

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

**Разработка информационной подсистемы для
регистрации, обработки и контроля исполнения
заказов**

Студент _____ Овечкина Екатерина Александровна
(подпись, дата) (ФИО полностью)

Руководитель _____ Попова Н.А.
(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, 2026

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
«26» декабря 2025 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Овечкина Екатерина Александровна гр. 22ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Разработка информационной подсистемы для регистрации, обработки и контроля исполнения заказов

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

2. Руководитель работы к.т.н. доцент, доцент кафедры «МО и ПЭВМ» Попова Н.А.

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

1) Назначение: проектирование и разработка информационной подсистемы управления и мониторинга заказов ControlDesk для автоматизации операционной деятельности предприятия

2)Среда выполнения: Windows 10

Языковые средства: C#, XAML, SQL, UML.

Инструментальные средства: Среда разработки Microsoft Visual Studio, СУБД Microsoft SQL Server, средства проектирования диаграмм (Visual Paradigm).

Технологии разработки: Платформа Windows Presentation Foundation (WPF), технология объектно-реляционного отображения Entity Framework 6, архитектурный паттерн MVVM (Model-View-ViewModel), платформа .NET Framework 4.8.1, протокол передачи почты SMTP.

3)Основные функции приложения:

а) регистрация и ведение заказов с фиксацией контрагентов, ответственных лиц

б) управление жизненным циклом заказов через систему динамических статусов;

в) автоматизированный мониторинг исполнения и визуальная индикация;

г) ведение нормативно-справочной информации (базы клиентов и сотрудников);

д) разграничение прав доступа к функциям и данным на основе ролевой модели (Администратор, Менеджер, Исполнитель);

е) автоматическая отправка Email-уведомлений клиентам при изменении состояния их заказов;

ж) формирование аналитической отчетности в виде _____
сводных таблиц с возможностью экспорта в формат CSV. _____

4. Перечень подлежащих разработке вопросов _____
1. Анализ предметной области и постановка задачи _____
2. Проектирование; _____
3. Реализация; _____
4. Контроль качества системы; _____
5. Оформление пояснительной записки. _____

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20%	14.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20%	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20%	22.05.2026 – 02.06.2026	

Дата выдачи задания «26» декабря 2025г.

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 93 листа, 36 рисунков, 22 таблиц, 21 использованный источник и 6 приложений.

ИНФОРМАЦИОННАЯ ПОДСИСТЕМА, УПРАВЛЕНИЕ ЗАКАЗАМИ, МОНИТОРИНГ ИСПОЛНЕНИЯ, МНОГОСЛОЙНАЯ АРХИТЕКТУРА, MVVM, UML, VISUAL STUDIO, C#, WPF, .NET FRAMEWORK, ENTITY FRAMEWORK, SQL SERVER.

Объект исследования – процессы автоматизации операционного управления заказами и контроля исполнительской дисциплины в коммерческих организациях.

Цель выпускной квалификационной работы – спроектировать и разработать информационную подсистему ControlDesk для автоматизации регистрации, сопровождения и аналитического учета заказов.

Технологии разработки – проектирование и программная реализация подсистемы проводились на языке программирования C# с использованием графической платформы WPF в среде разработки Microsoft Visual Studio. Взаимодействие с данными реализовано с помощью технологии объектно-реляционного отображения Entity Framework 6. База данных спроектирована и развернута в Microsoft SQL Server. Визуальное моделирование архитектуры и процессов системы выполнено с использованием языка UML.

Среда исполнения – Windows 10.

Результаты работы – в ходе выполнения выпускной квалификационной работы была создана подсистема для управления заказами. Реализованы модули авторизации с разграничением прав доступа, система Email-уведомлений и конструктор отчетности. Проведенное тестирование подтвердило полную работоспособность системы и её соответствие всем заявленным требованиям.

					ПГУ 09.03.02 – 08BP221.14 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Овечкина Е.А.			Разработка. информационной подсистемы для регистрации, обработки и контроля исполнения заказов Пояснительная записка	Лит.	Лист	Листов
Пров.		Попова Н.А..					4	93
Реценз.						Группа 22ВИ1		
Н. Контр.		Такташкин Д.В						
Утв.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ предметной области и постановка задачи.....	9
1.1 Постановка задачи.....	9
1.2 Анализ информационных систем аналогичного назначения.....	11
1.2.1 Анализ системы Битрикс24.....	11
1.2.2 Анализ системы amoCRM.....	12
1.2.3 1C:CRM.....	12
1.3 Анализ функциональных требований к системе.....	14
1.4 Анализ нефункциональных требований	18
1.5 Выбор технологий проектирования системы	20
1.6 Выбор инструментальных средств	22
1.7 Планирование разработки и оценка бюджета.....	23
2 Проектирование информационной подсистемы для регистрации обработки и контроля исполнения заказов.....	27
2.1 Проектирование архитектуры приложения	27
2.2 Проектирование структур данных	29
2.2.1 Концептуальная модель базы данных.....	29
2.2.2 Логическая модель базы данных	31
2.3 Проектирование программных средств	31
2.4 Проектирование пользовательского интерфейса	36
3 Реализация информационной подсистемы для регистрации, обработки и контроля исполнения заказов.....	39
3.1 Реализация базы данных.....	39
3.2 Программная реализация функциональных модулей и механизмов управления данными.....	40
3.3 Программная реализация интерфейса и логики представления подсистемы	42
3.4 Реализация интерфейса информационной подсистемы для регистрации, обработки и контроля исполнения заказов.....	44
4 Контроль качества программного обеспечения.....	53
4.1 Выбор режима тестирования.....	53
4.2 Модульное тестирование подсистемы ControlDesk.....	54
4.3 Функциональное тестирование подсистемы ControlDesk.....	56

4.4	Статический анализ качества программного кода	61
	Заключение.....	63
	Список использованных источников.....	65
	Приложение А.....	67
	Приложение Б	71
	Приложение В.....	77
	Приложение Г	81
	Приложение Д.....	85
	Приложение Ж.....	89

Введение

В условиях цифровой экономики предприятия разных масштабов и отраслей вынуждены повышать скорость и качество обработки заказов, сокращать время на рутинные операции и усиливать контроль над бизнес-процессами. Заказ является одним из ключевых элементов взаимодействия с клиентами и партнёрами. От того, насколько оперативно и точно происходит регистрация, обработка и контроль исполнения заказов, зависит удовлетворённость потребителей, эффективность персонала и устойчивость предприятия на рынке [1].

Особенно остро проблема ручной или плохо организованной обработки заказов стоит перед малым и средним бизнесом. Согласно Федеральному закону № 209-ФЗ «О развитии малого и среднего предпринимательства в Российской Федерации», к малым предприятиям относятся организации с численностью до 100 человек и годовым доходом до 800 млн рублей [с. 56]. При ограниченных ресурсах такие компании не могут позволить себе большие отделы для ручной обработки заказов и дорогие корпоративные системы. Поэтому именно автоматизация обработки заказов становится для малого бизнеса критически важным условием, без которого сложно конкурировать с крупными компаниями.

При этом именно в малом и среднем бизнесе обработка заказов чаще всего остаётся ручной. Такой метод работы часто приводит к ошибкам, потерям данных и задержкам. Проблема усугубляется при большом количестве заказов и необходимости постоянного отслеживания их статуса. Программные средства позволяют хранить данные в единой системе, оперативно распределять задачи и контролировать соблюдение сроков [2].

Развитие информационных систем для управления заказами напрямую связано с общим трендом на цифровизацию и объединение бизнес-процессов. Современные программы позволяют автоматически принимать заказы, отслеживать их выполнение, менять статусы, согласовывать работу разных отделов и готовить отчёты [3]. Для компании это означает более прозрачные процессы, более быструю реакцию на клиентские запросы, меньшую нагрузку

на сотрудников и в итоге – более качественный сервис. Особенно заметный эффект дают системы, которые собирают и анализируют данные по заказам в одном месте, а потом используют их для дальнейшей настройки работы предприятия.

Актуальность темы выпускной квалификационной работы обусловлена тем, что во многих организациях процессы регистрации, обработки и контроля исполнения заказов до сих пор осуществляются либо вручную, либо с использованием разрозненных инструментов (электронные таблицы, мессенджеры, бумажные журналы), что снижает эффективность работы и увеличивает вероятность ошибок. Разработка информационной подсистемы для автоматизации этих операций позволяет повысить оперативность управления заказами, обеспечить прозрачный контроль, улучшить качество обслуживания клиентов и создать основу для дальнейшего развития информационной инфраструктуры предприятия [4].

Таким образом, тема работы является своевременной и практически значимой для современного предпринимательства в условиях цифровой экономики.

1 Анализ предметной области и постановка задачи

1.1 Постановка задачи

Тема выпускной квалификационной работы посвящена разработке информационной подсистемы для регистрации, обработки и контроля исполнения заказов. Внедрение информационных технологий позволяет существенно повысить качество обработки данных, обеспечить их централизованное хранение и организовать оперативное взаимодействие между сотрудниками.

Для обеспечения структурированной работы и разграничения зон ответственности в разрабатываемой подсистеме необходимо предусмотреть следующие категории пользователей:

- администратор – обладает полными правами доступа к системе;
- менеджер – отвечает за операционную работу с заказами;
- исполнитель – имеет доступ к просмотру назначенных на него заказов.

Реализация прав доступа тесно связана с функциональным наполнением системы. Исходя из анализа требований к автоматизации процесса управления заказами, программный продукт должен обеспечивать выполнение следующих функций:

- регистрация новых заказов с указанием данных клиента, описания задачи, приоритета и планового срока исполнения;
- ведение и актуализация справочников клиентов, сотрудников и штатных должностей;
- управление жизненным циклом заказа через систему последовательных статусов (новый, в обработке, выполнен, отменен);
- автоматическая рассылка E-mail уведомлений ответственным лицам при изменении состояния заказа;
- мониторинг соблюдения дедлайнов и визуальное выделение просроченных заказов в интерфейсе;

- хранение истории изменений по каждой заявке для обеспечения прослеживаемости процесса выполнения;
- формирование отчетов с возможностью гибкой фильтрации данных и их последующего экспорта в формат CSV.

Предметом выпускной квалификационной работы является разработка информационной подсистемы для регистрации, обработки и контроля исполнения заказов.

Объектом работы выступает процесс автоматизации учета и управления заказами в организации.

Цель данной работы заключается в проектировании и программной реализации информационной подсистемы, получившей название «ControlDesk», которая предназначена для обеспечения полного цикла учета и контроля исполнения заказов.

Для достижения поставленной цели определены следующие задачи:

- проведение анализа требований к системе и изучение специфики предметной области;
- рассмотрение существующих программных продуктов аналогичного назначения;
- выбор технологий проектирования и инструментальных средств разработки (C#, WPF, SQL Server);
- разработка логической структуры базы данных и архитектуры подсистемы;
- программная реализация механизмов регистрации заказов и управления их статусами;
- разработка модуля автоматического E-mail оповещения пользователей;
- проверка работоспособности и оценка качества готового программного продукта.

В результате выполнения работы должна быть разработана информационная подсистема «ControlDesk», обеспечивающая упорядоченное

хранение данных, автоматизацию рутинных операций и оперативное информирование клиентов по электронной почте. Реализация данного решения позволит минимизировать количество ошибок, повысить скорость обработки информации и обеспечить эффективный контроль за соблюдением сроков выполнения заказов в организации.

1.2 Анализ информационных систем аналогичного назначения

В настоящий момент существует значительное количество программных решений для управления заказами и задачами. Некоторые из них являются полноценными CRM-системами, другие – узкоспециализированными инструментами для командной работы. Далее будут рассмотрены наиболее распространённые и актуальные продукты, функционально близкие к разрабатываемой подсистеме.

1.2.1 Анализ системы Битрикс24

Первым рассматриваемым продуктом является Битрикс24 - российская платформа для управления бизнесом, включающая CRM, задачи, проекты и коммуникации. Активно используется как малым, так и средним бизнесом [5].

Система обладает широким набором функциональных возможностей:

- ведение воронки продаж и учёт заказов в CRM;
- назначение ответственных сотрудников, контроль дедлайнов;
- разграничение прав доступа по ролям;
- интеграция с почтой, телефонией и мессенджерами;
- встроенные отчёты и аналитика.

Достоинства платформы: функциональная полнота, наличие бесплатного тарифа, развитая система интеграций, регулярные обновления. Вместе с тем у Битрикс24 есть и существенные недостатки. Система требует постоянного подключения к интернету, поскольку является облачным решением. Интерфейс перегружен для пользователей, которым нужна только работа с заказами. Платные тарифы с расширенным функционалом достаточно дороги. Кроме того,

внедрение и настройка системы могут занимать значительное время, а освоение функционала требует дополнительного обучения сотрудников. Для небольших компаний часть возможностей системы может оказаться избыточной, что усложняет повседневную работу и снижает удобство использования.

1.2.2 Анализ системы amoCRM

amoCRM – это специализированная CRM-система для отдела продаж. Ориентирована на автоматизацию воронки и коммуникаций с клиентами [6].

К ключевым возможностям системы относятся:

- ведение сделок и задач по этапам воронки;
- уведомления о событиях и напоминания о задачах;
- разграничение прав пользователей;
- интеграция с мессенджерами и почтой;
- аналитика по воронке продаж и сотрудникам.

К достоинствам amoCRM можно отнести удобный интерфейс, ориентированность на продажи. Из недостатков выделяются: облачная модель без возможности локальной установки, ежемесячная абонентская плата, ограниченные возможности для формирования произвольных отчётов, избыточный акцент на продажи при слабой поддержке производственных заказов. Кроме того, при увеличении числа пользователей стоимость использования существенно возрастает.

1.2.3 1C:CRM

Третьим рассматриваемым продуктом является «1C:CRM». Решение на базе платформы 1С:Предприятие, предназначенное для автоматизации работы с клиентами и заказами [7].

Система предоставляет следующие возможности:

- ведение базы клиентов и истории взаимодействия;
- управление заказами и контроль сроков;
- разграничение прав по ролям пользователей;

- интеграция с другими конфигурациями 1С (торговля, бухгалтерия);
- гибкое формирование отчётности.

Несомненные достоинства продукта: возможность локальной установки, глубокая интеграция с экосистемой 1С, широкие возможности настройки. Вместе с тем 1С:CRM имеет высокий порог вхождения, требует значительных затрат на внедрение и лицензии, а интерфейс нередко воспринимается пользователями как устаревший и перегруженный.

Сравнительный анализ рассмотренных систем и разрабатываемой подсистемы ControlDesk представлен в таблице 1.

Таблица 1 – Сравнительный анализ программных продуктов

Критерий	Битрикс24	amoCRM	1С:CRM	ControlDesk (разрабатываемый)
Качество интерфейса	Среднее	Высокое	Низкое	Высокое
Локальная установка	Нет	Нет	Да	Да
Стоимость	Есть бесплатный тариф	Платная подписка	Высокая	Бесплатно (собственная разработка)
Гибкость отчётности	Средняя	Низкая	Высокая	Высокая
Разграничение ролей	Да	Да	Да	Да
Работа без интернета	Нет	Нет	Да	Да
Контроль просроченных	Да	Да	Да	Да

В результате анализа были выявлены сильные и слабые стороны существующих аналогов. Облачные решения (Битрикс24, amoCRM) требуют постоянного подключения к интернету и предполагают ежемесячную оплату, что неприемлемо для ряда организаций. Платформа 1С:CRM функционально богата, но сложна в освоении и дорогостояща в сопровождении. Разрабатываемая

подсистема ControlDesk ориентирована на конкретный круг задач: регистрацию и контроль исполнения заказов, ведение справочников клиентов и сотрудников, гибкую отчетность без избыточного функционала и с возможностью локальной работы.

1.3 Анализ функциональных требований к системе

Разрабатываемая информационная подсистема ControlDesk должна обеспечивать обработку заказов от момента их регистрации до завершения или отмены. Основные функциональные требования к подсистеме сформированы исходя из анализа предметной области и типовых сценариев работы с заказами.

Управление заказами является ключевым функциональным блоком. Подсистема должна обеспечивать регистрацию новых заказов с указанием клиента, ответственного сотрудника, описания, приоритета и срока исполнения. Заказ должен проходить через статусы, отражающие актуальное состояние работы: новый, в обработке, ожидает уточнения, выполнен. Отмена заказа реализуется не через его удаление, а путём смены статуса на «Отменён», что позволяет сохранять историю и обеспечивать прослеживаемость. Все изменения по заказу должны фиксироваться в истории.

Управление справочниками предполагает ведение базы клиентов и сотрудников. Список клиентов доступен для редактирования менеджерам и администраторам. Список сотрудников может редактировать только администратор, который также управляет назначением должностей.

Разграничение прав доступа – обязательное требование к подсистеме. В зависимости от роли пользователь получает доступ только к тем функциям, которые необходимы ему для выполнения служебных обязанностей: администратор управляет всеми данными системы, менеджер работает с заказами и клиентами, исполнитель видит только назначенные ему заказы.

Контроль исполнения и работа с просроченными заказами. Подсистема должна автоматически определять просроченные заказы на основе установленного срока исполнения. Пользователь должен иметь возможность

фильтровать список заказов по статусу, периоду, ответственному сотруднику и контрагенту.

Формирование отчётности реализуется в виде конструктора. Пользователь самостоятельно выбирает столбцы, которые должны отображаться в отчёте, а также задаёт параметры фильтрации: период дат заказов, клиент, ответственный сотрудник, статус заказа, вывод только просроченных записей. Сформированный отчёт должен быть доступен для сохранения в формате CSV.

Таким образом, сформированный перечень требований включает весь необходимый функционал: управление заказами с историей изменений, ведение справочников, разграничение прав доступа, контроль сроков и гибкую отчётность. Эти требования стали основой для проектирования и реализации подсистемы ControlDesk.

Перечень функциональных требований:

- авторизация и разграничение прав: обеспечение входа в систему по логину и паролю с использованием хэширования SHA256 и предоставление функционала согласно ролям (администратор, менеджер, исполнитель);
- учет и регистрация заказов: создание карточек заказов с фиксацией контрагента, ответственного лица, приоритета, описания и плановых сроков;
- управление составом заказа: возможность добавления перечня товаров или услуг в каждый заказ с указанием их количества и стоимости;
- автоматический расчет стоимости: динамическое вычисление общей суммы заказа на основе добавленных позиций;
- управление жизненным циклом (статусы): реализация переходов заказа между состояниями: «новый», «в обработке», «ожидает уточнения», «выполнен» и «отменен»;
- e-mail информирование: автоматическая отправка уведомлений на электронную почту клиента при обновлении статуса его заявки;
- ведение истории изменений: регистрация всех действий пользователей в журнале событий с указанием даты, времени и автора изменений;

- управление справочниками: ведение реестров клиентов и сотрудников с возможностью их добавления, редактирования и удаления;
- мониторинг исполнения: автоматическое определение просроченных заказов и их визуальное выделение в интерфейсе (цветовая индикация);
- поиск и фильтрация: многокритериальный поиск по базе заказов (по номеру, клиенту, сотруднику, датам и текущему статусу);
- формирование и экспорт отчетности: создание сводных отчетов по ключевым показателям деятельности с возможностью выгрузки данных в формат CSV.

