МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 05 01 Информационные системы и технологии

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема: Программное средство «Сервис продажи туристических путевок»

Исполнитель

студент (ка) 3 курса группы 2 Дрозд Алексей Игоревич

(Ф.И.О.)

Руководитель работы преподаватель-стажер Якунович А. В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель

(подпись)

Минск 2023

**Содержание**

[Введение 4](#_Toc154148274)

[1 Обзор литературы и аналогов 6](#_Toc154148275)

[1.1 АталанТур 6](#_Toc154148276)

[1.2 Coral Travel 7](#_Toc154148277)

[1.3 Вывод 8](#_Toc154148278)

[2 Анализ требований к программному средству и разработка функциональных требований 9](#_Toc154148279)

[2.1 Определение требований к программному средству 9](#_Toc154148280)

[2.2 Описание средств разработки 10](#_Toc154148281)

[2.3 Описание функциональности программного средства 11](#_Toc154148282)

[2.4 Спецификация функциональных требований 11](#_Toc154148283)

[2.5 Паттерн Репозиторий 12](#_Toc154148284)

[2.6 Вывод по разделу 13](#_Toc154148285)

[3 Проектирование программного средства 14](#_Toc154148286)

[3.1 Модель базы данных 14](#_Toc154148287)

[3.2 Общая структура программного средства 15](#_Toc154148288)

[3.2.1 Структура приложения администратора 15](#_Toc154148289)

[3.2.2 Структура пользовательского приложения 18](#_Toc154148290)

[3.3 Взаимоотношение между классами 21](#_Toc154148291)

[3.4 Проектирование последовательности проекта 21](#_Toc154148292)

[3.5 Вывод по разделу 22](#_Toc154148293)

[4 Реализация программного средства 23](#_Toc154148294)

[4.1 Авторизация и регистрация 23](#_Toc154148295)

[4.2 Оплата бронирования 23](#_Toc154148296)

[4.3 Отмена бронирования 24](#_Toc154148297)

[4.4 Вывод по разделу 24](#_Toc154148298)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 25](#_Toc154148299)

[6 Руководство по использованию 29](#_Toc154148300)

[Заключение 33](#_Toc154148301)

[Список использованных источников 34](#_Toc154148302)

[Приложение А 35](#_Toc154148303)

[Приложение Б 36](#_Toc154148304)

[Приложение В 37](#_Toc154148305)

# Введение

Целью данного курсового проекта является разработка современного приложения для продажи туристических путевок, ориентированного на удовлетворение потребностей клиентов и облегчение работы туроператоров. Основной задачей проекта является создание десктопного приложения, предоставляющего возможность просмотра, выбора и бронирования различных путевок, находящихся на сервисе.

В ходе выполнения проекта особое внимание будет обращено на создании инновационного приложения, способного удовлетворить потребности как клиентов, так и туроператоров, а также обеспечить безопасность и надежность взаимодействия между ними.

Для разработки приложения был выбран обширный API-интерфейс — Windows Presentation Foundation (WPF). Данный инструмент предназначен для создания настольных программ с насыщенным графическим интерфейсом пользователя. В процессе разработки использовался язык программирования C#, который обладает синтаксисом, схожим с языком программирования C, и предназначен для создания приложений на платформе Microsoft .NET Framework.

Хранение данных осуществляется в системе управления базами данных Microsoft SQL Server. Это обеспечивает надежное и эффективное хранение информации о номерах, бронированиях и клиентах отеля, обеспечивая приложение надежностью и быстродействием в работе с данными.

# 1 Обзор литературы и аналогов

Путешествия — это одно из самых захватывающих приключений в жизни. Каждый раз, когда мы отправляемся в путешествие, мы получаем незабываемые впечатления, расширяем свой кругозор, насыщаем себя новыми знаниями и узнаем много интересного о мире, в котором мы живем.

Путешествия могут быть разнообразными - от коротких выходных на природе до масштабных мировых туров. Независимо от того, где вы едете и сколько времени проводите в пути, путешествие всегда представляет собой возможность вырваться из обыденности и погрузиться в новые впечатления.

