

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждения образования «БЕЛОРУССКИЙ  
ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий  
Кафедра программной инженерии  
Специальность 6-05-0612-01 Программная инженерия

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии  
программирования и стандарты проектирования»  
Тема: Программное средство «Бронирование билетов для  
маршрутных транспортных средств»

Исполнитель  
Студент (ка) 2 курса группы 9 Хуторцов Кирилл  
Владимирович (Ф.И.О.)  
Руководитель работы асс. Ромыш А.С.  
(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой \_\_\_\_\_  
Председатель Смелов В.В.  
(подпись)

Минск 2025

## Оглавление

Введение .....	4
1 Аналитический обзор литературы и формирование требований .....	5
1.1 Анализ прототипов .....	5
1.2 Аналитический обзор источников .....	10
1.3 Требования к проекту .....	11
2 Анализ требований к программному средству и разработка функциональных требований .....	12
2.1 Описание средств разработки .....	12
2.1.1 Microsoft Visual Studio 2022 .....	12
2.1.2 Программная платформа .NET 8.0 .....	12
2.1.3 Язык программирования C# .....	13
2.1.4 Технология WPF .....	13
2.1.5 Расширяемый язык разметки XAML .....	13
2.1.6 Технология ADO.NET .....	13
2.1.7 Microsoft SQL Server .....	14
2.1.8 Паттерн MVVM .....	14
2.2 Архитектура системы .....	15
2.3 Диаграмма вариантов использования .....	16
3 Проектирование программного средства .....	18
3.1 Общая структура приложения .....	18
3.2 Взаимоотношение между классами .....	19
3.3 Проектирование логической структуры базы данных .....	20
3.4 Проектирование архитектуры приложения .....	23
3.5 Проектирование архитектуры окон .....	24
4 Реализация программного средства .....	27
4.1 Диаграмма классов .....	27
4.2 Реализация паттернов MVVM и Command. ....	27
4.3 Реализация методов регистрации и авторизации пользователей	29
5. Тестирование, проверка работоспособности и анализ полученных результатов .....	30

5.1 Тестирование регистрации и авторизации .....	30
5.2 Тестирование панелей поиска и бронирования билета .....	32
5.3 Тестирование панели администратора .....	34
6 Руководство по установке и использованию .....	35
ЗАКЛЮЧЕНИЕ .....	40
Список использованных источников .....	41
ПРИЛОЖЕНИЕ А .....	42
ПРИЛОЖЕНИЕ Б .....	43
ПРИЛОЖЕНИЕ В .....	44
ПРИЛОЖЕНИЕ Г .....	49

## Введение

Раньше для того, чтобы заказать билет приходилось проводить много времени в очередях и на вокзале в целом. Также многие не могли попасть на нужный автобус по причине того, что просто опоздали.

Поэтому для того, чтобы избавить заказчиков от ненужных хлопот и сохранить их время разрабатываются приложения, которые ускоряют процесс заказа билета, а также делают маршрут комфортным.

Данные приложения позволяют пользователям с легкостью заказать билеты между практически любыми точками мира и при этом они потратят гораздо меньше времени, чем если бы они приобретали билет на вокзале.

Целью данного курсового проекта является разработка ПО, осуществляющего поиск и бронирование билетов маршрутных транспортных средств, регистрацию и авторизацию пользователей. Также необходимо реализовать сохранение пользовательских маршрутов, возможность добавления маршрутов администратором.

Для успешной реализации курсового проекта необходимо:

- провести анализ соответствующей литературы;
- ознакомиться с прототипами программных средств выбранной тематики.
- определить функциональные требования.
- продумать структуру базы данных.
- продумать структуру проекта.
- реализовать программное средство.
- протестировать программное средство.
- написать руководство пользователя.

Главная задача данного курсового проектирования — это разработка программного средства, которое реализует все вышеперечисленные функции и решает поставленные задачи. Язык разработки проекта – C#. При выполнении курсового проекта будут использованы принципы и приемы ООП, база данных MS SQL Server, и технология Windows Presentation Foundation (WPF).

Содержание данной пояснительной записки отражает все этапы выполнения курсового проекта.

# 1 Аналитический обзор литературы и формирование требований

## 1.1 Анализ прототипов

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов и литературных источников.

Были проанализированы цели и задачи, поставленные в данном курсовом проекте, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

Были рассмотрены 3 аналога:

В качестве первого аналога был рассмотрен веб-сервис «INFOBUS.BY», специализирующийся на бронировании автобусных билетов. Данная платформа предоставляет пользователям возможность поиска и покупки билетов на междугородние и международные автобусные маршруты, интегрируясь с расписаниями перевозчиков.

Плюсы:

- широкий выбор маршрутов;
- удобный интерфейс: интуитивная навигация, визуализация мест в автобусе, пошаговое оформление заказа;
- гибкая система фильтров;
- поддержка онлайн-оплаты.

Минусы:

- проблемы с отменой заказа
- бывают проблемы с организацией перевозки

Чтобы найти билеты, нужно заполнить поля с направлениями и датой. Далее нужно дождаться конца поиска, чтобы увидеть все доступные варианты, в том числе самые быстрые и дешёвые.

Интерфейс для поиска билета представлен на рисунок 1.1.

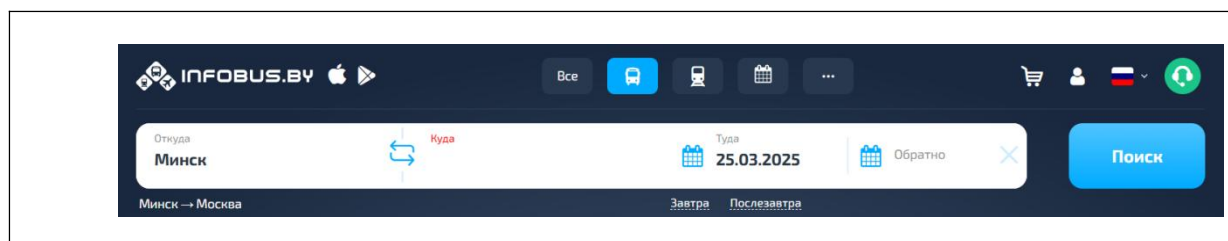


Рисунок 1.1 — Поиск билета в приложении «INFOBUS.BY»

Результаты поиска — это варианты билетов с ценами, датой и временем выезда и приезда, местом посадки и высадки.

Интерфейс для отображения результатов поиска представлен на рисунке 1.2.

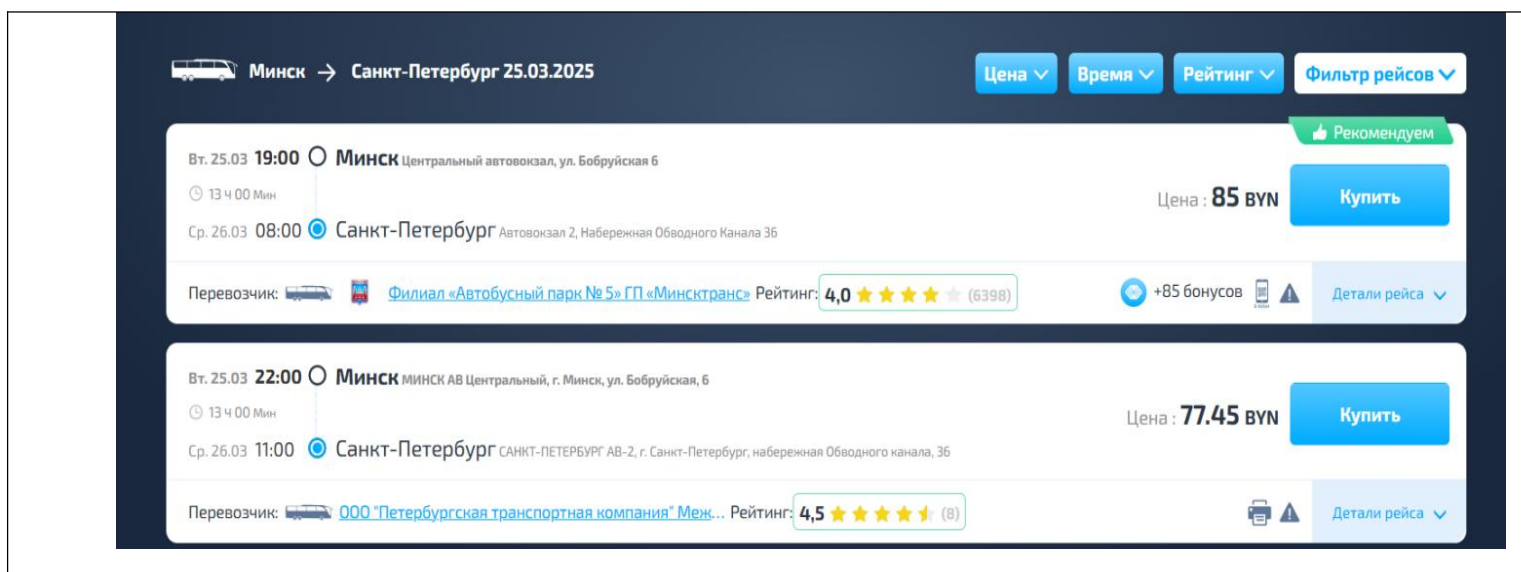


Рисунок 1.2 — Результат поиска билета в приложении «INFOBUS.BY»

В качестве второго аналога рассмотрен веб-сервис «SMILEBUS.BY», специализирующийся на бронировании автобусных билетов. Данная платформа предоставляет пользователям возможность поиска и покупки билетов между городами и городскими поселками, интегрируясь с расписаниями перевозчиков.

Плюсы:

- широкий выбор маршрутов;
- возможность стать в очередь при отсутствии мест;
- бонусная система баллов;
- удобный сайт и интуитивно понятный интерфейс;
- скидки для детей.

Минусы:

- при отмене заказа менее чем за 12 часов статус изменяется на плохой и пропадает возможность бронировать билет онлайн;
- отсутствует возможность оплаты онлайн;
- регулярное повышение цен.

В главном меню приложения пользователь может осуществить поиск необходимого ему маршрута.

Форма для поиска билетов представлена на рисунке 1.3.

Рисунок 1.3 — Поиск билета в приложении «SMILEBUS.BY»

Результаты поиска маршрута по введенным пользователем данным представлены на рисунке 1.4.

Отправление	Прибытие	Цена	Места
<b>Минск - Глуск - Октябрьский</b> <b>11:00</b> Минск, м. Институт Культуры	<b>13:32</b> Глуск, ПМК-249	2 часа 32 мин в пути 28 р.	Мест нет <a href="#">Стать в очередь</a>
<b>Минск - Глуск - Октябрьский</b> <b>14:00</b> Минск, м. Институт Культуры	<b>16:32</b> Глуск, ПМК-249	2 часа 32 мин в пути 28 р.	Мест нет <a href="#">Стать в очередь</a>
<b>Минск - Глуск - Октябрьский</b> <b>18:00</b> Минск, м. Институт Культуры	<b>20:32</b> Глуск, ПМК-249	2 часа 32 мин в пути 28 р.	3+ <a href="#">Заказать</a>

Рисунок 1.4 — Результат поиска билета в приложении «SMILEBUS.BY»

Рассматриваемое программное средство также предлагает возможность сохранения истории заказов пользователя. Пример представлен на рисунке 1.5.

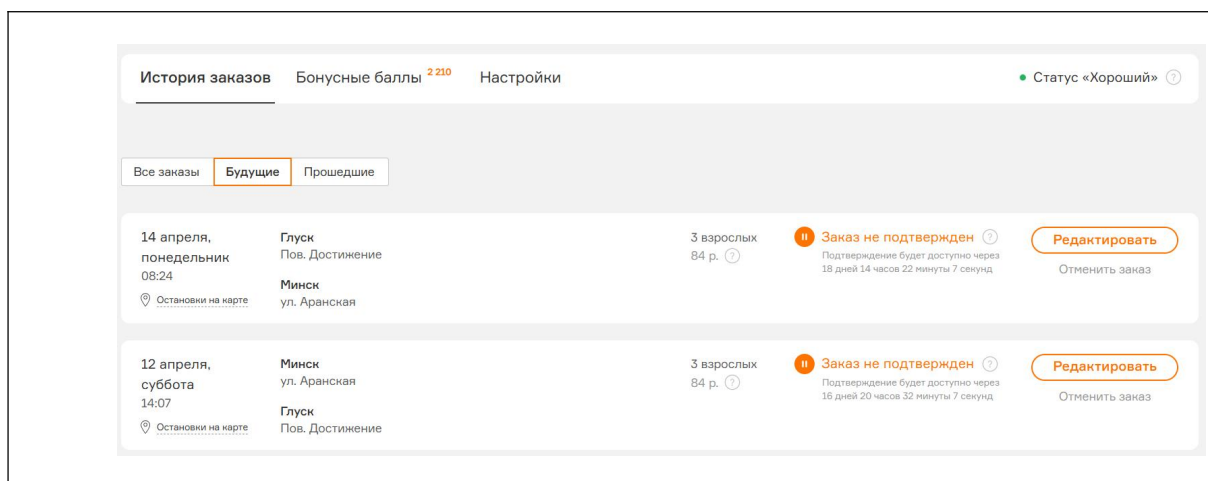


Рисунок 1.5 — Хранение истории заказов в приложении «SMILEBUS.BY»

«SMILEBUS.BY» также имеет систему регистрации и авторизации пользователей для синхронизации данных между сайтом и мобильным приложением. Страница регистрации и авторизации представлена на рисунке 1.6.