Диаграмма вариантов использования, составленная на основе выделенных требований, изображена на рисунке 1.

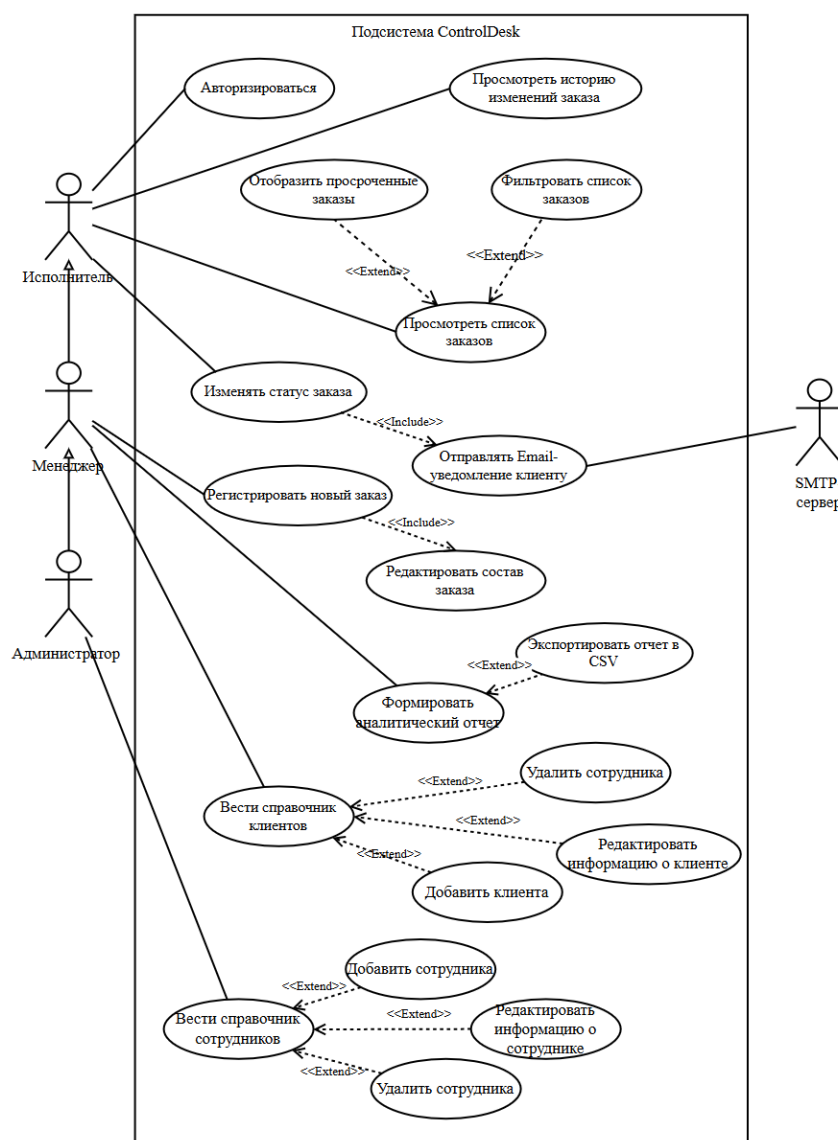


Рисунок 1 – Диаграмма вариантов использования подсистемы

Ниже приведено описание наиболее важных вариантов использования, связанных с основным функционалом подсистемы. Спецификация варианта использования «Регистрировать новый заказ» приведена на рисунке 2.

Прецедент: Регистрировать новый заказ
ID: 1
Краткое описание: Менеджер создает в системе новую карточку заказа, указывая параметры и состав работ
Главные актеры: Менеджер, Администратор
Второстепенные актеры: нет
Предусловие: 1. Пользователь авторизован с правами не ниже Менеджера 2. Открыто окно создания заказа
Основной поток: 1. Пользователь выбирает клиента и ответственного сотрудника из справочников 2. Пользователь вводит описание, выбирает приоритет и плановую дату исполнения 3. Пользователь добавляет позиции заказа (наименование, цена, количество) 4. Пользователь нажимает кнопку «Сохранить»

Рисунок 2 – Спецификация варианта использования «Регистрировать новый заказ»

Спецификация варианта использования «Изменить статус заказа» приведена на рисунке 3.

Прецедент: Изменить статуса заказа
ID: 2
Краткое описание: Пользователь обновляет текущий этап обработки заказа, что сопровождается уведомлением клиента
Главные актеры: Исполнитель, Менеджер, Администратор
Второстепенные актеры: Почтовый сервер (SMTP)
Предусловие: 1. Пользователь авторизован 2. Выбран конкретный заказ из списка
Основной поток: 1. Пользователь выбирает новый статус из выпадающего списка 2. Пользователь вводит комментарий к изменениям (опционально) 3. Пользователь нажимает кнопку «Обновить статус» 4. Система сохраняет изменения в БД 5. Система в автоматическом режиме отправляет Email-уведомление клиенту через SMTP-сервер

Рисунок 3 – Спецификация варианта использования «Изменить статус заказа»

Спецификация варианта использования «Формировать аналитический отчет» приведена на рисунке 4.

Прецедент: Формировать аналитический отчет
ID: 3
Краткое описание: Пользователь формирует сводную таблицу данных по заданным критериям и экспортирует её в файл
Главные актеры: Менеджер, Администратор
Второстепенные актеры: нет
Предусловие: 1. Пользователь авторизован 2. Открыт раздел «Отчетность»
Основной поток: 1. Пользователь задает параметры фильтрации (период, клиент, статус) 2. Пользователь выбирает столбцы, необходимые для отображения в отчете 3. Система генерирует предварительный просмотр данных 4. Пользователь выбирает функцию «Экспорт в CSV»

Рисунок 4 – Спецификация варианта использования «Формировать аналитический отчет»

В ходе анализа функциональных возможностей подсистемы ControlDesk были определены ключевые сценарии взаимодействия пользователей с программным продуктом. К ним относятся процессы авторизации, управления полным циклом обработки заказов и ведения нормативно-справочной информации о клиентах и персонале. Для наиболее значимых функциональных блоков подготовлены детальные спецификации, описывающие последовательность действий пользователя и соответствующий алгоритм реакции системы».

1.4 Анализ нефункциональных требований

Помимо реализации основных бизнес-задач, разрабатываемая подсистема ControlDesk должна соответствовать ряду нефункциональных требований. Данные требования определяют качественные характеристики программного продукта, такие как удобство использования, надежность, производительность и информационная безопасность.

Требования к эргономике и интерфейсу (Usability). Исходя из анализа целевой аудитории, подсистема должна иметь интуитивно понятный

графический интерфейс, не требующий от пользователя специальной технической подготовки. Основной упор сделан на визуализацию данных. В частности, для повышения оперативности принятия решений необходимо реализовать механизмы цветовой индикации:

- разграничение заказов по их текущему состоянию (использование различных цветов для статусов «Новый», «В обработке» и т.д.);
- автоматическое выделение просроченных записей контрастным цветом: интерфейс должен строиться по принципу минимизации действий пользователя для доступа к основным функциям.

Требования к надежности (Reliability). Подсистема должна обеспечивать стабильную работу в условиях ежедневной эксплуатации. Ключевым аспектом надежности является сохранность и целостность данных. Для этого в системе предусмотрены следующие механизмы:

- автоматическая инициализация и проверка структуры базы данных при запуске приложения;
- валидация вводимых данных на уровне интерфейса и программного кода, что исключает попадание некорректной информации в базу данных;
- система должна корректно обрабатывать исключительные ситуации, такие как отсутствие связи с почтовым сервером или недоступность базы данных, информируя пользователя о возникшей ошибке без аварийного завершения работы.

Требования к производительности (Performance). Программный продукт ориентирован на работу в рамках локальной сети предприятия. Учитывая использование технологии Entity Framework 6 для взаимодействия с данными, к производительности предъявляются следующие требования:

- время отклика интерфейса на типовые запросы (фильтрация списка, открытие карточки заказа) не должно превышать 1–2 секунд;
- процесс генерации и экспорта аналитической отчетности при штатном объеме данных (до нескольких тысяч записей) должен занимать не более 5 секунд;

- минимальное потребление системных ресурсов, позволяющее запускать приложение на рабочих станциях со средними техническими характеристиками.

Требования к информационной безопасности (Security). Безопасность системы базируется на ролевой модели управления доступом и защите персональных данных сотрудников и клиентов. Основные требования включают:

- обязательную идентификацию и аутентификацию пользователей при входе;
- криптографическую защиту паролей – в системе не должны храниться пароли в открытом виде, для их защиты необходимо использовать алгоритм хэширования (например, SHA256);
- разграничение прав доступа на уровне функциональных модулей (например, ограничение доступа исполнителей к финансовой отчетности или управлению персоналом).

Технические требования к среде (Environmental Requirements). Подсистема проектируется как классическое Desktop-приложение. Для её функционирования необходимо наличие установленной платформы .NET Framework версии 4.8.1 или выше. Система должна поддерживать работу с локальными или сетевыми экземплярами SQL Server. Важным требованием является возможность автономной работы в локальной сети предприятия без обязательного доступа к глобальной сети Интернет (за исключением функции отправки Email-уведомлений).

Таким образом, соблюдение указанных требований позволит создать качественное программное обеспечение, соответствующее современным стандартам разработки и эксплуатации корпоративных информационных систем.

1.5 Выбор технологий проектирования системы

При проектировании информационной подсистемы были изучены и применены средства визуального моделирования, базирующиеся на унифицированном языке моделирования UML. Этот язык является

общепризнанным стандартом в области проектирования программного обеспечения и позволяет описать структуру системы и её поведение с помощью набора диаграмм [8].

В ходе работы рассматривались два инструмента: Draw.io (diagrams.net) и Visual Paradigm Community Edition. Draw.io – бесплатный онлайн-редактор диаграмм с открытым исходным кодом, доступный как в браузерной версии, так и в виде настольного приложения для Windows, macOS и Linux [2]. Инструмент поддерживает построение диаграмм UML, блок-схем, ER-диаграмм и других типов визуальных моделей. Интерфейс прост и не требует предварительного обучения: рабочая область с набором фигур на боковой панели и возможностью экспорта в форматы PNG, SVG и PDF делает его удобным для быстрого прототипирования. Вместе с тем Draw.io не осуществляет валидацию нотации UML и не предупреждает об ошибках в построенных моделях, что может быть существенным при разработке технически точной документации.

Visual Paradigm Community Edition – профессиональная CASE-среда, поддерживающая полный стандарт UML 2.x. Инструмент предоставляет широкий набор типов диаграмм, включая диаграммы вариантов использования, классов, последовательностей и деятельности. Одним из ключевых преимуществ является продуманный интерфейс с поддержкой drag-and-drop, возможность экспорта диаграмм в высоком разрешении, а также встроенная валидация нотации, что исключает ошибки при построении моделей. Бесплатная версия Community Edition предоставляет функциональность, достаточную для решения учебных и прикладных задач [9].

В итоге в качестве основного средства проектирования была выбрана среда Visual Paradigm Community Edition. Выбор обусловлен полноценной поддержкой стандарта UML, встроенной проверкой корректности моделей и качеством экспортируемых изображений, которые соответствуют требованиям к оформлению пояснительной записки.

1.6 Выбор инструментальных средств

Разработка информационных систем включает в себя использование набора взаимосвязанных технологий, таких как язык программирования, фреймворк для создания пользовательского интерфейса, система управления базами данных, а также инструменты для обработки и работы с данными на уровне программного кода. При выборе инструментальных средств для реализации подсистемы ControlDesk были приняты во внимание следующие факторы:

- доступность и распространённость используемых технологий;
- возможность создания полноценного настольного приложения с удобным интерфейсом;
- поддержка объектно-реляционного отображения для упрощения работы с базой данных;
- производительность и надёжность при работе с локальными данными;
- наличие качественной документации.

В качестве языка программирования был выбран C# - современный объектно-ориентированный язык платформы .NET. C# обладает строгой типизацией, развитыми средствами обобщённого программирования и активно поддерживается компанией Microsoft. Язык хорошо зарекомендовал себя при разработке корпоративных приложений, обеспечивает высокую читаемость кода и предоставляет богатую стандартную библиотеку [10].

Для построения пользовательского интерфейса выбран WPF (Windows Presentation Foundation) - фреймворк, входящий в состав .NET и предназначенный для создания настольных приложений под операционную систему Windows. WPF основан на декларативном языке разметки XAML, что позволяет чётко разделить логику приложения и представление. Фреймворк поддерживает привязку данных (data binding), шаблоны элементов управления,

стили и анимации. Это даёт возможность строить современные, масштабируемые интерфейсы без значительных затрат на разработку [11].

В качестве системы управления базами данных использован Microsoft SQL Server. Это реляционная СУБД корпоративного уровня, обеспечивающая надёжное хранение данных, поддержку транзакций, гибкие средства индексирования и развитые механизмы обеспечения безопасности. SQL Server хорошо масштабируется и подходит как для небольших локальных баз, так и для решений уровня предприятия [4].

Entity Framework – объектно-реляционный маппер (ORM) для платформы .NET. Entity Framework позволяет работать с данными через объектную модель, не прибегая к написанию SQL-запросов вручную. Связи между таблицами базы данных отражаются в виде навигационных свойств классов-сущностей, что упрощает сопровождение кода и повышает его читаемость. Подход Code First позволяет описывать структуру базы данных непосредственно в коде на C# с последующей автоматической генерацией схемы [12].

Таким образом, выбранный стек технологий представляет собой целостное и хорошо проработанное решение для разработки настольных информационных систем. Все компоненты интегрированы в рамках единой платформы Microsoft .NET, что обеспечивает согласованность инструментов и упрощает разработку и сопровождение приложения.

1.7 Планирование разработки и оценка бюджета

Успешная разработка проекта напрямую зависит от того, насколько грамотно выстроены рабочие процессы и рационально распределены имеющиеся ресурсы.

Процесс подготовки к созданию информационной подсистемы для регистрации, обработки и контроля исполнения заказов обязательно включает в себя этапы детального планирования и оценки необходимых затрат. С целью визуализации временных рамок проекта и определения стоимости всех работ была составлена диаграмма Ганта в программной среде ProjectLibre.

Весь цикл разработки был разделен на четыре последовательных блока:

- анализ предметной области и постановка задачи – подразумевает детальное изучение специфики обработки заказов, исследование существующих программ-аналогов и формирование перечня функциональных требований к будущей системе;
- проектирование информационной системы – включает в себя выбор архитектурных решений, моделирование структуры базы данных, а также проработку внешнего вида интерфейсов и внутренней логики работы приложения;
- реализация информационной системы – охватывает непосредственное программирование серверных и клиентских модулей, а также техническое создание и наполнение базы данных;
- контроль качества информационной системы – предполагает выбор наиболее эффективной методики тестирования и проведение комплексной проверки работоспособности всех функций системы.

В общей сложности план разработки содержит 15 индивидуальных задач. Датой начала работ определено 16 марта 2026 года, а завершение всего проекта намечено на 4 мая 2026 года. Подробный состав работ по каждому направлению и последовательность их выполнения отражены на рисунке 5.

Название	Начало	Окончание	Продолжительность
Начало проекта	16.03.26 8:00	16.03.26 8:00	0 дней
Анализ предметной области и постановка задачи	16.03.26 8:00	23.03.26 17:00	6 дней
Исследование предметной области	16.03.26 8:00	16.03.26 17:00	1 день
Анализ информационных систем аналогичного назначения	17.03.26 8:00	17.03.26 17:00	1 день
Анализ функциональных требований	18.03.26 8:00	19.03.26 17:00	2 дней
Анализ нефункциональных требований	20.03.26 8:00	20.03.26 17:00	1 день
Определение технических средств реализации	23.03.26 8:00	23.03.26 17:00	1 день
Проектирование информационной системы	24.03.26 8:00	07.04.26 13:00	10,5 дней
Определение архитектуры приложения	24.03.26 8:00	26.03.26 13:00	2,5 дней
Проектирование структур данных	26.03.26 13:00	30.03.26 13:00	2 дней
Проектирование программных средств	30.03.26 13:00	02.04.26 13:00	3 дней
Проектирование пользовательского интерфейса	02.04.26 13:00	07.04.26 13:00	3 дней
Реализация информационной системы	07.04.26 13:00	24.04.26 13:00	13 дней
Реализация базы данных	07.04.26 13:00	09.04.26 13:00	2 дней
Реализация модулей бизнес-логики	09.04.26 13:00	16.04.26 13:00	5 дней
Реализация интерфейса приложения	16.04.26 13:00	24.04.26 13:00	6 дней
Контроль качества информационной системы	24.04.26 13:00	04.05.26 13:00	6 дней
Выбор режима тестирования	24.04.26 13:00	27.04.26 13:00	1 день
Модульное тестирование внутренней логики	27.04.26 13:00	29.04.26 13:00	2 дней
Функциональное тестирование интерфейса	29.04.26 13:00	04.05.26 13:00	3 дней
Конец проекта	04.05.26 13:00	04.05.26 13:00	0 дней

Рисунок 5 – Описание этапов разработки системы

На рисунке 6 представлена диаграмма Ганта, на которой отображена последовательность выполнения всех задач и продолжительность каждого этапа.

