**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc184825135)

[1. ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ 6](#_Toc184825136)

[1.1 Анализ предметной области 6](#_Toc184825137)

[1.2 Выбор средств разработки 7](#_Toc184825138)

[1.2.1 Язык программирования 7](#_Toc184825139)

[1.2.2 Система управления базами данных 10](#_Toc184825140)

[1.2.3 Среда разработки 12](#_Toc184825141)

[1.3 Описание предметной области 16](#_Toc184825142)

[2. ПРАКТИЧЕСКИЙ РАЗДЕЛ 18](#_Toc184825143)

[2.1 Разработка базы данных 18](#_Toc184825144)

[2.2 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИ 20](#_Toc184825145)

[**2.2.1 Разработка формы главного окна** 20](#_Toc184825146)

[**2.2.2 Разработка формы авторизации** 21](#_Toc184825147)

[**2.2.3 Разработка формы регистрации** 21](#_Toc184825148)

[**2.2.4 Разработка формы личного кабинета** 22](#_Toc184825149)

[**2.2.5 Разработка формы управления для администратора** 23](#_Toc184825150)

[**2.2.6 Разработка формы Отчетов** 24](#_Toc184825151)

[**2.2.7 Разработка формы каталога услуг** 24](#_Toc184825152)

[**2.2.8 Разработка формы товара и оформления заказа** 25](#_Toc184825153)

[**2.2.9 Разработка формы управления для менеджера** 26](#_Toc184825154)

[Заключение 28](#_Toc184825155)

[Список источников 30](#_Toc184825156)

# ВВЕДЕНИЕ

Актуальность разработки программного обеспечения для предоставления строительных услуг обусловлена ростом спроса на автоматизацию бизнес-процессов в строительной отрасли и необходимостью эффективного управления проектами, ресурсами и заказами. Строительные проекты часто требуют координации большого количества участников и ресурсов, что делает важным внедрение современных технологий для учета и контроля процессов строительства.

Предмет исследования – разработка и внедрение автоматизированной информационной системы для управления строительными проектами и предоставлением строительных услуг.

Цель курсовой работы – разработка программного обеспечения, которое позволит автоматизировать управление строительными услугами, обеспечивая эффективное хранение информации и упрощение работы с клиентами и проектами.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Проанализировать предметную область и существующие программные решения для управления строительными проектами и услугами.
2. Определить функциональные и нефункциональные требования к программному обеспечению.
3. Спроектировать архитектуру системы и структуру базы данных.
4. Реализовать основные модули приложения, включая управление проектами, клиентами, сотрудниками и ресурсами.

# 1. ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

## 1.1 Анализ предметной области

Разработка программного обеспечения по предоставлению строительных услуг представляет собой организацию, которая специализируется на выполнении широкого спектра строительных работ для частных лиц и компаний. Основной задачей предприятия является обеспечение эффективного управления проектами строительства, своевременное выполнение заказов и поддержание актуальной информации о клиентах, услугах и задействованных ресурсах.

Разрабатываемое программное обеспечение должно автоматизировать ключевые процессы предприятия, включая:

* управление проектами строительства;
* оформление заказов на предоставление услуг;
* отслеживание выполнения работ;
* взаимодействие с клиентами;

Предприятие предоставляет несколько основных услуг:

* строительство: клиенты могут заказать строительные услуги, такие как возведение зданий, монтаж инженерных систем и другие строительные работы;
* проектирование и планирование: услуги по разработке проектов, планированию строительства, расчету стоимости материалов и трудозатрат.

Целевая аудитория:

* менеджеры проекта: сотрудники, отвечающие за управление строительными проектами, которые будут использовать программное обеспечение для планирования работ, управления ресурсами, составления графиков и контроля выполнения задач;
* клиенты: физические лица или компании, заинтересованные в заказе строительных услуг. Они будут использовать систему для выбора услуг, оформления заказов и получения информации о ходе выполнения работ;
* рабочие бригады: сотрудники, непосредственно выполняющие строительные работы, включая прорабов, начальников участков и рабочих. Они будут использовать систему для получения информации о назначенных задачах и сроках выполнения;
* администраторы: сотрудники, управляющие базой данных заказов, услуг и ресурсов, которые будут вести учет и обновление информации.

Таким образом, разрабатываемое программное обеспечение должно обеспечить прозрачное и эффективное взаимодействие между всеми категориями пользователей, предоставляя удобные и интуитивно понятные инструменты для выполнения их задач.

## 1.2 Выбор средств разработки

При разработке программного обеспечения для предоставления строительных услуг необходимо выбрать инструменты, которые обеспечат эффективное выполнение проекта. В этом разделе будут рассмотрены основные компоненты разработки, включая язык программирования, операционную среду, систему управления базами данных и среду разработки.

### 1.2.1 Язык программирования

Язык программирования — формальный язык, предназначенный для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (обычно — ЭВМ) под её управлением.

Выбор языка программирования играет основоположную роль в проектировании и реализации ПО. Рассмотрим два наиболее подходящих для данного проекта языка:

1. Python:

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нем программ.

Рассмотрим преимущества и недостатки данного языка программирования

Преимущества:

1. Простота и читабельность: Python имеет понятный синтаксис, близкий к естественному языку, что делает его идеальным для новичков и способствует быстрой разработке.
2. Большая стандартная библиотека: Python поставляется с обширной библиотекой модулей, что упрощает выполнение различных задач, таких как работа с файлами, обработка данных и сетевое программирование.

Недостатки:

1. Медленное выполнение: по сравнению с компилируемыми языками, Python работает медленнее из-за интерпретируемого характера.
2. Высокое потребление памяти: Python может требовать больше оперативной памяти, особенно при работе с большими объемами данных или в высоконагруженных системах.
3. C#:

С# – объектно-ориентированный язык программирования общего назначения. Разработанный для платформы .NET, подходящий для разработки приложений на Windows. С его помощью можно быстро создавать приложения с богатым пользовательским интерфейсом, а также серверные решения.

Рассмотрим преимущества и недостатки данного языка программирования

Преимущества:

1. Интеграция с .NET: C# отлично интегрирован с платформой .NET, что предоставляет доступ к большому количеству библиотек, фреймворков и инструментов для разработки различных типов приложений, включая веб-приложения, десктопные программы и мобильные приложения.
2. Строгость типизации: C# — это статически типизированный язык, что помогает обнаруживать ошибки на этапе компиляции, улучшая стабильность и безопасность программ.

Недостатки:

1. Зависимость от платформы Windows: хотя с появлением .NET Core и .NET 5/6 C# стал кроссплатформенным, многие его функции исторически были сильно завязаны на экосистему Windows, что все еще может создавать сложности при разработке под другие ОС.
2. Большие ресурсоемкие приложения: программы на C# могут потреблять больше ресурсов, особенно в сравнении с языками низкого уровня (например, C++), что может быть проблемой для высокопроизводительных систем.

В результате анализа языков программирования для разработки программного обеспечения для предоставления строительных услуг было принято решение выбрать C#. Этот язык был выбран по следующим причинам:

* интеграция с .NET и поддержка Windows: C# отлично подходит для разработки бизнес-приложений, особенно тех, которые должны работать на Windows. Для программного обеспечения для предоставления строительных услуг, где, возможно, используются различные системы для управления заказами, учета клиентов и финансами, возможности платформы .NET предоставляют гибкость и надежность;
* объектно-ориентированный подход: для управления данными о строительных услугах C# предоставляет мощные инструменты объектно-ориентированного программирования, что позволяет легко моделировать реальные объекты, такие как строительные заказы и взаимодействия с клиентами;
* поддержка сложных бизнес-логик: C# позволяет создавать сложные системы с применением строгой типизации и расширенных возможностей управления данными. Это упрощает разработку надежного программного обеспечения с минимальным количеством ошибок, что особенно важно в системах управления заказами, складами и учетными системами, которые могут использоваться в программном обеспечении для предоставления строительных услуг;
* гибкие UI-технологии: C# поддерживает разработку пользовательских интерфейсов через Windows Forms, WPF или веб-приложения через ASP.NET, что позволяет создать удобное и функциональное программное обеспечение для сотрудников и клиентов.

Таким образом, выбор C# в сочетании с Windows обеспечивает максимальную продуктивность разработки и позволяет эффективно решать задачи, связанные с созданием надежного и функционального программного обеспечения для данного проекта.

### 1.2.2 Система управления базами данных

Система управления базами данных (СУБД) — это комплекс программно-языковых средств, позволяющих создать базы данных и управлять данными. СУБД обеспечивает пользователям и приложениям интерфейс для работы с данными, позволяя выполнять операции по их хранению, извлечению, обновлению и удалению.

Для хранения данных необходимо выбрать надежную систему управления базами данных. Рассмотрим два наиболее подходящих варианта для данного проекта:

1. PostgreSQL

PostgreSQL — это мощная объектно-реляционная система управления базами данных с открытым исходным кодом. Она известна своей гибкостью, надежностью и поддержкой стандартов SQL. PostgreSQL широко используется в приложениях, требующих сложной обработки данных, высокой производительности и масштабируемости.

Рассмотрим преимущества и недостатки данной СУБД

Преимущества:

1. Открытый исходный код: PostgreSQL является свободной и открытой СУБД, что позволяет пользователям свободно изменять и распространять её код в соответствии с лицензией.
2. Расширенные возможности: PostgreSQL поддерживает широкий спектр функций, включая сложные запросы, индексы, триггеры, хранимые процедуры и многое другое, что делает его мощным инструментом для различных задач.
3. Масштабируемость: СУБД хорошо масштабируется и способна обрабатывать большие объемы данных, что делает её подходящей для проектов с различными масштабами и потребностями.

Недостатки:

1. Высокая цена: PostgreSQL может требовать дополнительных ресурсов и инвестиций в оборудование для обеспечения высокой производительности, что может повлечь за собой значительные затраты для предприятий.
2. Высокое потребление оперативной памяти: при работе с крупными объемами данных PostgreSQL может потреблять значительное количество оперативной памяти, что может стать проблемой в случае ограниченных ресурсов сервера.
3. Microsoft SQL Server — это реляционная система управления базами данных, разработанная компанией Microsoft. SQL Server используется для хранения и управления данными, поддерживает стандартный язык запросов SQL и широко применяется в корпоративных и бизнес-системах. Эта СУБД предоставляет множество инструментов для обработки, анализа и визуализации данных.

Рассмотрим преимущества и недостатки данной СУБД

Преимущества:

1. Интеграция с продуктами Microsoft: SQL Server отлично взаимодействует с другими продуктами Microsoft, такими как Windows Server, Azure, Visual Studio и Power BI. Это делает его естественным выбором для компаний, использующих Microsoft-технологии.
2. Высокая производительность и масштабируемость: SQL Server поддерживает работу с большими объемами данных и обеспечивает высокую производительность для корпоративных приложений. Он оптимизирован для сложных транзакций и аналитических запросов, что делает его подходящим для масштабируемых решений.
3. Инструменты бизнес-аналитики: встроенные инструменты, такие как SQL Server Reporting Services (SSRS), Integration Services (SSIS) и Analysis Services (SSAS), делают SQL Server мощным инструментом для анализа данных и отчетности.

Недостатки:

1. Высокая стоимость: коммерческая версия SQL Server может быть дорогой, особенно для крупных предприятий, которым требуется лицензирование на основе количества ядер процессора. Хотя существуют бесплатные версии (SQL Server Express), они ограничены в функциональности и масштабируемости.
2. Зависимость от Windows: с появлением SQL Server для Linux зависимость от Windows уменьшилась, все же большинство функций и поддержка тесно интегрированы с Windows, что ограничивает гибкость при использовании других операционных систем.
3. Высокие системные требования: SQL Server требует значительных ресурсов для обеспечения высокой производительности, особенно при работе с большими базами данных и высокими нагрузками. Это может стать проблемой для небольших компаний или проектов с ограниченным бюджетом на инфраструктуру.

Таким образом, выбор Microsoft SQL Server в качестве системы управления базами данных обеспечивает необходимую производительность, надежность и гибкость для успешной реализации проекта, направленного на создание эффективного программного обеспечения для предоставления строительных услуг.

### 1.2.3 Среда разработки

**Среда разработки (Integrated Development Environment, IDE)** — это программное обеспечение, предназначенное для облегчения процесса разработки программного обеспечения. IDE объединяет в себе набор инструментов, необходимых для написания, редактирования, компиляции, отладки и тестирования кода, что делает процесс разработки более удобным и эффективным.

Выбор среды разработки (IDE) важен для удобства и эффективности работы над проектом. Рассмотрим две популярные среды:

1. Project Rider

JetBrains Rider — это мощная интегрированная среда разработки (IDE) от компании JetBrains, предназначенная для разработки на языке C# и других языках, поддерживаемых платформой .NET. Rider объединяет в себе возможности ReSharper и платформы IntelliJ IDEA, предоставляя удобный интерфейс и богатый функционал для разработки приложений на различных технологиях .NET, включая .NET Core, .NET Framework, ASP.NET, Xamarin и Unity.

Преимущества:

1. Поддержка множества платформ: Rider поддерживает разработку для .NET Framework, .NET Core, Mono и Unity, что делает его универсальным инструментом для разработки как настольных, так и мобильных приложений, игр и веб-приложений.
2. Быстрая и легкая IDE: Rider, как правило, работает быстрее и использует меньше ресурсов по сравнению с Visual Studio, что делает его подходящим выбором для разработчиков с более ограниченными системными ресурсами.
3. Встроенные инструменты рефакторинга и анализа кода: Rider использует возможности ReSharper, предоставляя мощные инструменты для улучшения качества кода, автоматического рефакторинга, быстрого поиска и исправления ошибок. Это помогает повысить продуктивность разработчика.