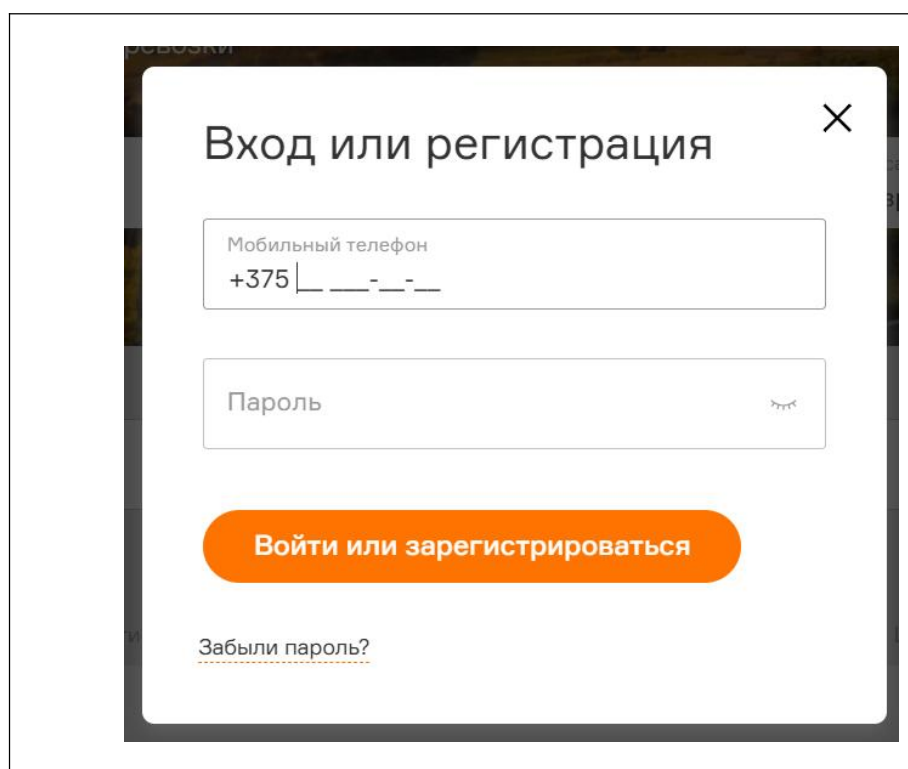


Рисунок 1.6 — Система регистрации и авторизации в приложении «SMILEBUS.BY»

В качестве третьего аналога рассмотрен веб-сервис «ATLASBUS.BY», специализирующийся на бронировании автобусных билетов. Данная платформа предоставляет



пользователям возможность поиска и покупки билетов на междугородние и международные автобусные маршруты, интегрируясь с расписаниями перевозчиков.

Плюсы:

- легкий удобный поиск, выбор и бронирование желаемого маршрута;
- подробная информация о маршрутах;
- поддерживает систему управления бронированиями;
- имеется программа поощрения пассажиров;
- регулярные скидки на менее популярные маршруты.

Минусы:

- введение в заблуждение информацией о маршрутах, хотя на самом деле билетов на них нету

На главной странице сайта пользователь может осуществить поиск интересующего его маршрута. Форма для поиска билета представлена на рисунке 1.7.

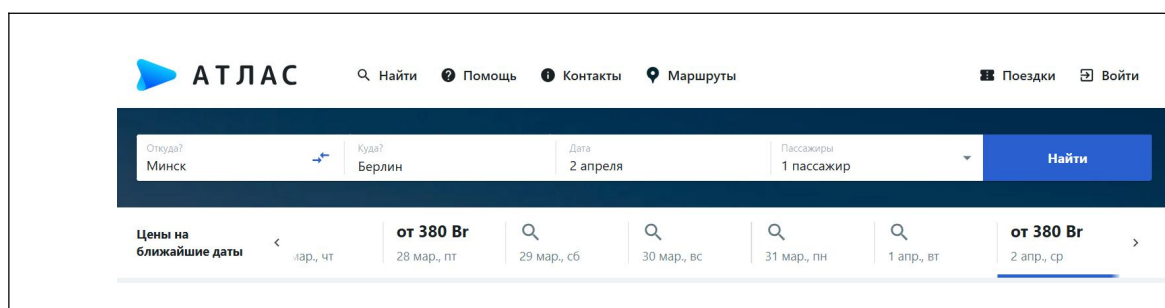
The screenshot shows the top section of the ATLAS website. At the top is the ATLAS logo and navigation links: 'Найти' (Find), 'Помощь' (Help), 'Контакты' (Contacts), 'Маршруты' (Routes), 'Поездки' (Trips), and 'Войти' (Login). Below this is a search bar with fields for 'Откуда?' (From) set to 'Минск', 'Куда?' (To) set to 'Берлин', 'Дата' (Date) set to '2 апреля', and 'Пассажиры' (Passengers) set to '1 пассажир'. A blue 'Найти' (Find) button is on the right. Below the search bar is a horizontal scrollable list of dates with prices, starting with 'от 380 Br' for '28 мар., пт'.

Рисунок 1.7 — Поиск билета на сайте «ATLASBUS.BY»

Результаты поиска маршрута по введенным пользователем данным представлены на рисунке 1.8.

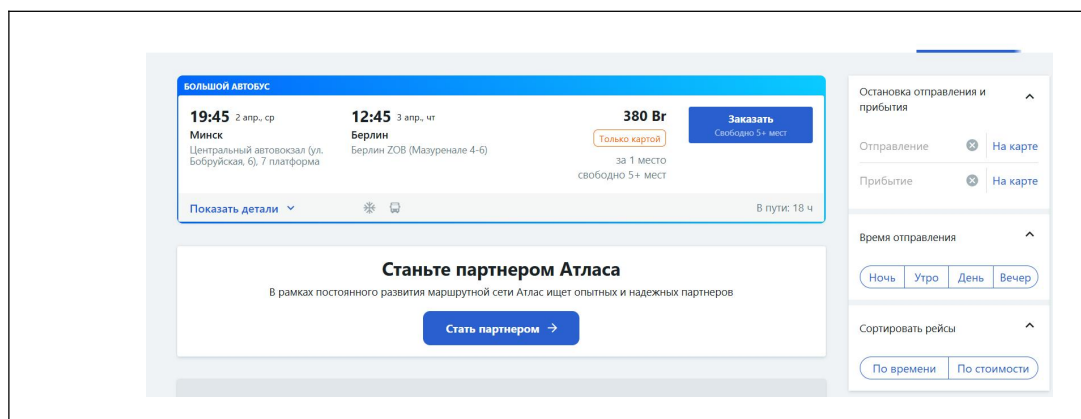
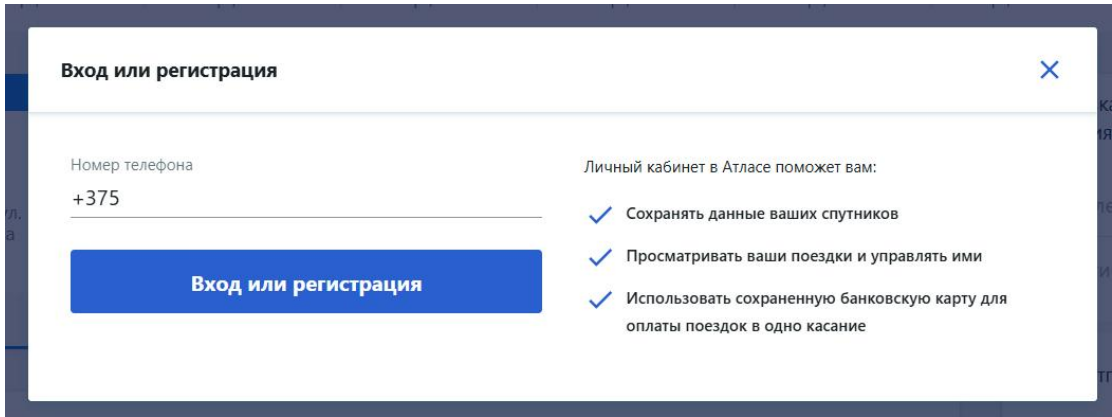
The screenshot shows the search results for a bus route. The main card displays the route 'Минск - Берлин' with departure at 19:45 on April 2nd and arrival at 12:45 on April 3rd. The price is 380 Br. A 'Заказать' (Order) button is visible. Below the card is a section 'Станьте партнером Атласа' (Become an Atlas partner). On the right side, there are filters for 'Остановка отправления и прибытия' (Departure and arrival stops), 'Время отправления' (Departure time) with options for Night, Day, and Evening, and 'Сортировать рейсы' (Sort trips) by time or cost.

Рисунок 1.8 — Результат поиска билета на сайте «ATLASBUS.BY»

Как видно на рисунке, результаты поиска можно сортировать по различным критериям.

Рассматриваемый сайт также поддерживает систему регистрации и авторизации пользователей. Форма для регистрации/авторизации представлена на рисунке 1.9.



Вход или регистрация

Номер телефона  
+375

Вход или регистрация

Личный кабинет в Атласе поможет вам:

- ✓ Сохранять данные ваших спутников
- ✓ Просматривать ваши поездки и управлять ими
- ✓ Использовать сохраненную банковскую карту для оплаты поездок в одно касание

Рисунок 1.9 — Система регистрации и авторизации на сайте «ATLASBUS.BY»

Анализируя аналоги, можно сделать вывод, что все три приложения обладают современным интерфейсом и стараются обеспечить удобство и легкость в использовании для клиентов. Однако, каждое приложение может иметь свои особенности и преимущества, которые следует учитывать.

## 1.2 Аналитический обзор источников

В ходе подготовки была изучена специальная техническая, учебно-методическая и справочная литература, статьи и материалы, опубликованные в сети интернет.

При разработке окон для регистрации и авторизации был использован подход, описанный в статье «WPF – система авторизации и регистрации». В статье были рассмотрены алгоритм работы системы авторизации и регистрации и пример создания окон регистрации и авторизации.

Принцип работы с SQL были получены из статьи «Подключение к базе данных». В статье было рассмотрено подключение необходимых библиотек, работа с SQL.

Дополнительная информации о принципах работы с WPF была получена из интернет-источника «Metanit», содержащего практические советы по работе с технологией.

### 1.3 Требования к проекту

После рассмотрения представленных выше аналогов было решено использовать следующий функционал: возможность авторизации и регистрации пользователей, поиск и бронирование билетов пользователями, просмотр истории заказов, а также возможность добавления администратором новых маршрутов.

Обзор вышеперечисленных аналогов позволяет проанализировать все преимущества и недостатки альтернативных возможностей и позволяет сформулировать список требований, предъявляемых к программному средству, разрабатываемому в данном курсовом проекте.

Программное средство должно обеспечивать возможность выполнения перечисленных ниже функций:

Функции администратора:

- поддерживать работу с базой данных;
- добавлять и удалять билеты, автобусы/маршрутки, места посадки/высадки, компании;
- просматривать и удалять заказы пользователей;
- просмотр статистики по разным направлениям.

Функции клиента:

- выполнять регистрацию и авторизацию;
- выполнять поиск маршрутов;
- выбор по дате, количеству мест, типу ТС(автобус/маршрутка);
- выбор места посадки и высадки;
- бронировать и покупать билеты;
- оценка поездки;
- просматривать историю заказов;
- сохранение избранных маршрутов.

Таким образом, в ходе работы над этим разделом были сформулированы основные функциональные требования для проектирования программного средства.

## **2 Анализ требований к программному средству и разработка функциональных требований**

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Основная цель анализа требований в проектах заключается в получении максимально полной информации о клиенте, его задачах и особенностях проекта. Анализ требований помогает определить границы проекта, выявить возможные риски и ключевые требования, как методологические, так и технологические. На этом этапе формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые будут использоваться для оценки результатов внедрения. Определение и описание требований является важным этапом, так как эти требования оказывают влияние на все последующие этапы проекта и определяют его успех.

### **2.1 Описание средств разработки**

При разработке приложения были использованы:

- интегрированная среда разработки Microsoft Visual Studio 2022;
- программная платформа .NET Framework 8.0;
- язык программирования C#;
- расширяемый язык разметки XAML;
- технология WPF;
- технология ADO.NET;
- СУБД Microsoft SQL Server;
- паттерн MVVM.

#### **2.1.1 Microsoft Visual Studio 2022**

Microsoft Visual Studio 2022 — это интегрированная среда разработки (IDE) от Microsoft, которая обеспечивает разработчиков инструментами и возможностями для создания приложений с использованием Windows Presentation Foundation (WPF).

#### **2.1.2 Программная платформа .NET 8.0**

Платформа .NET — это программный фреймворк, разработанный компанией Microsoft, который в основном работает на операционной системе Microsoft Windows. Он предоставляет среду выполнения для запуска приложений и набор библиотек, которые используются для создания программных приложений.

Фреймворк .NET состоит из нескольких компонентов, включая общую языковую среду (Common Language Runtime — CLR), библиотеку классов .NET Framework и различные инструменты разработки.

### **2.1.3 Язык программирования C#**

В качестве языка программирования используется C# — основной язык разработки в .NET. Язык объектно-ориентированный, имеет строгую статическую типизацию, поддерживает перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения. Используется как основной язык в технологии WPF.

### **2.1.4 Технология WPF**

Для предоставления пользовательского интерфейса и разграничения дизайна и бизнес-логики используется технология Microsoft WPF — аналог WinForms, система для построения клиентских приложений Windows с возможностями взаимодействия с пользователем и графическая подсистема в составе .NET, использующая язык разметки XAML.

### **2.1.5 Расширяемый язык разметки XAML**

WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language) элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление. XAML представляет собой язык декларативного описания интерфейса, основанный на XML.

### **2.1.6 Технология ADO.NET**

ADO.NET (ActiveX Data Objects .NET) — это набор технологий, предоставляющих программистам доступ к данным из различных источников данных в среде .NET. ADO.NET является частью .NET Framework и обеспечивает эффективное взаимодействие с базами данных и другими источниками данных.

ADO.NET поддерживает различные источники данных, включая реляционные базы данных, XML-файлы и объекты в памяти. Он также предоставляет возможности для выполнения транзакций, обработки ошибок, кэширования данных и других задач, связанных с доступом к данным.

### 2.1.7 Microsoft SQL Server

Для организации баз данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта. Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL, должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения. Основным используемым языком запросов — Transact-SQL — реализован на структурированном языке запросов (SQL) с расширениями.

### 2.1.8 Паттерн MVVM

MVVM (Model-View-ViewModel) - это паттерн архитектуры, используемый в разработке программного обеспечения для разделения пользовательского интерфейса (View) от бизнес-логики (Model) и связывания их через промежуточный слой ViewModel.

