) REFERENCES Manufacturer (IDManufacturer);
ALTER TABLE Medicine ADD CONSTRAINT FKMedicine392514 FOREIGN KEY
(IDActiveSub) REFERENCES ActiveSub (IDActiveSub);
ALTER TABLE Medicine_Category ADD CONSTRAINT FKMedicine_C663064
FOREIGN KEY (MedicineIDMedicine) REFERENCES Medicine (IDMedicine);
ALTER TABLE Medicine_Category ADD CONSTRAINT FKMedicine_C637754
FOREIGN KEY (CategoryIDCategory) REFERENCES Category (IDCategory);
ALTER TABLE ConcretMed ADD CONSTRAINT FKConcretMed849091 FOREIGN
KEY (IDMedicine) REFERENCES Medicine (IDMedicine);
ALTER TABLE ConcretMed ADD CONSTRAINT FKConcretMed281218 FOREIGN
KEY (IDForm) REFERENCES Form (IDForm);
ALTER TABLE PartOfMOrder ADD CONSTRAINT FKPartOfMOrd704112 FOREIGN
KEY (IDConcretMed) REFERENCES ConcretMed (IDConcretMed);

```

```
ALTER TABLE PartOfBOrder ADD CONSTRAINT FKPartOfBOrd783136 FOREIGN
KEY (IDConcretMed) REFERENCES ConcretMed (IDConcretMed);
```

Листинг А.2 – Скрипты запросов к базе данных

```
SELECT dbo.Buyer.PhoneNumber AS Номер_телефона_покупателя,
dbo.Category.Category AS Категория,
count(dbo.PartOfBOrder.IDOrder) as Количество
FROM      dbo.Buyer INNER JOIN
dbo.BuyerOrder ON dbo.Buyer.PhoneNumber =
dbo.BuyerOrder.PhoneNumber INNER JOIN
dbo.PartOfBOrder ON dbo.BuyerOrder.IDOrder =
dbo.PartOfBOrder.IDOrder INNER JOIN
dbo.ConcretMed ON dbo.PartOfBOrder.IDConcretMed =
dbo.ConcretMed.IDConcretMed INNER JOIN
dbo.Medicine ON dbo.ConcretMed.IDMedicine =
dbo.Medicine.IDMedicine INNER JOIN
dbo.Medicine_Category ON dbo.Medicine.IDMedicine =
dbo.Medicine_Category.MedicineIDMedicine INNER JOIN
dbo.Category ON dbo.Category.IDCategory =
dbo.Medicine_Category.CategoryIDCategory Group by
dbo.Buyer.PhoneNumber, dbo.Category.Category;
```

```
SELECT dbo.Buyer.PhoneNumber AS Номер_телефона_покупателя,
dbo.Buyer.NOrders AS Количество_заказов
FROM dbo.Buyer INNER JOIN dbo.BuyerOrder ON dbo.Buyer.PhoneNumber
= dbo.BuyerOrder.PhoneNumber order by dbo.Buyer.NOrders Desc;
```

```
Create View AboutBOrder AS
SELECT dbo.BuyerOrder.IDOrder AS Номер_заказа,
dbo.BuyerOrder.DateOrder AS Дата,
dbo.Buyer.PhoneNumber AS Номер_телефона_покупателя,
dbo.Medicine.Name AS Название_лекарства,
dbo.PartOfBOrder.Count AS Количество,
dbo.PartOfBOrder.Cost AS Стоимость
FROM dbo.Buyer INNER JOIN dbo.BuyerOrder ON dbo.Buyer.PhoneNumber
= dbo.BuyerOrder.PhoneNumber
INNER JOIN dbo.PartOfBOrder ON dbo.BuyerOrder.IDOrder =
dbo.PartOfBOrder.IDOrder
INNER JOIN dbo.ConcretMed ON dbo.PartOfBOrder.IDConcretMed =
dbo.ConcretMed.IDConcretMed
INNER JOIN dbo.Medicine ON dbo.ConcretMed.IDMedicine =
dbo.Medicine.IDMedicine;
```