Недостатки:

1. Коммерческая лицензия: Rider является платным продуктом, существует бесплатный пробный период, его стоимость может быть значительным фактором для индивидуальных разработчиков или малых компаний.
2. Проблемы с интеграцией: в некоторых случаях Rider может иметь проблемы с интеграцией с определенными инструментами или библиотеками, что может потребовать дополнительных усилий для настройки и исправления ошибок.
3. Зависимость от ресурсов: несмотря на то, что Rider оптимизирован для производительности, он все же требует значительное количество ресурсов, особенно при работе с большими проектами, что может быть проблемой на менее мощных машинах.
4. Visual Studio

Visual Studio — это интегрированная среда разработки (IDE) от Microsoft, предназначенная для создания приложений на различных языках программирования, включая C#, VB.NET, C++, F# и другие. Она поддерживает разработку для платформ .NET, ASP.NET, Azure, а также создание настольных, веб- и мобильных приложений. Visual Studio предлагает множество инструментов и функций для упрощения процесса разработки, от редактирования кода до отладки и тестирования.

Преимущества:

1. Широкая функциональность: Visual Studio предлагает множество встроенных инструментов для разработки, включая редактор кода, отладчик, профайлер, средства для работы с базами данных и интеграцию с системами контроля версий.
2. Интеграция с другими продуктами Microsoft: IDE прекрасно интегрируется с другими решениями Microsoft, такими как Azure, Office и Power BI, что упрощает разработку и развертывание облачных приложений.
3. Поддержка различных языков программирования: Visual Studio поддерживает множество языков, включая C#, VB.NET, C++, F#, JavaScript и Python, что делает её универсальным инструментом для разработки.

Недостатки:

1. Сложность освоения для новичков: из-за обширного функционала и большого количества возможностей, новичкам может потребоваться время для освоения всех аспектов среды.
2. Проблемы с совместимостью: иногда возникают проблемы с совместимостью между различными версиями Visual Studio и проектами, особенно при использовании устаревших технологий.
3. Частые обновления: хотя обновления обычно приносят улучшения, иногда они могут вызывать нестабильность или проблемы с совместимостью, требуя дополнительного времени для исправления.

В результате анализа интегрированных сред разработки для создания программного обеспечения подходит Visual Studio для проекта по предоставлению строительных услуг.

Кроме того, встроенные средства для отладки и тестирования помогают выявлять и устранять ошибки на ранних этапах разработки. Это гарантирует высокое качество конечного продукта, что особенно важно в строительной отрасли, где ошибки могут привести к значительным затратам и задержкам.

Интуитивно понятный интерфейс делает процесс разработки более удобным и эффективным, что будет полезно для команд, работающих над проектом. Также стоит отметить возможность расширяемости: Visual Studio поддерживает множество расширений и библиотек, что позволяет легко интегрировать дополнительные функции и инструменты для управления проектами и ресурсами в строительстве.

Широкое сообщество разработчиков и обширная документация обеспечивают поддержку и ресурсы, которые могут быть полезны при решении проблем или внедрении новых функций. Наконец, возможность работы с системами контроля версий упрощает совместную работу нескольких разработчиков над проектом, что делает процесс разработки более организованным.

Таким образом, использование Visual Studio в проекте по предоставлению строительных услуг позволит создать эффективное, надежное и масштабируемое программное обеспечение, соответствующее требованиям вашей компании и клиентов.

## 1.3 Описание предметной области

Предметная область данной курсовой работы охватывает управление магазином по продаже строительных услуг. Основная цель системы — обеспечить эффективное взаимодействие между клиентами и услугами, предлагаемыми магазином, а также управлять внутренними процессами, связанными с предоставлением строительных услуг.

1. Пользователи системы

Система строительных услуг предполагает наличие различных пользователей, каждый из которых имеет свою роль и уровень доступа:

* администраторы: ответственные за управление пользователями. Администраторы имеют полный доступ к функционалу системы, включая управление проектами, бригадами, ресурсами, а также аналитическими отчетами и финансовыми данными;
* менеджеры: работают непосредственно с клиентами, оформляют их заказы и управляют их выполнением. Менеджеры имеют доступ к информации о клиентах, проектах и рабочих бригадах. Они контролируют процесс выполнения строительных работ и координируют работу между клиентом и бригадами;
* клиенты: могут просматривать каталог строительных услуг, заказывать необходимые работы. Клиенты имеют ограниченный доступ, позволяющий им управлять своими заказами и просматривать их статус;

1. Услуги по проекту

Магазин по предоставлению строительных услуг предлагает широкий спектр строительных и ремонтных услуг. Услуги делятся на категории в зависимости от типа работ, что упрощает выбор для клиентов.

Список услуг может включать:

* проектирование: разработка чертежей и планов для строительства и ремонта зданий;
* строительные работы: выполнение строительных работ, включая возведение зданий, ремонт помещений, реконструкцию объектов;
* ремонтные работы: восстановление жилых и коммерческих помещений, ремонт фасадов, крыши и других частей здания;
* ландшафтные работы: услуги по благоустройству территорий, создание садов и парков.

Каждая услуга имеет детальное описание, цену и сроки выполнения.

1. Заказы и статусы

Клиенты могут оформлять заказы на строительные услуги. Каждый заказ включает в себя название проекта, услуга, стоимость и сроки выполнения. Заказы также связаны с конкретным клиентом и ответственным менеджером.

Заказы могут иметь различные статусы, такие как:

* «Создан»;
* «В обработке»;
* «Назначена бригада»;
* «Выполняется»;
* «Завершен»;
* «Отменен».

Эти статусы помогают отслеживать текущее состояние заказа и обеспечивают эффективное управление процессом выполнения работ.

1. Цели и задачи системы

Основные цели системы по предоставлению строительных услуг:

* обеспечение удобного интерфейса для пользователей с возможностью быстрого доступа к информации о строительных услугах;
* автоматизация процессов оформления заказов, управления проектами и назначением бригад;
* обеспечение актуальности информации о предоставляемых услугах и сроках выполнения работ.

Таким образом, данная система используется для оптимизирования управления строительными проектами, улучшить взаимодействие с клиентами и повысить качество услуг, предоставляемых компанией.

# 2. ПРАКТИЧЕСКИЙ РАЗДЕЛ

## Разработка базы данных

Для разработки базы данных в рамках курсовой работы была использована реляционная модель хранения данных с использованием системы управления базами данных Microsoft SQL Server. В процессе проектирования базы данных были определены основные сущности, которые являются ключевыми для функционирования системы. Каждая сущность была реализована в виде таблицы, с заданием соответствующих полей и связей между ними.

База данных включает следующие основные таблицы:

* Role — таблица ролей, которая определяет типы пользователей (менеджер, администратор). Содержит поля: ID\_Role (идентификатор роли), roll\_name (наименование роли), salary (заработная плата сотрудника). SQL – скрипт для создания данной таблицы представлен в Приложении 1 на рисунке 2.1.
* Users — таблица пользователей системы. Содержит поля: ID\_Users (идентификатор пользователя), Email (электронная почта), Password (пароль), Number\_Phone (номер телефона), Address (адрес), First\_name (Имя), Last\_name (фамилия), Patronymic (Отчество), Passport\_details (паспортные данные). SQL – скрипт для создания данной таблицы представлен в Приложении 1 на рисунке 2.2.
* Status\_Orders — таблица статусов заказов. Содержит поля: ID\_ Status\_Orders (идентификатор статуса), Name\_Status (наименование статуса: "в обработке", "доставлен" и т.д.). SQL – скрипт для создания данной таблицы представлен в Приложении 1 на рисунке 2.3.
* Staff – таблица сотрудников. Содержит поля: ID\_Staff (идентификатор сотрудника), ID\_Role (идентификатор роли сотрудника, который связывает сотрудника с его правами в системе), Email (электронная почта), Password (пароль), Image (фото сотрудника), First\_name (Имя), Last\_name (фамилия), Patronymic (Отчество), Passport\_details (паспортные данные), Date\_birth (дата рождения), Date\_employment (дата устройства). SQL – скрипт для создания данной таблицы представлен в Приложении 1 на рисунке 2.4.
* Foremen — таблица начальники участков, которые предлагаются в системе. Содержит поля: ID\_Foreman (идентификатор начальника участка), First\_name (Имя), Last\_name (фамилия), Patronymic (Отчество), (ссылка на модель), Number\_of\_Workers (количество строителей), Number\_phone (номер телефона). SQL – скрипт для создания данной таблицы представлен в Приложении 1 на рисунке 2.5.
* Services – таблица услуг, которые предлагаются в системе. Содержит поля: ID\_Service (идентификатор услуги), Item\_Name (название), Item\_Description (описание), Price (цена), Image (фото). SQL – скрипт для создания данной таблицы представлен в Приложении 1 на рисунке 2.6.
* Requests — таблица заявок, написанных пользователями. Содержит поля: ID\_Requests (идентификатор заявки), ID\_Service (ссылка на услугу, выбранный в заявке), First\_name (Имя), Last\_name (Фамилия), Email (электронная почта), Additional\_Info (вопрос по заявки), Status (статус заявки). SQL – скрипт для создания данной таблицы представлен в Приложении 1 на рисунке 2.7.
* Customer\_orders – таблица заказов пользователей. Содержит поля: ID\_ Customer\_orders (идентификатор заказа), ID\_Service (ссылка на услугу), ID\_Userы (ссылка на пользователя), Order\_Date (дата заказа), Status (статус заказа). SQL – скрипт для создания данной таблицы представлен в Приложении 1 на рисунке 2.8.
* Processed\_customer\_orders – таблица обработанных заказов пользователей. Содержит поля: ID\_ Processed\_сustomer\_orders (идентификатор заказа), ID\_Customer\_orders (ссылка на заказ), ID\_Staff (ссылка на сотрудника), ID\_Foreman (ссылка на начальников участка), ID\_Status\_Orders (ссылка на статус заказа), Date\_Start (дата начала), Date\_Ending (дата окончания), Final\_sum (окончательная цена заказа). SQL – скрипт для создания данной таблицы представлен в Приложении 1 на рисунке 2.9.

Логическая модель данных системы представлена на рисунке 2.10 в Приложении 1. Диаграмма отображает основные таблицы базы данных, их ключевые поля, а также связи между ними через первичные и внешние ключи.

Данная структура базы данных позволяет эффективно хранить и обрабатывать данные о пользователях, сотрудниках, заказах, статусах и т.д. В дальнейшем, в процессе разработки программного обеспечения, будет реализован доступ к этим данным через объектную модель с использованием **Entity Framework**, что упростит взаимодействие с базой данных.

## 2.2 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИ

### **2.2.1 Разработка формы главного окна**

Главная форма приложения предоставляет пользователю доступ к ключевым разделам системы и формирует первое визуальное впечатление.

Внешний вид интерфейса представлен в Приложении 2 на рисунке 2.11. Элементы главного окна включают:

* Навигационное меню для быстрого перехода между разделами;
* Почта пользователя (отображается после авторизации).

Обработчик событий для перехода к разделам «Услуги», «О нас», «Контакты», «Регистрация», «Авторизация» реализован методом MouseDown и ButtonClick (код представлен на рисунке 2.12, Приложение 2).

### **2.2.2 Разработка формы авторизации**

Форма авторизации предназначено для входа в систему или создания новой учетной записи. Его интерфейс (Приложение 2, рисунок 2.13) включает:

* Поля для ввода логина и пароля;
* Кнопки «Войти в систему» и «Создать аккаунт».

Реализация функциональности формы авторизации включает:

1. Логика входа в систему. Код проверки введённых данных и определения роли пользователя представлен на рисунке 2.14 (Приложение 2).
2. Проверка данных через базу данных. Реализация на основе Entity Framework представлена в коде на рисунке 2.14 (Приложение 2).

### **2.2.3 Разработка формы регистрации**

Форма «Registration» предназначено для создания новой учетной записи пользователя и предоставляет удобный интерфейс для ввода данных.

Интерфейс окна регистрации (см. Приложение 2, рисунок 2.15) включает:

* Текстовые поля для ввода почты, пароля и его подтверждения;
* Кнопка «Обработку персональных данных», открывает договор о обработке персональных данных копании для ознакомления пользователю;
* Кнопка подтверждения «Обработки персональных данных», дает право пользователю продолжить регистрацию;
* Кнопку "Зарегистрироваться", завершает процесс регистрации.

Реализация функциональности формы регистрации включает:

1. Проверка подтверждения на обработку персональных данных и введённых данных на корректность. Логика проверки реализована в коде, представленном на рисунке 2.16 (Приложение 2).
2. Пользователю на почту отправляется 6-шестезный код для подтверждения его данных. Логика проверки реализована в коде, представленном на рисунке 2.17 (Приложение 2).
3. Взаимодействие с базой данных. Добавление нового пользователя в базу данных осуществляется через Entity Framework, функция RegisterButton\_Click. Код этой функции приведён на рисунке 2.16 (Приложение 2).
4. Договор об обработке персональных данных. Логика открытия реализована в коде, представленном на рисунке 2.18 (Приложение 2).

Данная форма обеспечивает безопасную и удобную регистрацию новых пользователей, включая валидацию данных и интеграцию с базой данных.

### **2.2.4 Разработка формы личного кабинета**

Форма «PersonalAccount» позволяет пользователю управлять учетной записью, редактировать данные, просматривать историю заказов, а также сохранять изменения.

На рисунках 2.19, 2.20, 2.21 (Приложении 2) представлен интерфейс вкладки «Профиль», «Заказы», «История».

Основной функционал:

1. Загрузка данных пользователя: при открытии вкладки осуществляется загрузка данных текущего пользователя из базы данных. Код, реализующий эту функцию, представлен на рисунке 2.22 (Приложение 2).
2. Редактирование и обновления личных данных: Пользователь может редактировать, обновлять и добавлять имя, фамилию, отчество, номер телефона, почту, паспортные данные, адрес, пароль. Все изменения проверяются и сохраняются в базу данных. Код обработки редактирования данных показан на рисунке 2.23 (Приложение 2).
3. Загрузка заказов: при открытии вкладки «Заказы», «История» осуществляется выборка из базы данных всех заказов, связанных с текущим пользователем. Код загрузки истории заказов представлен на рисунке 2.24 (Приложение 2).

### **2.2.5 Разработка формы управления для администратора**

Форма «AdminPage» предназначено для управления данными в административной панели системы. Его интерфейс включает функции работы с данными из базы данных, просмотр отчетов, выход из аккаунта.