В MVVM модель представляет данные и бизнес-логику приложения, которая может включать в себя операции чтения/записи данных из и в источники данных, такие как база данных или веб-сервисы.

Представление (View) отвечает за отображение данных и взаимодействие с пользователем. Оно не содержит бизнес-логику и должно быть максимально независимым от модели.

ViewModel представляет промежуточный слой между View и Model. Он содержит логику, необходимую для обработки пользовательских действий, обновления данных в модели и уведомления View об изменениях. ViewModel предоставляет свойства и команды, которые привязываются к элементам пользовательского интерфейса во View, позволяя им отображать данные и реагировать на действия пользователя.

MVVM позволяет достичь разделения ответственности между компонентами приложения, облегчает тестирование и повышает переиспользуемость кода. Он также способствует лучшей поддержке параллельной разработки пользовательского интерфейса и бизнес-логики, так как разработчики могут работать независимо над своими частями приложения.

## 2.2 Архитектура системы

Для курсового проекта был использован способ разработки программных модулей, основанный на архитектуре клиентского приложения. Приложение состоит из двух частей:

- Клиентская часть. В этом модуле осуществляется взаимодействие пользователя с приложением с помощью графического интерфейса. В клиентской части пользователь сможет получить полную информацию об объектах приложения, в зависимости от присвоенной ему роли (пользователь или администратор).

- База данных. В базе данных описываемого приложения хранится информация о билетах, пользователях, забронированных билетах, избранных маршрутах, оценках и комментариях.

Различают три уровня требований к проекту:

- бизнес-требования;
- пользовательские требования;
- функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований можно рассматривать общие требования к разрабатываемому средству. К их числу относятся:

- простота и лёгкость интерфейса;
- использование принципов ООП;
- использование архитектурных шаблонов проектирования;
- использование системы управления базами данных (СУБД).

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы.

Программное средство должно предоставлять следующие функциональные возможности:

Для пользователя:

- регистрация;
- авторизация;

- просмотр доступных билетов;
- поиск билетов;
- выбор по дате, количеству мест, типу ТС(автобус/маршрутка);
- выбор места посадки и высадки.;
- бронировать и покупать билеты;
- оценка поездки;
- просматривать историю заказов;
- сохранение избранных маршрутов.

Для администратора:

- поддерживать работу с базой данных;
- добавлять и удалять билеты, автобусы/маршрутки, места
- посадки/высадки, компании;
- просматривать и удалять заказы пользователей;
- просмотр статистики по разным направлениям.

После проведения анализа были выявлены следующие функциональные требования:

- вся информация должна храниться в базе данных;
- приложение должно производить валидацию вводимых пользователем данных;
- приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
- приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
- приложение должно предоставлять возможность пользователям проходить авторизацию и входить в систему.

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированным списком.

### **2.3 Диаграмма вариантов использования**

Диаграмма вариантов использования (Use Case Diagram) является графическим средством моделирования, которое позволяет представить функциональность системы или программного продукта с точки зрения ее пользователей. Она помогает идентифицировать и описать различные сценарии использования системы, включая взаимодействие между актерами (пользователями) и самой системой.



Диаграмма вариантов использования состоит из актеров, вариантов использования (Use Case) и связей между ними. Актеры представляют различных пользователей или внешние системы, которые взаимодействуют с программой. Варианты использования представляют собой функциональные возможности системы, которые могут быть выполнены пользователями или другими актерами.

Преимущества использования диаграммы вариантов использования включают:

Визуализацию функциональности системы: Диаграмма вариантов использования помогает визуализировать все возможные сценарии использования системы, что помогает лучше понять ее функциональность и взаимодействие с пользователями.

Определение требований: Диаграмма вариантов использования помогает идентифицировать и описать функциональные требования к системе на основе сценариев использования.

Улучшение коммуникации: Диаграмма вариантов использования служит средством коммуникации между разработчиками, заказчиками и другими заинтересованными сторонами, позволяя им лучше понять ожидания от данного приложения.

UML-диаграмма вариантов использования для программного средства представлена в приложении А.

## 3 Проектирование программного средства

### 3.1 Общая структура приложения

Общая структура проекта Windows Presentation Foundation (WPF) – это организация файлов и папок в рамках проекта WPF. Структура разрабатываемого проекта представлена на рисунке 3.1.



Рисунок 3.1 — Структура проекта

Описание структуры основных папок и файлов проекта и библиотеки классов представлено в таблице 3.1.

Таблица 3.1 — Описание структуры папок и файлов проекта

Имя файла/папки	Содержание
App.config	Файл с параметрами проекта
App.xaml	Определение глобальных ресурсов
Папка View	Директория, содержащая классы окон приложения, а также графическую информацию, использующуюся в окнах
Папка ViewModels	Директория, содержащая команды для связи Model и View для реализации MVVM
Папка Models	Директория, в которой располагаются файлы, отвечающие за логику приложения и всех страниц и окон, каждой странице соответствует свой Model класс.
Папка Images	Директория, содержащая графические материалы
Папка Languages	Директория, содержащая язык приложения
Папка Styles	Директория, содержащая словари ресурсов стилей

Таким образом, сформированная таблица помогает понять общую структуру проекта проектируемого программного средства.

Структура папки Models представлена на рисунке 3.2.

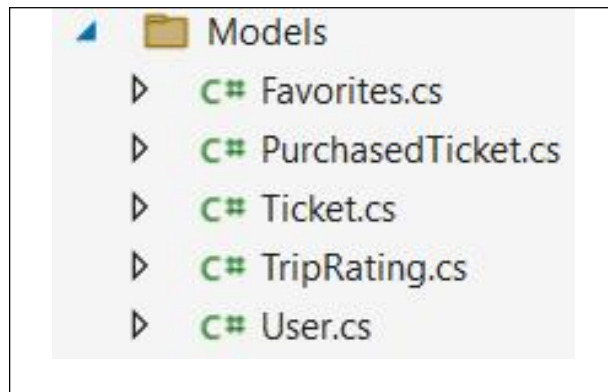


Рисунок 3.2 — Содержание папки Models

Более подробное описание структуры и файлов данной папки представлено в таблице 3.2.

Таблица 3.2 — Описание структуры папки Models

Имя файла	Содержание
Tickets.cs	Класс для определения и хранения информации о билетах
User.cs	Класс для определения и хранения учётной записи пользователя
PurchasedTickets.cs	Класс для определения и хранения информации об забронированных билетах
Favorites.cs	Класс для определения и хранения информации о избранных билетах
TripRating.cs	Класс для определения и хранения информации о оценках и комментариях

В целом, описание структуры проекта позволяет лучше понимать, как устроено программное средство и какие компоненты в нем присутствуют.

### 3.2 Взаимоотношение между классами

Для визуализации взаимосвязей между классами используется диаграмма UML — графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями).

Для представления внутренней структуры программы в виде классов и связей между ними используется диаграмма классов. Приложение спроектировано таким образом, что каждый класс выполняет свои функции и практически не зависит от других. Диаграмма классов представлена в приложении Б.

### 3.3 Проектирование логической структуры базы данных

Для создания базы данных описываемого приложения использовалась система управления реляционными базами данных Microsoft SQL Server 2022.

База данных — это совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимо от прикладных программ.

Система управления базами данных — совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами.

База данных данного курсового проекта состоит из 5 таблиц.

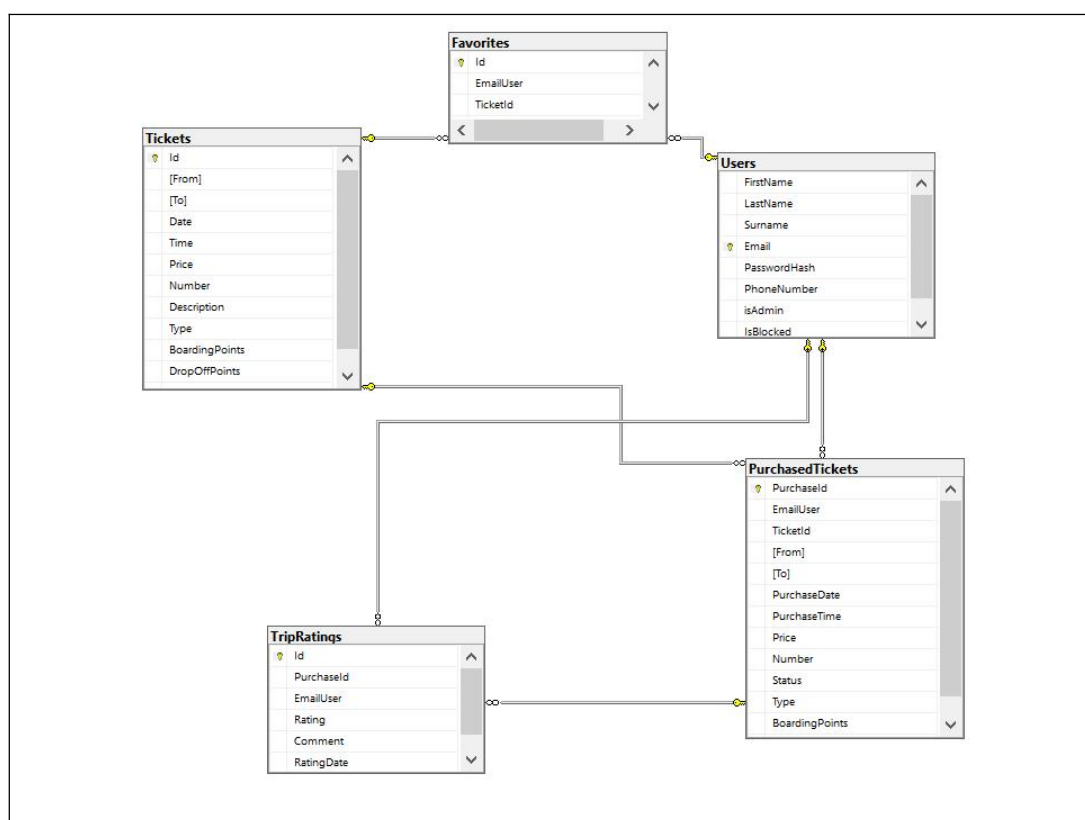


Рисунок 3.3 — Схема базы данных приложения

Остановимся более подробно на таблицах и рассмотрим, какие поля они содержат.

В таблице 3.3 проиллюстрирована структура таблицы «Tickets», которая содержит информацию о билетах.

Таблица 3.3 — Таблица «Tickets»

Имя столбца	Тип данных	Описание
Id	int	Содержит уникальный id билета
From	nvarchar(100)	Содержит название откуда идёт маршрут
To	nvarchar(100)	Содержит куда приходит маршрут
Date	date	Содержит дату отправления маршрута
Time	time	Содержит время отправления маршрута
Price	float	Содержит цену на билет
Number	int	Содержит кол-во мест
Description	nvarchar(max)	Содержит описание маршрута
Type	nvarchar(50)	Содержит тип транспорта
BoardingPoints	nvarchar(500)	Содержит места посадки маршрута
DropOffPoints	nvarchar(500)	Содержит места высадки маршрута
Company	nvarchar(100)	Содержит компании

В таблице 3.4 проиллюстрирована структура таблицы «Users», которая содержит информацию о пользователях

Таблица 3.4 — Таблица «Users»

Имя столбца	Тип данных	Описание
FirstName	nvarchar(50)	Содержит имя пользователя
LastName	nvarchar(50)	Содержит отчество пользователя
Surname	nvarchar(50)	Содержит фамилию пользователя
Email	nvarchar(100)	Содержит email пользователя
PasswordHash	nvarchar(100)	Содержит хеш пароля пользователя
PhoneNumber	nvarchar(20)	Содержит телефон пользователя
isAdmin	bit	Содержит является ли админом пользователь
IsBlocked	bit	Содержит заблокирован ли пользователь

В таблице 3.5 проиллюстрирована структура таблицы «PurchasedTickets», которая содержит информацию о заброннированных билетах пользователем.

Таблица 3.5 — Таблица «PurchasedTickets»

Имя столбца	Тип данных	Описание
PurchaseId	int	Содержит id заброннированного билета
EmailUser	nvarchar(100)	Содержит email пользователя, который забронировал билет
TicketId	int	Содержит id билета
From	nvarchar(100)	Содержит название откуда идёт маршрут
To	nvarchar(100)	Содержит куда приходит маршрут
PurchaseDate	date	Содержит дату бронирования билета
PurchaseTime	time	Содержит время бронирования билета
Price	float	Содержит цену на билет
Number	int	Содержит кол-во мест выбранное пользователями
Status	int	Содержит статус - забронирован/оплачен
Type	nvarchar(50)	Содержит выбранный пользователем тип транспорта
BoardingPoints	nvarchar(500)	Содержит выбранное место посадки
DropOffPoints	nvarchar(500)	Содержит выбранное место высадки

В таблице 3.6 проиллюстрирована структура таблицы «Favorites», которая содержит информацию о избранных маршрутах.

Таблица 3.6 — Таблица «Favorites»

Имя столбца	Тип данных	Описание
Id	int	Содержит id избранного билета
EmailUser	nvarchar(100)	Содержит email пользователя, который добавил билет в избранное
TicketId	int	Содержит id билета
AddedDate	date	Содержит дату добавления в избранное

В таблице 3.7 проиллюстрирована структура таблицы «TripRatings», которая содержит информацию об оценках и комментариях.

Таблица 3.7 — Таблица «TripRatings»

Имя столбца	Тип данных	Описание
Id	int	Содержит id рейтинга
PurchaseId	int	Содержит id заброннированного билета
EmailUser	nvarchar(100)	Содержит email пользователя, который забронировал билет
Rating	int	Содержит оценку от 1 до 5
Comment	nvarchar(500)	Содержит комментарий
RatingDate	date	Содержит дату добавления рейтинга

В целом, структура базы данных проекта предоставляет необходи́мую основу для хранения и управления информацией в приложении.