```
Create View AboutMed AS SELECT
dbo.Medicine.Name AS Название_лекарства,
dbo.ActiveSub.ActiveSub AS Действующее_вещество,
dbo.Manufacturer.Manufacturer AS Производитель,
dbo.Category.Category AS Категория,
dbo.Form.Form as Форма_выпуска,
dbo.ConcretMed.Cost AS Стоимость
FROM dbo.Medicine
```

```

INNER JOIN dbo.ActiveSub ON dbo.Medicine.IDActiveSub =
dbo.ActiveSub.IDActiveSub
INNER JOIN dbo.Manufacturer ON dbo.Medicine.IDManufacturer =
dbo.Manufacturer.IDManufacturer
INNER JOIN dbo.Medicine_Category ON dbo.Medicine.IDMedicine =
dbo.Medicine_Category.MedicineIDMedicine
INNER JOIN dbo.Category ON
dbo.Medicine_Category.CategoryIDCategory = dbo.Category.IDCategory
INNER JOIN dbo.ConcretMed ON dbo.Medicine.IDMedicine =
dbo.ConcretMed.IDMedicine
INNER JOIN dbo.Form ON dbo.ConcretMed.IDForm = dbo.Form.IDForm;

```

Листинг А.3 – Исходные коды хранимых процедур и функций модификации данных

```

Create procedure AddNewBuyerOrder
@DO date, @PN varchar(11), @CM int, @Count int, @Pl int, @W int as
declare @cst real
declare @i int
Set @cst = (Select ConcretMed.Cost*@Count as cst from
APTEKA.dbo.ConcretMed where ConcretMed.IDConcretMed=@CM)
IF exists (SELECT * from BuyerOrder WHERE BuyerOrder.DateOrder =
@DO and BuyerOrder.PhoneNumber = @PN and BuyerOrder.IDPlace=@Pl)
begin
    Select @i=(Select BuyerOrder.IDOrder as i from BuyerOrder
where BuyerOrder.DateOrder = @DO and BuyerOrder.PhoneNumber = @PN
and BuyerOrder.IDPlace=@Pl)
    PRINT @i
    Exec NewPartOfBOrder @i,@CM,@Count,@cst
    Exec UpdateBOrder @i,@cst
end
ELSE begin
    IF exists (SELECT * from Buyer WHERE Buyer.PhoneNumber = @PN)
        Exec AddNumberOrders @PN
    ELSE
        Exec NewBuyer @PN
    Exec NewBuyerOrder @PN,@DO,@cst,@Pl,@W
    Select @i=(Select BuyerOrder.IDOrder from BuyerOrder where
BuyerOrder.DateOrder = @DO and BuyerOrder.PhoneNumber = @PN and
BuyerOrder.IDPlace=@Pl)
    Exec NewPartOfBOrder @i,@CM,@Count,@cst
End

Create procedure NewPartOfBOrder
@IDO int,
@IDCM int,
@Count int,
@W real
As INSERT INTO PartOfMOrder values
(@IDO,@IDCM,@Count,@W);

```

```
Create procedure UpdateBOrder
@ON int,
@C real as
UPDATE BuyerOrder SET BuyerOrder.Cost=BuyerOrder.Cost+@C
WHERE BuyerOrder.IDOrder = @ON;
```

```
Create procedure UpdatePartOfBorder
@ON int,
@Col int as
UPDATE PartOfBOrder SET PartOfBOrder.Count=@Col
WHERE PartOfBOrder.IDPartOfOrder = @ON;
```

```
Create procedure AddNumberOrders
@PN varchar(11) as
UPDATE Buyer SET Buyer.NOrders=Buyer.NOrders+1
WHERE Buyer.PhoneNumber = @PN;
```

```
Create procedure DelBuyerOrder
@W int
As Delete from BuyerOrder where BuyerOrder.IDOrder=@W;
```

```
Create procedure NewBuyerOrder
@NPhone varchar(11),
@Dat date,
@Cost real,
@Pl int,
@W int
As INSERT INTO BuyerOrder values
(@Dat,@NPhone,@Cost,@Pl,@W) Select * from BuyerOrder;
```

```
Create Procedure NewBuyer
@PNum varchar(11) as
INSERT INTO dbo.Buyer Values (@PNum, 1, 0) Select * from Buyer;
```

Листинг А.4 – Исходные коды триггеров