Интерфейс формы представлен на рисунке 2.26 (Приложение 2) и включает:

* 1. Кнопки управления:
* Добавить — открывает окно добавления новой услуги.
* Сохранить — сохраняет изменение в базе данных.
* Удалить — удаление записи из базы данных.
* Клиенты – отображается таблица «User».
* Сотрудники – отображается таблица «Staff».
* Услуги – отображается таблица «Service».
* Заказы – отображается таблица «Orders».
* Заявки – отображается таблица «Request».
* Роли – отображается таблица «Roll».
* Начальники участков – отображается таблица «Foramen».
* Статусы – отображается таблица «Orders».
* Отчеты – открывает форму с графиками и выгрузкой отчетов в Excel.
* Выйти из аккаунта — закрывает окно и возвращает в главное меню.

**Реализация функционала:**

1. **Добавление новой услуги**: Метод Button\_Click\_Add открывает окно добавления услуг. Реализация метода показана на рисунке 2.27 (Приложение 2).
2. Код для добавления записи в базу данных представлен на рисунках 2.28, 2.29, 2.30, 2.31, 2.32 (Приложение 2).
3. **Выйти из аккаунта**: Метод leave\_Click закрывает текущее окно и возвращает в главное меню. Код реализации показан на рисунке 2.33 (Приложение 2).
4. **Сохранить**: Метод Button\_Click\_Save сохраняет запись, которую редактировал администратор. Код реализации показан на рисунке 2.34 (Приложение 2).
5. **Удалить:** Метод DeleteUser\_Click удаляет запись, которую выбрал администратор. Код реализации показан на рисунке 2.35 (Приложение 2).
6. **Отчеты**: Метод Load\_Report\_Click открывает форму отчетов. Код реализации показан на рисунке 2.36 (Приложение 2).

### **2.2.6 Разработка формы Отчетов**

Форма «Reports» предназначено для составления отчетов в виде графиков и выгрузки таблиц из базы данных. Его интерфейс включает функции работы с данными из базы данных.

Интерфейс окна представлен на рисунке 2.37 (Приложение 2) и включает:

1. Кнопки управления:

* Excel — открывает окно добавления новой услуги.
* Назад — возвращает на форму AdminPage.

**Реализация функциональности:**

1. **Назад**: Метод bac\_Click возвращает администратора на AdminPage. Код реализации показан на рисунке 2.38 (Приложение 2).
2. **Excel**: Метод Button\_Click\_Excel использует библиотеку Microsoft.Office.Interop для генерации отчета с базы данных. Код представлен на рисунке 2.39 (Приложение 2).
3. **Графики**: Класс LoadChartData выводит информацию с базы данных в виде графика. Код представлен на рисунке 2.40 (Приложение 2).

### **2.2.7 Разработка формы** **каталога услуг**

Форма «Service» предоставляет пользователю возможность просматривать услуги, применять фильтрацию и сортировку. В интерфейсе также реализована универсальна карта с подробностью о товаре.

На рисунке 2.41 (Приложение 2) представлен интерфейс форма «Каталог услуг».

Основной функционал:

1. Загрузка данных о услугах: при открытии вкладки происходит загрузка всех доступных услуг из базы данных. Код, реализующий эту функцию, представлен на рисунке 2.42 (Приложение 2).
2. Фильтрация услуг: Пользователь может ввести поисковый запрос в текстовом поле и применить фильтрацию по названию услуг. Все товары, не соответствующие запросу, скрываются. Код реализации фильтрации показан на рисунке 2.43 (Приложение 2).
3. Переход на карту товара: Пользователь нажимает кнопку подробнее после чего открывается форма со всей информаций о услуги и возможности заказать его. Код для выборки товара из базы данных представлен на рисунке 2.44 (Приложение 2).

### **2.2.8 Разработка формы товара и оформления заказа**

**Форма «ServiceCard» позволяет пользователю подробно узнать информацию о услуги, а также оформлять заказы.**

На рисунке 2.45 (Приложение 2) представлен интерфейс формы «Карта товара».

**Основной функционал:**

1. **Загрузка товаров из базы данных:** при открытии формы «**ServiceCard»** осуществляется загрузка только выбранного товара, который нужен пользователь. Код, реализующий эту функцию, представлен на рисунке 2.46 (Приложение 2).
2. **Оформление заказа:** для оформления заказа пользователю нужно быть авторизованным. Код оформления заказа и проверки авторизации представлен на рисунке 2.47 (Приложение 2).
3. При успешном оформлении заказа пользователю отправляться электронный чек об успешном заказе, после чего заказ будет отправлен менеджеру для его обработки. Код оформления заказа и проверки авторизации представлен на рисунке 2.48 (Приложение 2).

### **2.2.9 Разработка формы управления для менеджера**

Форма «ManagerPage» предназначено для менеджера с помощью ее он может взять заказ, узнать всю информацию о заказе, изменить данные заказа. Его интерфейс включает функции работы с данными из базы данных, возможность взять новый заказ, просмотр информации о его заказах, выход из аккаунта.

Интерфейс формы представлен на рисунке 2.49 (Приложение 2) и включает:

* 1. Кнопки управления:
* Взять заказ — открывает окно добавления новой услуги;
* Заказы — показывает доступные заказы для обработки в базе данных;
* Мои заказы — показывает уже взятые заказы менеджера из базы данных;
* Сохранить заказ — сохраняет изменения, внесенные в заказ;
* Подробнее — открывает форму со всей информацией о заказе;
* Выйти из аккаунта — закрывает окно и возвращает в главное меню.
  1. **Реализация функционала:**
* **Взять заказ**: Метод TakeOrder\_Click открывает форму где менеджер вводит основную информацию о заказе. Реализация метода показана на рисунке 2.50 (Приложение 2).
* **Выйти из аккаунта**: Метод leave\_Click закрывает текущее окно и возвращает в главное меню. Код реализации показан на рисунке 2.51 (Приложение 2);
* **Мои заказы:** Метод MyOrders\_Click открывает форму уже обработанными зеками, в ней менеджер может редактировать заказ и узнать всю информацию о заказе. Код реализации показан на рисунке 2.52 (Приложение 2);
* **Сохранить заказ**: Метод SaveOrder\_Click сохраняет изменения, внесенные в заказ менеджером. Код реализации показан на рисунке 2.53 (Приложение 2);
* Подробнее: Метод Orderdetails\_Click открывает форму со всей информацией об этом заказе информация берётся с помощью выборки из базы данных. Код реализации показан на рисунке 2.39 (Приложение 2);

# Заключение

В рамках выполнения курсового проекта была достигнута цель — разработано программное обеспечение для автоматизации управления салоном винтажных автомобилей. В процессе работы выполнены все поставленные задачи, включая анализ предметной области, проектирование структуры базы данных, создание архитектуры приложения и реализация его функциональности.

В результате проведенной работы было создано приложение, которое обеспечивает эффективное взаимодействие пользователей с системой, улучшает процесс управления данными и оптимизирует ключевые бизнес-процессы салона. Основные возможности системы включают управление каталогом автомобилей, оформление заказов, ведение учета клиентов и сотрудников, а также генерацию отчетов. Реализован дружественный интерфейс, который упрощает использование программы для всех категорий пользователей: администраторов, менеджеров и клиентов.

Особое внимание уделено выбору инструментов разработки. Использование языка программирования C#, среды разработки Visual Studio и системы управления базами данных Microsoft SQL Server позволило создать надежное и производительное решение. Entity Framework обеспечил удобную работу с базой данных, минимизируя сложность обработки данных и ускоряя разработку.

Практическая значимость работы заключается в том, что разработанное программное обеспечение может быть использовано не только в салонах винтажных автомобилей, но и адаптировано для других бизнес-процессов, связанных с продажей и обслуживанием транспортных средств. Система обладает гибкостью и масштабируемостью, что позволяет вносить изменения и расширять функциональность в соответствии с изменяющимися потребностями бизнеса.

В ходе тестирования системы были подтверждены её функциональность и соответствие установленным требованиям. Программное обеспечение продемонстрировало высокую производительность и устойчивость при выполнении операций с данными. Предложенные функции, такие как управление заказами, обновление информации о товарах и анализ отзывов клиентов, делают систему удобным инструментом для оптимизации работы салона.

Дальнейшее развитие проекта может включать добавление мобильного приложения для клиентов, интеграцию с платежными системами и внедрение аналитических инструментов для прогнозирования продаж и улучшения качества услуг. Эти улучшения позволят повысить конкурентоспособность бизнеса и удовлетворить потребности целевой аудитории.

В заключение можно отметить, что выполненная работа отвечает требованиям профессионального модуля и демонстрирует применение теоретических знаний на практике. Проект достиг поставленных целей и является успешным примером разработки программного обеспечения для управления бизнес-процессами.

# Список источников

1. Бьяликов, И. А. Разработка программного обеспечения: учебное пособие / И. А. Бьяликов. — М.: Юрайт, 2020. — 412 с.
2. Скотт, А. Проектирование баз данных: принципы и практика / А. Скотт, М. Брайант. — СПб.: Питер, 2018. — 640 с.
3. Бут, Р. UML: руководство пользователя / Р. Бут, Г. Армстронг. — М.: Вильямс, 2017. — 496 с.
4. Троелсен, Э. Язык программирования C# 9.0 и платформы .NET 5.0 / Э. Троелсен, Ф. Джепикс. — М.: Вильямс, 2021. — 960 с.
5. Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений / Г. Буч. — СПб.: Питер, 2017. — 432 с.
6. Microsoft SQL Server Management Studio. Электронный ресурс. URL: <https://learn.microsoft.com/en-us/sql/ssms/> (дата обращения: 03.12.2024).
7. Visual Studio: Integrated Development Environment. Электронный ресурс. URL: <https://visualstudio.microsoft.com/> (дата обращения: 03.12.2024).
8. Metanit.com. Руководство по C# и .NET. Электронный ресурс. URL: https://metanit.com/sharp/ (дата обращения: 03.12.2024).
9. Entity Framework Documentation. Электронный ресурс. URL: <https://learn.microsoft.com/en-us/ef/> (дата обращения: 03.12.2024).
10. Общие рекомендации по проектированию интерфейсов. Электронный ресурс. URL: <https://uxdesign.cc/> (дата обращения: 03.12.2024).

**ПРИЛОЖЕНИЕ**

Приложение 1. Разработка базы данных

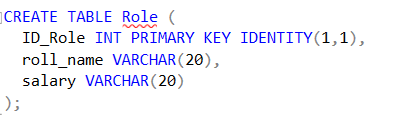


Рисунок 2.1 – SQL – скрипт создания таблицы Role.

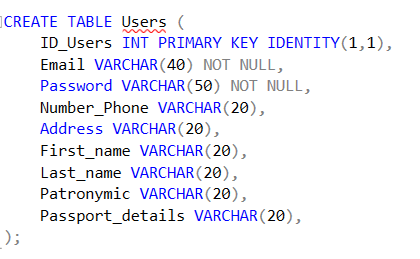


Рисунок 2.2 – SQL – скрипт создания таблицы Users.

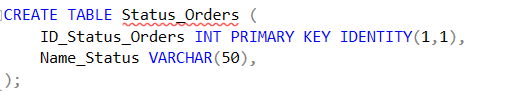


Рисунок 2.3 – SQL – скрипт создания таблицы Status\_Orders.

Продолжение Приложения 1

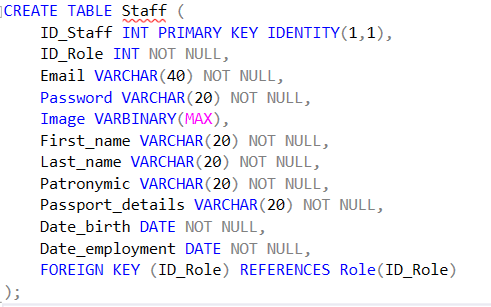


Рисунок 2.4 – SQL – скрипт создания таблицы Staff.

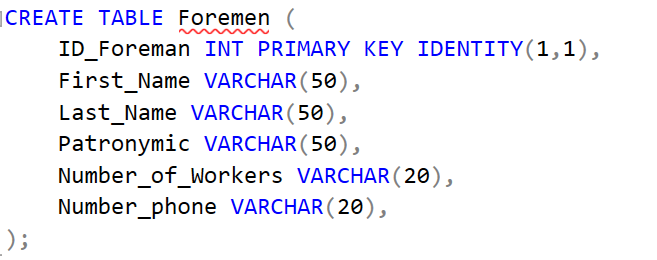


Рисунок 2.5 – SQL – скрипт создания таблицы Foremen.

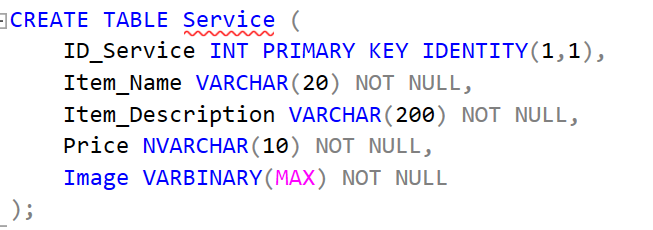


Рисунок 2.6 – SQL – скрипт создания таблицы Service

Продолжение Приложения 1

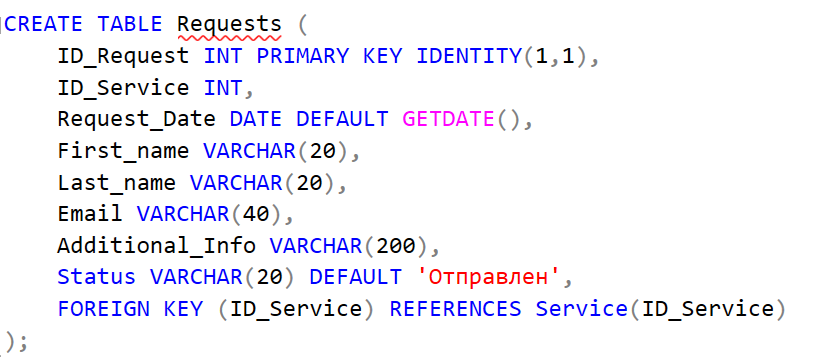


Рисунок 2.7 – SQL – скрипт создания таблицы Requests.