### 3.4 Проектирование архитектуры приложения

Архитектура программного обеспечения представляет собой совокупность ключевых решений, касающихся организации программной системы. Она включает выбор структурных элементов и их интерфейсов, определение их взаимодействия, а также объединение этих элементов в более крупные системы. Однако, центральную роль играет архитектурный стиль, который определяет всю организацию системы, включая элементы, их интерфейсы, их сотрудничество и способы их соединения.

Для удовлетворения требований проектируемой системы с точки зрения различных атрибутов качества, применяются различные архитектурные шаблоны, так называемые паттерны. В разрабатываемом нами приложении мы используем архитектурный шаблон Model-View-ViewModel (MVVM). Этот шаблон состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View). Паттер MVVM позволяет добиться четкого разделения ответственности: бизнес-логика инкапсулируется в модели, интерфейс управляется через представление, а ViewModel обеспечивает их слабую связанность за счет механизмов двустороннего связывания данных. Такой подход упрощает тестирование компонентов, повышает гибкость приложения и способствует эффективному параллельному развитию интерфейса и логики. На рисунке 3.4 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.

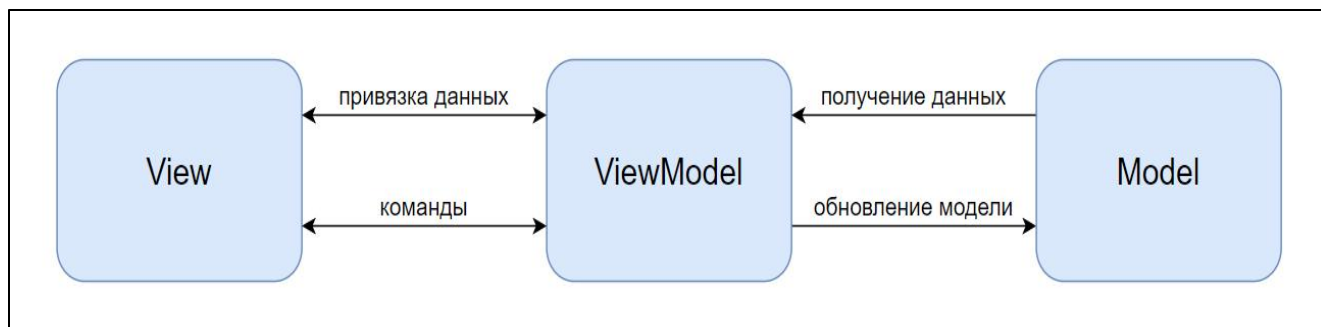


Рисунок 3.4 — Структура шаблона MVVM

Model (модель) описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную с этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления. Часто модель реализует интерфейс `INotifyPropertyChanged`, который позволяет системе автоматически обнаруживать изменения свойств модели и облегчает привязку к представлению, хотя сама модель не взаимодействует напрямую с представлением.

View (представление) определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление — это код XAML, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов. Представление не обрабатывает события, а в основном выполняет действия посредством команд.

ViewModel (модель представления) связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса `INotifyPropertyChanged` автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны. Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд.

### 3.5 Проектирование архитектуры окон

Приложение включает в себя 11 страниц.

При запуске программного средства пользователь сразу попадает на страницу входа. Если у пользователя еще нет аккаунта, он может нажать на кнопку "Создать аккаунт", чтобы перейти на форму регистрации. Интерфейс стартового окна представлен на рисунке 3.5.



The image shows a login form titled "Вход в систему" (Login to the system) in green text. Below the title are two input fields: "Email" and "Пароль" (Password). Under the input fields are two buttons: a green button labeled "Войти" (Login) and a dark blue button labeled "Создать аккаунт" (Create account).

Рисунок 3.5 – Форма входа

Если нужно зарегистрировать аккаунт, то пользователь это сможет сделать это на форме регистрации, представленной на рисунке 3.6.

The image shows a registration form titled "Регистрация" (Registration) in green text. Below the title are seven input fields: "Имя" (Name), "Фамилия" (Surname), "Отчество" (Patronymic), "Email", "Пароль" (Password), "Подтвердите пароль" (Confirm password), and "Телефон" (Phone). Under the input fields are two buttons: a green button labeled "Создать аккаунт" (Create account) and a dark blue button labeled "Назад к входу" (Back to login).

Рисунок 3.6 – Форма регистрации

После входа в приложение, в зависимости от роли, пользователь переходит либо к окну администратора, либо к окну стандартного пользователя.

Окно стандартного пользователя дает доступ к стандартному функционалу приложения. Пункты меню окна изображены на рисунке 3.7.

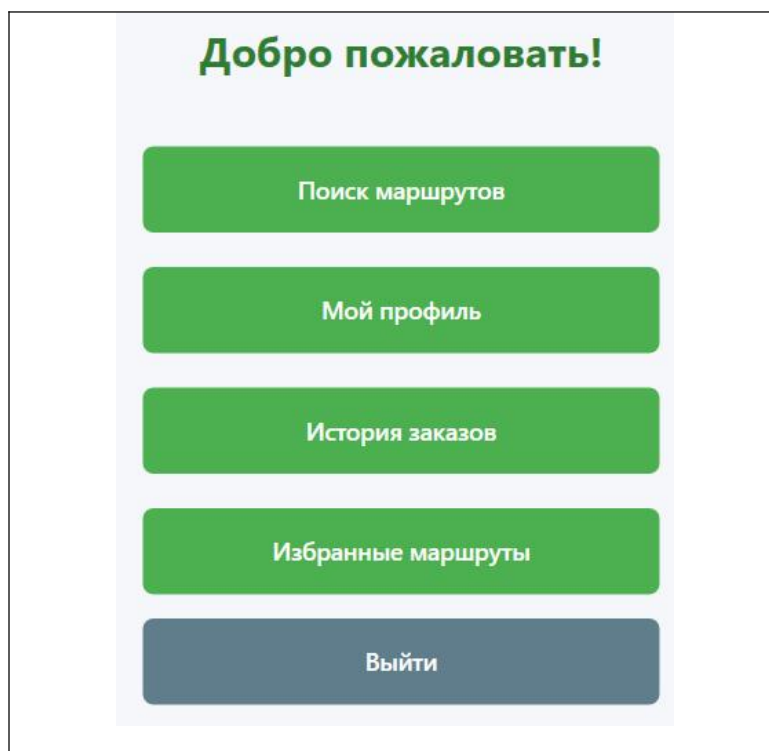


Рисунок 3.7 — Пункты меню для пользователя

При авторизации под логином и паролем администратора, можно попасть на страницу администрирования, пункты меню изображены на рисунке 3.8. Администратор имеет больше технических возможностей, чем остальные пользователи. Об этом свидетельствуют пункты меню, которые позволяют перейти на страницы с возможностью просмотра и редактирования таблиц БД.

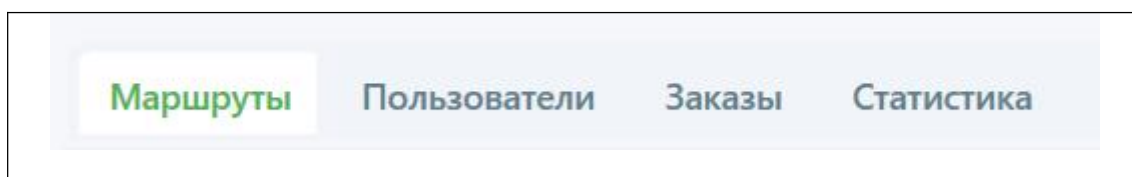


Рисунок 3.8 — Пункты меню для администратора

Таким образом, мы можем заметить, что для разных ролей пользователей приложение предоставляет разный функционал и, соответственно, разные окна и страницы.

## **4 Реализация программного средства**

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

### **4.1 Диаграмма классов**

В рамках разработки программного решения для организации задач и управления проектами, диаграмма классов является одним из ключевых инструментов моделирования и проектирования системы. Диаграмма классов представляет структуру системы, ее классы, атрибуты и отношения между классами.

В процессе проектирования программы для организации задач и управления проектами, диаграмма классов помогает определить основные классы, их свойства и методы. Она позволяет увидеть структуру данных, которые будут использоваться в программе, а также взаимосвязи между классами. Диаграмма классов также помогает лучше понять концептуальную модель системы и обеспечивает основу для дальнейшей разработки.

Диаграмма классов представлена в приложении Б.

### **4.2 Реализация паттернов MVVM и Command.**

Программное средство основано на архитектурном паттерне MVVM, для более простого взаимодействия с разметкой реализован паттерн Command

Для паттерна MVVM были созданы следующие классы:

- Users – класс реализует сущность пользователя;
- Tickets – класс реализует сущность билета;
- PurchaseTickets – класс реализует сущность забронированного билета;
- Favorites - класс реализует сущность избранного билета.
- TripRatings - класс реализует сущность добавления оценки и комментария
- UserProfileViewModel – класс реализует логику личного кабинета пользователя ;
- SearchRoutesViewModel – класс реализует поиск билета пользователем;
- RouteDetailsViewModel – класс позволяет выбрать детали маршрута, добавлять в избранное, бронировать билет, переходить на страницу оплаты;

- PurchaseHistoryViewModel – класс позволяет просмотреть историю заказов пользователя, удалить билет и перейти на страницу рейтинга;
- FavoritesViewModel – класс позволяет просматривать и удалять билеты из избранного;
- AdminViewModel – класс позволяет просматривать, редактировать, добавлять билеты и пользователей, просматривать и удалять заказы пользователей, просматривать статистику по разным направлениям.

Для паттерна Command создавался был создан класс:

- RelayCommand – класс реализует встроенный интерфейс ICommand. Реализация класса приведена в листинге 4.1;

```
using System;
using System.Windows.Input;
namespace RouteBookingSystem
{
    public class RelayCommand : ICommand
    {
        private readonly Action<object> _execute;
        private readonly Func<object, bool> _canExecute;

        public RelayCommand(Action<object> execute,
            Func<object, bool> canExecute = null)
        {
            _execute = execute;
            _canExecute = canExecute;
        }

        public bool CanExecute(object parameter) =>
            _canExecute == null || _canExecute(parameter);
        public void Execute(object parameter) =>
            _execute(parameter);

        public event EventHandler CanExecuteChanged
        {
            add => CommandManager.RequerySuggested +=
value;
            remove => CommandManager.RequerySuggested -=
value;
        }
    }
}
```

Листинг 4.1 – Реализация класса RelayCommand

Класс был создан для того чтобы было более удобно управлять данными и чтобы их отображение было корректно выведено.

### **4.3 Реализация методов регистрации и авторизации пользователей**

При запуске приложения пользователь первым делом видит стартовое окно Входа. Если у пользователя есть аккаунт, он нажимает на кнопку «Вход» и в случае ввода корректных данных он переходит к функционалу приложения, который отличается в зависимости от роли пользователя. Листинг реализации входа приводится в приложении В.

При отсутствии аккаунта пользователь может зарегистрироваться. В приложении Г приводится реализация регистрации.

## 5. Тестирование, проверка работоспособности и анализ полученных результатов

Тестирование приложения — это проведение проверки и оценки соответствия между реальным и ожидаемым поведением программы.

### Цели тестирования:

- убедиться, что ПО отвечает заявленным требованиям;
- выявить ошибки, несоответствия или нежелательное поведение.

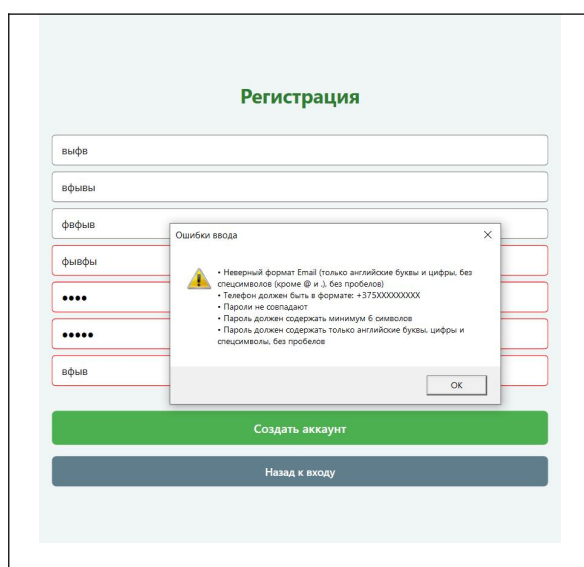
### Задачи тестирования:

- предотвратить как можно больше дефектов;
- проверить, что известные дефекты устранены;
- проверить, что при устранении известных дефектов, не было внесены новые дефекты.

Для корректной работы программы обрабатываются возможные ошибки. Поскольку приложение использует базу данных, некорректные или отсутствующие данные могут привести к сбоям.

## 5.1 Тестирование регистрации и авторизации

Сначала протестируем регистрацию, введя неверные данные. В результате отобразится сообщение об ошибках, как показано на рисунке 5.1.



### Рисунок 5.1 — Ошибка регистрации

Попробуем зарегистрироваться под email пользователя, который уже существует в базе данных приложения.

На рисунке 5.2 демонстрируется окно ошибки, в котором выводится соответствующее сообщение.

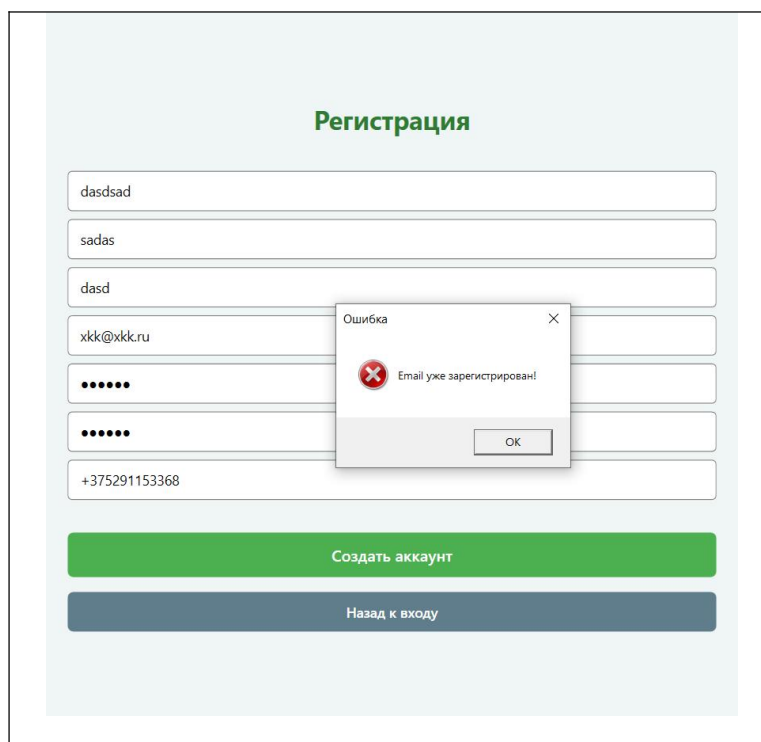


Рисунок 5.2 — Ошибка регистрации (почта занята другим пользователем)

Если же пользователь вводит несуществующий email появится соответствующее сообщение, показанное на рисунке 5.3.