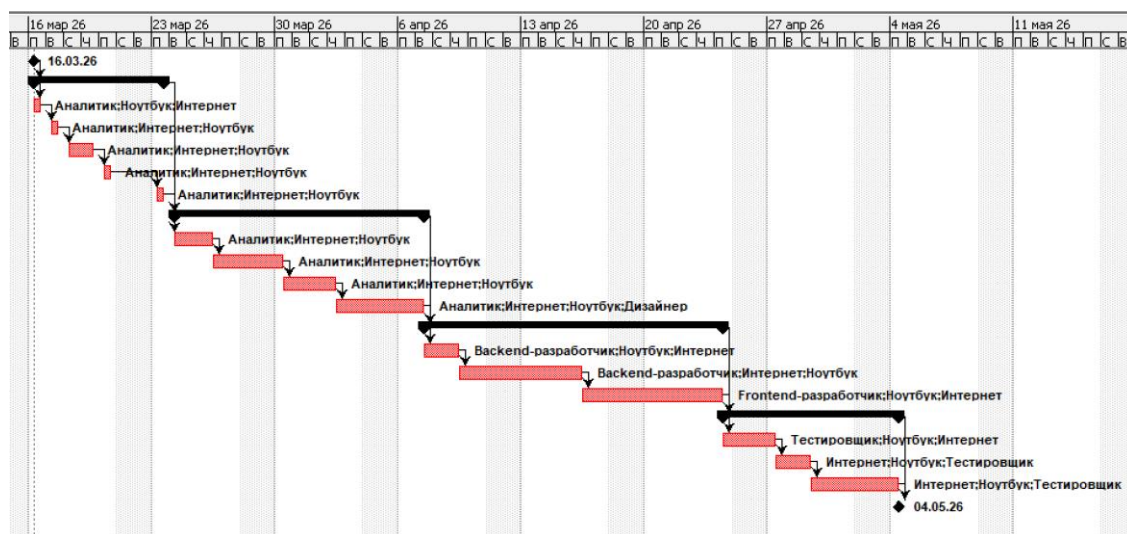


Рисунок 6 – Диаграмма Ганта проекта

Весь цикл разработки занял 36 рабочих дней. Этапы выстроены последовательно: переход к новому блоку работ возможен только после полного завершения всех задач предыдущего.

На рисунке 7 показана стоимость рабочего времени использованных в проекте ресурсов.

	🔧	Название	Стандартная ставка	Тип
1	🔧	Аналитик	450/час Работа	
2	🔧	Дизайнер	400/час Работа	
3	🔧	Backend-разработчик	500/час Работа	
4	🔧	Frontend-разработчик	450/час Работа	
5	🔧	Тестировщик	450/час Работа	
6	🔧	Ноутбук	0 Материал	
7	🔧	Интернет	0 Материал	

Рисунок 7 – Стоимость ресурсов

Итоговая стоимость проекта составила 130 000 рублей. Распределение затрат по этапам выглядит следующим образом:

- анализ предметной области и постановка задачи – 21 600 руб;
- проектирование информационной системы – 37 200 руб;
- реализация информационной системы – 49 600 руб;
- контроль качества информационной системы – 21 600 руб.

Сводные данные о затратах проекта с привязкой к продолжительности работ приведены на рисунке 8.

Название	Стоимость
Начало проекта	0 руб.
Анализ предметной области и постановка задачи	21600 руб.
Исследование предметной области	3600 руб.
Анализ информационных систем аналогичного назначения	3600 руб.
Анализ функциональных требований	7200 руб.
Анализ нефункциональных требований	3600 руб.
Определение технических средств реализации	3600 руб.
Проектирование информационной системы	37200 руб.
Определение архитектуры приложения	9000 руб.
Проектирование структур данных	7200 руб.
Проектирование программных средств	10800 руб.
Проектирование пользовательского интерфейса	10200 руб.
Реализация информационной системы	49600 руб.
Реализация базы данных	8000 руб.
Реализация модулей бизнес-логики	20000 руб.
Реализация интерфейса приложения	21600 руб.
Контроль качества информационной системы	21600 руб.
Выбор режима тестирования	3600 руб.
Модульное тестирование внутренней логики	7200 руб.
Функциональное тестирование интерфейса	10800 руб.
Конец проекта	0 руб.
	ИТОГО: 130000 руб.

Рисунок 8 – Общие затраты проекта

Выполненное планирование даёт возможность ещё до начала работ видеть полную структуру этапов и оптимально задействовать ресурсы.

2 Проектирование информационной подсистемы для регистрации обработки и контроля исполнения заказов

2.1 Проектирование архитектуры приложения

Ключевая задача этапа проектирования сводится к разработке архитектуры подсистемы ControlDesk, обеспечивающей модульность, надежность и простоту сопровождения программного продукта. С учетом требований к автономной работе в локальной сети и специфики настольных приложений, система строится на базе архитектуры «клиент-сервер».

В данной схеме клиентская часть представляет собой «толстый клиент» - полнофункциональное приложение, содержащее логику обработки данных и интерфейс пользователя. Серверная часть представлена СУБД Microsoft SQL Server, которая отвечает за централизованное хранение информации и обеспечение целостности данных. Диаграмма развертывания подсистемы представлена на рисунке 9.

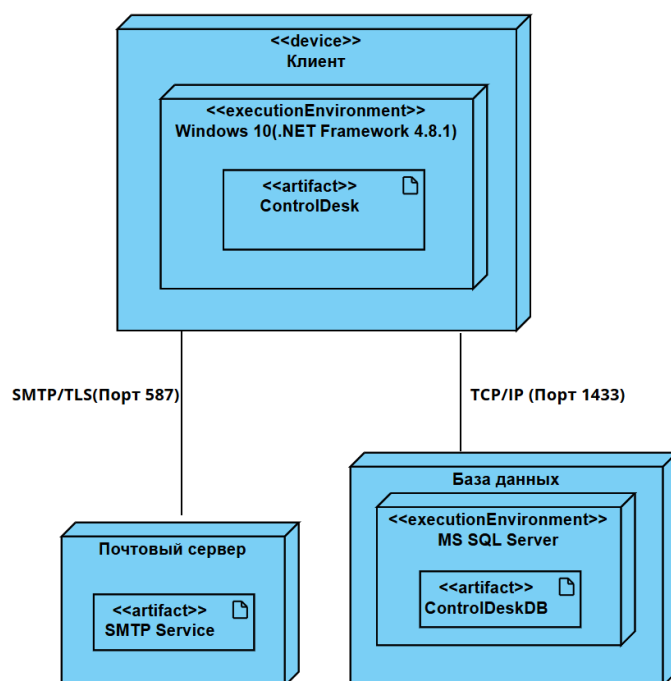


Рисунок 9 - Диаграмма развертывания подсистемы

Для реализации программного кода выбрана многослойная архитектура на базе паттерна MVVM (Model-View-ViewModel). Применение данного паттерна позволяет четко разделить ответственность между визуальным представлением

и бизнес-логикой системы. Архитектурное решение включает в себя следующие уровни:

- уровень представления (View): отвечает за визуализацию данных и взаимодействие с пользователем, на базе интуитивно понятных элементов управления, обеспечивающих доступ к функциям регистрации и мониторинга заказов;
- уровень логики представления (ViewModel): выступает посредником между интерфейсом и данными, на этом уровне сосредоточена логика обработки команд пользователя, а также механизмы уведомления интерфейса об изменениях;
- слой бизнес-сервисов: реализует бизнес-правила (например, авторизацию и отправку e-mail уведомлений);
- слой данных (Data Access Layer / Repository): включает описание структур сущностей (заказов, клиентов, сотрудников) и контекст взаимодействия с СУБД на базе технологии Entity Framework 6.

Взаимодействие между спроектированными уровнями представлено на рисунке 10.

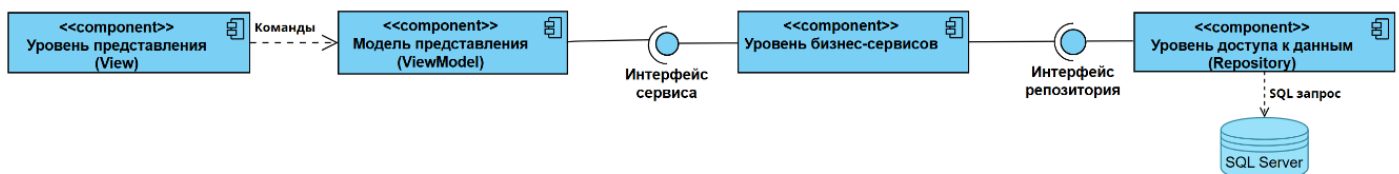


Рисунок 10 – Диаграмма компонентов подсистемы

Основные функциональные компоненты подсистемы:

- модуль авторизации: обеспечивает идентификацию пользователей с разграничением прав доступа (администратор, менеджер, исполнитель);
- модуль управления заказами: основной блок системы, отвечающий за создание, редактирование и проведение заявок по жизненному циклу (статусам);
- модуль ведения справочников: предоставляет инструменты для актуализации данных о контрагентах и кадровом составе организации;

– сервис оповещений: реализует автоматическую отправку уведомлений клиентам через внешний SMTP-сервер при изменении состояния их заказов;

– модуль истории: фиксирует все действия пользователей в системе для обеспечения прослеживаемости процесса выполнения работ.

Проектирование системы на базе указанных принципов обеспечивает необходимую гибкость: логика работы с данными отделена от интерфейса, что позволяет модифицировать функционал подсистемы без риска нарушения работы её визуальной части.

2.2 Проектирование структур данных

2.2.1 Концептуальная модель базы данных

Концептуальная модель представляет собой обобщённое отображение выбранной предметной сферы, фиксирующее состав ключевых сущностей, их характерные признаки и систему взаимосвязей. Данная модель строится без привязки к конкретным инструментам реализации баз данных и выступает в роли фундаментальной платформы для последующего построения архитектуры информационной системы.

Формирование концептуальной модели данных начинается с выделения основных сущностей, для рассматриваемой предметной области. Описание выделенных объектов приводится в таблице 2.

Таблица 2 – Сущности БД

Название сущности	Идентификатор сущности	Назначение сущности
Клиент	Client	Содержит сведения о контрагентах организации (наименование, контактные данные, адрес).
Сотрудник	Employee	Предназначена для хранения персональных и контактных данных персонала, а также сведений о должностях.

Продолжение таблицы 2

Название сущности	Идентификатор сущности	Назначение сущности
Учетная запись	User	Содержит данные для аутентификации пользователей в системе (логин, хэш пароля) и информацию о назначенной роли.
Заказ	Order	Центральная сущность системы, фиксирующая параметры заявки: номер, статус, приоритет и сроки исполнения.
Позиция заказа	OrderItem	Хранит детальный перечень товаров или услуг, включенных в конкретный заказ, с указанием их количества и цены.
История изменений	OrderHistory	Предназначена для ведения журнала аудита: фиксирует сведения о действиях пользователей и смене статусов заказов.

Графическое представление концептуальной модели базы данных представлено на рисунке 11.

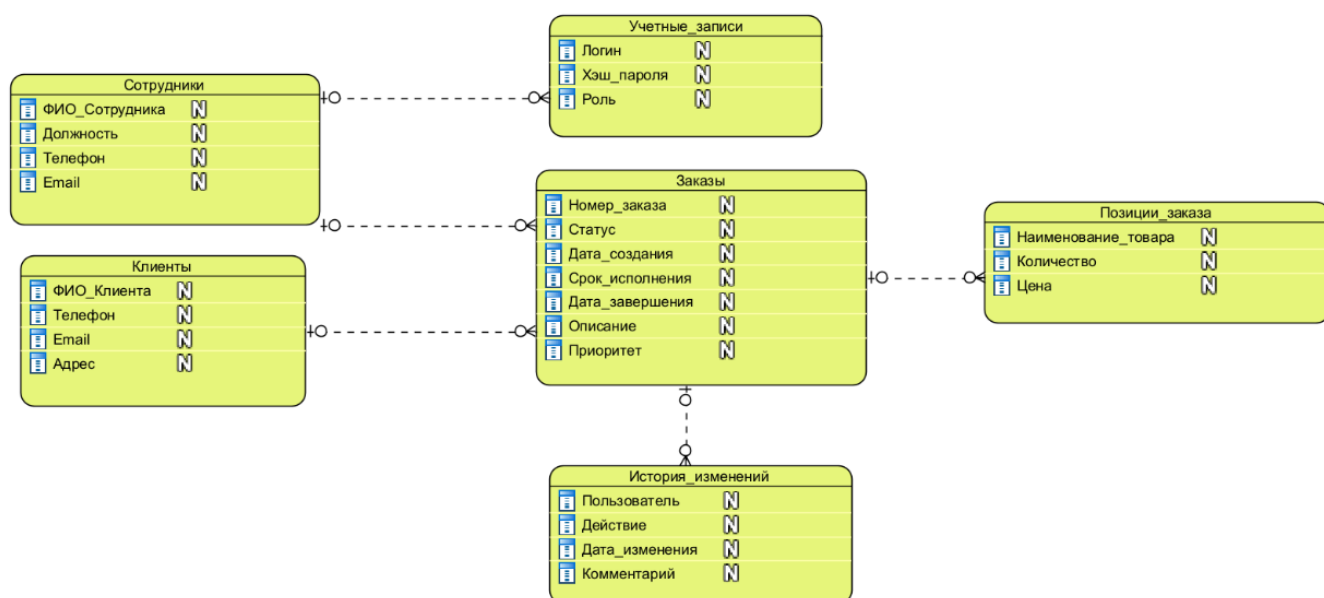


Рисунок 11 – Концептуальная модель базы данных подсистемы

Таким образом была разработана концептуальная модель разрабатываемой базы данных информационной подсистемы для регистрации, обработки и контроля исполнения заказов.

2.2.2 Логическая модель базы данных

Логическая модель данных представляет собой детализированное описание структуры базы данных в терминах сущностей, их атрибутов и связей между ними. В отличие от концептуального уровня, логическая модель учитывает специфику реляционного подхода. Она служит промежуточным звеном между абстрактным представлением предметной области и физической реализацией базы данных, обеспечивая переход от смыслового описания к формализованной структуре, которая необходима для последующего развертывания в конкретной СУБД.

Логическая модель проектируемой базы данных представлена на рисунке 11.

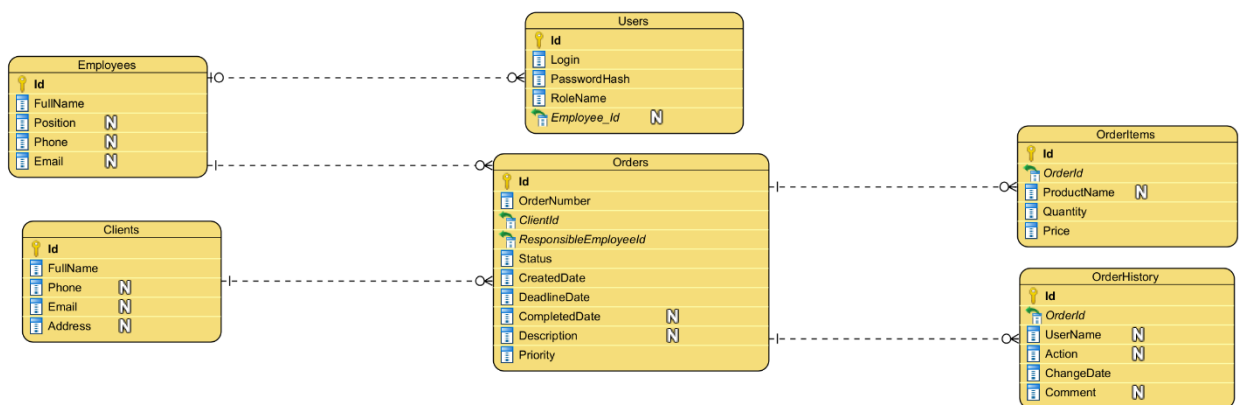


Рисунок 11 – Логическая модель базы данных системы

Таким образом, построенные модели в полной мере описывают структуру предметной области и служат основой для перехода к этапу физического проектирования. Составы атрибутов каждой сущности и их параметры приведены в приложении Д.

2.3 Проектирование программных средств

На основании разработанной архитектуры для каждого из программных модулей подсистемы ControlDesk были построены диаграммы классов, описывающие внутреннюю структуру системы и связи между объектами.

Программная реализация выполняется на языке C# (.NET Framework 4.8.1) с использованием технологии WPF и многослойного разделения ответственности

в рамках паттерна MVVM. Проектируемое решение включает в себя следующие уровни программных компонентов:

- уровень представлений (Presentation Layer): отвечает за визуальное отображение графического интерфейса пользователя и получение входных данных через систему окон, разработанных на языке разметки XAML;
- уровень моделей представления (ViewModel Layer): обеспечивает логическую связь между интерфейсом и данными, управляя состоянием визуальных компонентов и обрабатывая пользовательские действия через механизм команд;
- уровень команд (Command Layer): реализует логику вызова функций системы из интерфейса, инкапсулируя алгоритмы обработки событий в независимые классы для обеспечения чистоты программного кода;
- уровень бизнес-сервисов (Service Layer): содержит реализацию ключевых прикладных задач, таких как процедуры авторизации пользователей, механизмы отправки почтовых уведомлений и логирование системных процессов;
- уровень репозитория (Repository Layer): предоставляет абстрактный интерфейс для управления данными, позволяя основной программе выполнять операции с базой заказов независимо от конкретной реализации хранилища;
- уровень объектных моделей (Model Layer): определяет структуру сущностей предметной области, включая описание всех атрибутов заказов, контрагентов и сотрудников организации;
- уровень доступа к данным (Data Layer): обеспечивает физическое взаимодействие с СУБД на базе технологии Entity Framework 6, отвечая за объектно-реляционное отображение и инициализацию структуры базы данных;
- уровень преобразователей (Converter Layer): реализует вспомогательную логику трансформации программных состояний в визуальные свойства интерфейса, обеспечивая динамическую цветовую индикацию статусов и сроков исполнения.

Краткое описание компонентов уровня представлений, обеспечивающих графический интерфейс системы, приведено в таблице 3.

Таблица 3 – Компоненты уровня представлений (Views)

Название класса	Назначение
LoginWindow	Окно первичной авторизации пользователя для доступа к системе
MainWindow	Главное окно приложения, содержащее основной интерфейс управления заказами
AddOrderWindow	Форма для ввода данных при регистрации или редактировании заказа
ClientWindow	Интерфейс для работы со справочником контрагентов (клиентов)
EmployeeWindow	Интерфейс для управления реестром сотрудников организации
OrderItemsWindow	Вспомогательное окно для просмотра списка позиций конкретного заказа
OrderItemWindow	Форма для добавления или корректировки характеристик отдельной позиции заказа

Для обеспечения независимости логики от способа хранения данных используются компоненты уровня репозитория (Таблица 4)

Таблица 4 – Компоненты уровня репозитория (Repositories)

Название класса	Назначение
IOrderRepository	Определяет единый набор методов для доступа к заказам и справочникам
EfOrderRepository	Реализует хранение и извлечение данных с использованием Entity Framework
InMemoryOrderRepository	Обеспечивает работу с временным хранилищем данных в оперативной памяти
RepositoryFactory	Управляет созданием нужного типа репозитория в зависимости от настроек системы

Механизмы обработки пользовательских действий и визуальной трансформации данных описаны в таблице 5.