```
Create trigger OnBOrderAdd on BuyerOrder after insert as
declare @id int if @@ROWCOUNT=1
begin
Select @id-IDOrder from Inserted
print 'Заказ успешно добавлен!'
Select * from BuyerOrder where BuyerOrder.IDOrder=@id
end;
```

```
Create trigger OnBOrderUpd on PartOfBOrder after update as
declare @id int, @pr real if @@ROWCOUNT=1
begin
Select @id-IDOrder from Deleted
Select @pr=(select ConcretMed.Cost from ConcretMed, PartOfBOrder
where ConcretMed.IDConcretMed=PartOfBOrder.IDConcretMed)
```

```
update PartOfBOrder set Cost=(select PartOfBOrder.Count from
PartOfBOrder where PartOfBOrder.IDOrder=@id)*@pr where
PartOfBOrder.IDOrder=@id
print 'Заказ успешно обновлен!'
end;
```

```
Create trigger OnBOrderDel on BuyerOrder instead of delete as
declare @id int if @@ROWCOUNT=1
begin
Select @id=IDOrder from Deleted
Delete from PartOfBOrder where @id=PartOfBOrder.IDOrder
Delete from BuyerOrder where @id = BuyerOrder.IDOrder
print 'Заказ и его составляющие успешно удалены!'
end;
```

Приложение Б.
Листинг приложения
(рекомендуемое)

Листинг Б.1 – CartActivity.kt

```

package com.example.ecomercetutorial

import android.content.Context
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.AdapterView
import android.widget.Button
import android.widget.ListView
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import models.OrderPart
import java.sql.Connection

class CartActivity : AppCompatActivity() {
    var mssqlConnection: MSSQLConnection? = null
    var connection: Connection? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_cart)

        val buttonCatalog: Button =
            findViewById(R.id.buttonToCatalog)
        val buttonPlace: Button =
            findViewById(R.id.buttonToPlace)
        val listCartParts: RecyclerView =
            findViewById(R.id.listCartParts)
        val listCartPlaces: ListView =
            findViewById(R.id.listCartPlaces)

        mssqlConnection = MSSQLConnection()
        connection = mssqlConnection!!.getConnection()

        val sharedPref = getSharedPreferences("PREFERENCES_USER",
            Context.MODE_PRIVATE)
        val cartID = sharedPref.getInt("cartID", 0)

        if (cartID != 0) {
            // Позиции заказа
            var part: OrderPart
            var parts = arrayListOf<OrderPart>()
            val sqlParts = "SELECT
p.IDPartOfOrder,p.IDOrder,p.IDConcretMed,med.Name,p.Count
,p.Cost " +
                "FROM PartOfBOrder p " +
                "INNER JOIN ConcretMed cmed on p.IDConcretMed =
cmed.IDConcretMed " +
                "INNER JOIN Medicine med on cmed.IDMedicine =
med.IDMedicine " +

```

```

        "WHERE p.IDOrder = $cartID"
        val partSet =
mssqlConnection!!.executeMyQuery(sqlParts)
        while (partSet!!.next()) {
            part = OrderPart(
                partSet.getInt("IDPartOfOrder"),
                partSet.getInt("IDOrder"),
                partSet.getInt("IDConcretMed"),
                partSet.getString("Name"),
                partSet.getInt("Count"),
                partSet.getFloat("Cost").toInt(),
            )
            parts.add(part)
        }
        listCartParts.layoutManager = LinearLayoutManager(this)
        listCartParts.adapter = OrderPartAdapter(parts, this)

        // Пункты доставки
        var checkedPlaceId = 0
        var places = arrayListOf<String>()
        var placesID = arrayListOf<Int>()

        val sqlPlaces = "SELECT * FROM Place"
        val placeSet =
mssqlConnection!!.executeMyQuery(sqlPlaces)
        while (placeSet!!.next()) {
            places.add(placeSet.getString("Place"))
            placesID.add(placeSet.getInt("IDPlace"))
        }
        listCartPlaces.adapter = ArrayAdapter(this,
            android.R.layout.simple_list_item_single_choice, places)
        listCartPlaces.setChoiceMode(ListView.CHOICE_MODE_SINGLE)
        listCartPlaces.setOnItemClickListener { parent, view,
position, id ->
            checkedPlaceId = placesID[position]
        }
    }