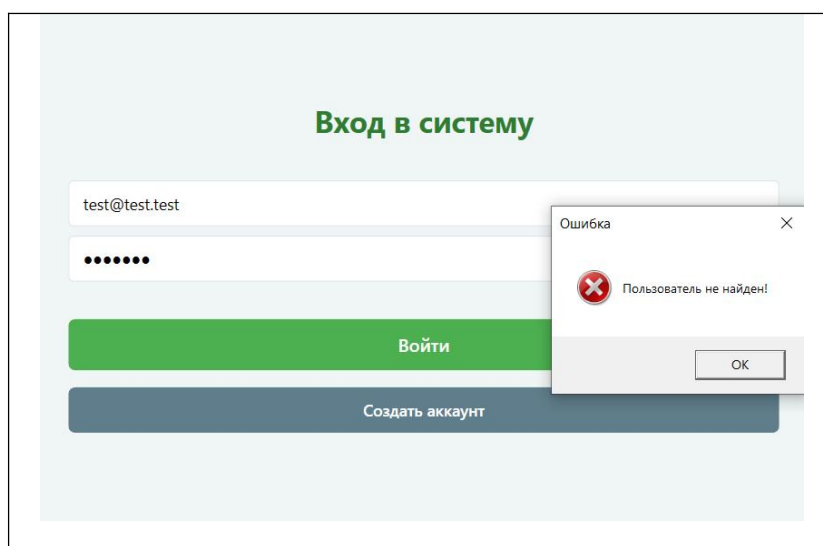


Рисунок 5.3 — Ошибка входа(введён несуществующий email)

Если же пользователь вводит корректный email, но неверный пароль появится соответствующее сообщение, показанное на рисунке 5.4.

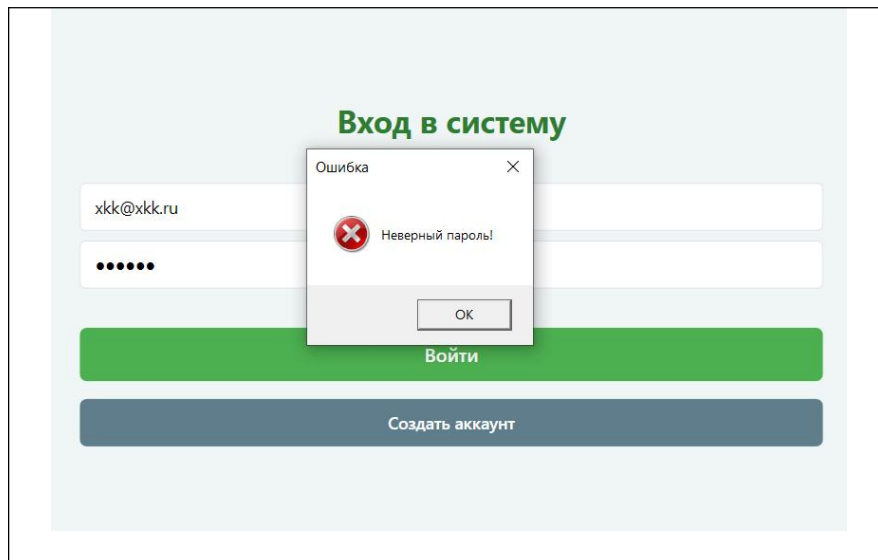


Рисунок 5.4 — Ошибка входа(введён некорректный пароль)

Таким образом, приложение предотвращает поступление некорректных данных в базу данных и препятствует возникновению пользователей с одинаковыми адресами электронной почты.

## 5.2 Тестирование панелей поиска и бронирования билета

Допустим, пользователь случайно выбирает прошедшую дату для предотвращения недопонимания выдаём соответствующее сообщение. Результат тестирования представлен на рисунке 5.5.

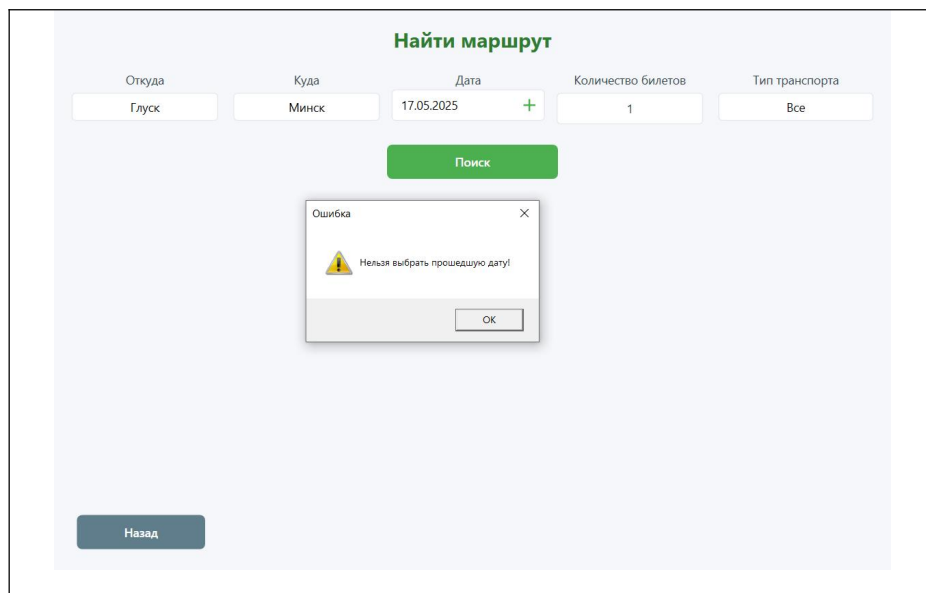


Рисунок 5.5 — Ошибка выбора прошедшей даты

При попытке повторного добавления в избранное получаем соответствующее сообщение(при повторном бронировании поведение



аналогичное). В результате тестирования мы получим сообщение, представленное на рисунке 5.6.

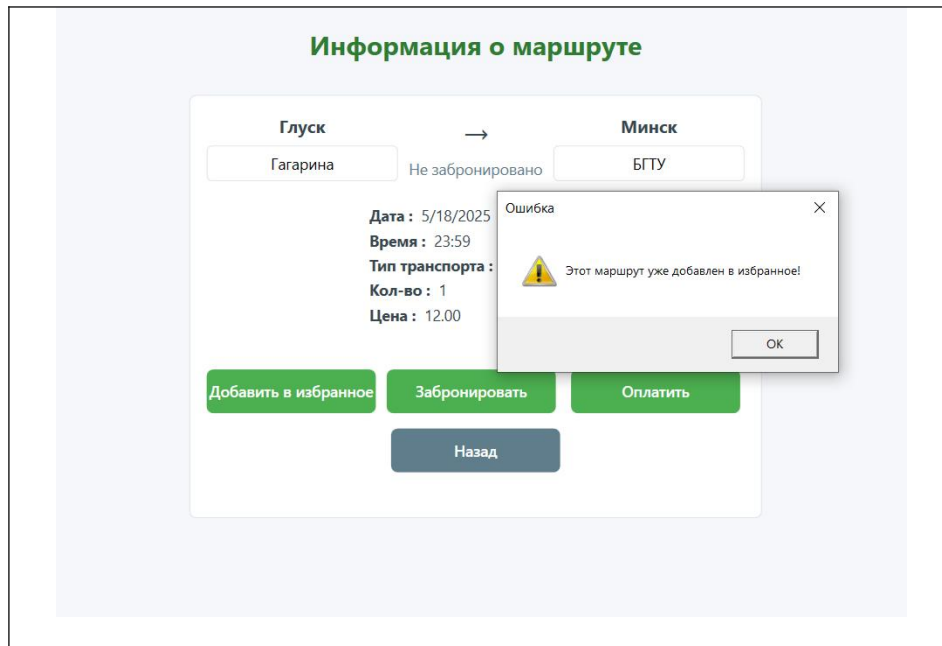


Рисунок 5.6 — Ошибка повторного добавления

При попытке оплаты незабронированного билета получаем сообщение, что сначала нужно забронировать маршрут. В результате тестирования мы получим сообщение, представленное на рисунке 5.7.

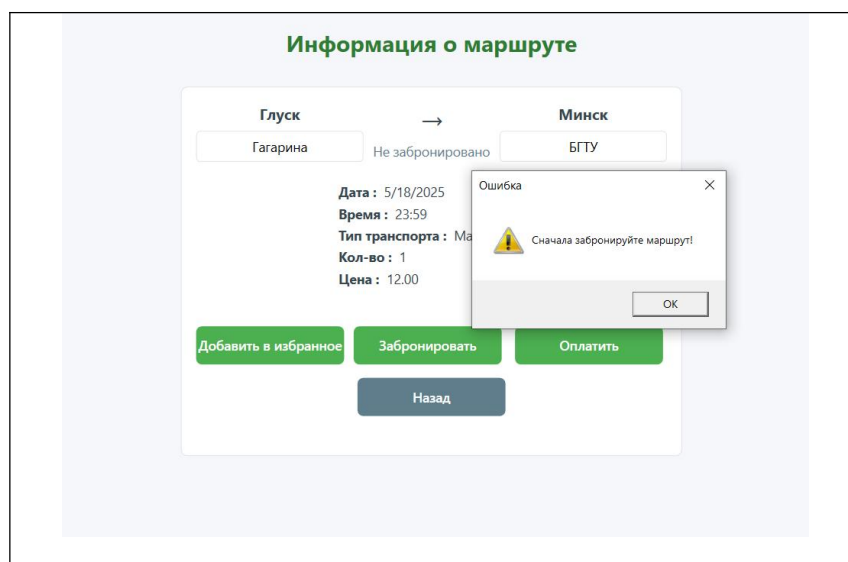


Рисунок 5.7 — Ошибка оплаты до бронирования

Таким образом мы протестировали работоспособность панелей поиска и бронирования билета.

### 5.3 Тестирование панели администратора

Теперь проведём тестирование панели администратора. Для начала попробуем создать маршрут с некорректными данными. Результат тестирования в виде сообщения об ошибке представлен на рисунке 5.8.

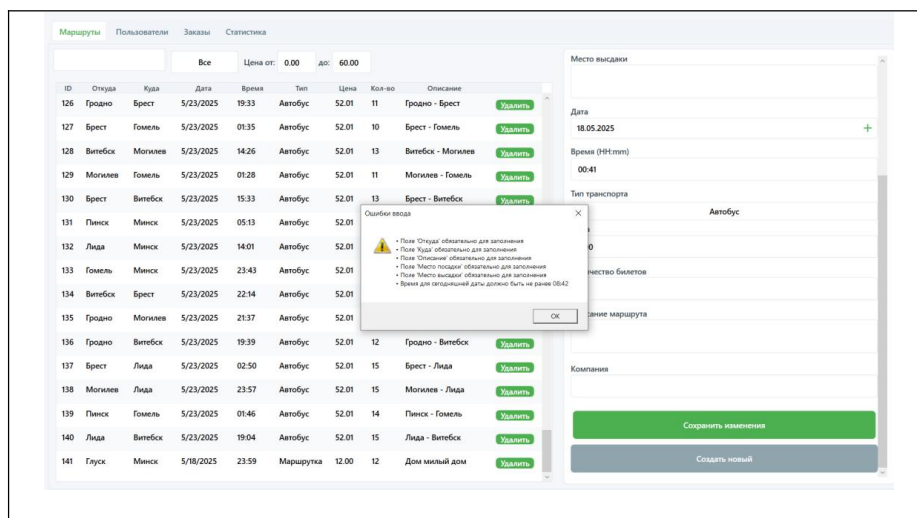


Рисунок 5.8 — Ошибка добавления маршрута

Теперь попробуем создать пользователя с некорректными данными. Результат тестирования в виде сообщения об ошибке представлен на рисунке 5.9.

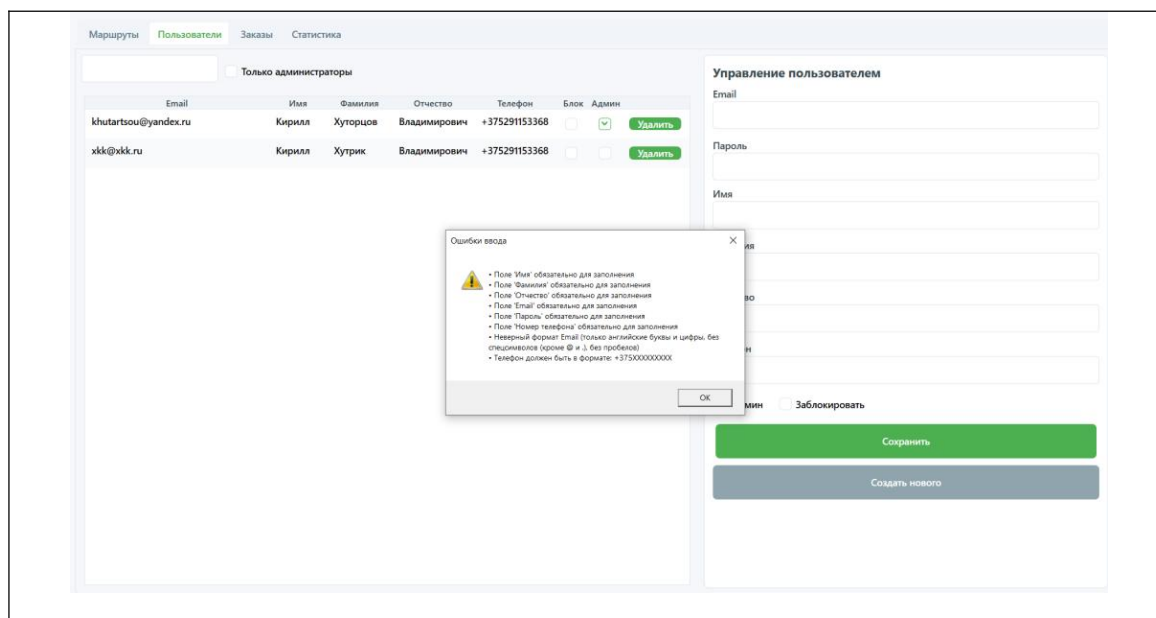


Рисунок 5.9 — Ошибка добавления пользователя

Таким образом мы провели тестирование попытки некорректных действий в панели администратора.