Таблица 5 – Компоненты уровней команд и преобразователей (Commands)

Название класса	Назначение
RelayCommand	Универсальная реализация интерфейса ICommand для привязки действий к кнопкам
StatusBrushConverter	Преобразует текстовое значение статуса заказа в соответствующий цветовой маркер
OverdueBrushConverter	Выполняет динамическую смену цвета шрифта для визуализации просроченных заказов

Логика управления состоянием интерфейса и связь с данными реализованы в компонентах, представленных в таблице 6.

Таблица 6 – Компоненты уровня моделей представления (ViewModels)

Название класса	Назначение
MainViewModel	Обеспечивает связь между данными и главным окном, управляет навигацией и списками
ObservableObject	Базовый класс, реализующий интерфейс INotifyPropertyChanged для обновления интерфейса

Реализация прикладных бизнес-задач сосредоточена в слое сервисов, который представлен в таблице 7

Таблица 7 – Компоненты уровня бизнес-сервисов (Services)

Название класса	Назначение
AuthService	Выполняет аутентификацию пользователей и обеспечивает безопасность паролей
EmailService	Реализует отправку автоматических уведомлений клиентам через SMTP-протокол
ConsoleLogService	Осуществляет регистрацию системных событий и ошибок в консоль отладки

Структура данных представлены в таблице 8.

Таблица 8 – Компоненты уровня объектных моделей (Models)

Название класса	Назначение
Order	Модель, содержащая полные сведения о заказе и его текущем состоянии

Продолжение таблицы 8

Название класса	Назначение
AuthService	Выполняет аутентификацию пользователей и обеспечивает безопасность паролей
EmailService	Реализует отправку автоматических уведомлений клиентам через SMTP-протокол
ConsoleLogService	Осуществляет регистрацию системных событий и ошибок в консоль отладки
Client	Описывает атрибуты контрагента (наименование, контакты, адрес)
Employee	Описывает персональные и профессиональные данные сотрудника
OrderItem	Определяет структуру позиции (товара или услуги) внутри заказа

В таблице 9 представлены правила взаимодействия с базой данных на физическом уровне

Таблица 9 – Компоненты уровня объектных моделей (Models)

Название класса	Назначение
ControlDeskDbContext	Основной класс контекста данных, обеспечивающий связь моделей с таблицами СУБД
ControlDeskInitializer	Реализует механизм первичного наполнения базы данных демонстрационными данными
Configuration	Содержит настройки миграций для автоматического обновления структуры БД

Однако статическое описание классов не даёт полного представления о том, как именно эти компоненты взаимодействуют во время выполнения бизнес-логики. Для этого применяется динамическое моделирование.

Диаграмма последовательности языка UML используется для моделирования динамических аспектов работы системы – она показывает, как программные объекты обмениваются сообщениями и вызовами в хронологическом порядке при выполнении того или иного бизнес-сценария.

Данный инструмент позволяет наглядно представить алгоритм взаимодействия компонентов, выявить последовательность операций и распределение ответственности между уровнями архитектуры.

На рисунке 12 представлена диаграмма последовательности для сценария изменения статуса заказа с автоматической отправкой уведомления клиенту.

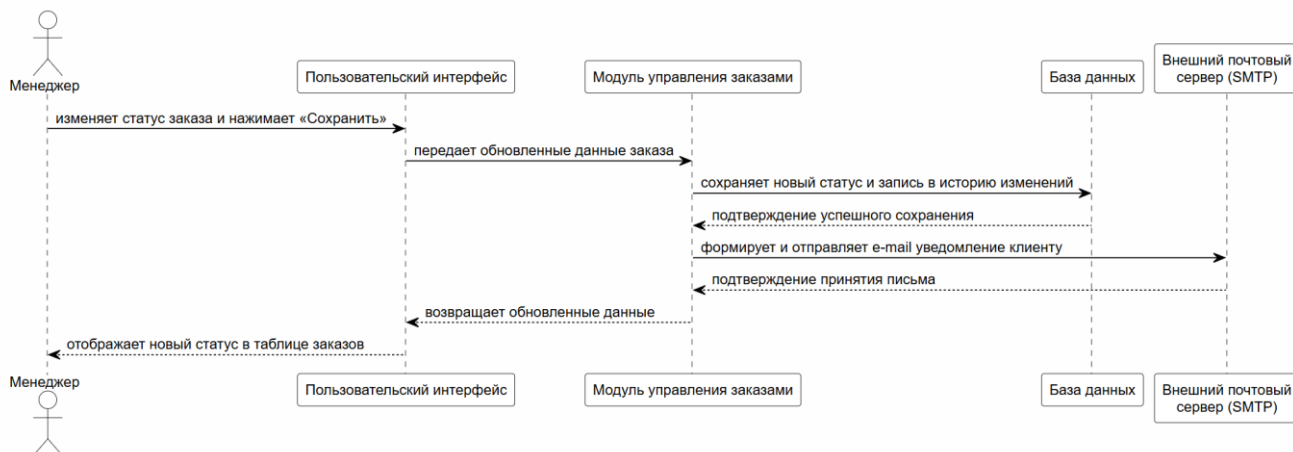


Рисунок 12 – Диаграмма последовательности

Спроектированная архитектура подсистемы ControlDesk охватывает все необходимые уровни программной реализации — от интерфейса до физического доступа к данным. Использование паттерна MVVM, репозиториев, сервисов и механизмов миграций обеспечивает гибкость и поддерживаемость системы. Диаграмма последовательности подтверждает корректность динамического взаимодействия компонентов в рамках ключевого бизнес-сценария. В целом, предложенное решение является модульным, масштабируемым и готовым к дальнейшей разработке.

2.4 Проектирование пользовательского интерфейса

Пользовательский интерфейс разрабатываемой подсистемы управления заказами ControlDesk представляет собой настольное приложение, функционирующее в среде ОС Windows. Интерфейс создан с использованием технологий XAML и WPF, что обеспечивает высокую производительность графической подсистемы, поддержку современных визуальных стилей и отзывчивость элементов управления при работе с большими массивами данных.

Целью проектирования интерфейса является создание простой, понятной и эргономичной среды взаимодействия для всех категорий сотрудников: администраторов, менеджеров и исполнителей. Интерфейс спроектирован по принципу «Dashboard» (информационная панель), что позволяет пользователю без специальной подготовки быстро оценить текущую ситуацию по заказам и приступить к выполнению служебных обязанностей.

В системе предусмотрены следующие основные окна и элементы интерфейса:

- а) окно авторизации – предназначено для идентификации пользователя и ограничения доступа к функциям системы в зависимости от его роли (администратор, менеджер, исполнитель);
- б) главная панель мониторинга – содержит сводные карточки показателей (общее количество заказов, активные, просроченные и общая сумма), позволяющие мгновенно оценить объем работы;
- в) реестр заказов – центральный элемент интерфейса, представляющий собой таблицу заказов с интерактивными инструментами:
 - 1) визуальная индикация статусов – использование цветовых маркеров (зеленый, оранжевый, красный) для быстрого считывания состояния заявок;
 - 2) система фильтрации и поиска – возможность отбора заказов по тексту, статусу или признаку просрочки;
- г) карточка заказа и управление составом:
 - 1) основные данные – редактирование параметров заявки, выбор клиента и ответственного сотрудника;
 - 2) спецификация (позиции заказа) – отдельный интерфейс для управления перечнем товаров или услуг в рамках конкретного заказа с расчетом итоговой стоимости;
- д) управление справочниками:
 - 1) для менеджеров и администраторов – доступ к справочнику клиентов для ведения базы контрагентов;

2) только для администратора – управление списком сотрудников и их должностными полномочиями;

е) аналитическая отчетность – интерактивный конструктор, позволяющий выбирать параметры фильтрации, настраивать состав колонок и выполнять экспорт сформированных данных во внешний формат CSV.

Каждый элемент интерфейса взаимодействует с программной логикой системы на основе паттерна MVVM с использованием механизма привязки данных (Data Binding). В отличие от веб-приложений, запросы обрабатываются внутри приложения через слой репозитория, взаимодействующий с базой данных SQL Server. Все изменения объектов в памяти мгновенно отражаются в визуальных компонентах интерфейса.

В интерфейсе реализована развитая система обратной связи и обработки исключений:

- при попытке сохранения некорректных или неполных данных (например, пустых обязательных полей) система блокирует действие и выводит уведомление;

- реализована цветовая индикация критических состояний;

- предусмотрены подтверждающие сообщения при успешном выполнении ключевых операций (регистрация заказа, удаление или экспорт отчета).

Навигация по системе реализована через боковую панель, элементы которой размещены логично и последовательно, что минимизирует количество переключений между окнами и упрощает повседневное взаимодействие с подсистемой.

3 Реализация информационной подсистемы для регистрации, обработки и контроля исполнения заказов

3.1 Реализация базы данных

По результатам концептуального и логического проектирования разработана физическая модель базы данных, представленная на рисунке 12. Физическая модель –это схема, регламентирующая конкретную реализацию базы данных в среде выбранной СУБД. Она включает описание всех таблиц, столбцов, первичных и внешних ключей, а также механизмов индексации и способов хранения данных на дисковых носителях. Данный этап является завершающим в процессе перехода от абстрактного описания предметной области к работоспособной базе данных [13].

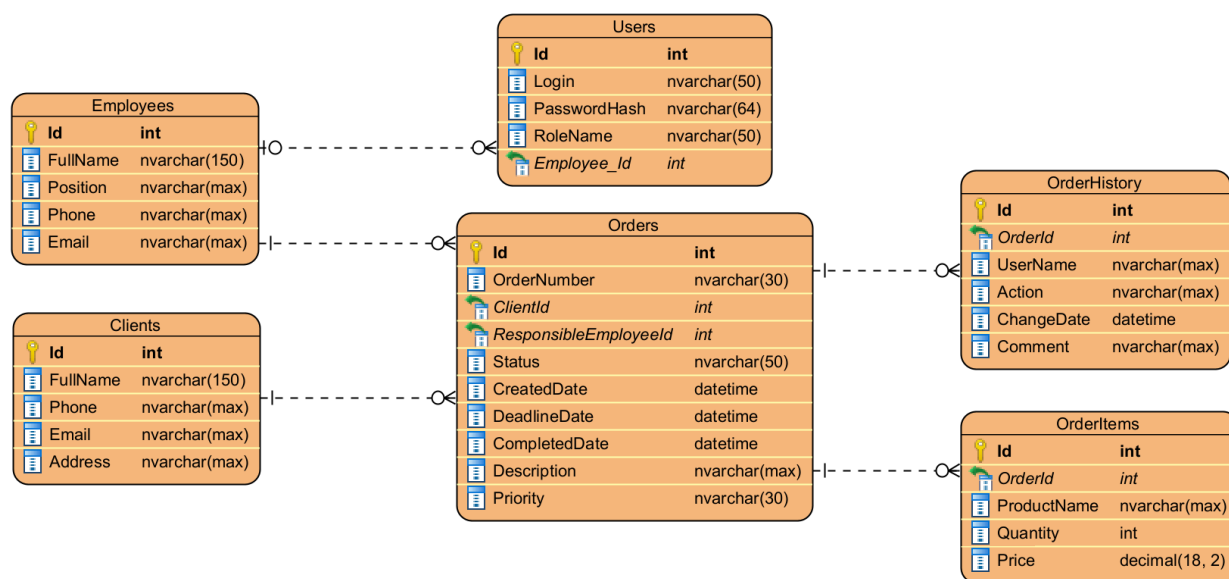


Рисунок 13 – Физическая модель базы данных информационной подсистемы для регистрации, обработки и контроля исполнения заказов

Согласно физической модели с помощью СУБД MS SQL Server была развернута база данных информационной подсистемы для регистрации, обработки и контроля исполнения заказов – ControlDeskDB. В процессе развертывания были созданы таблицы, соответствующие логической схеме и настроены первичные и внешние ключи для обеспечения целостности данных. Диаграмма развернутой базы данных представлена на рисунке 13.

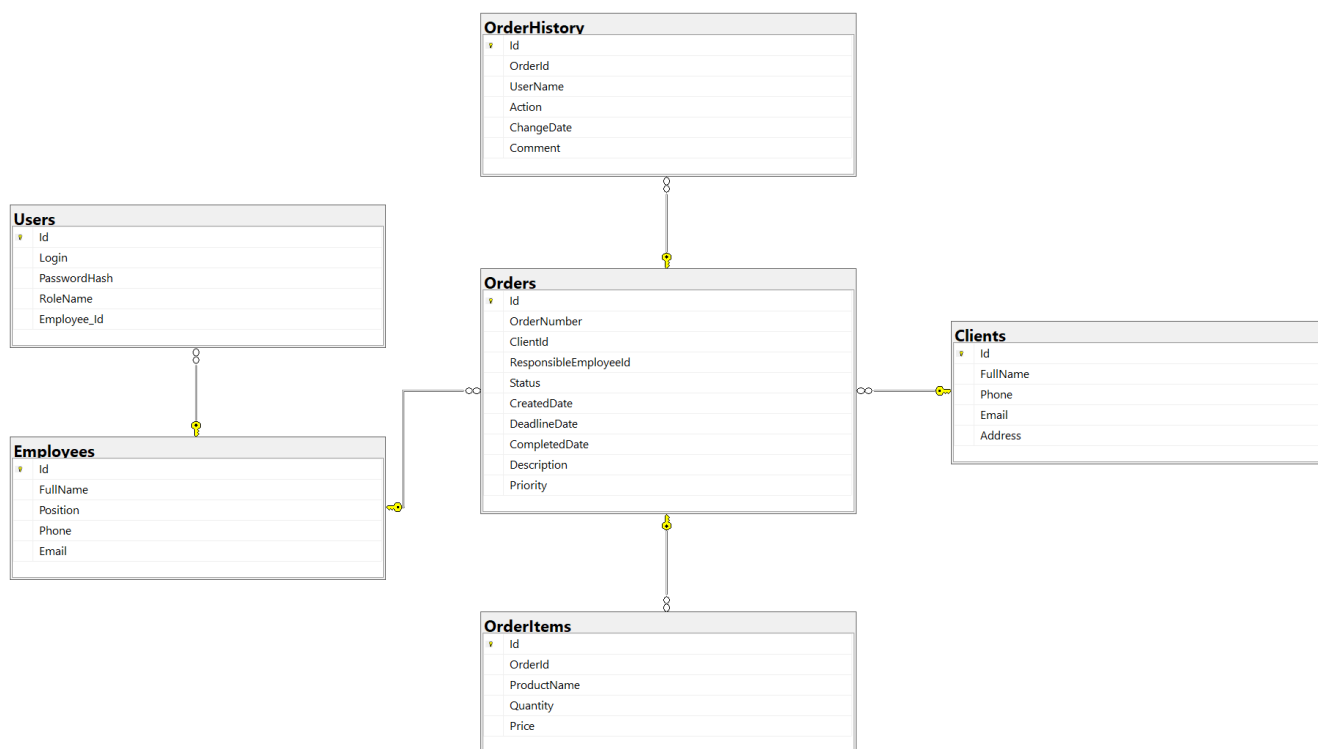


Рисунок 14 – Диаграмма базы данных информационной подсистемы для регистрации, обработки и контроля исполнения заказов

Таким образом, была реализована база данных ControlDeskDB для информационной системы по работе с заказами. Она обеспечивает хранение всей необходимой информации о клиентах, номенклатуре, статусах заказов и ответственных исполнителях, а также поддерживает обработку и контроль выполнения заявок на всех этапах. Структура базы данных позволяет эффективно вести учет и оперативно получать актуальные данные для управления процессами.

3.2 Программная реализация функциональных модулей и механизмов управления данными

Программная реализация логики обработки данных и взаимодействия с базой данных подсистемы ControlDesk выполнена в среде разработки Visual Studio на языке программирования C# с использованием платформы .NET Framework 4.8.1. Для работы с данными применена технология объектно-реляционного отображения (ORM) Entity Framework 6.

Архитектура внутренней части приложения построена по многослойному принципу, что обеспечивает независимость бизнес-логики от механизмов хранения. Основные компоненты этого уровня включают:

- модели данных, определяющие структуру таблиц БД (заказы, клиенты, сотрудники);
- контекст данных (DbContext), отвечающий за связь программных объектов с базой данных SQL Server;
- репозитории, реализующие методы доступа к данным и скрывающие детали SQL-запросов;
- сервисы, инкапсулирующие специфические бизнес-задачи (авторизация, рассылка уведомлений).

Взаимодействие между компонентами организовано через интерфейсы, что позволяет легко подменять реализацию (например, использовать InMemoryOrderRepository для тестирования вместо EfOrderRepository).

Основные методы управления данными, реализованные в модуле репозитория заказов, представлены в таблице 9.

Таблица 9 – Компоненты уровня объектных моделей (Models)

Название метода	Тип операции	Описание
GetOrders()	Чтение	Получение полного списка заказов с загрузкой связанных данных (клиентов, исполнителей)
SaveOrder(Order)	Запись/Обновление	Сохранение нового заказа или обновление существующей записи в базе данных
DeleteOrder(int)	Удаление	Физическое или логическое удаление записи о заказе по его идентификатору
GetClients()	Чтение	Извлечение списка всех контрагентов из справочника клиентов
SaveClient(Client)	Запись	Регистрация нового клиента или изменение данных существующего

Для обеспечения безопасности и информирования пользователей реализованы специализированные сервисы. Описание методов сервиса авторизации и уведомлений приведено в таблице 10.

Таблица 10 – Компоненты уровня объектных моделей (Models)

Название метода	Класс сервиса	Описание
Authenticate()	AuthService	Проверка пары логин/пароль и получение объекта пользователя с его ролью
HashPassword()	AuthService	Криптографическое преобразование пароля по алгоритму SHA256
SendStatusChanged()	EmailService	Формирование и отправка SMTP-сообщения клиенту при обновлении статуса заказа

В рамках раздела определена модульная структура подсистемы ControlDesk с распределением функций между компонентами. Для доступа к данным применены паттерны «Репозиторий» и «Фабрика» совместно с Entity Framework 6. В состав подсистемы включены сервис защиты информации и модуль автоматического оповещения. На основе выполненных решений возможен переход к разработке пользовательского интерфейса.

3.3 Программная реализация интерфейса и логики представления подсистемы

Визуальная часть ControlDesk выполняет две основные функции: отображение данных пользователю и передачу его команд в бизнес-логику. При обновлении данных интерфейс перестраивается, а действия пользователя преобразуются в вызовы соответствующих методов программного слоя, отвечающего за обработку логики. Архитектура этого уровня спроектирована с использованием паттерна MVVM (Model-View-ViewModel), что обеспечивает четкое разделение между визуальным описанием элементов и программным кодом обработки данных.