    buttonCatalog.setOnClickListener {
        val intent = Intent(this, CatalogActivity::class.java)
        startActivity(intent)
    }

    buttonPlace.setOnClickListener {

    }

}

```

Листинг Б.2 – CatalogActivity.kt

```
package com.example.ecomercetutorial
```

```

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.Button
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import models.Product
import java.sql.Connection

class CatalogActivity : AppCompatActivity() {

    var connection: Connection? = null
    var mssqlConnection: MSSQLConnection? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_catalog)

        val productsView : RecyclerView =
            findViewById(R.id.product_list)
        val buttonCart : Button = findViewById(R.id.buttonCart)
        val products = arrayListOf<Product>()
        // TODO: to model
        mssqlConnection = MSSQLConnection()
        connection = mssqlConnection!!.getConnection()
        var productSet = mssqlConnection!!.executeMyQuery("SELECT
* FROM BayerCatalog")

        var product: Product
        while(productSet!!.next()) {
            product = Product(
                productSet.getInt("IDMedicine"),
                productSet.getInt("IDConcretMed"),
                productSet.getString("Name"),
                productSet.getString("Manufacturer"),
                productSet.getString("Form"),
                productSet.getString("ActiveSub"),
                productSet.getDate("BestBefore"),
                productSet.getString("Categories"),
                productSet.getFloat("Cost").toInt()
            )
            products.add(product)
        }

        productsView.layoutManager = LinearLayoutManager(this)
        var adapter = ProductsAdapter(products, this)
        adapter.onClick = { product ->
            val intent = Intent(this,
                ProductActivity::class.java)
            intent.putExtra("EXTRA_PRODUCT", product)
            startActivity(intent)
        }
    }
}

```

```

    }
    productsView.adapter = adapter

    buttonCart.setOnClickListener {
        val intent = Intent(this, CartActivity::class.java)
        startActivity(intent)
    }
}
}

```

Листинг Б.3 – LoginActivity.kt

```

package com.example.ecomercetutorial

import android.content.Context
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import java.sql.Connection
import java.sql.ResultSet
import java.sql.Statement

class LoginActivity : AppCompatActivity() {

    var connection: Connection? = null
    var mssqlConnection: MSSQLConnection? = null
    var userSet: ResultSet? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        mssqlConnection = MSSQLConnection()
        connection = mssqlConnection!!.getConnection()

        val sharedPreferences =
            getSharedPreferences("PREFERENCES_USER" ,Context.MODE_PRIVATE)
            with (sharedPreference.edit()) {
                putString("phoneNumber","88003332211")
                apply()
            }

        val intent = Intent(this, CatalogActivity::class.java)
        intent.putExtra("EXTRA_USER_PHONE", "88001112233")
        startActivity(intent)

        var editPhone: EditText = findViewById(R.id.edit_phone)
    }
}

```

```

var editPass: EditText = findViewById(R.id.edit_password)
var buttonReg: Button = findViewById(R.id.button_login)

buttonReg.setOnClickListener {
    val phone = editPhone.text.toString().trim()
    val pass = editPass.text.toString().trim()

    if (phone != "" && pass != "") {
        //TODO: to model
        var isAuth = false
        userSet = mssqlConnection!!.executeMyQuery("SELECT
COUNT(*) FROM [Buyer] WHERE [PhoneNumber] = $phone")
        userSet!!.next()
        if (userSet!!.getInt(1) != 0) {
            userSet =
mssqlConnection!!.executeMyQuery("SELECT COUNT(*)
FROM [Buyer] WHERE [PhoneNumber] = $phone AND
[Password] = $pass")
            userSet!!.next()
            if (userSet!!.getInt(1) != 0) {
                isAuth = true
            } else {
                Toast.makeText(this, "Не правильный логин или
пароль", Toast.LENGTH_SHORT).show()
            }
        } else {
            var statment: Statement =
connection!!.createStatement()
            if (statment.executeUpdate("INSERT into [Buyer]
([PhoneNumber],[Password],[NOrders],[Permanent]) VALUES
($phone, $pass, 0, 0)") != 0) {
                isAuth = true
            }
        }
        if (isAuth) {
            val intent = Intent(this,
CatalogActivity::class.java)
            startActivity(intent)
        }
    } else {
        Toast.makeText(this, "Введите учетные данные чтобы
продолжить", Toast.LENGTH_SHORT).show()
    }
}
}

```

Листинг Б.4 – MSSQLConnection.kt