## 6 Руководство по установке и использованию

При запуске приложения мы видим стартовое окно входа(рисунок 6.1), если у нас есть учётная запись нужно ввести email и пароль затем нажать на кнопку «Войти», если учётной записи нету нажимаем кнопку «Создать аккаунт».

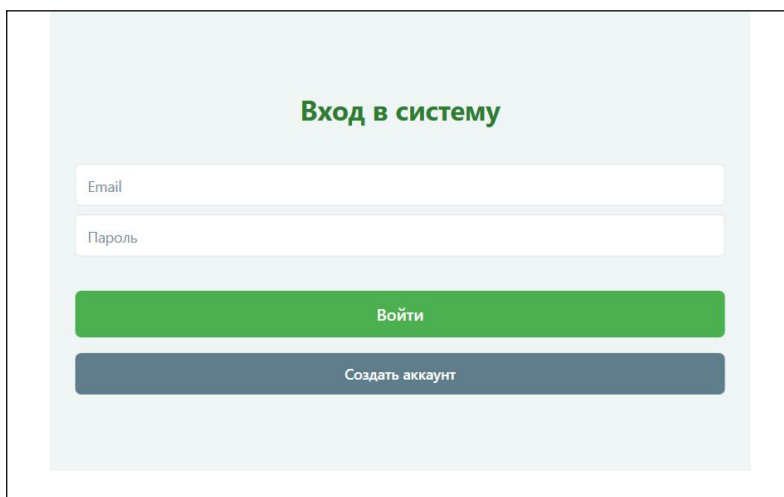
The image shows a login interface with a light blue background. At the top, the title "Вход в систему" is displayed in green. Below the title are two white input fields: "Email" and "Пароль". Underneath these fields are two buttons: a green button labeled "Войти" and a dark blue button labeled "Создать аккаунт".

Рисунок 6.1 — Стартовое окно

Вид окна регистрации показан на рисунке 6.2.

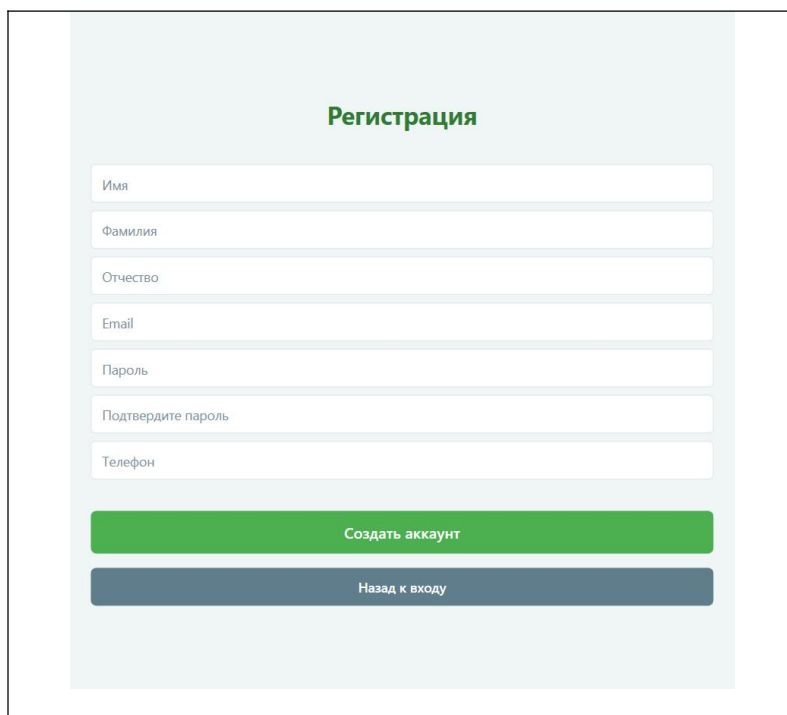
The image shows a registration interface with a light blue background. At the top, the title "Регистрация" is displayed in green. Below the title are seven white input fields: "Имя", "Фамилия", "Отчество", "Email", "Пароль", "Подтвердите пароль", and "Телефон". Underneath these fields are two buttons: a green button labeled "Создать аккаунт" and a dark blue button labeled "Назад к входу".

Рисунок 6.2 — Окно регистрации

После успешной регистрации нам нужно будет вернуться в окно входа и ввести введённые при регистрации email и пароль.

Если мы войдём в приложение под аккаунтом обычного пользователя, то увидим страницу с меню. Данная страница представлена на рисунке 6.3.

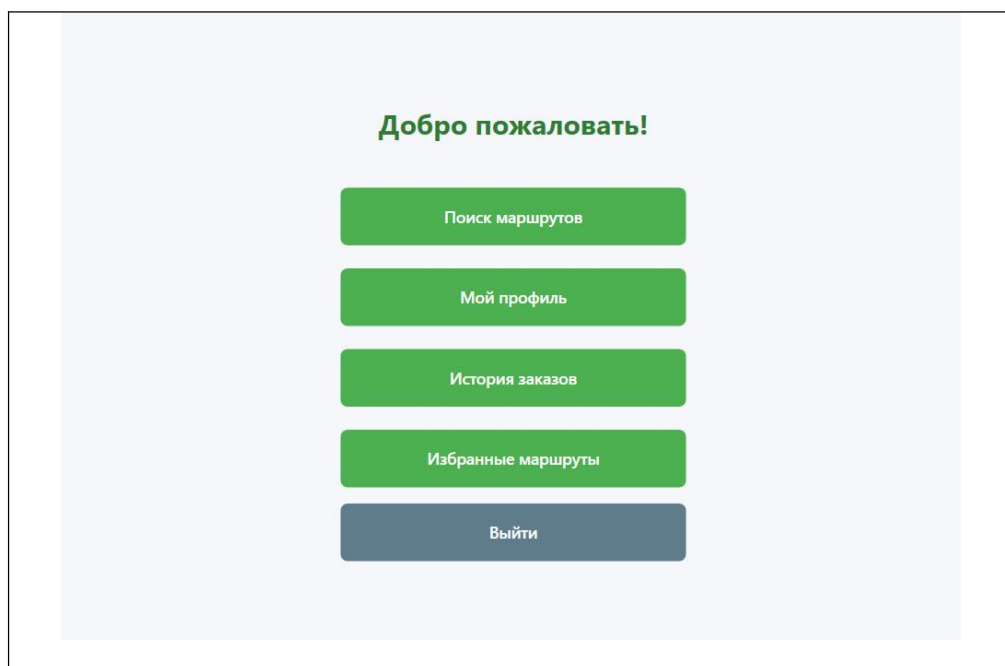


Рисунок 6.3 — Страница меню

При нажатии кнопки «Поиск маршрутов» соответственно перейдём на страницу поиска маршрутов. Данная страница представлена на рисунке 6.4.

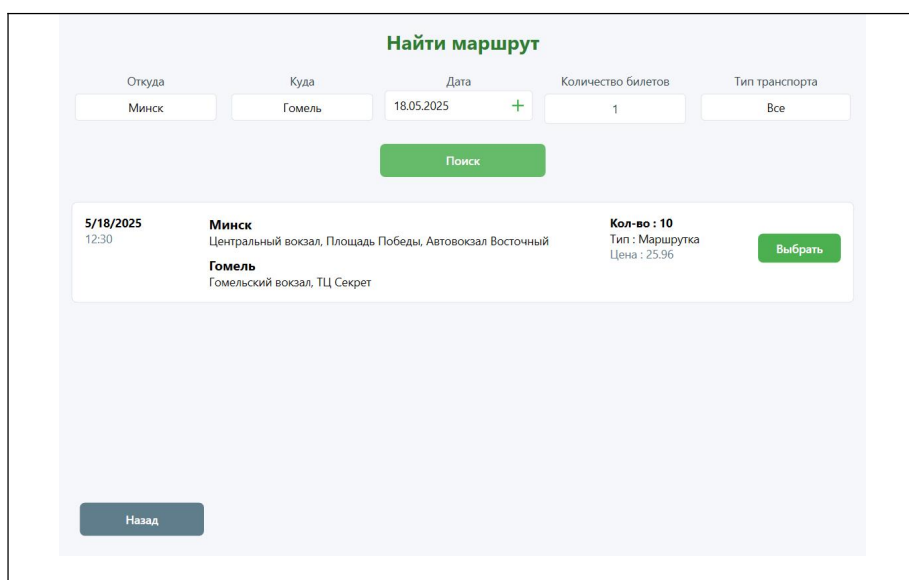


Рисунок 6.4 — Страница поиска маршрутов

При нажатии кнопки «Выбрать» мы перейдём на страницу с выбором места посадки и места высадки, а также с возможностью

добавления в избранное, бронирования и перехода на страницу оплаты. Данная страница представлена на рисунке 6.5.

The screenshot shows a web interface titled "Информация о маршруте" (Route Information). It displays a route from "Минск" (Minsk) to "Гомель" (Gomel). The starting point is "Центральный вокзал" (Central Station) and the ending point is "Гомельский вокзал" (Gomel Station). The status is "Не забронировано" (Not booked). The trip details are: "Дата: 5/18/2025" (Date: 5/18/2025), "Время: 12:30" (Time: 12:30), "Тип транспорта: Маршрутка" (Vehicle type: Shuttle), "Кол-во: 1" (Quantity: 1), and "Цена: 25.96" (Price: 25.96). At the bottom, there are four buttons: "Добавить в избранное" (Add to favorites), "Забронировать" (Book), "Оплатить" (Pay), and "Назад" (Back).

Рисунок 6.5 — Страница с действиям с маршрутом

При нажатии кнопки «Добавить в избранное» маршрут добавится в избранное, при нажатии кнопки «Забронировать» маршрут добавится в историю заказов, после бронирования и нажатии кнопки «Оплатить» мы перейдём на страницу оплаты. Вид страницы оплаты представлен на рисунке 6.6.

The screenshot shows a web interface titled "Оплата билета" (Payment). On the left, there is a dark blue credit card placeholder with labels "Держатель" (Cardholder) and "Срок действия" (Expiration date). On the right, there is a form with the following fields: "Сумма (BYN)" (Amount (BYN)) with the value "25.96", "Номер карты" (Card number), "Имя держателя" (Cardholder name), "Срок действия (ММYY)" (Expiration date (MMYY)), and "CVV". Below the form, there is a small note: "\* Номер карты: 16 цифр, имя: латинские буквы (макс. 30 символов), срок: ММYY, CVV: 3-4 цифры". At the bottom right, there are two buttons: "Оплатить" (Pay) and "Отмена" (Cancel).

Рисунок 6.6 — Страница оплаты

После успешной оплаты пользователь возвращается на страницу меню и при нажатии кнопки «История заказов» попадает

соответственно на страницу с историей заказов, где может просмотреть, отменить и оценить заказ. Страница истории заказов представлена на рисунке 6.7.

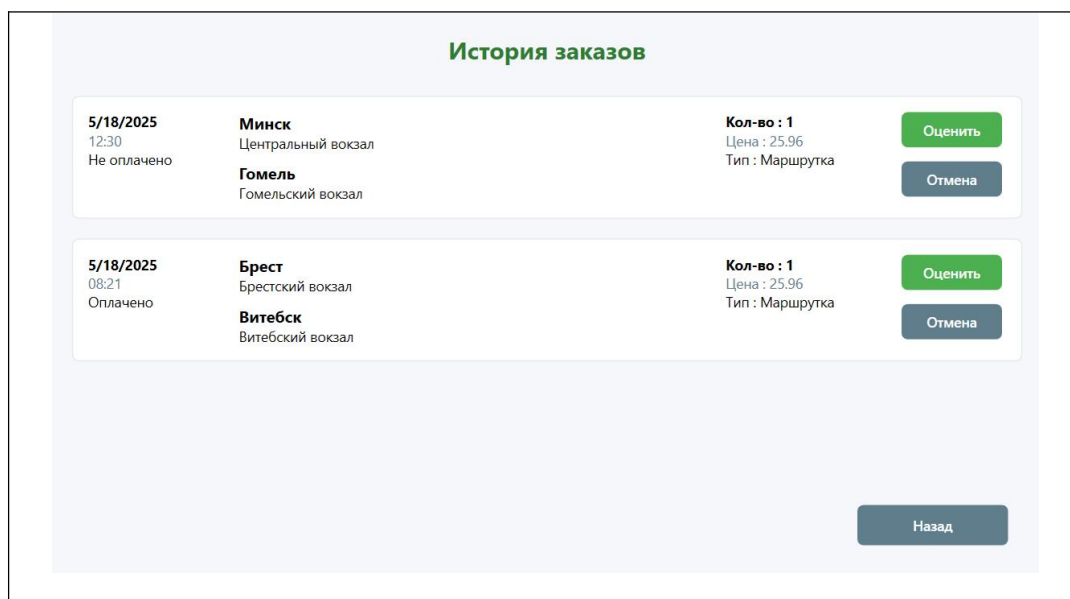


Рисунок 6.7 — Страница истории заказов

При нажатии кнопки «Оценить» перейти на страницу с возможностью выставления оценки и написания комментария. Страница с оценкой и комментарием представлена на рисунке 6.8.