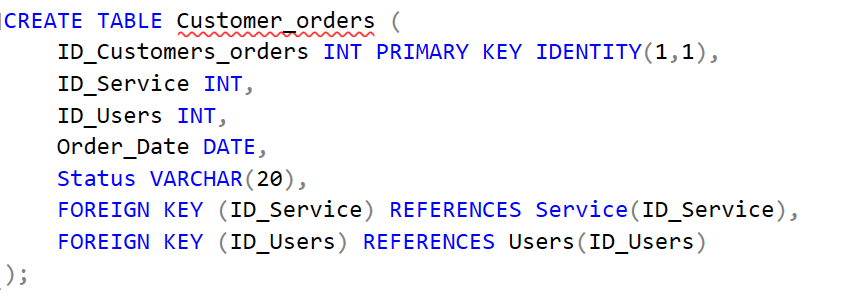


Рисунок 2.8 – SQL – скрипт создания таблицы Customer\_orders.

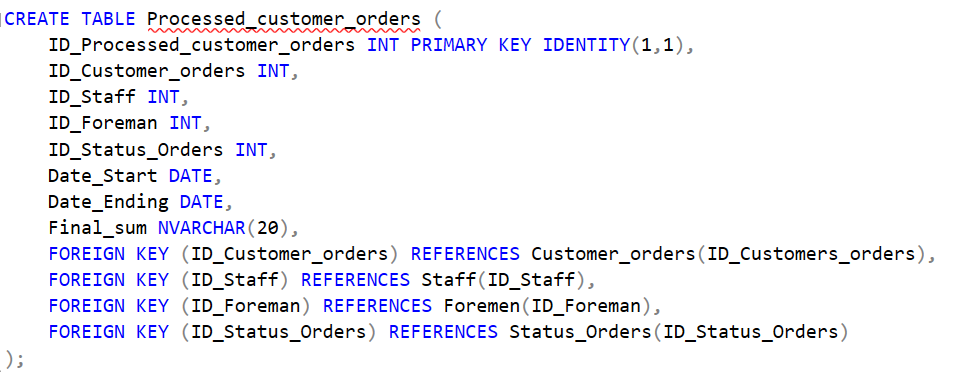


Рисунок 2.9 – SQL – скрипт создания таблицы Processed\_customer\_orders.

Продолжение Приложения 1

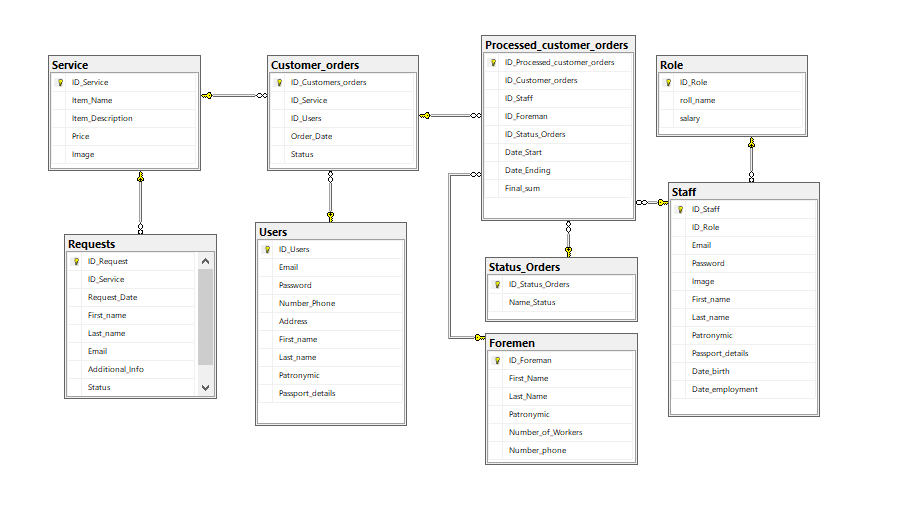


Рисунок 2.10 – Логическая модель данных.

Приложение 2. Разработка программного обеспечения

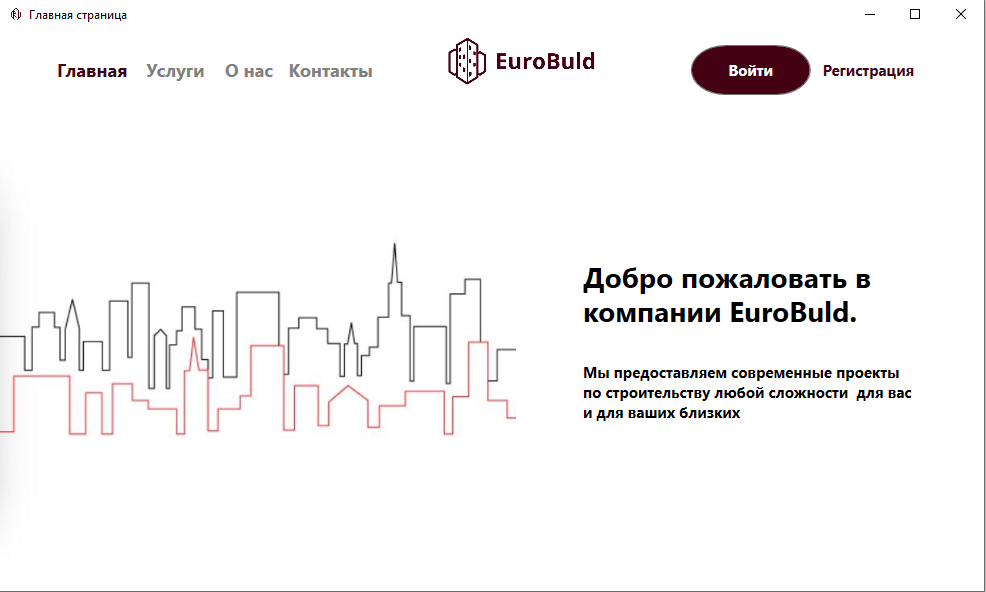


Рисунок 2.11 – интерфейс формы главного окна.

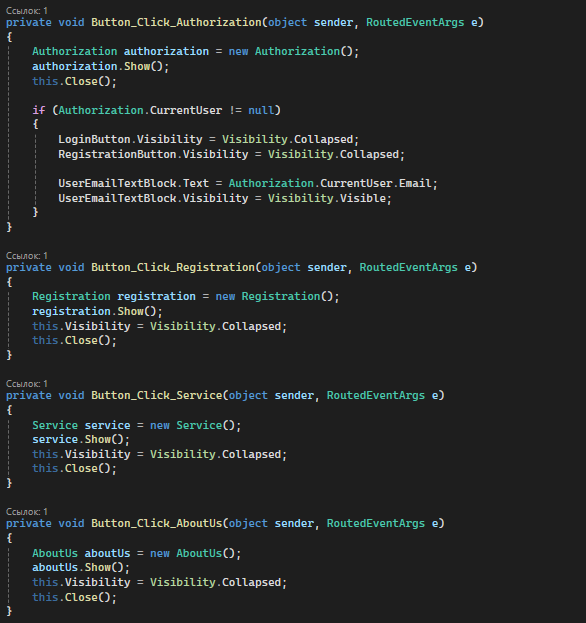


Рисунок 2.12 – код перехода к формам «Регистрация», «Авторизация», «Регистрация», «Услуги», «О нас», «Контакт».

Продолжение Приложения 2

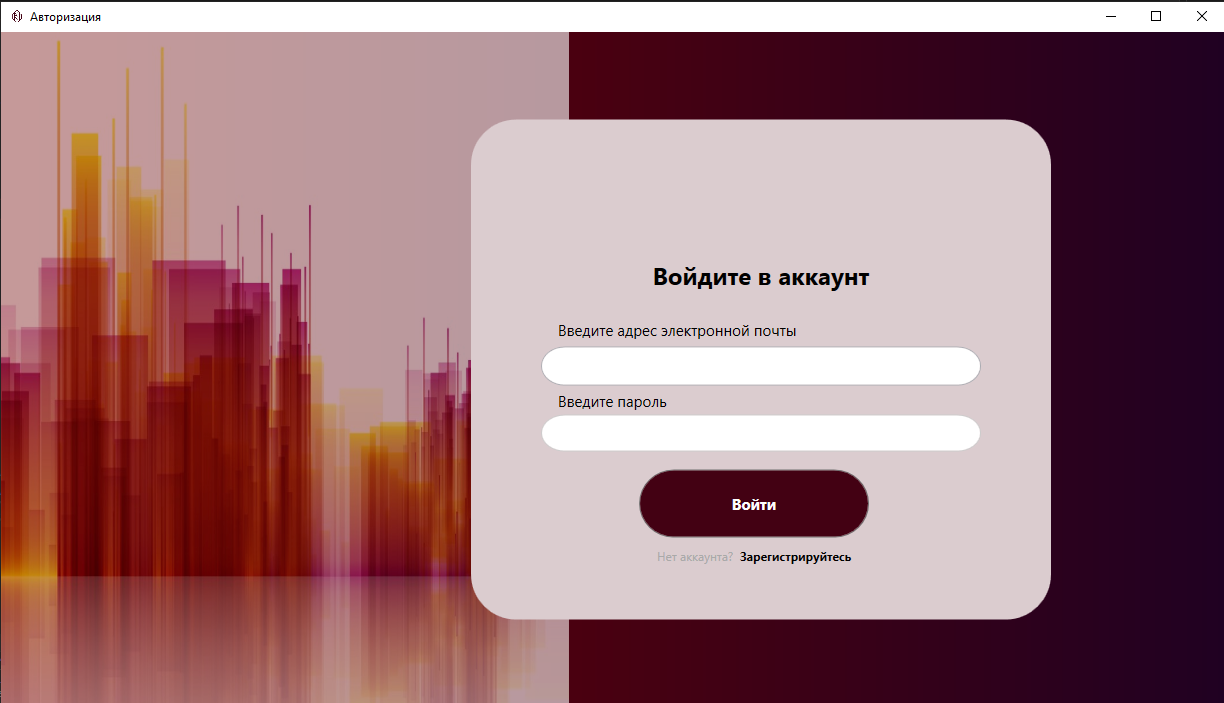
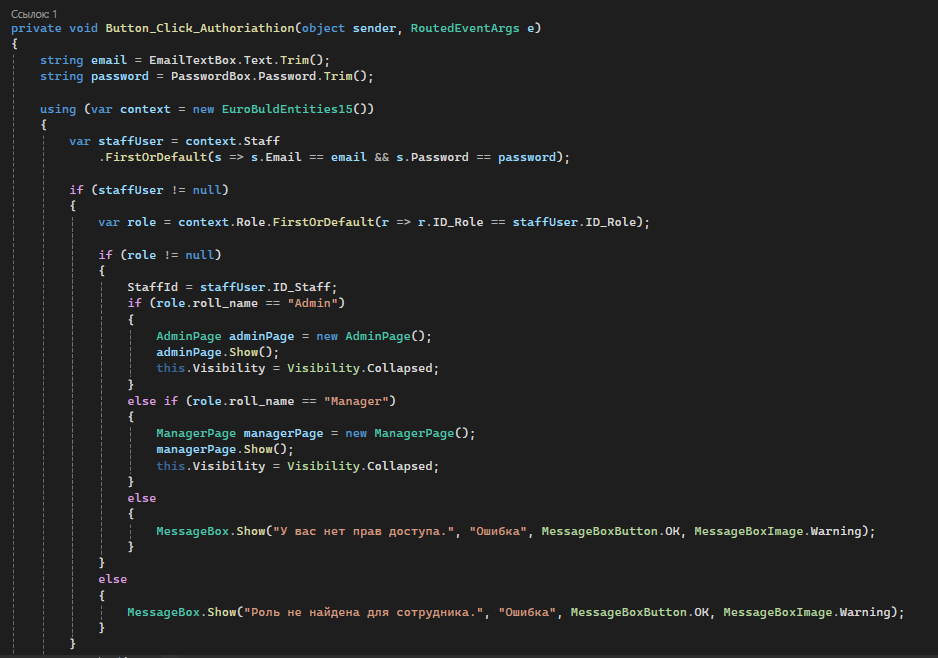


Рисунок 2.13 – интерфейс формы авторизации.



Продолжение Приложения 2

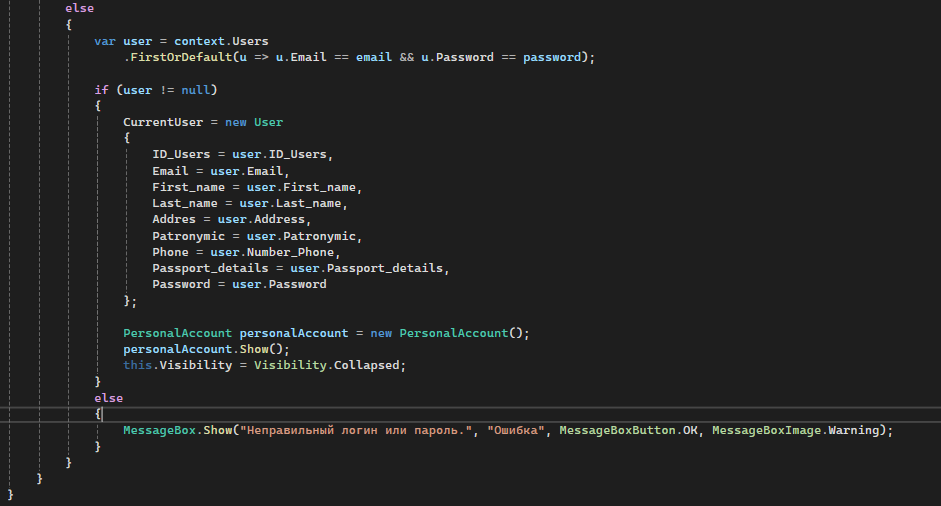


Рисунок 2.14 – код проверки введенных данных.