```

package com.example.ecomercetutorial
import android.os.StrictMode

```

```

import java.lang.Exception
import java.sql.Connection
import java.sql.DriverManager
import java.sql.ResultSet
import java.sql.SQLException
import java.sql.Statement
//import java.sql.*
import java.util.Properties
class MSSQLConnection {
    var conn: Connection? = null
    //private val ip = "192.168.251.17"
    //private val ip = "10.166.125.127"
    private val ip = "192.168.1.6" // database server ip
    private val port = "1433" // database server port
    private val db = "APTEKA" // database name
    private val username = "test" // database username
    private val password = "test" // database password
    fun getConnection(): Connection? {
        if (conn != null)
            return conn
        val policy =
            StrictMode.ThreadPolicy.Builder().permitAll().build()
            StrictMode.setThreadPolicy(policy)
        val connectionUrl: String =
            "jdbc:jtds:sqlserver://" + ip + ":" + port + "/" + db
        try {
            Class.forName("net.sourceforge.jtds.jdbc.Driver").newInstance()
            conn = DriverManager.getConnection(connectionUrl,
                username, password)
        } catch (ex: SQLException) {
            ex.printStackTrace()
        } catch (ex: Exception) {
            ex.printStackTrace()
        }
        return conn
    }
    fun executeMyQuery(queryString: String): ResultSet? {
        var stmt: Statement? = null
        var resultset: ResultSet? = null
        try {
            stmt = conn!!.createStatement()

            if (stmt.execute(queryString)) {
                resultset = stmt.resultSet
            }

        } catch (ex: SQLException) {
            ex.printStackTrace()
        }
        return resultset
    }
}

```

Листинг Б.5 – OrderPartAdapter.kt

```
package com.example.ecomercetutorial

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import models.OrderPart

class OrderPartAdapter(val items: List<OrderPart>, val context:
Context): RecyclerView.Adapter<OrderPartAdapter.ViewHolder>() {
    class ViewHolder(itemView: View):
RecyclerView.ViewHolder(itemView) {
        var name: TextView =
            itemView.findViewById(R.id.textPartName)
        var cost: TextView =
            itemView.findViewById(R.id.textPartCost)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): OrderPartAdapter.ViewHolder {
        var view =
            LayoutInflater.from(parent.context).inflate(R.layout.order_par
t_list_item, parent, false)
        return ViewHolder(view)
    }

    override fun onBindViewHolder(holder:
OrderPartAdapter.ViewHolder, position: Int) {
        holder.name.text = items[position].name
        holder.cost.text = items[position].cost.toString() + " ₸"
    }

    override fun getItemCount(): Int {
        return items.count()
    }
}
```

Листинг Б.6 – ProductsAdapter.kt

```
package com.example.ecomercetutorial

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
```

```

import models.Product

class ProductsAdapter(
    var items: List<Product>,
    var context: Context,
): RecyclerView.Adapter<ProductsAdapter.ViewHolder>() {

    var onItemClick: ((Product) -> Unit)? = null

    inner class ViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        var name: TextView = itemView.findViewById(R.id.textName)
        var form: TextView = itemView.findViewById(R.id.textForm)
        var category: TextView =
itemView.findViewById(R.id.textCategory)
        var cost: TextView = itemView.findViewById(R.id.textCost)

        init {
            itemView.setOnClickListener {
                onItemClick?.invoke(items[adapterPosition])
            }
        }

        override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): ViewHolder {
            var view =
LayoutInflater.from(parent.context).inflate(R.layout.product_1
ist_item, parent, false)
            return ViewHolder(view)
        }

        override fun getItemCount(): Int {
            return items.count()
        }

        override fun onBindViewHolder(holder: ViewHolder, position:
Int) {
            holder.name.text = items[position].name
            holder.form.text = items[position].form
            holder.category.text = items[position].category
            holder.cost.text = items[position].cost.toString() + " ₺"
        }
    }
}

```