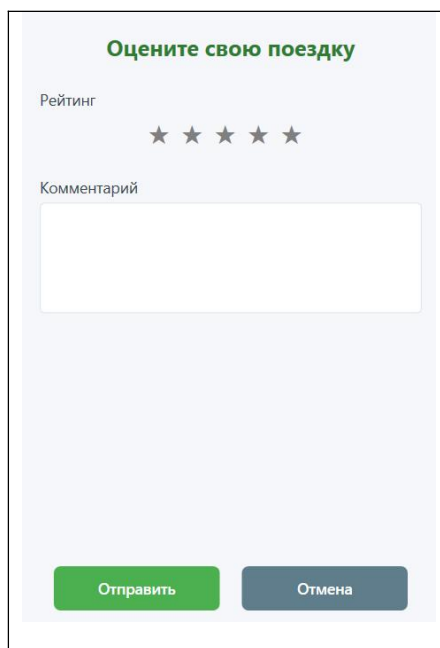
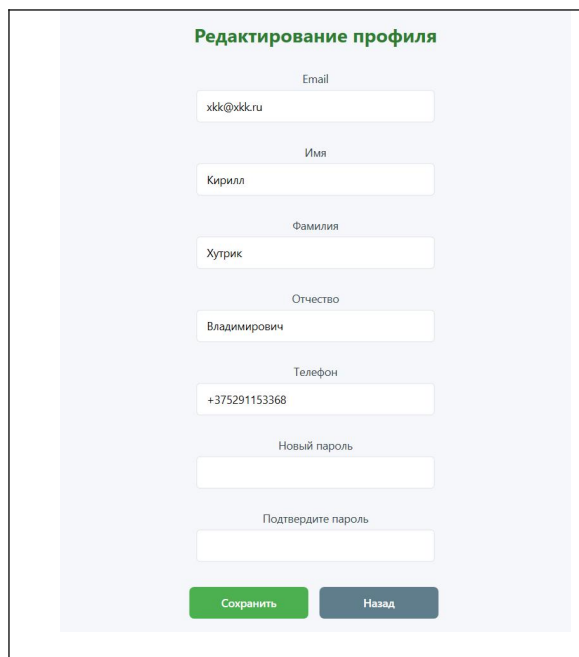


Рисунок 6.8 — Страница с оценкой и комментарием

Далее рассмотрим страницу личного кабинета, которая открывается при нажатии кнопки «Мой профиль» в меню, на

которой пользователь может просмотреть и изменить свои данные. Страница личного кабинета представлена на рисунке 6.9.



**Редактирование профиля**

Email  
xxx@xxx.ru

Имя  
Кирилл

Фамилия  
Хутрик

Отчество  
Владимирович

Телефон  
+375291153368

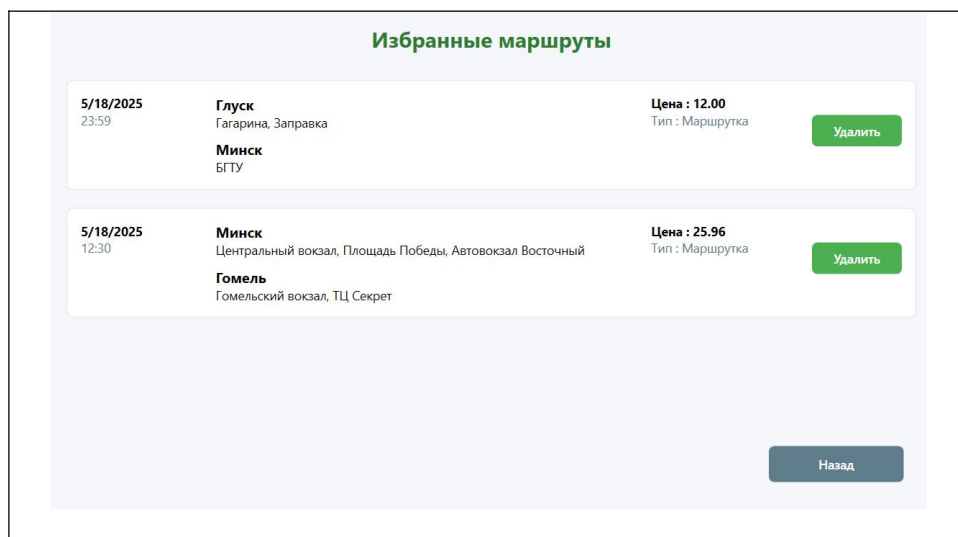
Новый пароль

Подтвердите пароль

Сохранить Назад

Рисунок 6.9 — Страница личного кабинета

Теперь рассмотрим страницу избранных маршрутов, которая открывается при нажатии кнопки «Избранные маршруты» в меню, на которой пользователь может просмотреть и удалить избранные маршруты. Данная страница представлена на рисунке 6.10.



**Избранные маршруты**

5/18/2025 23:59	<b>Глуск</b> Гагарина, Заправка <b>Минск</b> БГТУ	Цена : 12.00 Тип : Маршрутка	Удалить
5/18/2025 12:30	<b>Минск</b> Центральный вокзал, Площадь Победы, Автовокзал Восточный <b>Гомель</b> Гомельский вокзал, ТЦ Секрет	Цена : 25.96 Тип : Маршрутка	Удалить

Назад

Рисунок 6.10 — Страница избранных маршрутов

Можно было заметить, что при разработке пользовательского интерфейса, особый акцент был направлен на проработку интуитивно понятного интерфейса для любой аудитории в любом возрасте.

## ЗАКЛЮЧЕНИЕ

Данное курсовое проектирование было выполнено с целью разработки программного средства, которое позволяет пользователям искать и бронировать билеты на маршрутные транспортные средства, а также администратору управлять базой данных и обеспечивать техническую поддержку.

В процессе выполнения проекта были достигнуты следующие результаты:

- разработано интуитивно понятное пользовательское приложение с удобным интерфейсом для просмотра и бронирования билетов;

- разработана административная часть для управления маршрутами, пользователями, заказами пользователей и просмотра статистики по разным направлениям;

- произведено тестирование программного средства для проверки его функциональности и корректности работы.

Для реализации приложения был выбран язык программирования C# и технология Windows Presentation Foundation (WPF), обеспечивающая разработку удобного и современного пользовательского интерфейса. В качестве базы данных была использована MS SQL Server.

В процессе разработки программного средства был применен паттерн проектирования MVVM (Model-View-ViewModel). MVVM является одним из наиболее популярных паттернов для разработки пользовательских интерфейсов в технологии WPF.

В процессе разработки программного средства ViewModel была активно использована для связи пользовательского интерфейса с бизнес-логикой и данными. Благодаря MVVM, разработка стала более структурированной и управляемой, а код стал более поддерживаемым и расширяемым.

Таким образом, применение паттерна MVVM в программном средстве обеспечило эффективное управление данными и логикой приложения, улучшенную архитектуру и обеспечило высокую отзывчивость и удобство использования пользовательского интерфейса.

В результате выполнения данного курсового проекта было создано функциональное и удобное программное средство. Программа имеет потенциал для дальнейшего развития и расширения функциональности в соответствии с потребностями пользователей.



### **Список использованных источников**

1. Microsoft Visual Studio [Электронный ресурс] – [https://ru.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio) – Дата доступа 13.03.2024
2. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. Режим доступа: <https://metanit.com/sharp/tutorial/> – Дата доступа: 13.03.2024
3. Руководство по WPF // [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/wpf/> – Дата доступа: 15.03.2024
4. Руководство по XAML // [Электронный ресурс]. – Режим доступа: <https://www.tutorialspoint.com/xaml/index.htm> – Дата доступа: 15.04.2024
5. Блинова, Е.А. Курс лекций по Бадам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.

## ПРИЛОЖЕНИЕ А

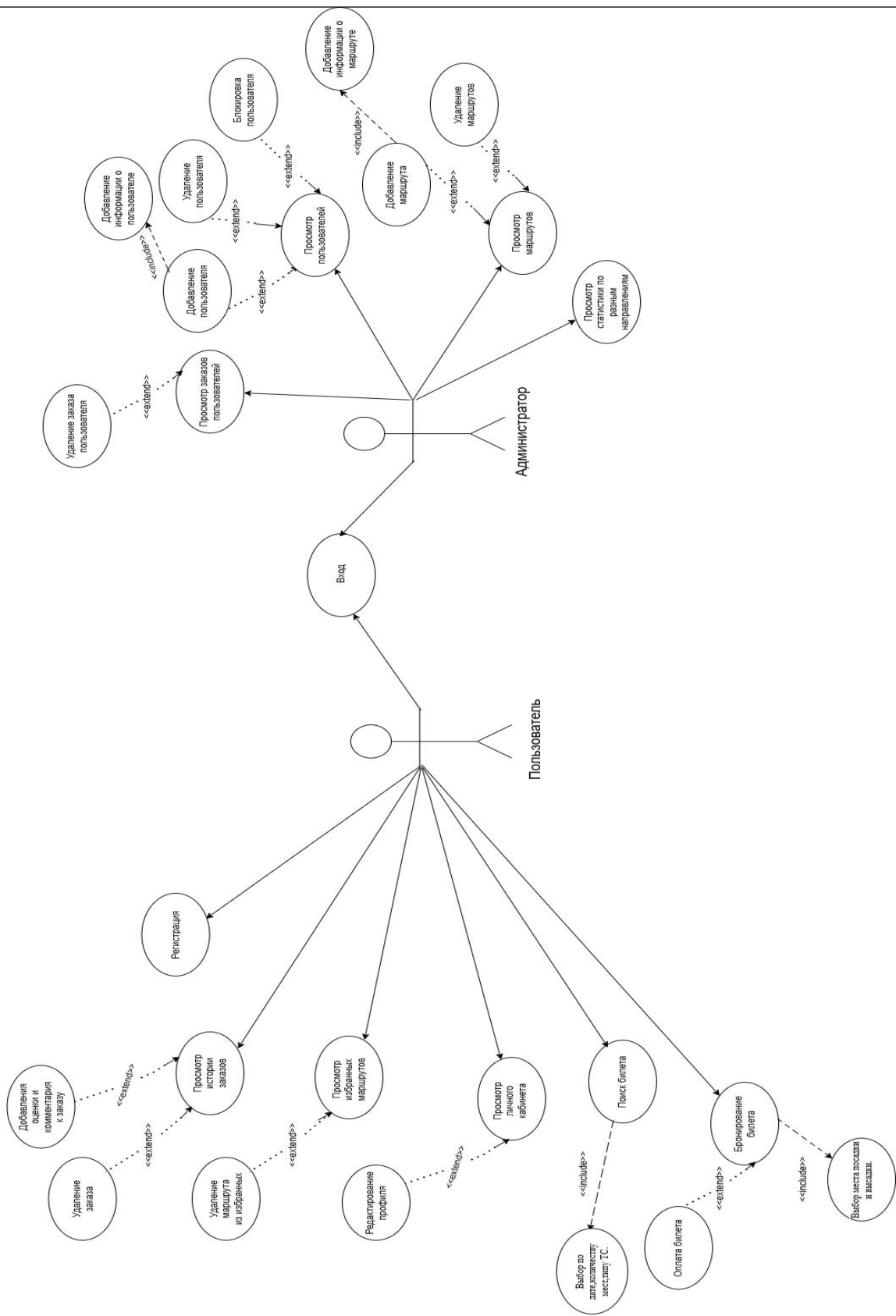


Рисунок 1 — Диаграмма вариантов использования

## ПРИЛОЖЕНИЕ Б



Рисунок 1 – Диаграмма классов (сгенерированная)

## ПРИЛОЖЕНИЕ В

```
using lab4wpf5oop;
using System;
using System.Data.SqlClient;
using System.Security.Cryptography;
using System.Text;
using System.Text.RegularExpressions;
using System.Windows;
using System.Windows.Controls;
using lab4wpf5oop.Models;

namespace RouteBookingSystem
{
    public partial class LoginWindow : Window
    {
        private readonly string connectionString =
"Server=DESKTOP-
DMJJTUE\\SQLEXPRESS;Database=XKKBUS;Trusted_Connection=Tr
ue;";

        public LoginWindow()
        {
            InitializeComponent();
            this.Icon = new
System.Windows.Media.Imaging.BitmapImage(new
Uri("C:/BSTU/SEM4/OOP/lab4wpf5oop/lab4wpf5oop/Images/bus_
icon-icons.com_76529.ico"));
        }

        private void LoginButton_Click(object sender,
RoutedEventArgs e)
        {
            string email = txtEmail.Text.Trim();
            string password =
txtPassword.Password.Trim();

            if (string.IsNullOrEmpty(email) ||
string.IsNullOrEmpty(password))
            {
                MessageBox.Show("Введите email и
пароль!", "Ошибка", MessageBoxButton.OK,
MessageBoxImage.Warning);
                return;
            }

            if (!IsValidEmail(email))
            {
                MessageBox.Show("Неверный формат Email
(только английские буквы и цифры, без спецсимволов (кроме
@ и .), без пробелов)", "Ошибка", MessageBoxButton.OK,
```

```

MessageBoxImage.Error);
        return;
    }

    if (!IsValidPassword(password))
    {
        MessageBox.Show("Неверный формат пароля  
(минимум 6 символов, только английские буквы, цифры и  
спецсимволы, без пробелов)", "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }

    try
    {
        using (SqlConnection conn = new
SqlConnection(connectionString))
        {
            conn.Open();
            string query = "SELECT FirstName,
LastName, Surname, Email, PasswordHash, PhoneNumber,
isAdmin, IsBlocked FROM Users WHERE Email = @Email";
            using (SqlCommand cmd = new
SqlCommand(query, conn))

cmd.Parameters.AddWithValue("@Email", email);
            using (SqlDataReader reader =
cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    string storedHash =
reader.GetString(4); // PasswordHash
                    bool isAdmin =
reader.GetBoolean(6); // isAdmin
                    bool isBlocked =
reader.GetBoolean(7); // IsBlocked

                    if (isBlocked)
                    {
                        MessageBox.Show("Ваш  
аккаунт заблокирован!", "Ошибка", MessageBoxButton.OK,
                        MessageBoxImage.Error);

                        return;
                    }

                    if
(VerifyPassword(password, storedHash))
                    {
                        // Создаем объект