Кроме того, путешествия могут быть полезны не только для души, но и для физического и психического здоровья. Они позволяют расслабиться, отвлечься от стрессов и проблем, а также поддерживать физическую форму за счет активного отдыха и разнообразной деятельности.

Одним из основных преимуществ туристических фирм является возможность приобретения готового тура в интересующую страну без необходимости самостоятельно организовывать поездку (бронирование отеля, поиск билетов на самолет и т. д.). Это делает туристические агентства очень востребованными среди большинства людей по всему миру.

## 1.1 АталанТур

Туристическая компания АлатанТур успешно работает на рынке с 1993 года и является одной из крупнейших в Беларуси. На данный момент АлатанТур располагает тремя офисами в Минске, а также семью офисами в крупных городах Республики Беларусь. Пример интерфейса данного сервиса представлен на рисунке 1.1.

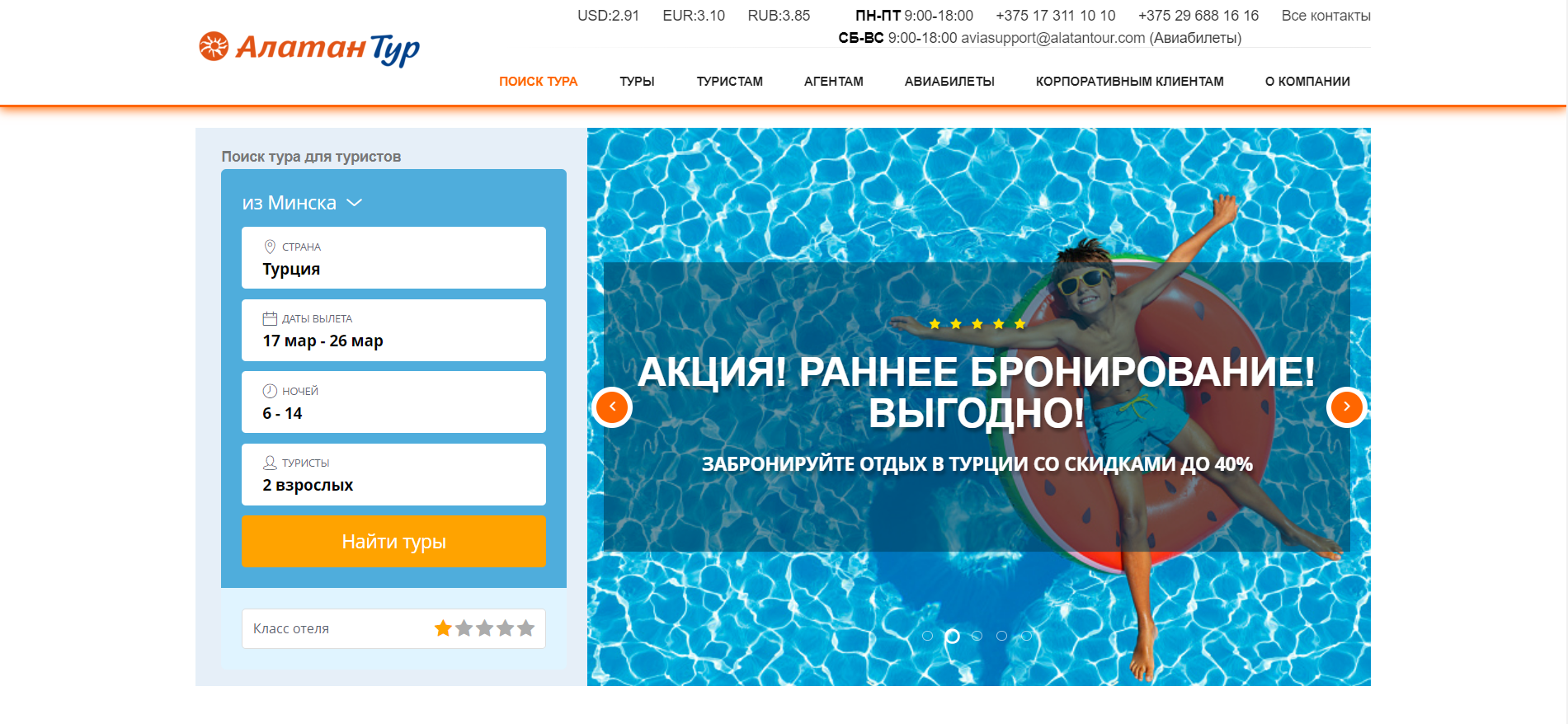


Рисунок 1.1 – Интерфейс сервиса АталанТур

Одной из основных функций, предложенных на сервисе АталанТур является поиск организованных туров. Пользователи могут искать интересующий тур исходя из предложенных агентством.