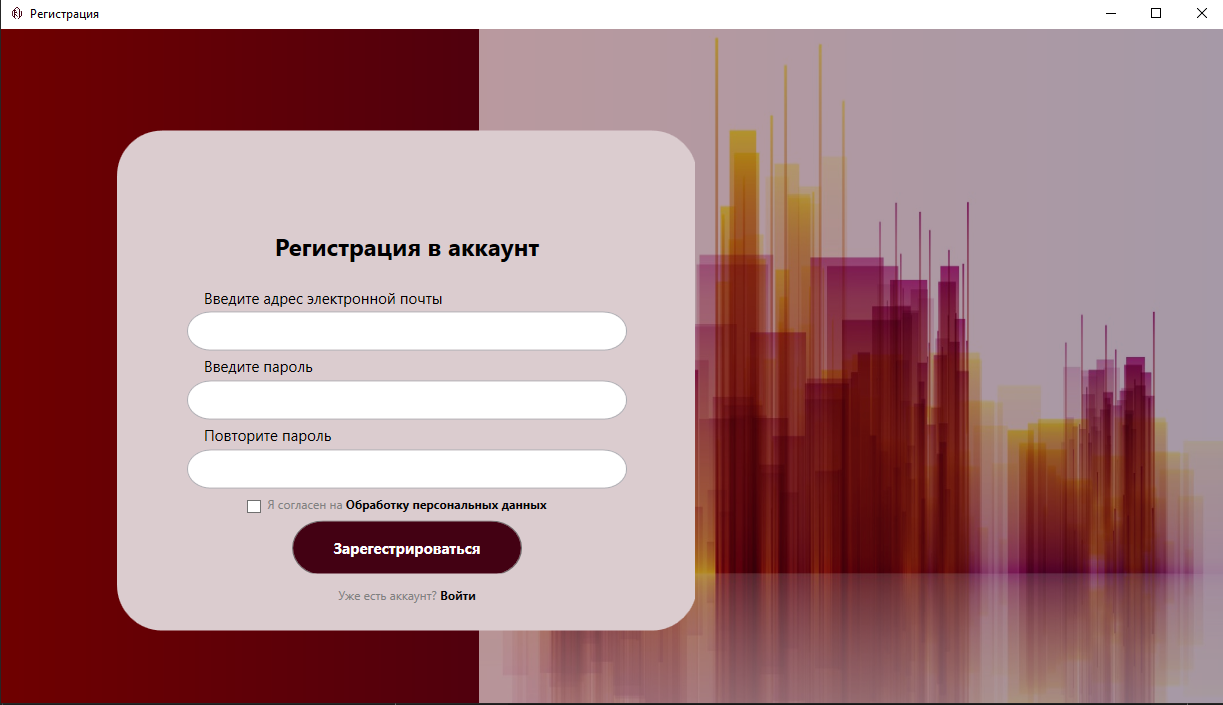


Рисунок 2.15 – интерфейс формы регистрации.

Продолжение Приложения 2

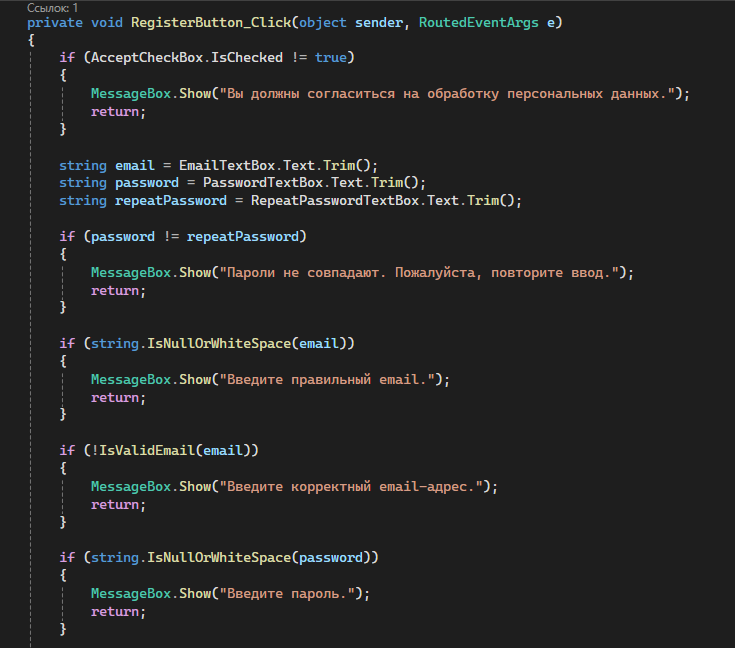




Рисунок 2.16 – код проверки корректности введенных данных.

Продолжение Приложения 2

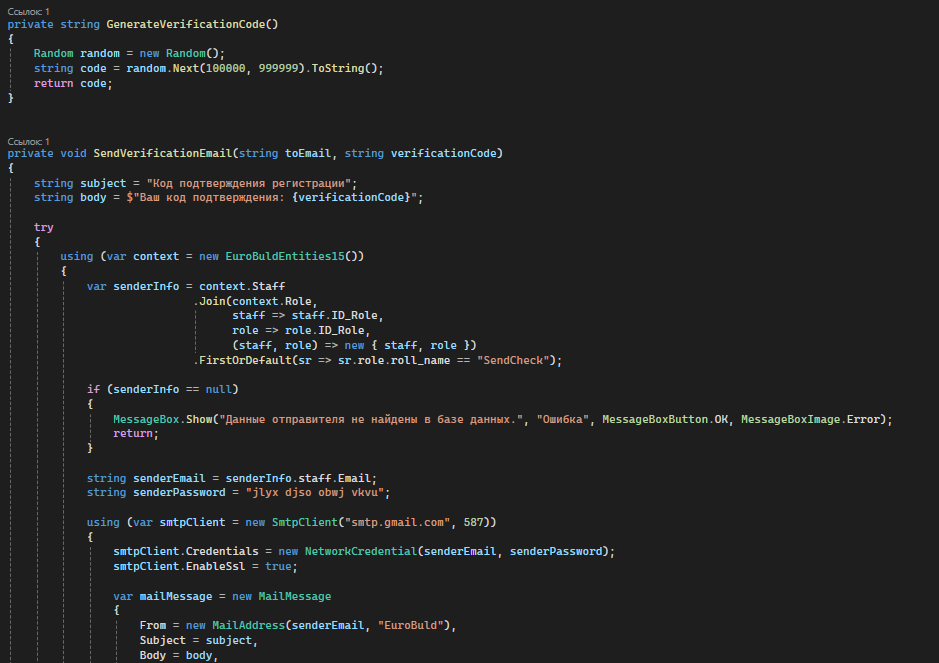




Рисунок 2.17 – код отправки 6-тизначного, когда для подтверждения почты.

Продолжение Приложения 2

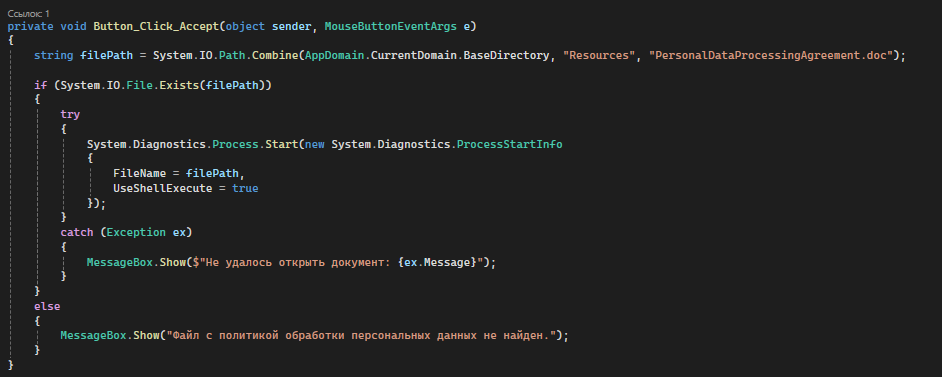


Рисунок 2.18 – код открытия договора о персональных данных.

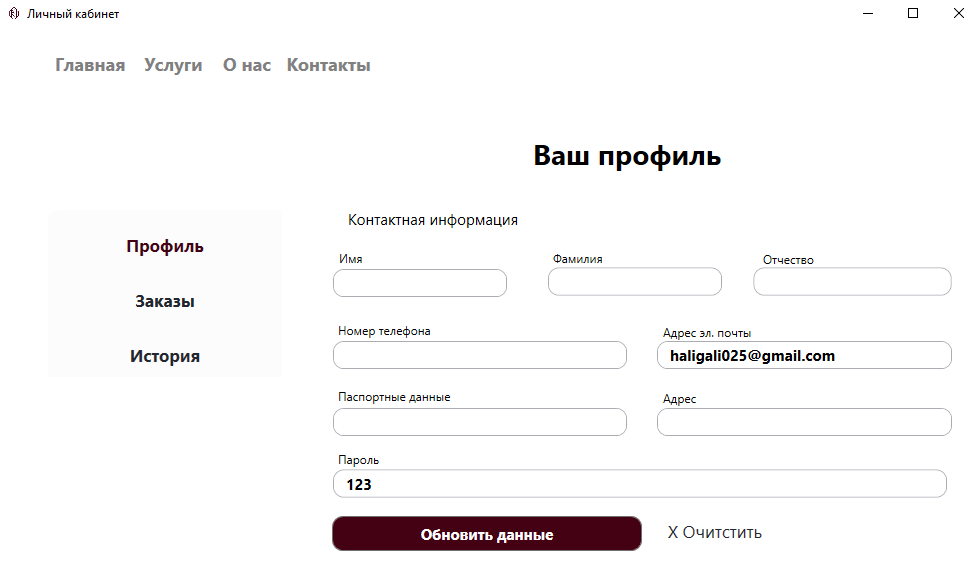


Рисунок 2.19 – интерфейс вкладки «Профиль» окна личный кабинет.

Продолжение Приложения 2

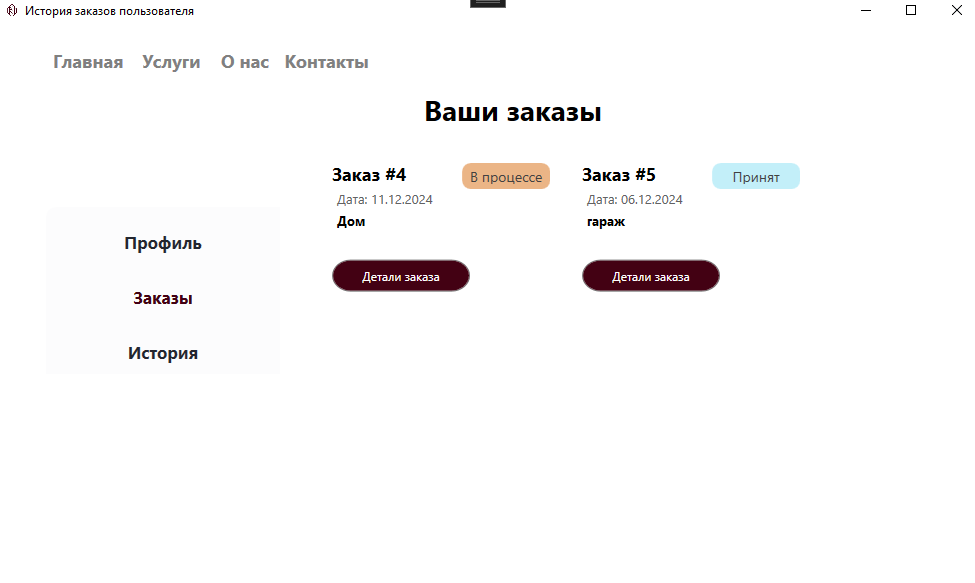


Рисунок 2.20 – интерфейс вкладки «Заказы» окна личный кабинет.

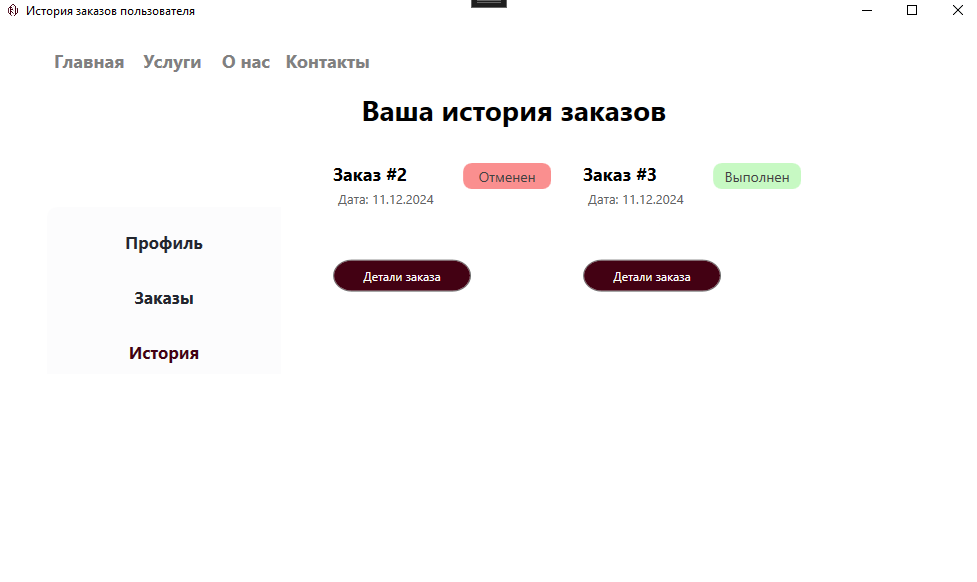


Рисунок 2.21 – интерфейс вкладки «История» окна личный кабинет.

Продолжение Приложения 2

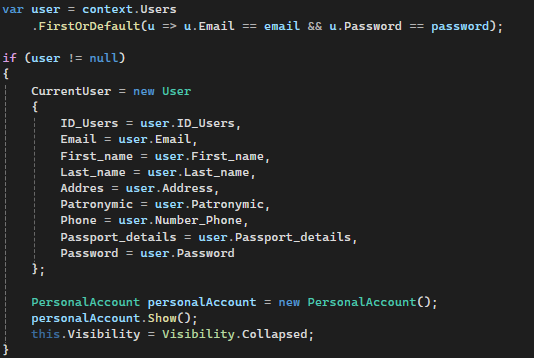


Рисунок 2.22 – код загрузки данных текущего пользователя из базы данных.