```

```

User с данными из базы

new User

reader.GetString(0),
reader.GetString(1),
reader.GetString(2),
reader.GetString(3),
storedHash,
reader.IsDBNull(5) ? null : reader.GetString(5),
isAdmin,
isBlocked // Добавляем новое поле

User currentUser =
{
    FirstName =
    LastName =
    Surname =
    Email =
    PasswordHash =
    PhoneNumber =
    IsAdmin =
    IsBlocked =
};

MessageBox.Show("Вход
выполнен успешно!", "Успех", MessageBoxButtons.OK,
MessageBoxImage.Information);

OpenUserWindow(isAdmin, currentUser);
}
else
{
    MessageBox.Show("Неверный пароль!", "Ошибка",
    MessageBoxButtons.OK, MessageBoxImage.Error);
}
else
{
    MessageBox.Show("Пользователь не найден!", "Ошибка",
    MessageBoxButtons.OK, MessageBoxImage.Error);
}
}
}
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка подключения к
базе данных: {ex.Message}", "Ошибка",
    MessageBoxButtons.OK, MessageBoxImage.Error);
}
}

```

```

    }

    private void OpenUserWindow(bool isAdmin, User
currentUser)
    {
        Window nextWindow = isAdmin ? new
AdminWindow(currentUser) : new
UserDashboardWindow(currentUser.Email);
        nextWindow.Show();
        this.Close();
    }

    private bool VerifyPassword(string inputPassword,
string storedHash)
    {
        using (SHA256 sha256 = SHA256.Create())
        {
            byte[] inputBytes =
Encoding.UTF8.GetBytes(inputPassword);
            byte[] hashedBytes =
sha256.ComputeHash(inputBytes);
            string hashedInput =
Convert.ToBase64String(hashedBytes);

            return hashedInput == storedHash;
        }
    }

    private bool IsValidEmail(string email)
    {
        return Regex.IsMatch(email, @"^[a-zA-Z][a-zA-
Z0-9]*@[a-zA-Z][a-zA-Z0-9]*\.[a-zA-Z]+$");
    }

    private bool IsValidPassword(string password)
    {
        return !string.IsNullOrEmpty(password) &&
password.Length >= 6 &&
            Regex.IsMatch(password, @"^[a-zA-Z0-
9!@#$$%^&*()+\-=\[\]\{\};:'\"", .<?>]+$") &&
            !password.Contains(" ");
    }

    private void RegisterButton_Click(object sender,
RoutedEventArgs e)
    {
        var registrationWindow = new
RegistrationWindow();
        registrationWindow.Show();
        this.Close();
    }

```

```
        private void txtEmail_TextChanged(object sender,
TextChangedEventArgs e)
        {
            if (txtEmailPlaceholder != null)
            {
                txtEmailPlaceholder.Visibility =
string.IsNullOrEmpty(txtEmail.Text) ?
Visibility.Visible : Visibility.Collapsed;
            }
        }
        private void txtPassword_PasswordChanged(object
sender, RoutedEventArgs e)
        {
            if (txtPasswordPlaceholder != null)
            {
                txtPasswordPlaceholder.Visibility =
string.IsNullOrEmpty(txtPassword.Password) ?
Visibility.Visible : Visibility.Collapsed;
            }
        }
    }
}
```



## ПРИЛОЖЕНИЕ Г

```
using System;
using System.Data.SqlClient;
using System.Security.Cryptography;
using System.Text;
using System.Text.RegularExpressions;
using System.Windows;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Input;

namespace RouteBookingSystem
{
    public partial class RegistrationWindow : Window,
    INotifyPropertyChanged
    {
        private string _firstName, _lastName, _surname,
        _email, _phone;

        public string FirstName
        {
            get => _firstName;
            set { _firstName = value;
OnPropertyChanged(); }
        }

        public string LastName
        {
            get => _lastName;
            set { _lastName = value;
OnPropertyChanged(); }
        }

        public string Surname
        {
            get => _surname;
            set { _surname = value;
OnPropertyChanged(); }
        }

        public string Email
        {
            get => _email;
            set { _email = value; OnPropertyChanged(); }
        }

        public string Phone
```

```

        {
            get => _phone;
            set { _phone = value; OnPropertyChanged(); }
        }

        private readonly string _connectionString =
"Server=DESKTOP-
DMJJTUE\\SQLEXPRESS;Database=XKKBUS;Trusted_Connection=Tr
ue;";

        public RegistrationWindow()
        {
            InitializeComponent();
            this.Icon = new BitmapImage(new
Uri("C:/BSTU/SEM4/OOP/lab4wpf5oop/lab4wpf5oop/Images/bus_
icon-icons.com_76529.ico"));
            DataContext = this;
            InitializePlaceholders();
        }

        private void InitializePlaceholders()
        {
            txtFirstName.TextChanged += (s, e) =>
UpdatePlaceholderVisibility(txtFirstName,
firstNamePlaceholder);
            txtLastName.TextChanged += (s, e) =>
UpdatePlaceholderVisibility(txtLastName,
lastNamePlaceholder);
            txtSurname.TextChanged += (s, e) =>
UpdatePlaceholderVisibility(txtSurname,
surnamePlaceholder);
            txtEmail.TextChanged += (s, e) =>
UpdatePlaceholderVisibility(txtEmail, emailPlaceholder);
            txtPhone.TextChanged += (s, e) =>
UpdatePlaceholderVisibility(txtPhone, phonePlaceholder);
        }

        private void UpdatePlaceholderVisibility(TextBox
textBox, TextBlock placeholder)
        {
            placeholder.Visibility =
string.IsNullOrEmpty(textBox.Text) ? Visibility.Visible :
Visibility.Collapsed;
        }

        private void RegisterButton_Click(object sender,
RoutedEventArgs e)
        {
            ResetFieldBorders();
            if (!ValidateInputs()) return;
        }

```

```

        // Дополнительная проверка перед выполнением
SQL
        if (string.IsNullOrEmpty(FirstName) ||
string.IsNullOrEmpty(LastName) ||
            string.IsNullOrEmpty(Surname) ||
string.IsNullOrEmpty(Phone) ||
            string.IsNullOrEmpty(Email) ||
string.IsNullOrEmpty(txtPassword.Password))
        {
            MessageBox.Show("Все обязательные поля
(Имя, Фамилия, Отчество, Email, Телефон, Пароль) должны
быть заполнены.",
                                "Ошибки          ввода",
MessageBoxButton.OK, MessageBoxImage.Warning);
            return;
        }

        try
        {
            if (IsEmailExists(Email))
            {
                MessageBox.Show("Email          уже
зарегистрирован!", "Ошибка", MessageBoxButton.OK,
MessageBoxImage.Error);
                HighlightErrorField(txtEmail);
                return;
            }

            using (var conn = new
SqlConnection(_connectionString))
            {
                conn.Open();
                const string query = @"INSERT INTO
Users
                                (FirstName,
LastName, Surname, Email, PasswordHash, PhoneNumber,
isAdmin, IsBlocked)
                                VALUES
                                (@FirstName,
@LastName, @Surname, @Email, @PasswordHash, @Phone, 0,
0)";

                using (var cmd = new
SqlCommand(query, conn))
                {

cmd.Parameters.AddWithValue("@FirstName", FirstName);
cmd.Parameters.AddWithValue("@LastName", LastName);
cmd.Parameters.AddWithValue("@Surname", Surname);

```

```

cmd.Parameters.AddWithValue("@Email", Email);

cmd.Parameters.AddWithValue("@PasswordHash",
    HashPassword(txtPassword.Password));

cmd.Parameters.AddWithValue("@Phone", Phone);

cmd.Parameters.AddWithValue("@IsBlocked", 0); //
Устанавливаем IsBlocked = 0 по умолчанию

        cmd.ExecuteNonQuery();
    }
}

    MessageBox.Show("Регистрация успешна!",
"Успех",
    MessageBoxImage.Information);
    new LoginWindow().Show();
    this.Close();
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка: {ex.Message}",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
}

private bool ValidateInputs()
{
    var errors = new StringBuilder();
    bool isValid = true;

    // Обязательные поля
    CheckField(FirstName, "Имя", txtFirstName,
errors);
    CheckField(LastName, "Фамилия", txtLastName,
errors);
    CheckField(Surname, "Отчество", txtSurname,
errors);
    CheckField(Email, "Email", txtEmail, errors);
    CheckField(Phone, "Телефон", txtPhone,
errors);

    // Пароль
    if
(string.IsNullOrEmpty(txtPassword.Password))
    {
        errors.AppendLine("• Поле 'Пароль'
обязательно");
        HighlightErrorField(txtPassword);
    }
}

```

```

        isValid = false;
    }

    // Проверка формата имени, фамилии, отчества
    (только русские и английские буквы)
    if (!string.IsNullOrEmpty(FirstName)
    && !Regex.IsMatch(FirstName, @"^[a-яA-ЯёЁa-zA-Z]+$"))
    {
        errors.AppendLine("• Имя должно содержать
только русские или английские буквы");
        HighlightErrorField(txtFirstName);
        isValid = false;
    }

    if (!string.IsNullOrEmpty(LastName)
    && !Regex.IsMatch(LastName, @"^[a-яA-ЯёЁa-zA-Z]+$"))
    {
        errors.AppendLine("• Фамилия должна
содержать только русские или английские буквы");
        HighlightErrorField(txtLastName);
        isValid = false;
    }

    if (!string.IsNullOrEmpty(Surname)
    && !Regex.IsMatch(Surname, @"^[a-яA-ЯёЁa-zA-Z]+$"))
    {
        errors.AppendLine("• Отчество должно
содержать только русские или английские буквы");
        HighlightErrorField(txtSurname);
        isValid = false;
    }

    // Формат Email
    if (!string.IsNullOrEmpty(Email)
    && !Regex.IsMatch(Email, @"^[a-zA-Z][a-zA-Z0-9]*@[a-zA-
Z][a-zA-Z0-9]*\.[a-zA-Z]+$"))
    {
        errors.AppendLine("• Неверный формат
Email (только английские буквы и цифры, без спецсимволов
(кроме @ и .), без пробелов)");
        HighlightErrorField(txtEmail);
        isValid = false;
    }

    // Формат телефона (+375XXXXXXXXXX)
    if (!string.IsNullOrEmpty(Phone)
    && !Regex.IsMatch(Phone, @"^\+375(25|29|33|44)\d{7}$"))
    {
        errors.AppendLine("• Телефон должен быть
в формате: +375XXXXXXXXXX");
        HighlightErrorField(txtPhone);
    }

```

```

        isValid = false;
    }

    // Пароли
    if (txtPassword.Password !=
txtConfirmPassword.Password)
    {
        errors.AppendLine("• Пароли не
совпадают");
        HighlightErrorField(txtPassword);
        HighlightErrorField(txtConfirmPassword);
        isValid = false;
    }

    if
(!string.IsNullOrEmpty(txtPassword.Password))
    {
        if (txtPassword.Password.Length < 6)
        {
            errors.AppendLine("• Пароль должен
содержать минимум 6 символов");
            HighlightErrorField(txtPassword);
            isValid = false;
        }

        if (!Regex.IsMatch(txtPassword.Password,
@"^[a-zA-Z0-9!@#$$%^&*()+\-=\[\]\{\};:'\"", .<>?]+") ||
txtPassword.Password.Contains(" "))
        {
            errors.AppendLine("• Пароль должен
содержать только английские буквы, цифры и спецсимволы,
без пробелов");
            HighlightErrorField(txtPassword);
            isValid = false;
        }
    }

    if (!isValid)
MessageBox.Show(errors.ToString(), "Ошибки ввода",
MessageBoxButton.OK, MessageBoxImage.Warning);
    return isValid;
}

private void CheckField(string value, string
fieldName, Control control, StringBuilder errors)
{
    if (string.IsNullOrEmpty(value))
    {
        errors.AppendLine($"• Поле '{fieldName}'
обязательно");
        HighlightErrorField(control);
    }
}

```

```

    }
}

private void HighlightErrorField(Control control)
{
    control.BorderBrush = Brushes.Red;
    control.BorderThickness = new Thickness(1);
}

private void ResetFieldBorders()
{
    Control[] fields = { txtFirstName,
txtLastName, txtSurname, txtEmail, txtPhone, txtPassword,
txtConfirmPassword };
    foreach (var field in fields)
    {
        field.BorderBrush = Brushes.Gray;
        field.BorderThickness = new Thickness(1);
    }
}

private string HashPassword(string password)
{
    using (var sha256 = SHA256.Create())
    {
        byte[] bytes =
Encoding.UTF8.GetBytes(password);
        byte[] hash = sha256.ComputeHash(bytes);
        return Convert.ToBase64String(hash);
    }
}

private bool IsEmailExists(string email)
{
    using (var conn = new
SqlConnection(_connectionString))
    {
        conn.Open();
        var cmd = new SqlCommand("SELECT COUNT(*)
FROM Users WHERE Email = @Email", conn);
        cmd.Parameters.AddWithValue("@Email",
email);
        return (int)cmd.ExecuteScalar() > 0;
    }
}

private void LoginButton_Click(object sender,
RoutedEventArgs e)
{
    new LoginWindow().Show();
    this.Close();
}

```

```

    }

    private void txtPassword_PasswordChanged(object
sender, RoutedEventArgs e)
    {
        passwordPlaceholder.Visibility
string.IsNullOrEmpty(txtPassword.Password)
Visibility.Visible : Visibility.Collapsed;
    }

    private void
txtConfirmPassword_PasswordChanged(object sender,
RoutedEventArgs e)
    {
        confirmPasswordPlaceholder.Visibility
string.IsNullOrEmpty(txtConfirmPassword.Password)
Visibility.Visible : Visibility.Collapsed;
    }

    public event PropertyChangedEventHandler
PropertyChanged;
    protected void
OnPropertyChanged([CallerMemberName] string name = null)
    {
        PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(name));
    }
}

```