Также пользователи могут сами указывать необходимые параметры для создания путевки. Для этого им необходимо заполнить всю информацию о создаваемом туре: выбор страны, выбор качества отеля, выбор города проживания, указание интересующих дат и прочее.

Дополнительно АталанТур предоставляет возможность покупки авиабилетов для юридических лиц. При выборе билетов клиент самостоятельно указывает необходимую дату, время, начальную и конечную точку маршрута.

## 1.2 Coral Travel

Еще одним из аналогов туристического агентства, рассмотренного в ходе обзора, является CORAL TRAVEL. CORAL TRAVEL предлагает лучшие курорты и отели в 30 странах мира, среди которых – Турция, Испания, Греция, Таиланд, Тунис, Марокко, Израиль, ОАЭ, Куба, Индия, Маврикий, Танзания, Доминиканская Республика, Индонезия, Мальдивы, Вьетнам, Сейшелы, Шри-Ланка, Сингапур, Мексика, Камбоджа, Иордания, Андорра, Австрия, Азербайджан, Грузия. Идет постоянная работа по открытию новых направлений. Туроператор организует групповые и индивидуальные туры FIT на базе собственных чартерных программ и регулярных рейсов, занимается развитием incentive-, congress-, спортивный и другие виды туризма.

Пользователям также предлагаются готовые туры, которые они могут приобрести в короткое время. Помимо этого, на сервисе присутствуют предложения «горячих» туров.

Аналогично агентству АталанТур, CORAL TRAVEL предоставляет пользователям возможность подбора собственного тура, настраиваемого лично пользователем. Пример интерфейса данного сервиса представлен на рисунке 1.2.