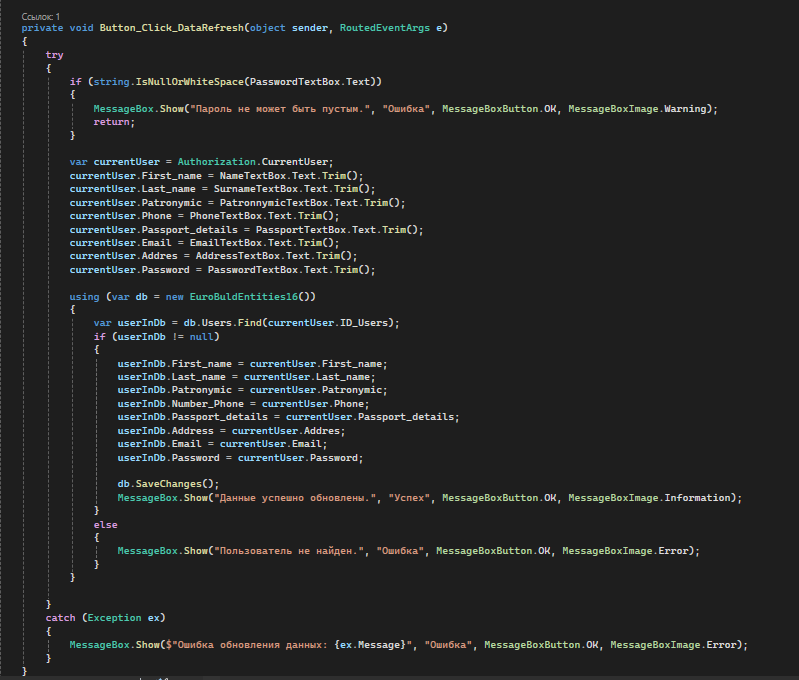


Рисунок 2.23 – код редактирования и обновления текущих данных пользователя из базы данных.

Продолжение Приложения 2



Рисунок 2.25 – код загрузки заказов

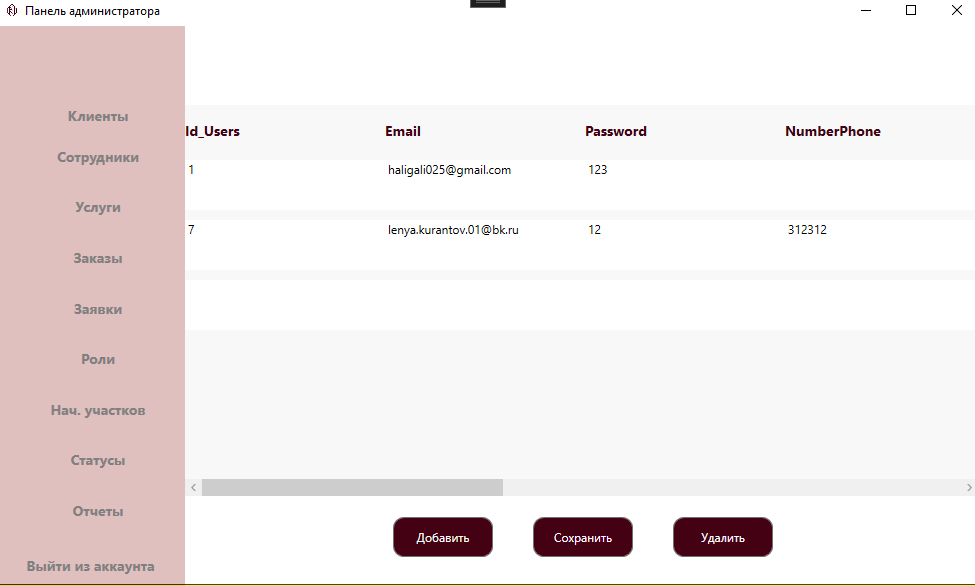


Рисунок 2.26 – интерфейс формы администратора.