Для представления общей структуры интерфейсного слоя была разработана диаграмма компонентов (рисунок X), которая иллюстрирует взаимосвязь между XAML-представлениями, моделями представлений и

ресурсами стилизации. В отличие от веб-ориентированных систем, модульность здесь достигается за счет разделения функционала на независимые окна (Windows) и соответствующие им классы логики.

В таблице 11 приведено описание основных компонентов, формирующих пользовательский интерфейс подсистемы ControlDesk.

Таблица 11 – Описание программных компонентов интерфейсного слоя

Компонент	Назначение
App.xaml	Главный файл приложения, определяющий глобальные ресурсы и стили оформления
MainWindow.xaml	Основная форма интерфейса с Dashboard-панелью и списком заказов
MainViewModel.cs	Центральный узел логики представления: обработка команд, фильтрация и навигация
LoginWindow.xaml	Форма аутентификации пользователей с валидацией учетных данных
AddOrderWindow.xaml	Диалоговый интерфейс для ввода и корректировки параметров заказа
ClientWindow.xaml	Экранная форма для управления справочником контрагентов
EmployeeWindow.xaml	Интерфейс ведения реестра сотрудников организации
OrderItemsWindow.xaml	Окно просмотра и управления перечнем позиций (товаров) в заказе
OrderItemWindow.xaml	Форма редактирования характеристик конкретной позиции спецификации
StatusBrushConverter.cs	Логический компонент для цветовой индикации текущих статусов заказов
OverdueBrushConverter.cs	Компонент автоматической подсветки просроченных дат исполнения
RelayCommand.cs	Базовый механизм реализации команд для связи кнопок с логикой C#

Интерфейсный слой подсистемы реализован на языке декларативной разметки XAML и языке программирования C# в среде разработки Visual Studio. Взаимодействие между элементами управления и данными осуществляется через механизм «Data Binding» (привязка данных).

Ключевой особенностью реализации является использование динамических ресурсов, что позволяет системе мгновенно реагировать на изменения в базе данных. Например, при смене статуса заказа в модели данных, интерфейс автоматически перекрашивает соответствующий индикатор без принудительной перезагрузки окна. Фрагменты программного кода, описывающие логику работы представлений и конвертеров, приведены в приложении В.

3.4 Реализация интерфейса информационной подсистемы для регистрации, обработки и контроля исполнения заказов

В соответствии с требованиями к пользовательскому интерфейсу и разработанной архитектурой, в рамках подсистемы ControlDesk были реализованы следующие графические формы:

- окно аутентификации пользователей (LoginWindow);
- главное окно управления и мониторинга (MainWindow);
- окно регистрации и редактирования заказов (AddOrderWindow);
- окно управления спецификацией заказа (OrderItemsWindow);
- форма редактирования отдельной позиции заказа (OrderItemWindow);
- окно ведения справочника клиентов (ClientWindow);
- окно управления реестром сотрудников (EmployeeWindow).

Работа с подсистемой начинается с окна авторизации, которое представлено на рисунке 22. Данный интерфейс обеспечивает защиту данных и разграничение прав доступа. Пользователь должен ввести уникальный логин и пароль, которые проходят проверку в базе данных с использованием алгоритма хэширования.

Элементы, которые доступны с окна авторизации:

- поле для ввода пароля;
- поле для ввода логина;
- кнопки для входа и выхода из системы.

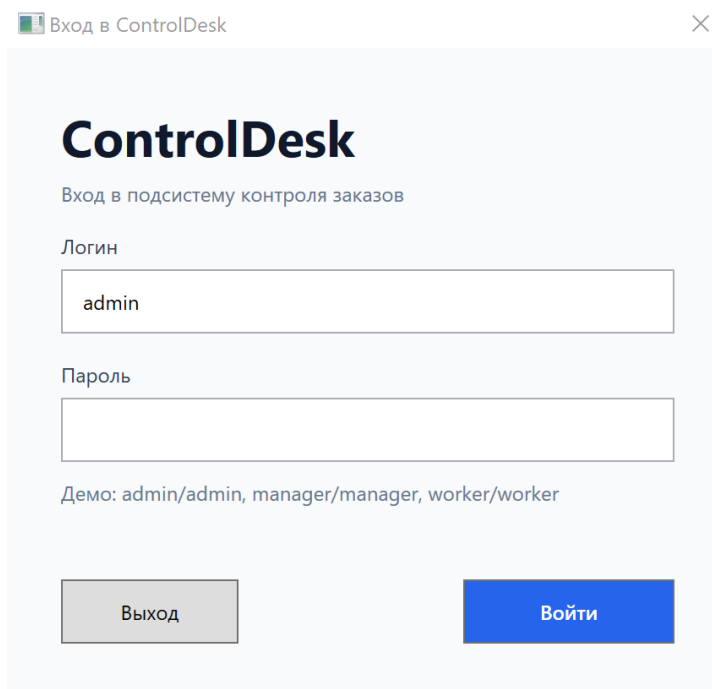


Рисунок 15 – Окно авторизации

После успешного прохождения процедуры аутентификации открывается главное окно приложения, представленное на рисунке 23. Оно спроектировано в виде интерактивной панели управления (Dashboard).

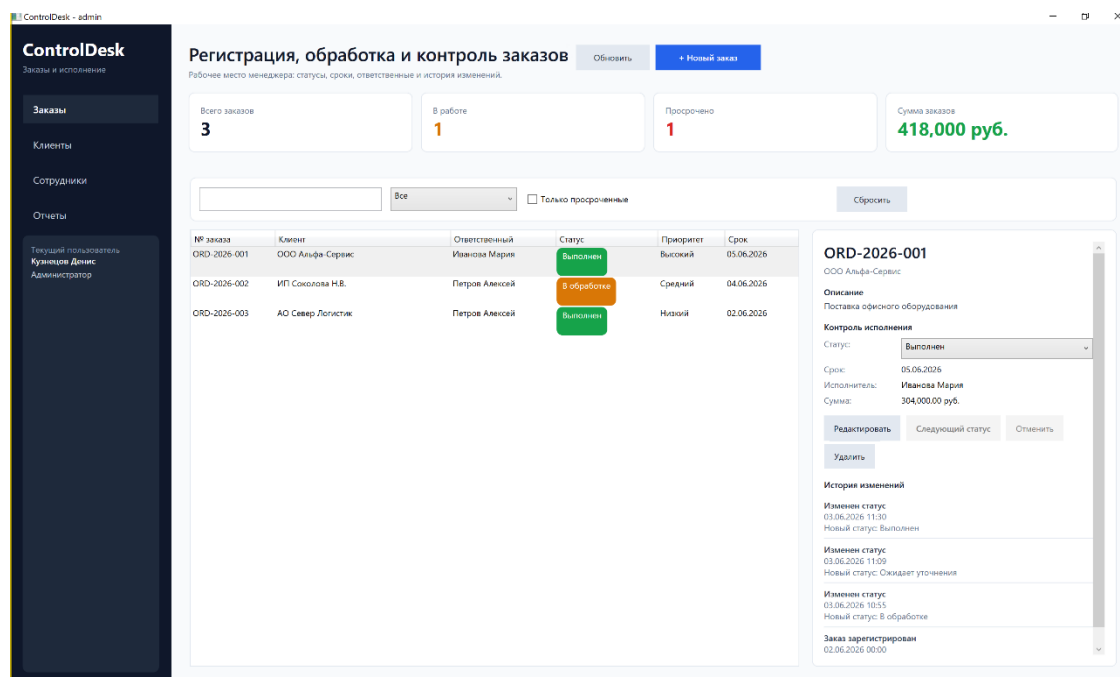


Рисунок 16 – Главное окно подсистемы

Функциональные зоны окна:

— навигационная панель: в левой части окна расположено вертикальное меню для переключения между модулями системы («Заказы», «Клиенты», «Сотрудники», «Отчеты»), в нижней части навигационной панели

реализован блок текущего пользователя, в котором отображаются ФИО («Кузнецов Денис») и роль («Администратор»), что позволяет визуально подтвердить уровень прав доступа к функциям;

- панель оперативных показателей: в верхней части рабочей области размещены информационные карточки, отображающие актуальную статистику в реальном времени.

- виджеты «Всего заказов», «В работе», «Просрочено» и «Сумма заказов» позволяют сотруднику мгновенно оценить общую нагрузку и финансовые показатели, особое внимание уделено цветовой индикации: количество просроченных заказов выделено красным цветом, а общая сумма зеленым, что облегчает визуальный контроль критических параметров;

- инструменты фильтрации и поиска: под блоком статистики расположена панель фильтров, включающая текстовое поле для быстрого поиска, по ключевым словам, выпадающий список выбора статуса и чекбокс «Только просроченные», кнопка «Сбросить» позволяет мгновенно вернуть таблицу в исходное состояние;

- реестр заказов: центральную часть занимает таблица (DataGrid), содержащая основные сведения: номер заказа, наименование контрагента, ответственного сотрудника, приоритет и срок исполнения.;

- статусы заказов («Выполнен», «В обработке») представлены в виде цветных «бейджей», что упрощает ориентирование в большом массиве данных;

- панель детальной информации и управления: справа расположена панель Master-Detail, отображающая полные сведения о выбранном в таблице заказе. В данном блоке доступны следующие элементы:

- описание задачи: подробный текст содержания заказа;
- блок контроля исполнения: интерфейс смены статуса через выпадающий список, отображение итоговой суммы и ответственного лица;
- кнопки управления: функциональные клавиши «Редактировать», «Следующий статус», «Отменить» и «Удалить», позволяющие оперативно управлять жизненным циклом заявки;

– журнал истории: нижний блок панели фиксирует хронологию изменений, в нем отображаются записи о регистрации заказа и всех последующих сменах статусов с указанием точной даты и времени операции.

В этом окне пользователю также доступны кнопки, позволяющие добавить новый заказ или обновить панель, при необходимости.

При помощи левого меню пользователь может свободно перемещаться по окнам приложения.

Раздел «Сотрудники» является важным элементом системы, так как именно на основе этих данных осуществляется назначение ответственных лиц за исполнение заказов и распределение ролей доступа.

На рисунке 23 представлен интерфейс модуля управления персоналом.

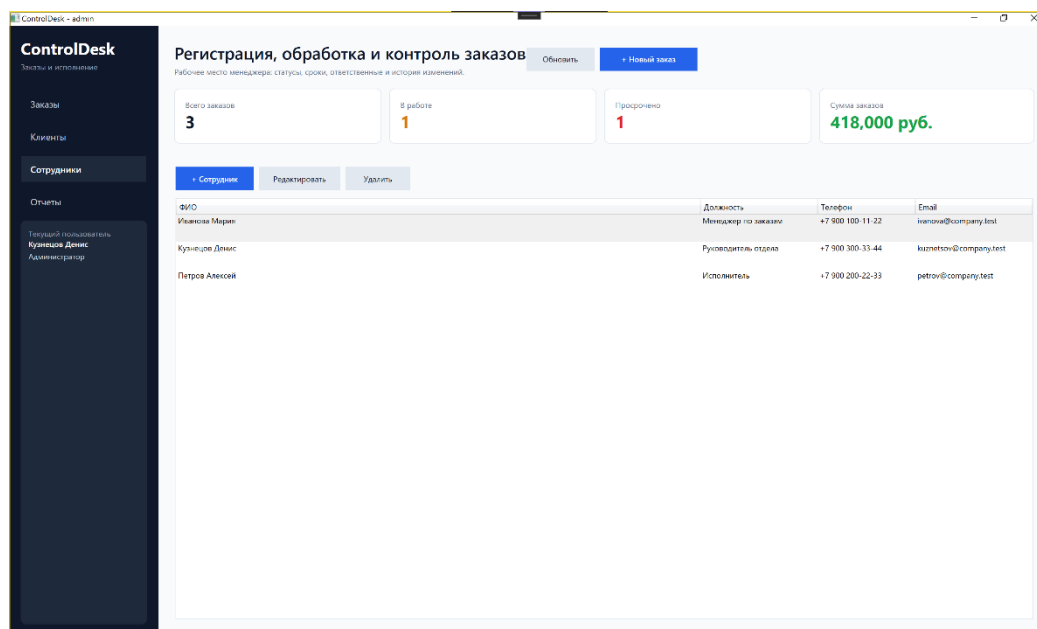


Рисунок 17 – Окно сотрудники подсистемы

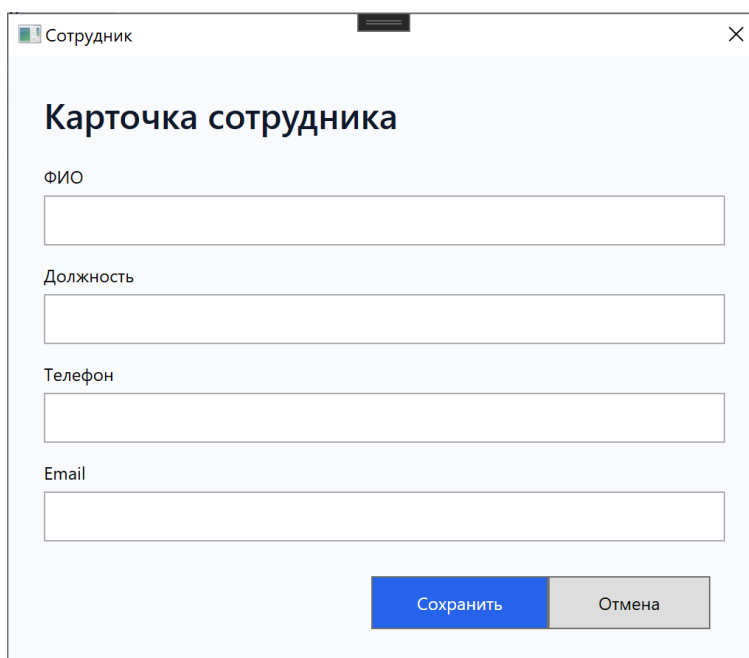
Визуальное оформление данного раздела полностью соответствует общей дизайн-концепции подсистемы ControlDesk. В верхней части окна сохраняется блок виджетов с ключевыми показателями, что позволяет администратору контролировать общую ситуацию по заказам в процессе работы с кадровым составом.

Функциональные особенности модуля «Сотрудники» включают:

– реестр персонала: таблица, содержащая ФИО сотрудников, их текущие должности и контактные данные (телефон, e-mail);

- панель инструментов: кнопки «+ Сотрудник», «Редактировать» и «Удалить» для оперативного изменения кадрового состава;
- интеграция с ролевой моделью: сопоставление каждого сотрудника с определенным уровнем полномочий (Администратор, Менеджер или Исполнитель);
- контроль целостности: программный запрет на удаление сотрудников, за которыми в базе данных закреплены активные или завершённые заказы;

При попытке добавления нового сотрудника открывается окно, отображенное на рисунке 24



Сотрудник

Карточка сотрудника

ФИО

Должность

Телефон

Email

Сохранить Отмена

Рисунок 18 – Окно добавления сотрудника подсистемы

Раздел «Клиенты» является фундаментальным элементом информационной базы подсистемы, так как содержит сведения о контрагентах, необходимые для регистрации и классификации заказов. Интерфейс данного раздела отображен на рисунке 25. Он спроектирован с соблюдением принципов преемственности дизайна, что обеспечивает удобство навигации при переключении между оперативной работой и ведением справочников.

В верхней части окна сохраняются информационные виджеты, со статистикой по заказам, в то время как центральная область полностью отведена под управление реестром клиентов.

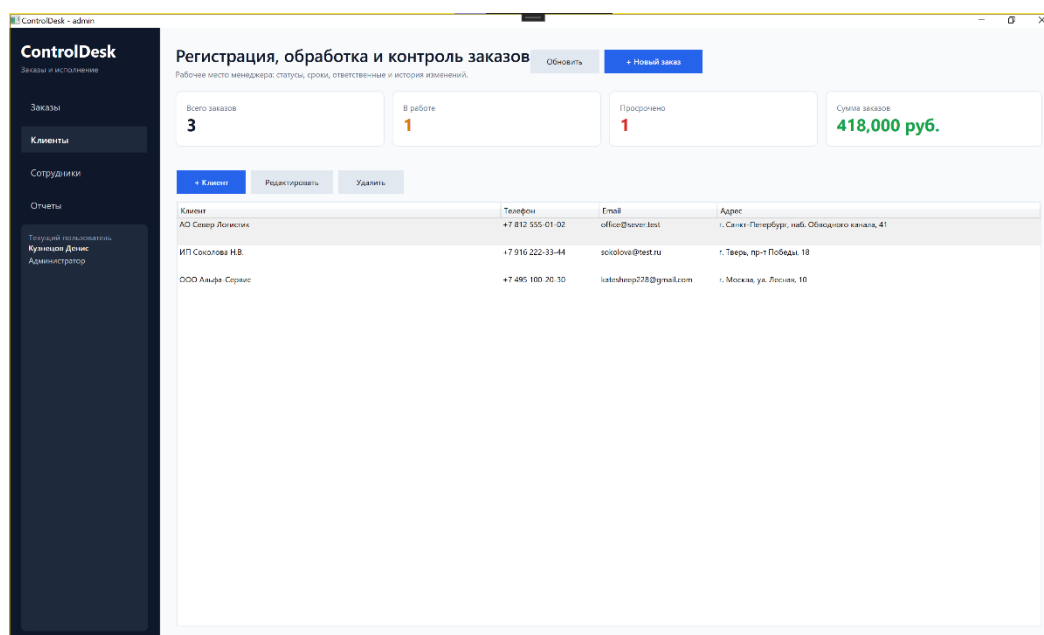


Рисунок 19 – Окно «Сотрудники» подсистемы

В рамках данного модуля реализованы следующие функциональные элементы:

- панель инструментов управления: кнопки «+ Клиент», «Редактировать» и «Удалить» обеспечивают полный цикл администрирования записей в справочнике;
- централизованный реестр контрагентов: информационная таблица отображает полные наименования организаций, их контактные телефоны, электронные адреса и фактические адреса расположения;
- механизм модального редактирования: при добавлении или изменении данных открывается специализированное окно ClientWindow, блокирующее основной интерфейс для исключения ошибок ввода;
- система валидации данных: встроенные алгоритмы проверяют заполнение обязательных полей и корректность формата электронного адреса перед сохранением записи в базу данных;
- контроль ссылочной целостности: программная логика запрещает удаление контрагентов, за которыми в системе закреплены активные или архивные заказы;

– интеграция с модулем заказов: данные из этого раздела автоматически подгружаются в выпадающие списки при регистрации новых заявок, исключая необходимость повторного ввода реквизитов.

Работа со справочником клиентов организована таким образом, чтобы минимизировать количество рутинных операций. Благодаря реализации через паттерн MVVM, любое обновление данных в этом разделе мгновенно становится доступным в главном окне управления заказами. Это позволяет поддерживать актуальность клиентской базы и обеспечивает высокую точность формирования аналитической отчетности.