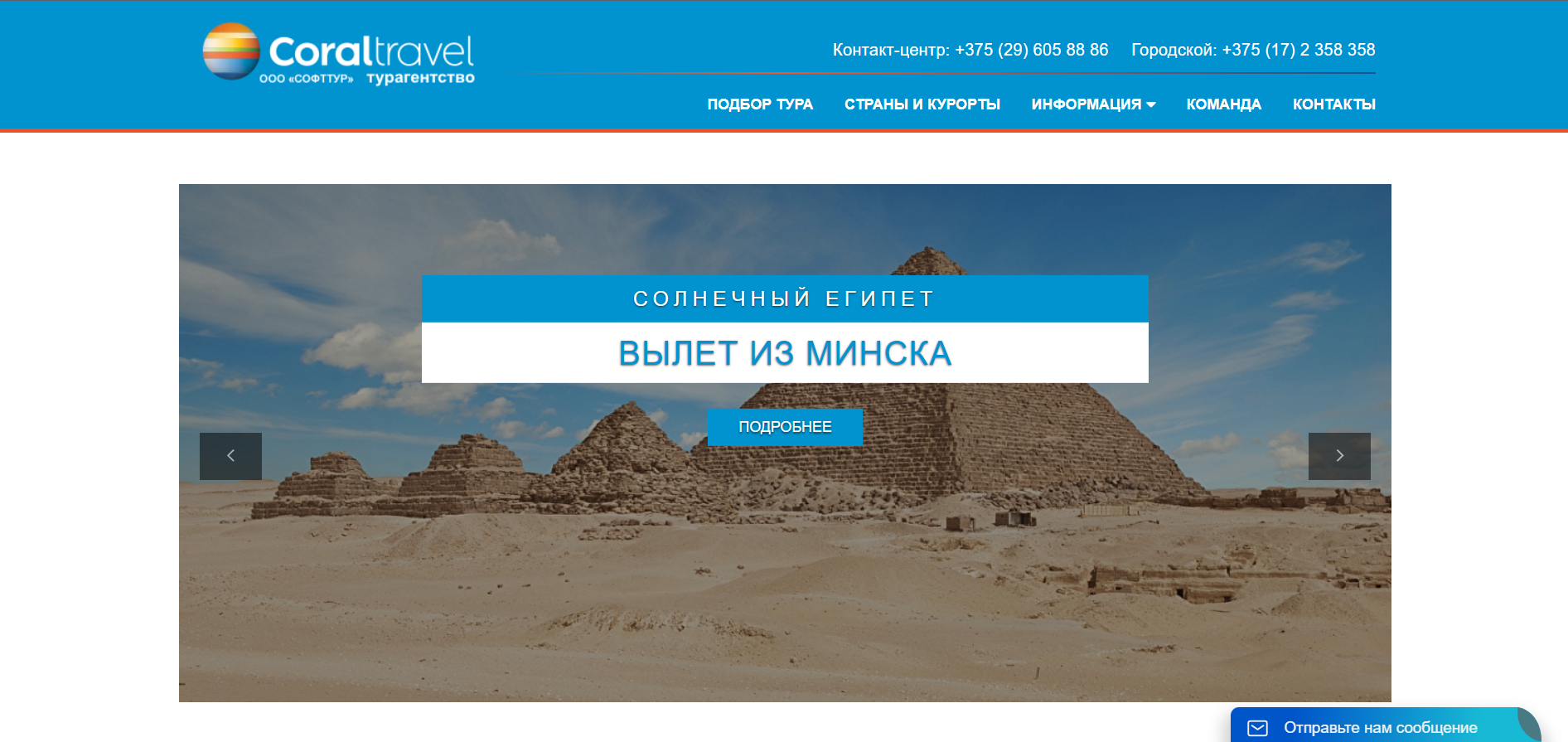


Рисунок 1.2 – Интерфейс сервиса CORAL TRAVEL

## 1.3 Вывод

В подразделах 1.1 и 1.2 были проведены аналитические обзоры двух туристических агентств – АталанТур и CORAL TRAVEL. Оба сервиса предоставляют пользователям возможность приобретения готовых туров или создания своего собственного с необходимыми параметрами. На основе аналитического обзора этих двух аналогов можно сделать несколько заметок, которые следует реализовать в приложении:

* отсутствие личного кабинета пользователя: на обоих представленных аналогах пользователю не предоставлена возможность доступа к своему личному кабинету, поэтому будет более разумно предоставить ему возможность иметь некоторую информацию о своих бронированиях и оплатах, произведенных ранее;
* простота и понятность пользовательского интерфейса: понимание, что сложные и запутанные процессы бронирования или оплаты могут отпугнуть клиентов, подчеркивает необходимость создания простого, понятного и удобного интерфейса. Это поможет сделать процессы более привлекательными и комфортными для пользователей;

Этот проект направлен на упрощение работы туристической компании и улучшение опыта клиентов при приобретении туристических путевок.

# 2 Анализ требований к программному средству и разработка функциональных требований

Проектирование архитектуры представляет собой ключевой этап в процессе разработки, который определяет выбор современных технологий и инструментов управления кодом. На этом этапе мы также структурируем базу данных, определяем типы данных и устанавливаем взаимосвязи между таблицами для эффективного хранения информации о клиентах, полисах и условиях страхования. Создание удобного интерфейса с окнами для заполнения заявок и просмотра полисов является важной составляющей этого процесса.

Добавление диаграммы вариантов использования (Use Case) помогает в наглядном представлении сценариев, охватывая действия как пользователей, так и системы, в различных ситуациях. Это дает возможность лучше понять потребности пользователей и гарантировать, что функциональные требования соответствуют их ожиданиям. Важно отметить, что этот этап не ограничивается только выбором инструментов и структурированием данных, но также включает в себя разработку удобного интерфейса для конечных пользователей и внедрение инструментов, способствующих более глубокому пониманию и соответствию потребностям пользователей, обеспечивая эффективность и функциональность всей системы.