Продолжение Приложения 2

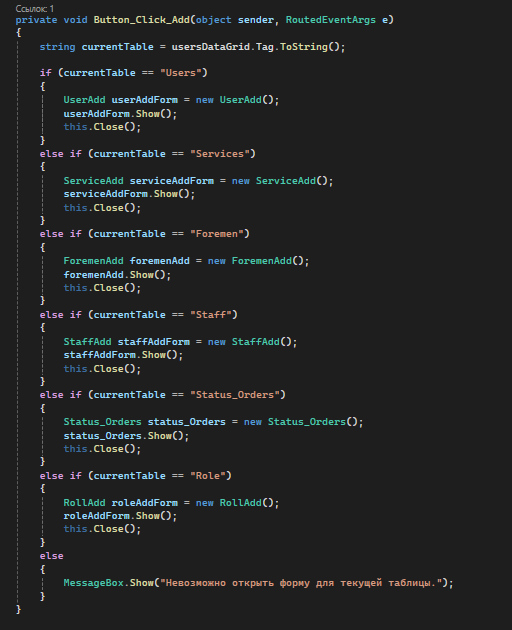


Рисунок 2.27 – код открытия форм для добавления новых записей

Продолжение Приложения 2



Рисунок 2.28 – код для добавления новой записи в базу данных

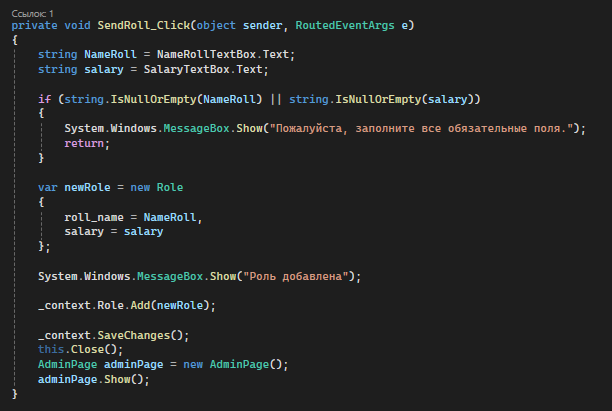


Рисунок 2.29 – код для добавления новой записи в базу данных

Продолжение Приложения 2

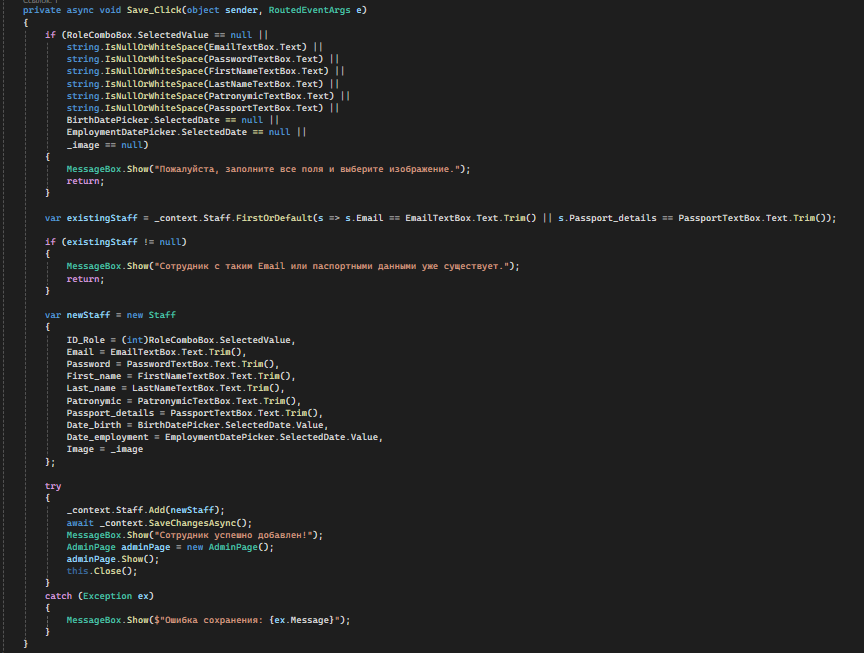


Рисунок 2.30 – код для добавления новой записи в базу данных

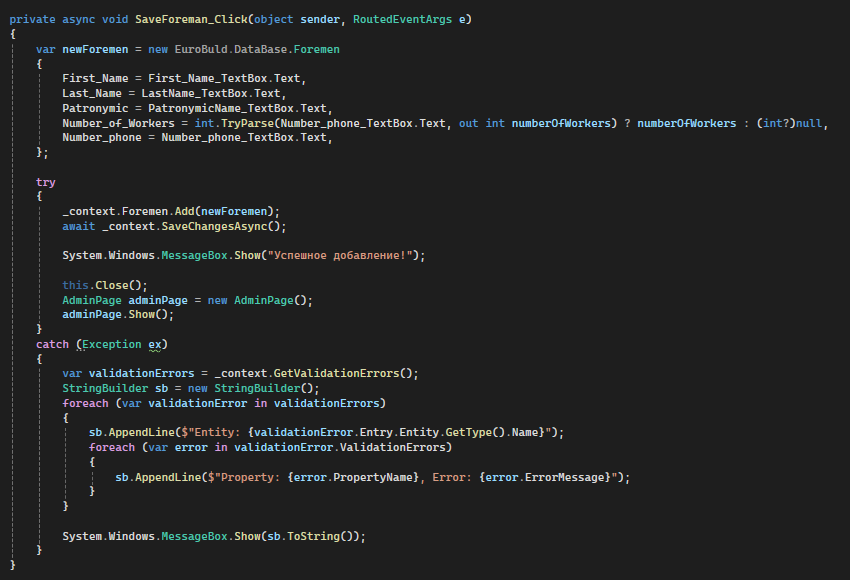


Рисунок 2.31 – код для добавления новой записи в базу данных

Продолжение Приложения 2

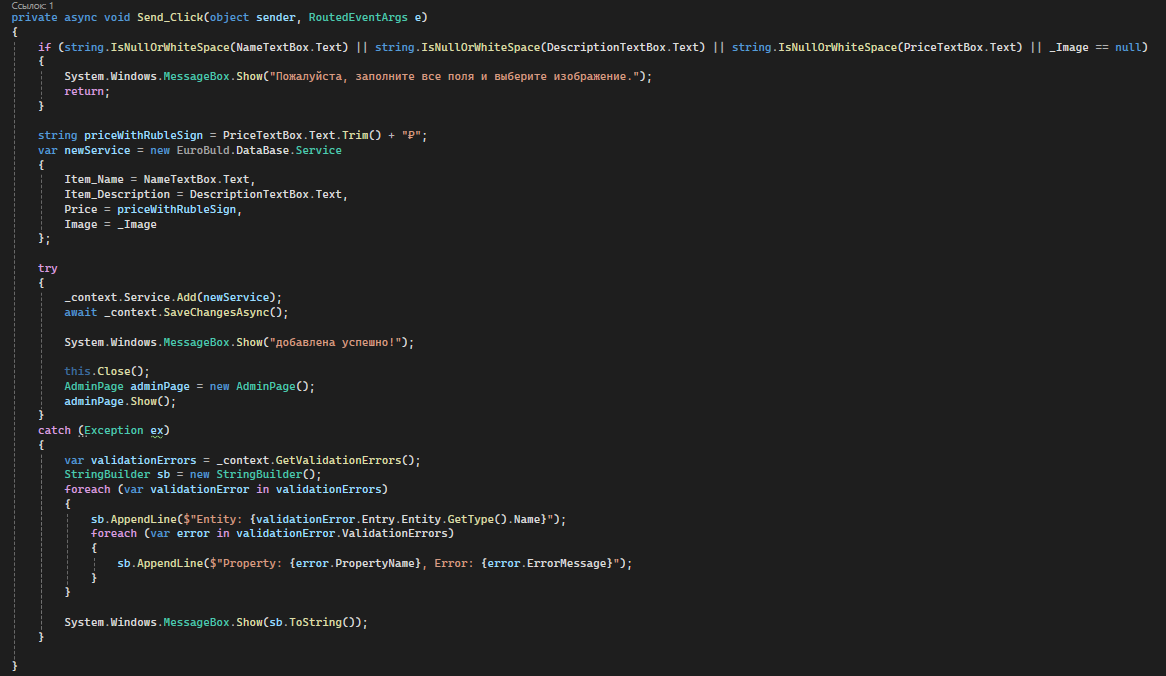


Рисунок 2.32 – код для добавления новой записи в базу данных

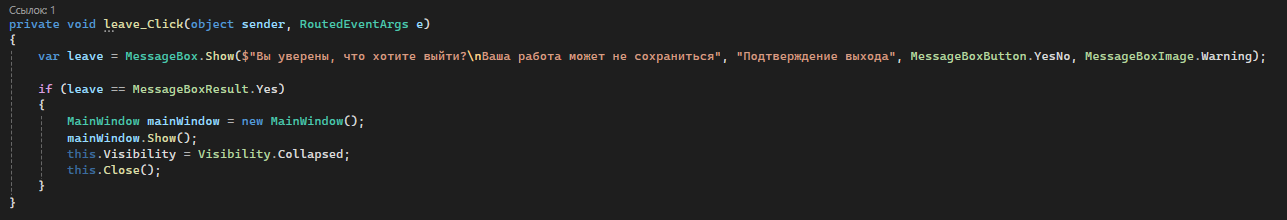


Рисунок 2.33 – код для выхода из аккаунта админа

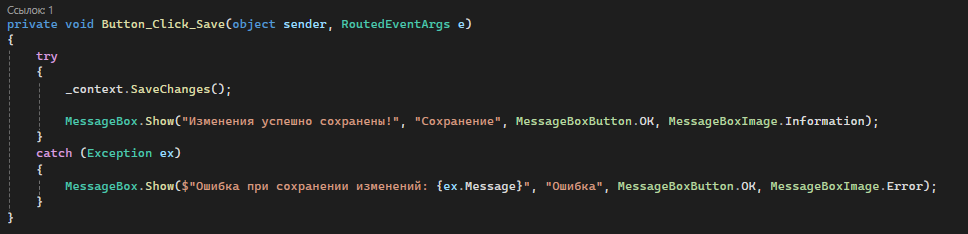


Рисунок 2.34 – код для сохранения изменений записей в базе данных

Продолжение Приложения 2

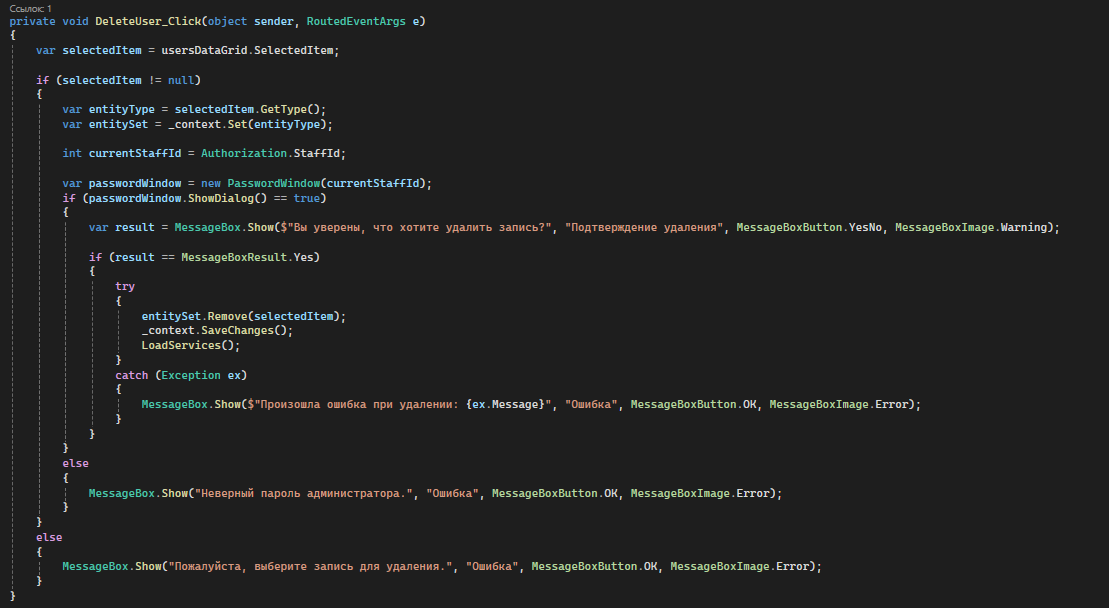


Рисунок 2.35 – код для удаления записей в базе данных

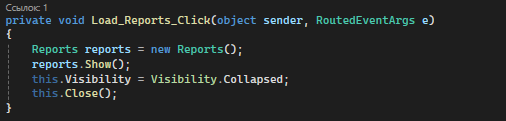


Рисунок 2.36 – код для открытия формы отчетов

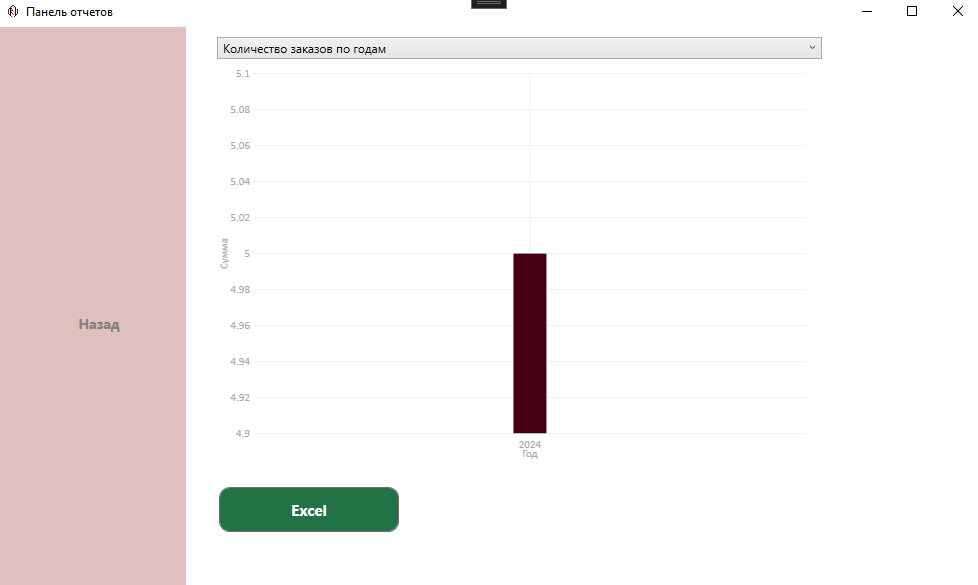


Рисунок 2.37 – интерфейс формы Отчетов.

Продолжение Приложения 2

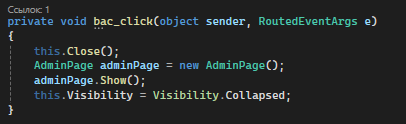


Рисунок 2.38 – код для возвращения на панель администратора



Продолжение Приложения 2



Продолжение Приложения 2



Продолжение Приложения 2

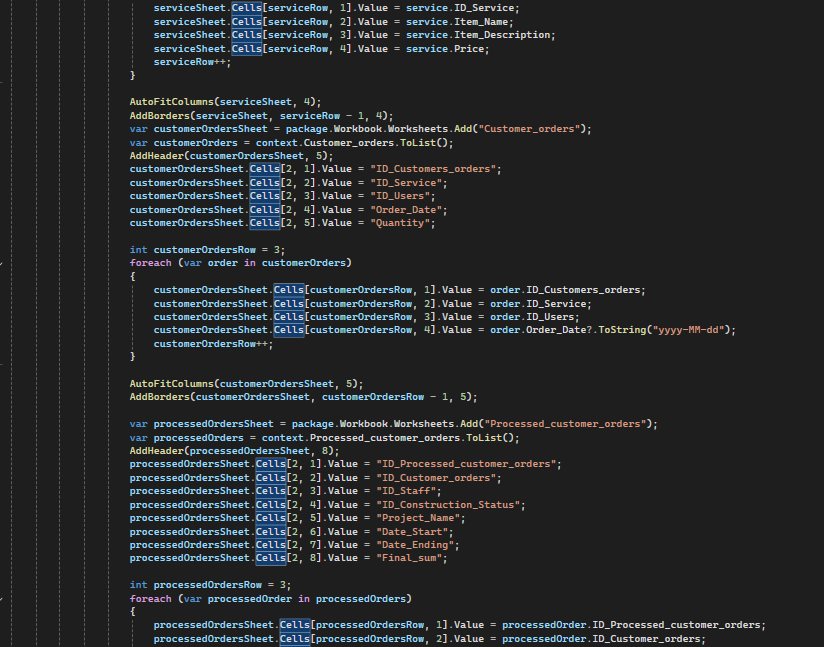
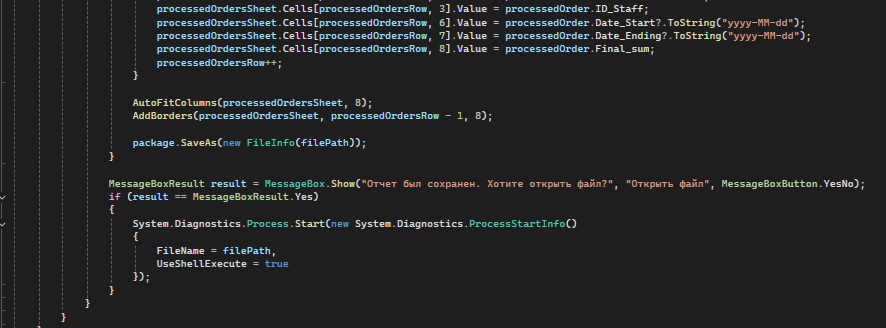
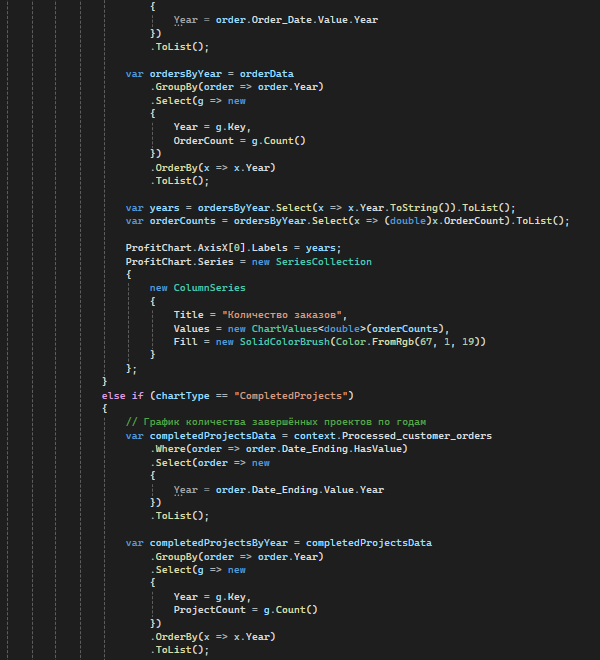
 

Рисунок 2.39 – код для выгрузки таблиц в Excel

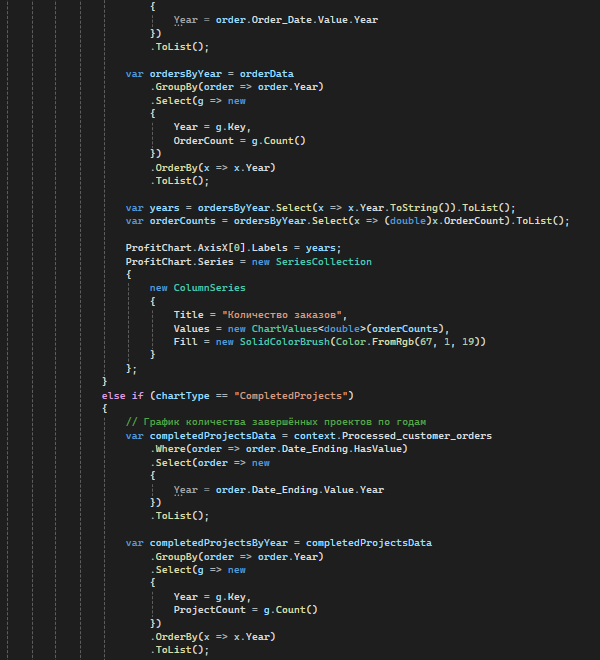
Продолжение Приложения 2



Продолжение Приложения 2



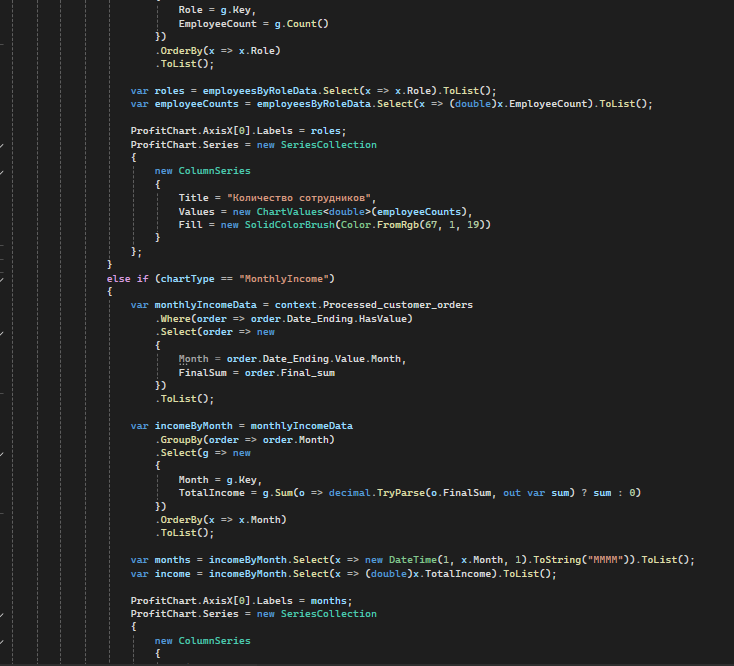
Продолжение Приложения 2



Продолжение Приложения 2



Продолжение Приложения 2



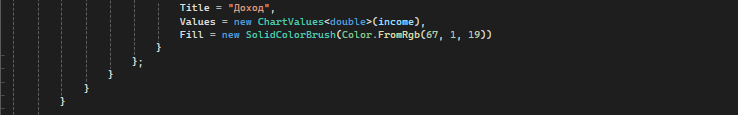


Рисунок 2.40 – код для отображения информации на графике

Продолжение Приложения 2

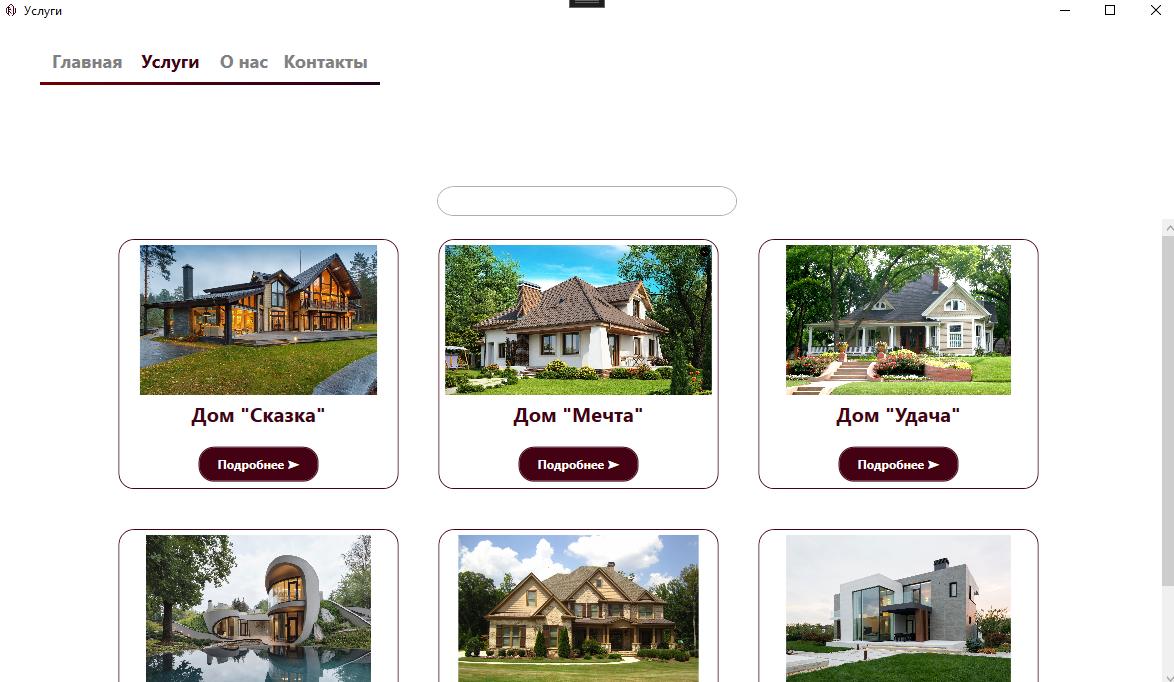


Рисунок 2.41 – интерфейс формы каталога товаров.