Для глубокого анализа показателей и подготовки данных к внешнему использованию в подсистеме реализован модуль аналитической отчетности. Интерфейс данного раздела представлен на рисунке 26. Окно спроектировано в виде интерактивного конструктора, который позволяет пользователю гибко настраивать параметры выборки и состав выходных данных.

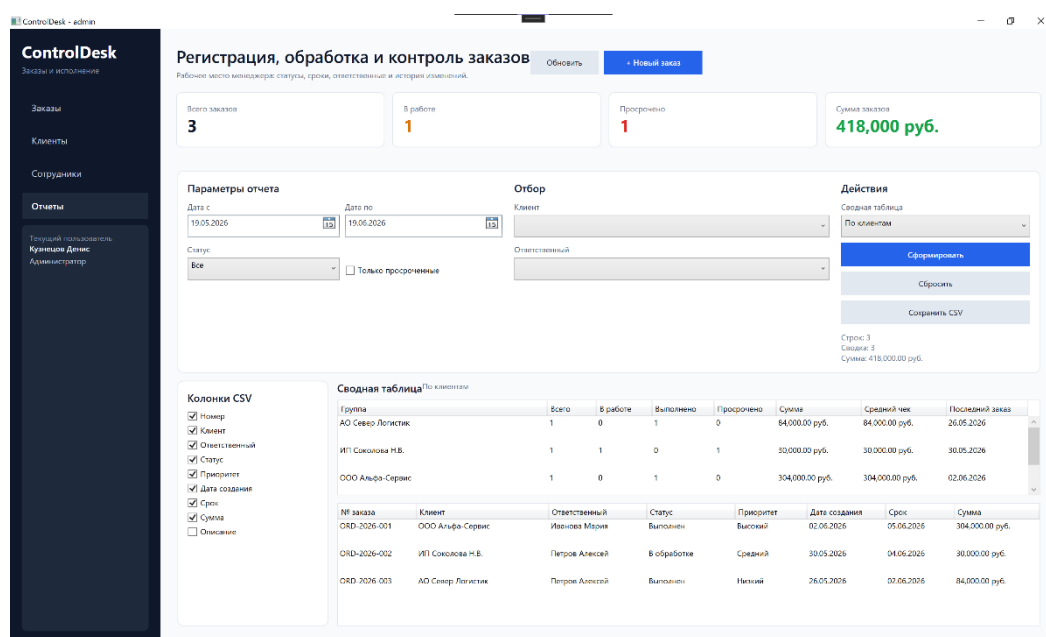


Рисунок 20 – окно «Отчеты» разрабатываемой системы

Модуль выполнен в едином стиле с основным интерфейсом: боковое меню и верхние виджеты статистики остаются на своих местах. Рабочая область разделена на несколько отдельных блоков, каждый из которых решает свою задачу.

В модуль входят следующие инструменты:

- фильтр по периоду задаёт временные границы выборки заказов;
- группировка данных доступна по клиентам или по исполнителям;
- в сводной таблице выводятся количество заказов, общая сумма и средний чек по каждой группе;
- кнопки экспорта позволяют сбросить настройки или сохранить отчёт в CSV.

Для отбора данных предусмотрено:

- выпадающие списки сужают выборку по контрагенту, сотруднику или статусу заявки;
- отдельный параметр выводит только записи с нарушенными сроками;
- переключатели в настройках CSV определяют состав колонок в выгружаемом файле.

При разработке раздела основное внимание уделено автоматизации расчётов. При смене фильтров суммы пересчитываются сразу, в отчёте показывается количество строк. Руководство получает актуальные данные о работе сотрудников и платежах клиентов без дополнительных программ. Готовый отчёт открывается в табличном редакторе и используется для анализа или печати.

Разработанный интерфейс ControlDesk соответствует поставленным задачам и удобен для повседневной работы. Применённые подходы к построению окон и управлению данными обеспечивают понятную и комфортную среду для работы с заказами.

В результате проектирования интерфейса получены следующие результаты:

- оформление всех зон выполнено в едином стиле, что сокращает время на обучение сотрудников;
- панели с аналитикой показывают ключевые данные сразу, без лишних запросов;

- цветовая маркировка статусов и просрочек ускоряет проверку текущих задач;
- модальные окна для ввода данных помогают сосредоточиться на операции и снижают риск ошибок.

Разработанный интерфейс является законченным программным решением, которое обеспечивает комфортное взаимодействие пользователя с базой данных и инструментами аналитики.

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования

Тестирование ПО – это процесс, в ходе которого фактическое поведение системы сверяется с ожидаемым на основе заранее подготовленных наборов входных данных. Его главная задача – удостовериться, что продукт отвечает установленным требованиям, гарантирует должный уровень качества и позволяет выявить критичные сбои на ранних этапах, не давая пользователям столкнуться с ними в процессе эксплуатации.

Для обеспечения высокого качества подсистемы ControlDesk и её соответствия техническому заданию в процессе разработки рассматривались различные методологии тестирования программного обеспечения. К наиболее популярным видам относятся функциональное, модульное, интеграционное и тестирование производительности.

Функциональное тестирование направлено на проверку выполнения системой конкретных пользовательских задач и оценку соответствия поведения программы установленным требованиям. Модульное тестирование ориентировано на проверку минимальных программных единиц, таких как классы, методы или сервисы, в изоляции от остальной части приложения. Интеграционное тестирование проверяет, как взаимодействуют компоненты системы, а тестирование производительности оценивает скорость работы и устойчивость при больших объёмах данных.

Для проверки подсистемы ControlDesk выбран комбинированный подход – функциональное и модульное тестирование. Функциональное тестирование оценивает систему с точки зрения пользователя. В ходе испытаний проверяют основные сценарии работы в интерфейсе: от авторизации до формирования отчётов. Это позволит убедиться в правильности навигации, работе Dashboard-панели и фильтров в главном окне.

Модульное тестирование применяется для проверки внутренней логики подсистемы на уровне классов и сервисов. Проверяются алгоритмы хэширования

паролей в сервисе безопасности и методы контроля сроков в модели заказов. Модульное тестирование позволяет проверять компоненты независимо друг от друга, что ускоряет поиск ошибок и обеспечивает точность расчётов без запуска графического интерфейса.

Такой комплексный подход позволил подтвердить надежность внутренней архитектуры подсистемы и удобство её практического использования в реальных рабочих процессах.

4.2 Модульное тестирование подсистемы ControlDesk

Для проверки внутренней логики подсистемы проводится модульное тестирование. Этот этап позволяет изолированно проверить работу отдельных классов и методов без влияния интерфейса и внешней среды, что обеспечивает надёжность системы на ранних этапах. Для тестирования используется фреймворк MSTest, встроенный в Visual Studio. В качестве объектов тестирования выбраны компоненты с ключевой бизнес-логикой: модель заказа (расчёт стоимости и контроль сроков) и сервис авторизации (защита паролей).

Успешное прохождение всех тест-кейсов подтвердило стабильность работы архитектуры и готовность функциональных модулей к интеграции с пользовательским интерфейсом». Полный код модульных тестов приведен в приложении Г.

Краткое описание проведенных модульных тестов представлено в таблице 12.

Таблица 12 – Модульные тесты программных компонентов

Название теста	Объект тестирования	Описание	Результат
CalculateTotalAmount	Модель Order	Проверка правильности суммирования стоимости всех позиций в заказе	Пройден

Продолжение таблицы 12

Название теста	Объект тестирования	Описание	Результат
CheckIsOverdue_True	Модель Order	Верификация логики определения просрочки при истекшем дедлайне	Пройден
CheckIsOverdue_False	Модель Order	Проверка отсутствия признака просрочки для выполненных заказов	Пройден
PasswordHash_Unique	Сервис AuthService	Проверка того, что хэширование одного и того же пароля дает стабильный результат	Пройден
Authenticate_Failure	Сервис AuthService	Проверка отказа в доступе при вводе неверных учетных данных	Пройден

Результаты модульного тестирования контроллера занятий приведены на рисунке 27.

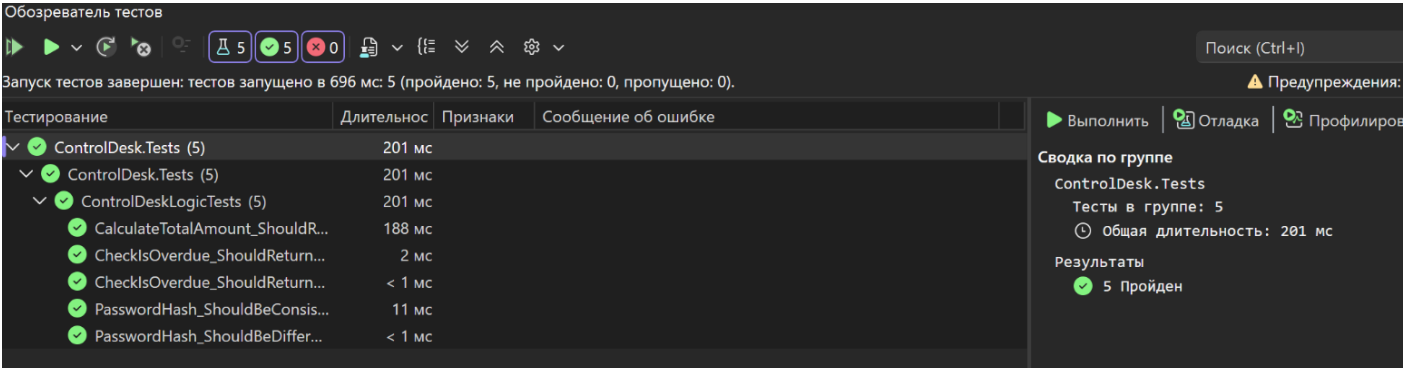


Рисунок 21 – Результаты тестирования в обозревателе решений

По результатам можно сделать вывод, что тесты пройдены успешно и подсистема работает корректно. Таким образом система прошла этап модульного тестирования.

4.3 Функциональное тестирование подсистемы ControlDesk

После завершения программной реализации было проведено комплексное функциональное тестирование подсистемы. Результаты выполнения тестовых сценариев приведены в приложении Ж.

Для проверки логики управления основным объектом системы – заказом, была разработана диаграмма состояний и переходов (рисунок 62). Она наглядно отображает жизненный цикл заявки от момента регистрации до завершения или отмены.

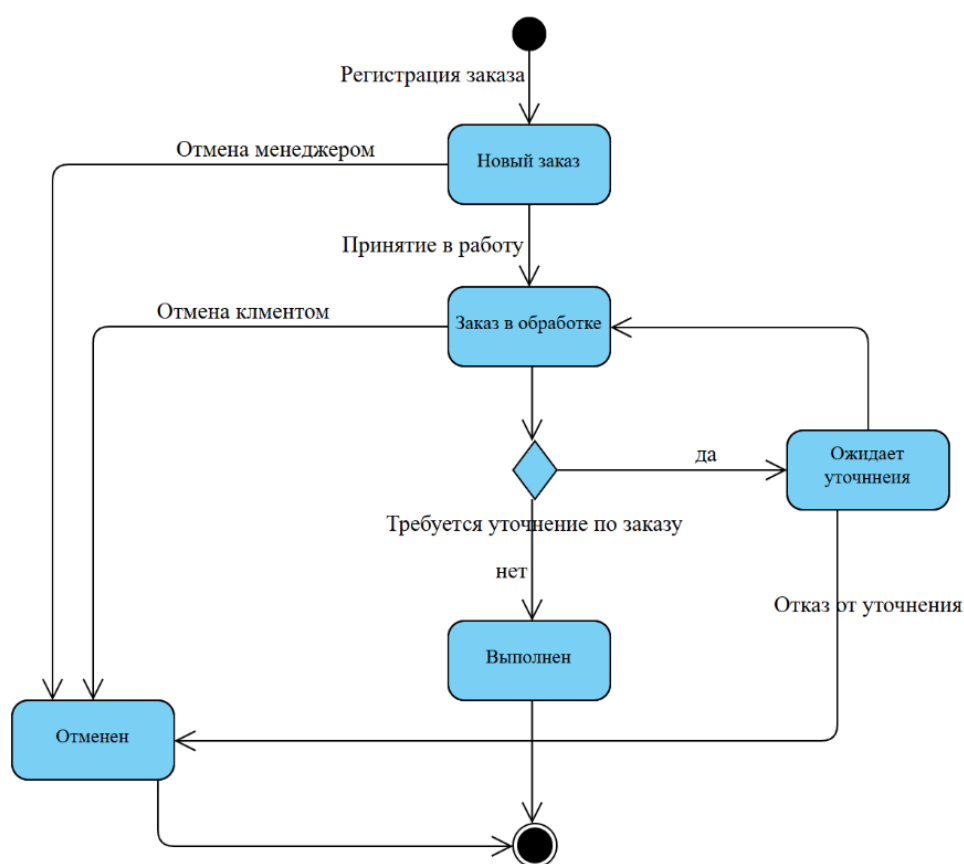


Рисунок 22 – Диаграмма состояний и переходов заказа

Использование диаграммы состояний в процессе функционального тестирования помогает отследить все возможные сценарии работы с заказом и проверить корректность реакции системы на действия пользователя. На основании данной диаграммы были сформированы тестовые сценарии, представленные в таблице 13.

Таблица 13 – Тест-кейсы функционала управления заказами

Название	Шаги	Ожидаемый результат	Фактический
Регистрация нового заказа	<ol style="list-style-type: none"> 1. Авторизоваться в системе. 2. Нажать кнопку «+ Новый заказ». 3. Ввести данные клиента и ответственного. 4. Нажать кнопку «Сохранить». 	В общем реестре появляется новая запись со статусом «Новый».	Соответствует ожидаемому (рисунки Ж.1 – Ж.3)
Смена статуса заказа	<ol style="list-style-type: none"> 1. Выбрать заказ в списке. 2. В панели управления выбрать статус «Выполнен». 3. Нажать кнопку «Обновить статус». 	Статус заказа меняется на выбранный, цвет индикатора обновляется.	Соответствует ожидаемому (рисунки Ж.4 – Ж.5)
Отмена заказа	<ol style="list-style-type: none"> 1. Выбрать активный заказ в реестре. 2. Нажать кнопку «Отменить» в панели управления. 3. Подтвердить действие в модальном окне. 	Заказу присваивается статус «Отменен», запись в истории фиксирует факт отмены.	Соответствует ожидаемому (рисунок Ж.6)

Для описания функционала справочников (клиенты и сотрудники) подготовлены сценарии тестирования, которые приведены в таблице 14.

Эта модель наглядно показывает переход объекта из режима просмотра в режим редактирования с последующей проверкой внесённых данных. Детальная проверка необходима для сохранения целостности данных и предотвращения удаления записей, связанных с действующими заказами. Такой подход исключает ошибки в учёте и обеспечивает полноту и достоверность информации. В ходе тестирования проверяются все этапы работы со справочником: от создания записи до её изменения и удаления.

Таблица 14 – Тест-кейсы функционала ведения справочников

Название	Шаги	Ожидаемый результат	Фактический
Добавление нового клиента	<ol style="list-style-type: none"> 1. Перейти в раздел «Клиенты». 2. Нажать кнопку «+ Клиент». 3. Заполнить контактные данные. 4. Нажать кнопку «Сохранить». 	Новая запись отображается в таблице справочника.	Соответствует ожидаемому (рисунок Ж.7, Ж.8)
Редактирование данных сотрудника	<ol style="list-style-type: none"> 1. Перейти в раздел «Сотрудники». 2. Выбрать запись и нажать «Редактировать». 3. Изменить номер телефона или должность. 4. Нажать кнопку «Сохранить». 	Информация в базе данных успешно обновлена.	Соответствует ожидаемому (рисунок Ж.9)
Валидация при удалении	<ol style="list-style-type: none"> 1. Выбрать клиента, у которого есть активные заказы. 2. Нажать кнопку «Удалить». 3. Подтвердить действие. 	Система выводит уведомление о невозможности удаления записи.	Соответствует ожидаемому (рисунок Ж.10)

Для наглядного представления процесса формирования аналитики разработана диаграмма состояний модуля отчетности (рисунок 64). На ней отражён полный цикл работы: от настройки фильтров до получения готового файла.

Создание этой модели позволило проследить логику действий пользователя при работе с инструментами отбора и группировки данных. На диаграмме выделены основные этапы проверки параметров – от выбора периода и контрагентов до предварительного просмотра результатов. Также

предусмотрен возврат к началу для корректировки условий поиска, что делает процесс гибким и снижает вероятность формирования неверных отчетов.

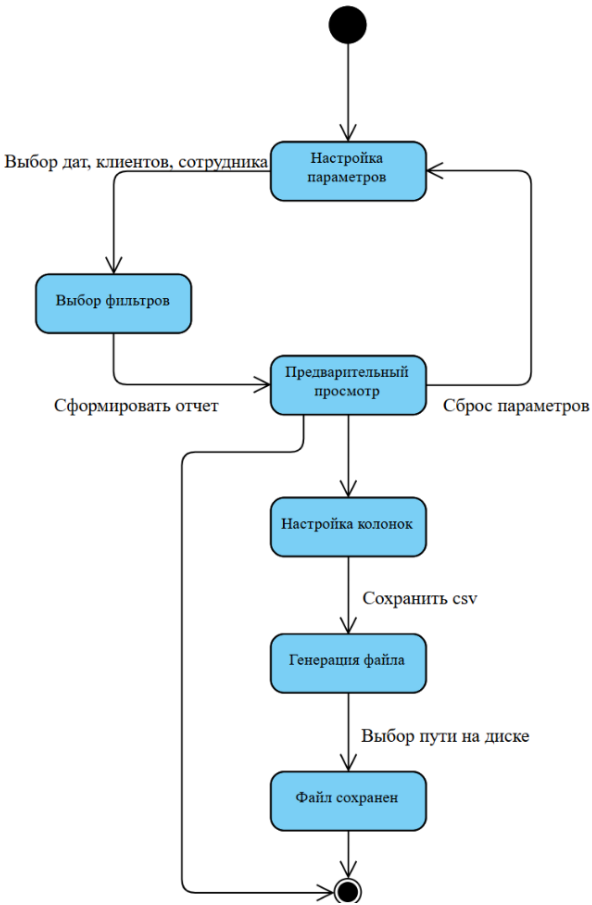


Рисунок 23 – Диаграмма состояний и переходов отчетов

На основании данной диаграммы были сформированы тестовые сценарии, представленные в таблице 15.