## 2.1 Определение требований к программному средству

При проведении анализа были вынесены следующие функциональные требования:

* осуществление валидации вводимых данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* функция поиска путевок с использованием различных фильтров, таких как место назначения, даты, стоимость и другие критерии;
* возможность просматривать детали о каждой туристической путевке, включая описание, цену, даты и место назначения;
* возможность для пользователей создать учетную запись и войти в систему;
* система должна предоставлять уведомления о бронированиях, изменениях и других событиях;
* возможность управления информацией о клиентах и туристических путевках, включая создание, редактирование и удаление профилей клиентов и путевок.

## 2.2 Описание средств разработки

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2022 [[1]](#_Список_использованных_источников);
* программная платформа .NET Framework 7;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework Core 6;
* СУБД Microsoft SQL Server.

В качестве интерфейса прикладного программирования был выбран обширный API-интерфейс – Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом. В основе графической технологии WPF лежит мощная инфраструктура, основанная на DirectX. Это является одним из основных отличий WPF от более ранней технологии создания пользовательских интерфейсов – Windows Forms [[2]](#_Список_использованных_источников).

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом – С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

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 [[3]](#_Список_использованных_источников).

Поскольку разработка приложения выполняется на языке программирования C#, требуется промежуточное программное средство для взаимодействия с базой данных. В данном случае была выбрана технология Entity Framework. Эта технология представляет собой объектно-ориентированный подход к доступу к данным и представляет собой решение для отображения объектов на реляционные базы данных (ORM) на платформе .NET Framework. Entity Framework позволяет выполнять запросы к данным с использованием LINQ to Entities, что упрощает и ускоряет процесс взаимодействия с базой данных [[4]](#_Список_использованных_источников).

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 10 на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения. Основной используемый язык запросов — Transact-SQL — реализован на структурированном языке запросов (SQL) с расширениями.

## 2.3 Описание функциональности программного средства

Описание функциональности программного средства представлено с помощью UML-диаграммы вариантов использования.

UML-диаграмма — это графическое представление модели объектно-ориентированной системы, используемое для описания ее структуры, поведения и взаимодействия между ее компонентами. UML-диаграммы могут включать в себя различные элементы, такие как классы, объекты, интерфейсы, отношения между объектами, состояния и переходы, взаимодействия между объектами и т.д.

Спецификация функциональности программного средства представлена в следующем пункте.

## 2.4 Спецификация функциональных требований

Для функциональности ПС необходимо создание базы данных для хранения информации приложения.

При запуске программного средства «Сервис продажи туристических путевок» необходимо реализовать регистрацию и авторизацию пользователей для дальнейшего использования приложения. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Для регистрации на сервисе необходимо придумать логин, пароль и личные данные, которые будут занесены в базу данных и могут быть использованы при следующем входе в приложении.

После авторизации пользователю будут доступны все функциональные возможности: бронирование определенного тура, поиск путевки по интересующим параметрам, просмотр своей истории платежей – все это будет реализовано посредством различных окон.

Для поиска интересующей путевки для пользователя будет добавлена отдельное окно, хранящее все данные о путевках, размещенных на сервисе в данный момент. На данной вкладке будут расположены множество фильтров: поиск по стране или городу, фильтрация по цене, поиск всех путевок, подходящих для указанного промежутка времени и многие другие.

При бронировании пользователю будет необходимо выбрать какой-либо определенный вариант путевки, удовлетворяющий всем его требованиям. После нажатия по выбранному варианту, пользователю будет показано более расширенная информация о самой путевки и кнопка бронирования. После нажатия по кнопке, пользователь получит уведомление о подтверждении его бронирования и добавление соответствующей записи в окно, хранящее все бронирования данного пользователя.

Весь функционал пользователя и администратора представлен на рисунке 2.1.