Рисунок 2.42 – код для загрузки информации о товарах на форму

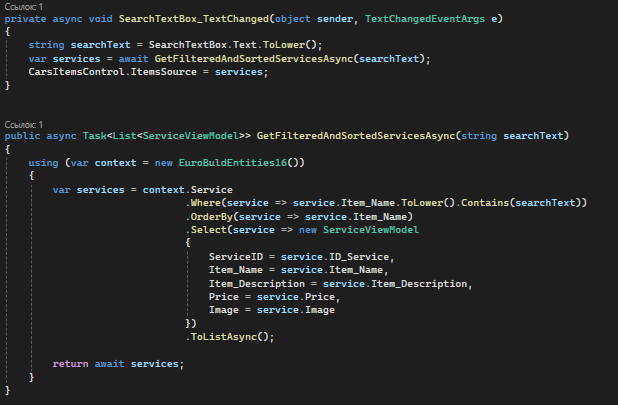


Рисунок 2.43 – код для сортировки товара по алфавиту

Продолжение Приложения 2

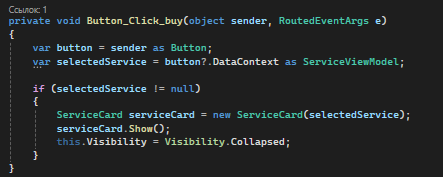


Рисунок 2.44 – код для открытия окна о товаре

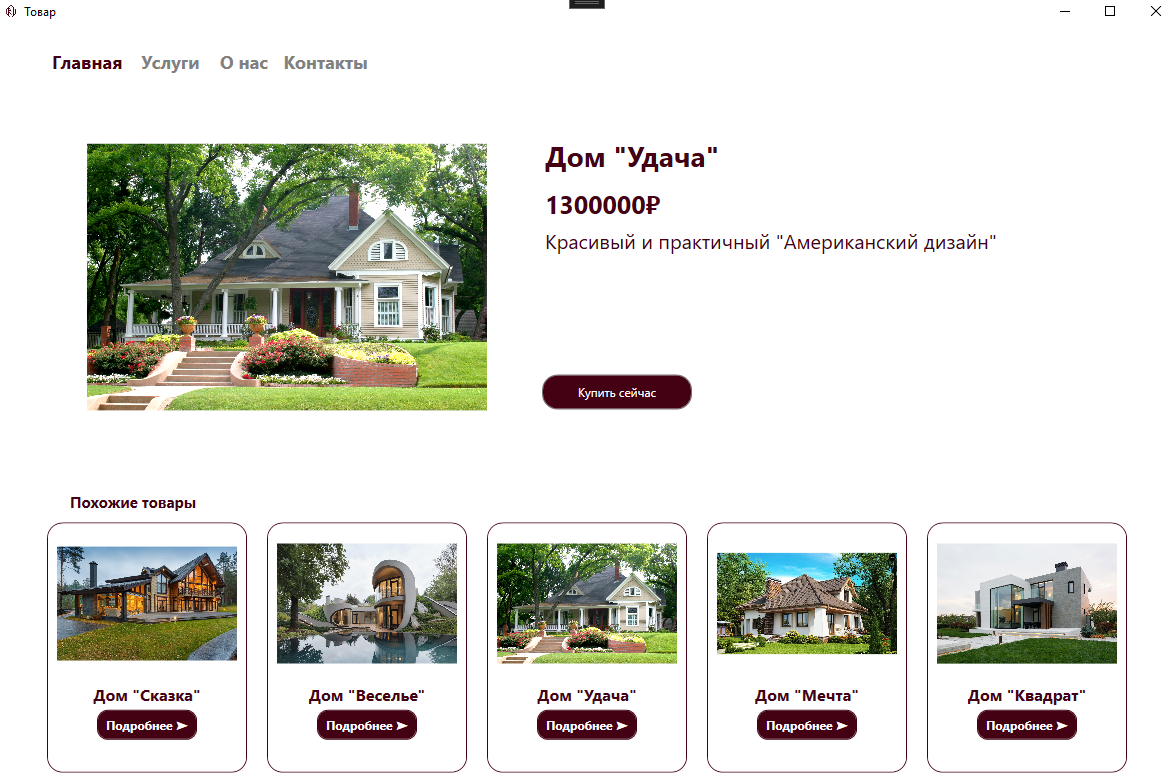


Рисунок 2.45 – интерфейс формы карточки товара.

Продолжение Приложения 2

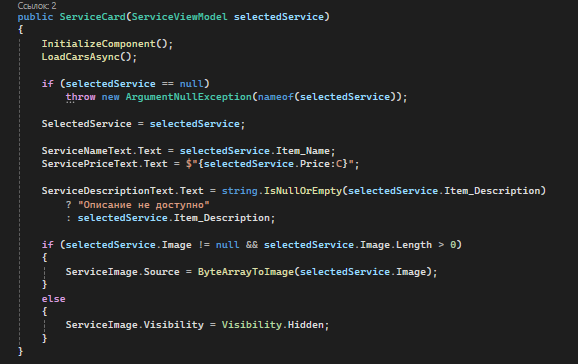
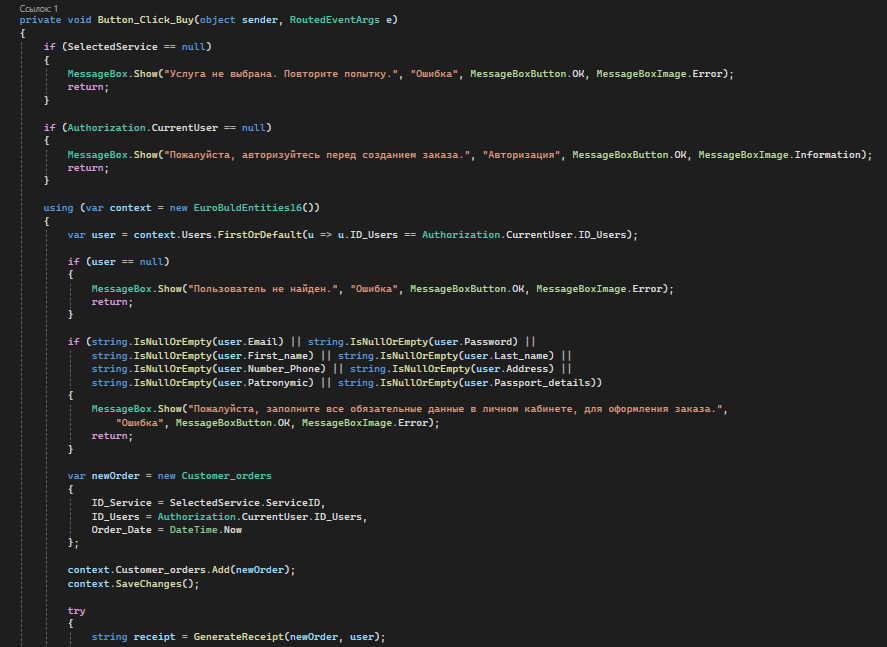


Рисунок 2.46 – код для загрузки товара из базы данных



Продолжение Приложения 2



Рисунок 2.47 – код для оформления заказа

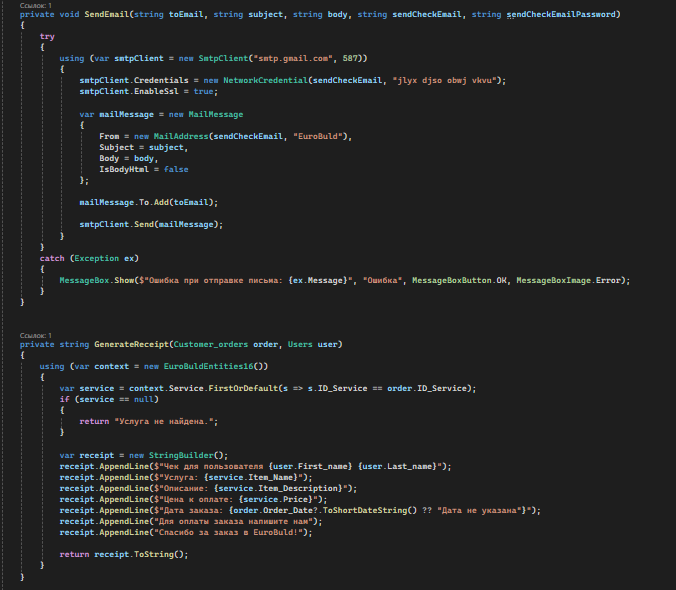


Рисунок 2.48 – код для отправки чека на электронную почту

Продолжение Приложения 2

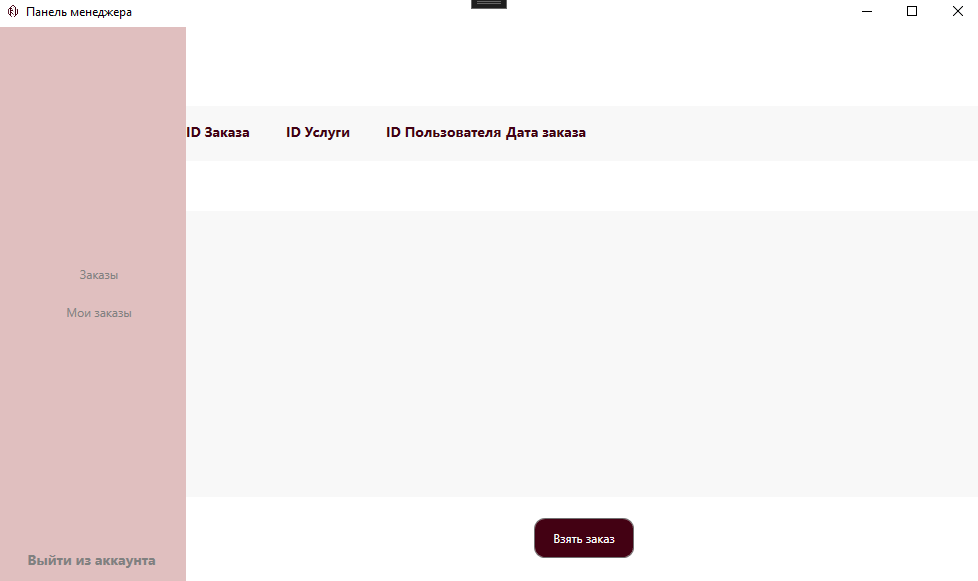


Рисунок 2.49 – интерфейс формы менеджера.

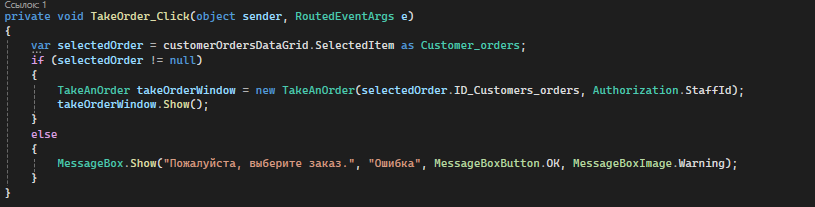


Рисунок 2.50 – код для взятия заказа

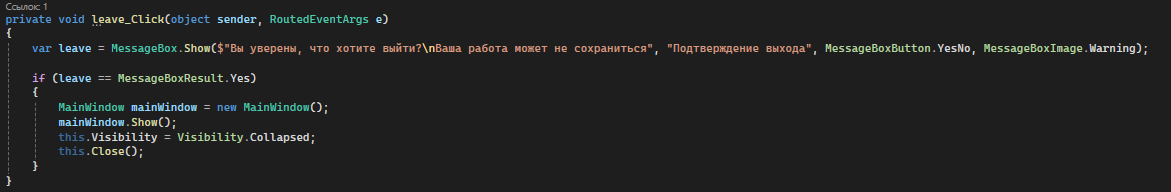


Рисунок 2.51 – код для выхода из аккаунта

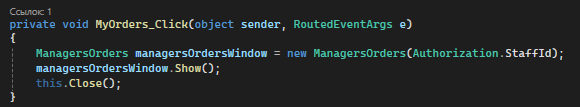


Рисунок 2.52 – код для открытия вкладки с заказами определенного менеджера

Продолжение Приложения 2

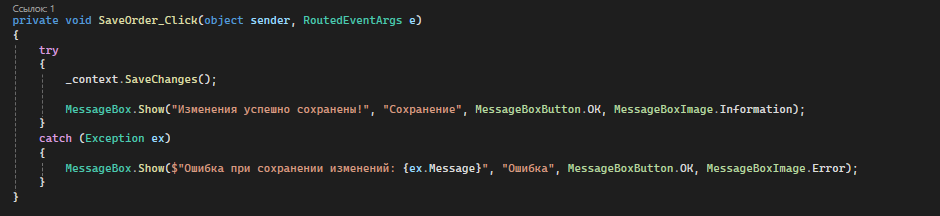


Рисунок 2.53 – код для сохранения изменений в заказе

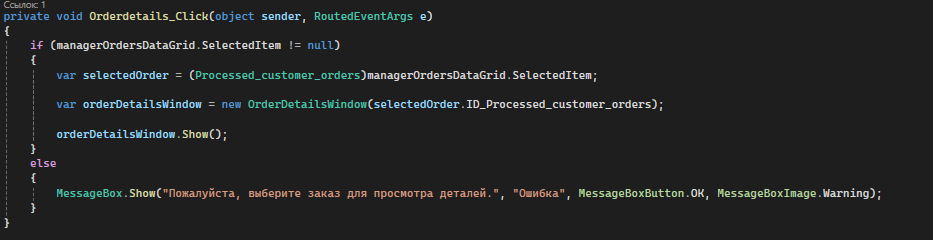


Рисунок 2.53 – код для открытия формы со всей подробностью о заказе