Таблица 15 – Тест-кейсы функционала ведения отчетов

Название	Шаги	Ожидаемый результат	Фактический
Применение фильтров отчета	1. Перейти в раздел «Отчеты». 2. Выбрать период и конкретного клиента. 3. Нажать кнопку «Сформировать».	В сводной таблице отображаются данные только по заданным критериям.	Соответствует ожидаемому (рисунок Ж.11)

Продолжение таблицы 15

Название	Шаги	Ожидаемый результат	Фактический
Настройка колонок экспорта	1. В блоке конструктора снять галочки с части колонок. 2. Нажать кнопку «Сформировать».	Предварительный просмотр таблицы меняется согласно выбранным полям.	Соответствует ожидаемому (рисунок Ж.12)
Сохранение в формат CSV	1. Нажать кнопку «Сохранить CSV». 2. Выбрать путь для сохранения файла. 3. Открыть сохраненный файл в табличном редакторе.	Файл содержит корректную структуру данных и заголовков.	Соответствует ожидаемому (рисунок Ж.13)

По итогам функционального тестирования все модули подсистемы ControlDesk работают штатно. Применение диаграмм состояний позволило проверить поведение системы в разных сценариях и подтвердить правильность реализации бизнес-процессов.

В ходе испытаний выявлено:

- управление заказами отрабатывает смену статусов и сохраняет историю изменений;
- справочная информация защищена от нарушений целостности за счёт встроенных проверок;
- отчётный модуль формирует данные любой сложности и корректно выгружает их в CSV.

Отдельно проверен интерфейс: он информативен, понятно реагирует на действия пользователя, а цветовые индикаторы статусов и просрочек снижают риск ошибок. Таким образом, проведённая проверка подтвердила соответствие системы техническим требованиям и возможность её внедрения в работу.

4.4 Статический анализ качества программного кода

Для объективной оценки качества архитектуры подсистемы ControlDesk был выполнен расчёт метрик кода средствами статического анализа Visual Studio.

Метрики кода представляют собой количественные показатели, с помощью которых оценивают качество программного обеспечения. Они дают представление о таких свойствах системы, как понятность исходного текста, сложность структуры, степень разделения на модули и лёгкость последующего сопровождения. Применение таких показателей позволяет своевременно обнаруживать недостатки на начальных стадиях разработки, а также облегчает дальнейшую поддержку и развитие продукта [21].

Результаты анализа, представленные на рисунке 30, подтверждают высокую степень готовности программного продукта к эксплуатации и сопровождению.

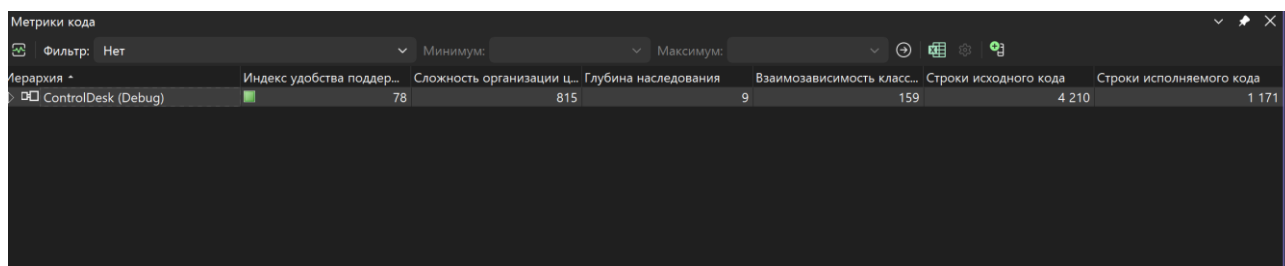


Рисунок 24 – Анализ метрик программного кода

Для более глубокого анализа полученных данных основные значения метрик были упорядочены и представлены в таблице 16.

Таблица 16 – Значения метрик кода подсистемы ControlDesk

Метрика	Значение	Трактовка результата
Индекс удобства поддержки	78	Высокий уровень качества (Зеленая зона). Код легко читаем и пригоден для модификации.
Сложность организации циклов	815	Суммарный показатель по всему проекту. Указывает на умеренную алгоритмическую сложность.
Глубина наследования	9	Отражает использование иерархии классов платформы .NET и WPF.

Продолжение таблицы 16

Метрика	Значение	Трактовка результата
Взаимозависимость классов	159	Общее количество связей между компонентами, что характерно для многослойной архитектуры.
Строки исходного кода	4 210	Реальный объем разработанного кода, включая XAML-разметку и C#-логику.
Строки исполняемого кода	1 171	Фактический объем логических программных инструкций.

Анализ полученных данных:

- индекс удобства поддержки (78): значение показывает, что код написан модульно, а это упрощает его доработку и обслуживание;
- сложность организации (815): общий показатель говорит о том, что в логике обработки заказов нет излишне запутанных алгоритмов;
- глубина наследования (9): отражает применение стандартных средств WPF, где пользовательские компоненты встроены в общую иерархию классов платформы;
- взаимозависимость классов (159): показывает, что обязанности между моделями, сервисами и представлениями распределены правильно;
- строки исходного и исполняемого кода (4210 и 1171): такое соотношение обычно встречается в WPF-проектах, где около 70% кода составляет разметка интерфейса в XAML, а оставшиеся 1171 строк — это программная логика на C#.

Таким образом, комплексный анализ метрик подтверждает, что подсистема ControlDesk имеет качественную программную архитектуру. Высокий индекс поддерживаемости и умеренная сложность исполняемого кода гарантируют надежность системы и низкие затраты на её развитие в будущем.

Заключение

В ходе выполнения выпускной квалификационной работы разработана информационная подсистема управления заказами «ControlDesk». Цель работы, состоявшая в проектировании и программной реализации инструмента для автоматизации процессов регистрации, сопровождения и анализа заказов, достигнута в полном объёме.

На этапе анализа предметной области определены основные потребности бизнеса в сфере оперативного управления заявками. Сформулированы функциональные требования, включающие управление жизненным циклом заказов, ведение справочников контрагентов и персонала, а также разграничение доступа. Учтены нефункциональные требования: удобство интерфейса, надёжность хранения данных и производительность при работе с большими объёмами информации.

При проектировании системы выбрана многослойная архитектура на основе паттерна MVVM. Это позволило разделить логику обработки данных и визуальное представление, что обеспечило гибкость и расширяемость продукта. С помощью языка UML построены диаграммы вариантов использования и состояний, которые легли в основу логической структуры приложения. База данных спроектирована с учётом нормализации и требований ссылочной целостности.

Программная реализация выполнена на C# с использованием .NET Framework 4.8.1 и WPF. Применение Entity Framework 6 позволило организовать взаимодействие с SQL Server на уровне объектных моделей. Среди ключевых особенностей реализации:

- сервис безопасности с криптографической защитой паролей (SHA256);
- модуль автоматических Email-уведомлений об изменении статуса заказов;

- система ведения истории изменений для отслеживания действий пользователей;
- конструктор отчётов с выгрузкой в CSV.

Особое внимание уделено интерфейсу. Dashboard-панель с виджетами статистики и цветовой индикацией статусов обеспечила высокую эргономику. Интуитивная навигация и модальные окна создали комфортные условия для сотрудников с разным уровнем подготовки.

На заключительном этапе выполнено тестирование системы. Модульные проверки показали, что алгоритмы подсчёта сумм, контроля сроков и аутентификации работают в соответствии с ожиданиями. Функциональное тестирование по методу «чёрного ящика» подтвердило, что система отвечает заданным сценариям использования. По результатам статического анализа метрик кода получены значения индекса поддерживаемости (78) и показателей сложности, которые не выходят за допустимые пределы.

Разработанная подсистема полностью работоспособна и готова к внедрению. Практическая значимость проекта состоит в повышении скорости обработки заказов, снижении ошибок персонала и получении аналитики для управленческих решений. Подсистема «ControlDesk» обладает потенциалом для дальнейшего масштабирования и интеграции с другими системами предприятия.

Список использованных источников

1. Федеральный закон от 24.07.2007 № 209-ФЗ «О развитии малого и среднего предпринимательства в Российской Федерации».
2. Федеральный закон от 27.07.2006 № 149-ФЗ «Об информации, информационных технологиях и о защите информации».
3. Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных».
4. Информационная система — сервис размещения заказов и услуг / Электронный архив ТПУ. URL: <https://earchive.tpu.ru/handle/11683/67016> (дата обращения: 23.06.2026).
5. Database First // Microsoft [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/ef/ef6/modeling/designer/workflows/database-first> (дата обращения: 25.05.2024)
6. Entity Framework 6. Использование Database First // Professor Web [Электронный ресурс]. URL: https://professorweb.ru/my/entity-framework/6/level1/1_6.php (дата обращения: 25.05.2026).
7. Программный код и его метрики // Habr.com [Электронный ресурс]. URL: <https://habr.com/ru/company/intel/blog/106082/> (дата обращения: 25.05.2026).
8. Аверьянов, А.В. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ / А.В. Аверьянов, И.Н. Кошель, В.В. Кузнецов // Известия высших учебных заведений. Приборостроение. – 2019. – №11. – С. 970 – 975.
9. Полякова, Э.Р. Исследование программного обеспечения встраиваемой вычислительной системы метриками Холстеда / Э.Р. Полякова, А.А. Поляков, А.Л.Шустов, Д.Г. Придачкин // Cloud of Science. – 2020. – №4. – С. 950 – 959.
10. Диаграмма вариантов использования: учебно-методическое пособие по дисциплине «Современные информационные технологии». – Пенза: Изд-во Пензенского государственного университета. – 14 с.

11. Создаем диаграмму Ганта // UNISENDER [Электронный ресурс] URL: <https://www.unisender.com/ru/blog/chto-takoe-diagramma-ganta-i-kak-ee-sdelat/> (Дата обращения: 15.05.2026).

12. Урок по диаграммам развертывания // LUCIDCHART [Электронный ресурс]. URL: <https://www.lucidchart.com/pages/ru/uml-deployment-diagram> (дата обращения: 16.05.2026).

13. Обзорная статья о видах тестирования IT-продуктов // ХАБР [Электронный ресурс] URL: <https://habr.com/ru/companies/otus/articles/720664/> (дата обращения 04.06.2026)

14. Draw.io // SKILLFACTORY MEDIA [Электронный ресурс]. URL: <https://blog.skillfactory.ru/draw-io/> (дата обращения: 04.06.2026).

Приложение А
(справочное)
Глоссарий

Автоматизация – применение технических и программных средств для частичного или полного освобождения человека от непосредственного участия в процессах получения, преобразования, передачи и использования информации.

Архитектура «клиент-сервер» – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

Информационная подсистема – выделенная часть информационной системы, отвечающая за выполнение определенного набора функциональных задач и обладающая свойствами самостоятельности в рамках общей архитектуры программного комплекса.

Паттерн проектирования (шаблон проектирования) – повторяемая архитектурная конструкция, представляющая собой формализованное и проверенное на практике решение типичной проблемы в рамках конкретного контекста проектирования программного обеспечения.

Привязка данных (Data Binding) – механизм синхронизации данных между визуальными элементами пользовательского интерфейса и источником данных (моделью), обеспечивающий автоматическое обновление интерфейса при изменении данных и наоборот.

Репозиторий (шаблон) – архитектурный паттерн проектирования, который изолирует слой доступа к данным от остальной части приложения, предоставляя объектно-ориентированный интерфейс для извлечения и сохранения объектов сущностей.

Толстый клиент (Fat client / Thick client) – тип архитектуры клиентского приложения, в котором графический интерфейс и значительная часть вычислительной бизнес-логики (обработка данных) выполняются на стороне пользователя, что позволяет программе функционировать независимо от постоянного соединения с центральным сервером.

Хэширование – процесс криптографического или математического преобразования массива данных произвольной длины в строку фиксированной

длины (хэш-код), применяемый для проверки целостности данных и безопасного хранения паролей.

.NET Framework – программная платформа, разработанная корпорацией Microsoft, предоставляющая среду выполнения (CLR) и обширную библиотеку базовых классов для создания и запуска программного обеспечения под управлением операционных систем семейства Windows.

C# (C Sharp) – современный объектно-ориентированный язык программирования со строгой типизацией, разработанный корпорацией Microsoft для создания прикладных программ, компонентов и служб в экосистеме .NET.

CSV (Comma-Separated Values) – текстовый формат, предназначенный для представления табличных данных, в котором каждая строка файла соответствует одной строке таблицы, а значения колонок разделяются специальным символом (как правило, запятой или точкой с запятой).

Entity Framework (EF) – технология объектно-реляционного отображения (ORM) от компании Microsoft, позволяющая разработчикам взаимодействовать с реляционными базами данных посредством строго типизированной объектной модели, абстрагируясь от прямого написания SQL-запросов.

MVVM (Model-View-ViewModel) – архитектурный паттерн проектирования пользовательских интерфейсов, обеспечивающий независимое и четкое разделение бизнес-логики приложения (модели), логики представления (модели представления) и визуального графического интерфейса (представления).

ORM (Object-Relational Mapping) – технология программирования, которая связывает реляционные базы данных с концепциями объектно-ориентированных языков, создавая виртуальную объектную базу данных для удобства работы с кодом.

SMTP (Simple Mail Transfer Protocol) – широко используемый сетевой протокол прикладного уровня, предназначенный для маршрутизации и передачи электронной почты в сетях TCP/IP.

UML (Unified Modeling Language) – стандартизированный унифицированный язык графического описания, предназначенный для визуализации, специфицирования, конструирования и документирования артефактов программных систем.

WPF (Windows Presentation Foundation) – графическая (презентационная) подсистема в составе платформы .NET, предназначенная для создания визуальных пользовательских интерфейсов настольных приложений с использованием механизмов аппаратного ускорения и декларативной разметки.

XAML (eXtensible Application Markup Language) – декларативный язык разметки на основе XML, разработанный компанией Microsoft, применяемый для инициализации структурированных значений и объектов, в частности, для конструирования визуального интерфейса в технологии WPF.

**Приложение Б
(Обязательное)**

Листинг бизнес-логики и управления данными

Листинг Б.1 – ControlDeskDbContext.cs

```
using ControlDesk.Models;
using System.Data.Entity;

namespace ControlDesk.Data
{
    public class ControlDeskDbContext : DbContext
    {
        public ControlDeskDbContext()
            : base("name=ControlDeskDb")
        {
        }

        public DbSet<Client> Clients { get; set; }
        public DbSet<Employee> Employees { get; set; }
        public DbSet<Order> Orders { get; set; }
        public DbSet<OrderItem> OrderItems { get; set; }
        public DbSet<OrderHistoryEntry> OrderHistory { get;
set; }

        public DbSet<User> Users { get; set; }

        protected override void
OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Client>().ToTable("Clients");

modelBuilder.Entity<Employee>().ToTable("Employees");
            modelBuilder.Entity<Order>().ToTable("Orders");

modelBuilder.Entity<OrderItem>().ToTable("OrderItems");

modelBuilder.Entity<OrderHistoryEntry>().ToTable("OrderHistory");
            modelBuilder.Entity<User>().ToTable("Users");


            modelBuilder.Entity<User>()
                .HasOptional(item => item.Employee)
                .WithMany()
                .HasForeignKey(item => item.EmployeeId)
                .WillCascadeOnDelete(false);

            modelBuilder.Entity<Client>()
                .Property(item => item.FullName)
                .IsRequired()
                .HasMaxLength(150);

            modelBuilder.Entity<Order>()
                .HasRequired(item => item.Client)
                .WithMany()
                .HasForeignKey(item => item.ClientId)
                .WillCascadeOnDelete(false);
        }
    }
}
```

```

        modelBuilder.Entity<Order>()
            .HasRequired(item =>
item.ResponsibleEmployee)
            .WithMany()
            .HasForeignKey(item =>
item.ResponsibleEmployeeId)
            .WillCascadeOnDelete(false);

        modelBuilder.Entity<Order>()
            .HasMany(item => item.Items)
            .WithRequired()
            .HasForeignKey(item => item.OrderId)
            .WillCascadeOnDelete(true);

        modelBuilder.Entity<Order>()
            .HasMany(item => item.History)
            .WithRequired()
            .HasForeignKey(item => item.OrderId)
            .WillCascadeOnDelete(true);

        base.OnModelCreating(modelBuilder);
    }
}

```

Листинг Б.2 – EmailService.cs

```

using ControlDesk.Models;
using System;
using System.Configuration;
using System.Net;
using System.Net.Mail;

namespace ControlDesk.Services
{
    public class EmailService
    {
        public bool TrySendOrderStatusChanged(Order order,
string changedBy, out string error)
        {
            error = null;
            ConsoleLogService.WriteLine("Email notification
requested.");

            if (!IsEnabled())
            {
                ConsoleLogService.WriteLine("Email
notification skipped: EmailNotificationsEnabled is false.");
                return false;
            }

            if (order == null)

```

```

        {
            ConsoleLogService.WriteLine("Email
notification skipped: order is null.");
            return false;
        }

        if (order.Client == null)
        {
            ConsoleLogService.WriteLine("Email
notification skipped: order client is null.");
            return false;
        }

        if
(string.IsNullOrEmpty(order.Client.Email))
        {
            ConsoleLogService.WriteLine("Email
notification skipped: client email is empty. Order: " +
order.OrderNumber);
            return false;
        }

        try
        {
            SendOrderStatusChanged(order, changedBy);
            ConsoleLogService.WriteLine("Email
notification sent. Order: " + order.OrderNumber + ", To: " +
order.Client.Email);
            return true;
        }
        catch (Exception ex)
        {
            error = ex.Message;
            ConsoleLogService.WriteLine("Email
notification failed. Order: " + order.OrderNumber + ", Error: " +
ex);
            return false;
        }
    }

    private static bool IsEnabled()
    {
        return
string.Equals(ConfigurationManager.AppSettings["EmailNotifications
Enabled"], "true", StringComparison.OrdinalIgnoreCase);
    }

    private static void SendOrderStatusChanged(Order
order, string changedBy)
    {
        var host = GetRequiredSetting("SmtpHost");
        var from = GetRequiredSetting("SmtpFrom");
    }