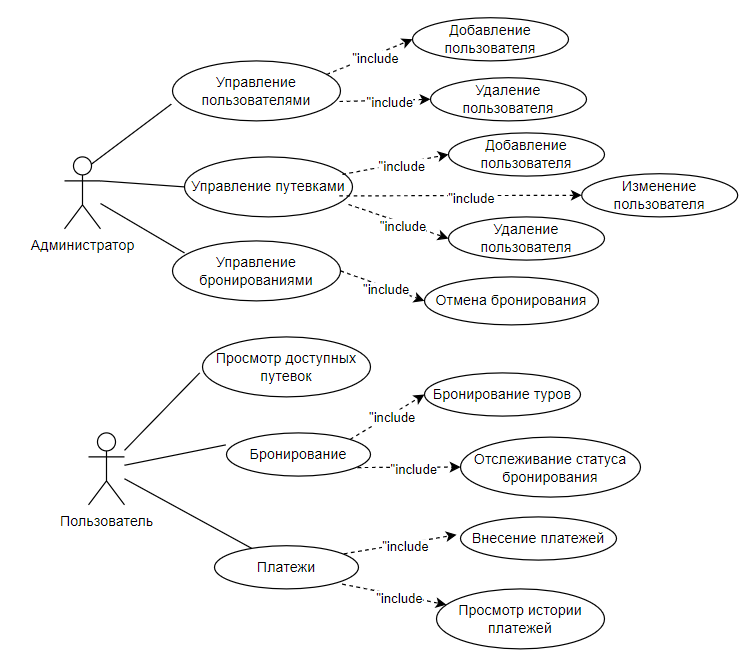


Рисунок 2.1 – UML-диаграмма вариантов использования сервиса для администратора и пользователя

При авторизации от имени администратора пользователю будет доступно полноценное влияние на работу сервиса. Это будет возможно посредством предоставления дополнительных кнопок и иных действий в интерфейсе.

## 2.5 Паттерн Репозиторий

Паттерн «Репозиторий» (Repository) является одним из важных структурных паттернов проектирования программного обеспечения. Его основной целью является изоляция логики доступа к данным от бизнес-логики приложения. Репозиторий предоставляет единый интерфейс для взаимодействия с данными, скрывая детали работы с хранилищем данных, будь то база данных, файловая система или другой источник.

Основные преимущества применения паттерна «Репозиторий» включают повышение модульности и гибкости приложения. Благодаря использованию репозитория, код, отвечающий за работу с данными, выделяется в отдельный слой, что упрощает поддержку и масштабирование системы. Этот подход также способствует повышению читаемости кода и облегчает тестирование, так как бизнес-логика приложения не зависит напрямую от деталей взаимодействия с данными.

Важно отметить, что паттерн «Репозиторий» не является универсальным решением и должен быть применен с учетом особенностей конкретного проекта. Однако, в контексте многих приложений, использование этого паттерна существенно улучшает архитектуру, делая код более чистым, управляемым и масштабируемым [[5]](#_Список_использованных_источников).

## 2.6 Вывод по разделу

Разработка архитектуры играет ключевую роль, поскольку включает в себя выбор передовых технологий и инструментов для управления кодом. Основное внимание уделяется созданию удобного интерфейса для пользователей и внедрению инструментов, которые способствуют лучшему пониманию и соответствию потребностям пользователей, обеспечивая тем самым эффективность и функциональность системы.

Применение современных технологий, таких как Microsoft Visual Studio 2022, .NET Framework, C#, WPF, Entity Framework и SQL Server Manager 2019, позволяет создать мощное приложение с современным интерфейсом и удобным доступом к базе данных. Внедрение паттерна «Репозиторий» упрощает и инкапсулирует логику обращений к модели, что обеспечивает простоту отладки, возможность внесения изменений и улучшает читаемость кода.

# 3 Проектирование программного средства

## 3.1 Модель базы данных

База данных сервиса продажи туристических путевок состоит из 6 таблиц: Payments, Bookings, TourOperators, Packages, Destinations, Customers. Схема данной БД представлена на рисунке 3.1.