```

```

        var user =
ConfigurationManager.AppSettings["SmtpUser"];
        var password =
ConfigurationManager.AppSettings["SmtpPassword"];
        var port = GetIntSetting("SmtpPort", 587);
        var enableSsl = GetBoolSetting("SmtpEnableSsl",
true);

        ConsoleLogService.WriteLine("SMTP settings:
Host=" + host + ", Port=" + port + ", EnableSsl=" + enableSsl + ",
From=" + from + ", User=" + user);
        ConsoleLogService.WriteLine("SMTP recipient: " +
order.Client.Email + ", Order: " + order.OrderNumber + ", Status:
" + order.Status);

        using (var message = new MailMessage())
        {
            message.From = new MailAddress(from);
            message.To.Add(order.Client.Email);
            message.Subject = "Изменение статуса заказа "
+ order.OrderNumber;
            message.Body =
                "Здравствуйте!\r\n\r\n" +
                "Статус вашего заказа изменен.\r\n" +
                "Номер заказа: " + order.OrderNumber +
"\r\n" +
                "Новый статус: " + order.Status + "\r\n"
+
                "Дата изменения: " +
DateTime.Now.ToString("dd.MM.yyyy HH:mm") + "\r\n" +
                "Изменил: " +
(string.IsNullOrEmpty(changedBy) ? "ControlDesk" : changedBy)
+ "\r\n\r\n" +
                "ControlDesk";

            using (var client = new SmtpClient(host,
port))
            {
                client.EnableSsl = enableSsl;
                client.DeliveryMethod =
SmtpDeliveryMethod.Network;

                if (!string.IsNullOrEmpty(user))
                {
                    client.Credentials = new
NetworkCredential(user, password);
                }

                client.Send(message);
            }
        }
    }

    private static string GetRequiredSetting(string key)

```

```

        {
            var value =
ConfigurationManager.AppSettings[key];
            if (string.IsNullOrEmpty(value))
            {
                throw new InvalidOperationException("Не
указана настройка " + key + " в App.config.");
            }

            return value;
        }

        private static int GetIntSetting(string key, int
defaultValue)
        {
            int value;
            return
int.TryParse(ConfigurationManager.AppSettings[key], out value) ?
value : defaultValue;
        }

        private static bool GetBoolSetting(string key, bool
defaultValue)
        {
            bool value;
            return
bool.TryParse(ConfigurationManager.AppSettings[key], out value) ?
value : defaultValue;
        }
    }
}

```

**Приложение В
(Обязательное)**

**Листинг пользовательского интерфейса
и логики представления**

Листинг В.1 – Order.cs

```
using System;
using System.Collections.ObjectModel;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;

namespace ControlDesk.Models
{
    public class Order : ObservableObject
    {
        private string status;
        private Employee responsibleEmployee;
        private DateTime? completedDate;

        public int Id { get; set; }
        public string OrderNumber { get; set; }
        public int ClientId { get; set; }
        public int ResponsibleEmployeeId { get; set; }
        public Client Client { get; set; }

        public Employee ResponsibleEmployee
        {
            get { return responsibleEmployee; }
            set { SetProperty(ref responsibleEmployee,
value); }
        }

        public string Status
        {
            get { return status; }
            set
            {
                if (SetProperty(ref status, value))
                {
                    OnPropertyChanged(nameof(IsOverdue));
                }
            }
        }

        public DateTime CreatedDate { get; set; }
        public DateTime DeadlineDate { get; set; }

        public DateTime? CompletedDate
        {
            get { return completedDate; }
            set
            {
                if (SetProperty(ref completedDate, value))
                {
                    OnPropertyChanged(nameof(IsOverdue));
                }
            }
        }
    }
}
```

```

    }

    public string Description { get; set; }
    public string Priority { get; set; }
    public ObservableCollection<OrderItem> Items { get;
set; } = new ObservableCollection<OrderItem>();
    public ObservableCollection<OrderHistoryEntry>
History { get; set; } = new
ObservableCollection<OrderHistoryEntry>();

    [NotMapped]
    public decimal TotalAmount => Items.Sum(item =>
item.Sum);

    [NotMapped]
    public bool IsOverdue => CompletedDate == null
        && Status != "Выполнен"
        && Status != "Отменен"
        && DeadlineDate.Date < DateTime.Today;
    }
}

```

Листинг В.2 – Order.cs

```

using System;
using System.Globalization;
using System.Windows.Data;
using System.Windows.Media;

namespace ControlDesk.Converters
{
    public class StatusBrushConverter : IValueConverter
    {
        public object Convert(object value, Type targetType,
object parameter, CultureInfo culture)
        {
            switch ((value ?? string.Empty).ToString())
            {
                case "Новый":
                    return new
SolidColorBrush(Color.FromRgb(37, 99, 235));
                case "В обработке":
                    return new
SolidColorBrush(Color.FromRgb(217, 119, 6));
                case "Ожидает уточнения":
                    return new
SolidColorBrush(Color.FromRgb(234, 88, 12));
                case "Выполнен":
                    return new
SolidColorBrush(Color.FromRgb(22, 163, 74));
                case "Отменен":
                    return new
SolidColorBrush(Color.FromRgb(100, 116, 139));
            }
        }
    }
}

```

```
        default:
            return new
SolidColorBrush(Color.FromRgb(51, 65, 85));
    }

    public object ConvertBack(object value, Type
targetType, object parameter, CultureInfo culture)
    {
        throw new NotSupportedException();
    }
}
```

Приложение Г
(Обязательное)
Листинг модульных тестов

Листинг Г.1 - UnitTest1.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using ControlDesk.Models;
using ControlDesk.Services;
using System;
using System.Collections.ObjectModel;

namespace ControlDesk.Tests
{
    [TestClass]
    public class ControlDeskLogicTests
    {
        // 1. Тест расчета общей суммы заказа
        [TestMethod]
        public void
CalculateTotalAmount_ShouldReturnCorrectSum()
        {
            // Arrange (Подготовка)
            var order = new Order();
            order.Items = new ObservableCollection<OrderItem>
            {
                new OrderItem { Price = 100, Quantity = 2 },
// 200
                new OrderItem { Price = 500, Quantity = 1 }
// 500
            };

            // Act (Действие)
            decimal total = order.TotalAmount;

            // Assert (Проверка)
            Assert.AreEqual(700, total);
        }

        // 2. Тест логики определения просроченного заказа
        (True)
        [TestMethod]
        public void
CheckIsOverdue_ShouldReturnTrue_WhenDeadlinePassed()
        {
            // Arrange
            var order = new Order
            {
                Status = "НОВЫЙ",
                DeadlineDate = DateTime.Today.AddDays(-1), //
Дедлайн был вчера
                CompletedDate = null
            };

            // Act
            bool overdue = order.IsOverdue;
```

```

        // Assert
        Assert.IsTrue(overdue);
    }

    // 3. Тест логики определения просрочки (False, если
выполнен)
    [TestMethod]
    public void
CheckIsOverdue_ShouldReturnFalse_WhenCompleted()
    {
        // Arrange
        var order = new Order
        {
            Status = "Выполнен",
            DeadlineDate = DateTime.Today.AddDays(-1),
            CompletedDate = DateTime.Today
        };

        // Act
        bool overdue = order.IsOverdue;

        // Assert
        Assert.IsFalse(overdue);
    }

    // 4. Тест уникальности и стабильности хэша пароля
    [TestMethod]
    public void PasswordHash_ShouldBeConsistent()
    {
        // Arrange
        string password = "admin_password";

        // Act
        string hash1 =
AuthService.HashPassword(password);
        string hash2 =
AuthService.HashPassword(password);

        // Assert
        Assert.AreEqual(hash1, hash2);
        Assert.AreNotEqual(password, hash1); // Пароль не
должен храниться в открытом виде
    }

    // 5. Тест работы хэширования для разных паролей
    [TestMethod]
    public void
PasswordHash_ShouldBeDifferentForDifferentPasswords()
    {
        // Act
        string hash1 = AuthService.HashPassword("12345");
        string hash2 = AuthService.HashPassword("54321");
    }

```

```
        // Assert
        Assert.AreNotEqual(hash1, hash2);
    }
}
```

Приложение Д
(рекомендуемое)
Атрибуты сущностей базы данных

Таблица Д.1 – Атрибуты отношения «Client» (Клиенты)

Идентификатор столбца	Описание	Тип данных	Размерность	Область допустимых значений	Null	Роль	Пример
Id	Идентификатор клиента	integer	10	от 1 до 2 147 483 647	нет	PK	1
FullName	ФИО или название	varchar	150	Строка текста	нет	-	ООО Альфа
Phone	Телефон	varchar	20	Формат: +7XXXXXXXXXX	да	-	+79001112233
Email	Электронная почта	varchar	100	Строка, email-адрес	да	-	client@test.ru
Address	Физический адрес	varchar	250	Строка текста	да	-	г. Москва
Address	Физический адрес	varchar	250	Строка текста	да	-	г. Москва

Таблица Д.2 – Атрибуты отношения «Employee» (Сотрудники)

Идентификатор столбца	Описание	Тип данных	Размерность	Область допустимых значений	Null	Роль	Пример
Id	Идентификатор сотрудника	integer	10	от 1 до 2 147 483 647	нет	PK	1
FullName	ФИО сотрудника	varchar	150	Строка текста	нет	-	Иванов И.И.
Position	Должность	varchar	100	Строка текста	да	-	Менеджер
Phone	Контактный телефон	varchar	20	Формат: +7XXXXXXXXXX	да	-	+79005556677
Email	Рабочий Email	varchar	100	Строка, email-адрес	да	-	ivanov@work.ru

Таблица Д.3 – Атрибуты отношения «User» (Учетные записи)

Идентификатор столбца	Описание	Тип данных	Размерность	Область допустимых значений	Null	Роль	Пример
Id	Идентификатор записи	integer	10	от 1 до 2 147 483 647	нет	PK	1
Login	Логин пользователя	varchar	50	Строка (уникально)	нет	-	admin
PasswordHash	Хэш-сумма пароля	varchar	64	Строка (SHA256)	нет	-	a665a4592...
RoleName	Наименование роли	varchar	50	Администратор, Менеджер	нет	-	Менеджер
Employee_Id	Идентификатор связи	integer	10	Ссылка на Employee.Id	да	FK	1

Таблица Д.4 – Атрибуты отношения «OrderItem» (Позиции заказа)

Идентификатор столбца	Описание	Тип данных	Размерность	Область допустимых значений	Null	Роль	Пример
Id	Идентификатор позиции	integer	10	от 1 до 2 147 483 647	нет	PK	1
OrderId	Ссылка на заказ	integer	10	Ссылка на Orders.Id	нет	FK	1
ProductName	Наименование товара	varchar	100	Строка текста	нет	-	Монитор
Quantity	Количество	integer	10	Целое число > 0	нет	-	5
Price	Цена за единицу	decimal	18, 2	Положительное число	нет	-	15500.00

Таблица Д.5 – Атрибуты отношения «OrderHistory» (История изменений)

Идентификатор столбца	Описание	Тип данных	Размерность	Область допустимых значений	Null	Роль	Пример
Id	Идентификатор лога	integer	10	от 1 до 2 147 483 647	нет	PK	1
OrderId	Ссылка на заказ	integer	10	Ссылка на Orders.Id	нет	FK	1
UserName	Автор изменения	varchar	50	Логин пользователя	нет	-	manager1
Action	Тип действия	varchar	100	Строка текста	нет	-	Смена статуса
ChangeDate	Дата изменения	datetime	8	Дата и время	нет	-	02.06.2026

Таблица Д.6– Атрибуты отношения «Order» (Заказы)

Идентификатор столбца	Описание	Тип данных	Размерность	Область допустимых значений	Null	Роль	Пример
Id	Идентификатор заказа	integer	10	от 1 до 2 147 483 647	нет	PK	1
OrderNumber	Номер заказа	varchar	30	Строка (уникально)	нет	-	ORD-2026-001
ClientId	Ссылка на клиента	integer	10	Ссылка на Clients.Id	нет	FK	1
RespEmployeeId	Ответственный	integer	10	Ссылка на Employees.Id	нет	FK	2
Status	Состояние заказа	varchar	50	Новый, В обработке и др.	нет	-	Новый
Priority	Приоритет	varchar	30	Низкий, Средний, Высокий	нет	-	Высокий
CreatedDate	Дата создания	datetime	8	Дата и время	нет	-	01.06.2026
DeadlineDate	Крайний срок	datetime	8	Дата и время	нет	-	05.06.2026
CompletedDate	Дата завершения	datetime	8	Дата и время	да	-	04.06.2026

Приложение Ж
(рекомендуемое)
Результаты функционального тестирования

Вход в ControlDesk

ControlDesk

Вход в подсистему контроля заказов

Логин

admin

Пароль

.....

Демо: admin/admin, manager/manager, worker/worker

Выход Войти

Рисунок Ж.1 – Авторизация

Регистрация заказа

Заполните основные данные и позиции заказа.

Номер заказа

ORD-2026-06220115

Клиент

АО Север Логистик

Ответственный

Иванова Мария

Приоритет

Средний

Срок исполнения

25.06.2026

Описание заказа

Позиции заказа

Добавлено позиций: 0

Наименование	Описание
--------------	----------

Сохранить Отмена

Рисунок Ж.2 – Добавление нового заказа

№ заказа	Клиент	Ответственный	Статус	Приоритет	Срок
ORD-2026-06220115	АО Север Логистик	Иванова Мария	Новый	Средний	25.06.2026

Рисунок Ж.3 – Добавление нового заказа

№ заказа	Клиент	Ответственный	Статус	Приоритет	Срок
ORD-2026-06220115	АО Север Логистик	Иванова Мария	Новый	Средний	25.06.2026

Рисунок Ж.4 – Заказ в статусе Новый

№ заказа	Клиент	Ответственный	Статус	Приоритет	Срок
ORD-2026-06220115	АО Север Логистик	Иванова Мария	Отменен	Средний	25.06.2026
ORD-2026-001	ООО Альфа-Сервис	Иванова Мария	Выполнен	Высокий	05.06.2026
ORD-2026-002	ИП Соколова Н.В.	Петров Алексей	В обработке	Средний	04.06.2026
ORD-2026-003	АО Север Логистик	Петров Алексей	Выполнен	Низкий	02.06.2026

ORD-2026-06220115

АО Север Логистик

Описание

Контроль исполнения

Статус: Отменен

Срок: Новый

Исполнитель: В обработке

Сумма: Ожидает уточнения

Выполнен

Отменен

Редактировать

Удалить

Рисунок Ж.4 – Заказ переведен в статус отменен

Клиент

Карточка клиента

Наименование / ФИО

Бакаева Алина Денисовна

Телефон

89093176548

Email

ndsijfi@gmail.com

Адрес

Пушкина 27

Сохранить

Отмена

Рисунок Ж.6 – Добавление нового клиента

Клиент	Телефон	Email	Адрес
АО Север Логистик	+7 812 555-01-02	office@sever.test	г. Санкт-Петербург, наб. Обводного канала, 41
Бакаева Алина Денисовна	89093176548	ndsifji@gmail.com	Пушкина 27

Рисунок Ж.7 – Клиент успешно добавлен

ФИО	Должность	Телефон	Email
Иванова Мария(ред.)	Менеджер по заказам	+7 900 100-11-22	ivanova@company.test

Рисунок Ж.8 – Данные отредактированы успешно

+ Клиент
Редактировать
Удалить

Клиент	Телефон	Email	Адрес
АО Север Логистик	+7 812 555-01-02	office@sever.test	г. Санкт-Петербург, наб. Обводного канала, 41
Бакаева Алина Денисовна	89093176548	ndsifji@gmail.com	Пушкина 27
ИП Соколова Н.В.	+7 916 222-33-44	sokolova@test.ru	г. Тверь, пр-т Победы, 18
ООО Альфа-Сервис			г. Москва, ул. Лесная, 10

Удаление клиента

Клиента нельзя удалить, если по нему есть заказы.

An error occurred while updating the entries. See the inner exception for details.

OK

Рисунок Ж.9 – Ошибка при удалении активного клиента

ControlDesk - admin

ControlDesk
Заказы и исполнение
Заказы
Клиенты
Сотрудники
Отчеты
Текущий пользователь: Кузнецов Денис, Администратор

Регистрация, обработка и контроль заказов

Обновить + Новый заказ

Всего заказов: 4

В работе: 1

Просрочено: 1

Сумма заказов: 418,000 руб.

Параметры отчета
Дата с: 22.05.2026, Дата по: 22.06.2026
Статус: Все, Только просроченные

Отбор
Клиент: АО Север Логистик
Ответственный:

Действия
Сводная таблица: По клиентам
Сформировать, Сбросить, Сохранить CSV
Строк: 2, Сводка: 1, Сумма: 84,000.00 руб.

Колонки CSV
[] Номер, [x] Клиент, [x] Ответственный, [x] Статус, [x] Приоритет, [x] Дата создания, [x] Срок, [x] Сумма, [] Описание

Сводная таблица По клиентам

Группа	Всего	В работе	Выполнено	Просрочено	Сумма	Средний чек	Последний заказ
АО Север Логистик	2	0	1	0	84,000.00 руб.	42,000.00 руб.	22.06.2026

№ заказа	Клиент	Ответственный	Статус	Приоритет	Дата создания	Срок	Сумма
ORD-2026-06220115	АО Север Логистик	Иванова Мария(ред.)	Отменен	Средний	22.06.2026	25.06.2026	0.00 руб.
ORD-2026-003	АО Север Логистик	Петров Алексей	Выполнен	Низкий	26.05.2026	02.06.2026	84,000.00 руб.

Рисунок Ж.10 – Сформирован корректный отчет

Колонки CSV

☒ Номер

☒ Клиент

☒ Ответственный

☐ Статус

☒ Приоритет

☒ Дата создания

☒ Срок

☒ Сумма

☐ Описание

Сводная таблицаПо клиентам

Группа	Всего	В работе	Выполнено	Просрочено	Сумма	Средний чек	Последний заказ
АО Север Логистик	2	0	1	0	84,000.00 руб.	42,000.00 руб.	22.06.2026

№ заказа	Клиент	Ответственный	Приоритет	Дата создания	Срок	Сумма
ORD-2026-06220115	АО Север Логистик	Иванова Мария(ред.)	Средний	22.06.2026	25.06.2026	0.00 руб.
ORD-2026-003	АО Север Логистик	Петров Алексей	Низкий	26.05.2026	02.06.2026	84,000.00 руб.

Рисунок Ж.11 – Сформирован отчет только по выбранным столбцам

orders-report.csv - Excel

Поиск

Общий доступ

Главная

Вставка

Разметка страницы

Формулы

Данные

Рецензирование

Вид

Справка

Acrobat

Вставить

Буфер обмена

Шрифт

Выравнивание

Число

Стили

Условное форматирование

Форматировать как таблицу

Стили ячеек

Ячейки

Редактирование

Надстройки

H12

fx

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Номер	Клиент	Ответствен	Приоритет	Дата созда	Срок	Сумма								
2	ORD-2026-	АО Север	/ Иванова М	Средний	22.06.2026	25.06.2026	0								
3	ORD-2026-	АО Север	/ Петров Ал	Низкий	26.05.2026	02.06.2026	84 000,00								
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															
24															
25															

orders-report

Готово

Специальные возможности: все в порядке

100 %

Рисунок Ж.12 – Корректный csv- файл