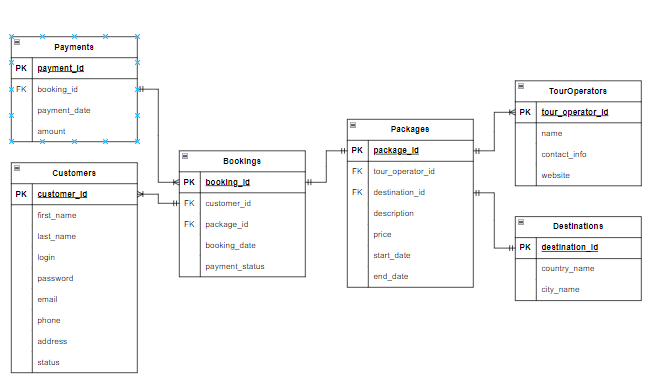


Рисунок 3.1 – Схема базы данных

Таблица Payments используется для хранения информации о произведенных оплатах забронированных путевок. Состоит из 4 столбцов: payment\_id (идентификатор платежа), booking\_id (идентификатор бронирования), payment\_date (дата платежа), amount (сумма платежа).

Таблица Customers хранит в себе данные клиентов сервиса, когда-либо приобретавших путевки. Состоит из 9 столбцов: customer\_id (идентификатор пользователя), first\_name (имя пользователя), last\_name (фамилия пользователя), login (логин пользователя), password (пароль пользователя), email (электронная почта пользователя), phone (телефон пользователя), address (адрес пользователя), status (указывает наличие привилегий администратора).

Таблица Bookings хранит в себе данные о бронировании путевок, их дате бронирования и статусе оплаты. Состоит из 5 столбцов: booking\_id (идентификатор бронирования), customer\_id (идентификатор пользователя), package\_id (идентификатор путевки), booking\_date (дата бронирования), payment\_status (статус оплаты).

Таблица Packages используется для хранения о всех путевках, предоставляемых туристическими операторами и доступными для бронирования пользователями. Состоит из 7 столбцов: package\_id (идентификатор путевки), tour\_operator\_id (идентификатор туроператора), destination\_id (идентификатор места назначения), description (описание путевки), price (цена путевки), start\_date (дата начала), end\_date (дата окончания).

Таблица TourOperators хранит в себе информацию о туристических операторах, предоставляющих путевки. Состоит из 4 столбцов: tour\_operator\_id (идентификатор туроператора), name (наименование туроператора), contact\_info (контактная информация), website (веб-сайт туроператора).

Таблица Destinations хранит в себе данные о странах и городах, используемых для указания конечной точки в путевке. Состоит из 3 столбцов: destination\_id (идентификатор места назначения), country\_name (название страны), city\_name (название города).

## 3.2 Общая структура программного средства

Данный проект создается в виде двух отдельных приложений – для пользователя и администратора. Это обусловлено тем, что пользователи и администраторы обычно имеют различные потребности и задачи при использовании программы. Разделение на два приложения позволяет создать удобный и интуитивно понятный интерфейс для конечных пользователей, облегчая им взаимодействие с функциональностью, специфичной для их потребностей. Также администраторы часто нуждаются в расширенных правах доступа для управления системой, настройки параметров безопасности, анализа журналов и решения проблем.

### 3.2.1 Структура приложения администратора

Приложение администратора имеет структуру, показанную на рисунке 3.2:

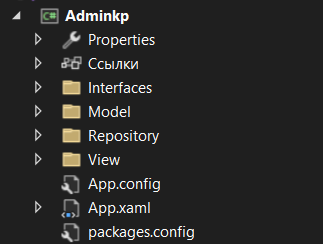


Рисунок 3.2 – Структура приложения администратора

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

Таблица 3.1 – Описание клиенсткой структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя пакета | Содержание |
| Папка Model | Здесь описаны модели, с которыми происходит вся работа в приложении. |
| Папка View | В этой папке хранятся файлы с xaml-разметкой, то есть окна и страницы, а также ресурсы для отображения данных страниц. |
| Папка Interface | Хранятся созданные интерфейсы для репозитория. |
| Папка Repository | В данной папке хранятся все классы для взаимодействия с моделью данных. |
| App.config | Файл конфигурации приложения. |
| App.xaml | Файл, который определяет ресурсы, которые являются общие для приложения. |
| Packages.config | Файл для управления зависимостями и установки сторонних библиотек. |

Более подробная структура содержимого папок «Model» и «Repository» показана на рисунке 3.3.

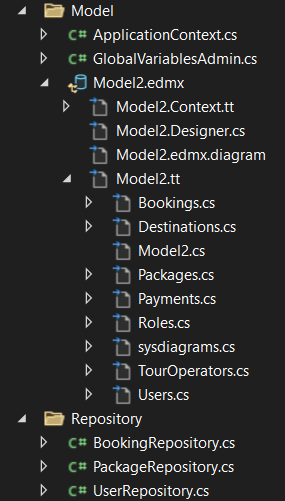
****

Рисунок 3.3 — Подробная структура папок «Model» и «Repository»

Здесь был использован Entity Framework, с помощью которого была создана модель данных на основе созданной базы данных. Описание файлов папки представлено в таблице 3.2.

Таблица 3.2 – Описание файлов папок «Model» и «Repository»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Model2.edmx | Файл модели Entity Framework, который содержит информацию об объектно-ориентированной модели базы данных, включая сущности, атрибуты и отношения между ними. |
| ApplicationContext.cs | Файл модели, который предсоставляет контекс данных, используется для доступа к модели базы данных. |
| GlobalVariablesAdmin.cs | Файл модели, который используется для хранения глобальной переменной идентификатора администратора. |
| BookingRepository.cs | Файл репозитория для доступа к временным сущностям Bookings в базе данных. Он содержит методы для создания, чтения, поиска и удаления объектов Bookings. |
| PackageRepository.cs | Файл репозитория для доступа к сущностям Packages в базе данных. Он содержит методы для создания, чтения, обновления и удаления объектов Packages. |
| UserRepository.cs | Интерфейс для репозитория доступа к сущностям Users в базе данных. Он содержит методы для создания, чтения, и удаления объектов Users. |

Рассмотрим папку «View». Ее подробная структура показана на рисунке 3.4.

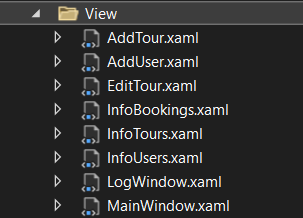


Рисунок 3.4 – Подробная структура папки «View»

Описание файлов папки представлено в таблице 3.3.

Таблица 3.3 – Описание файлов папки «View»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| AddTour.xaml | Окно, которое используется для добавления новой путевки. |
| AddUser.xaml | Окно, которое используется для добавления нового пользователя. |
| EditTour.xaml | Окно, которое используется для изменения параметров существующей путевки. |
| InfoBookings.xaml | Страница, хранящая данные о бронированиях. |
| InfoUsers.xaml | Страница, хранящая данные о пользователях. |
| InfoTours.xaml | Страница, хранящая данные о путевках. |
| LogWindow.xaml | Окно регистрации или авторизации. |
| MainWindow.xaml | Главное окно приложения. |

Структура административной части программного средства разделена на основные папки: Model для описания данных, View для файлов элементов интерфейса, Interface для интерфейсов репозитория, Repository для взаимодействия с данными.

### 3.2.2 Структура пользовательского приложения

Приложение пользователя имеет структуру, показанную на рисунке 3.5.

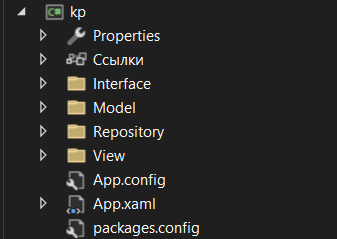


Рисунок 3.5 – Структура пользовательского приложения

Описание папок и файлов приложения пользователя показано в таблице 3.4.

Таблица 3.4 – Описание структуры папок и файлов проекта пользователя

|  |  |
| --- | --- |
| Имя пакета | Содержание |
| Папка Model | Здесь описаны модели, с которыми происходит вся работа в приложении. |
| Папка View | В этой папке хранятся файлы с xaml-разметкой, то есть окна и страницы, а также ресурсы для отображения данных страниц. |
| Папка Interface | Хранятся созданные интерфейсы для репозитория. |
| Папка Repository | В данной папке хранятся все классы для взаимодействия с моделью данных. |
| App.config | Файл конфигурации приложения. |
| App.xaml | Файл, который определяет ресурсы, которые являются общие для приложения. |
| Packages.config | Файл для управления зависимостями и установки сторонних библиотек. |

Более подробная структура содержимого папок «Model» и «Repository» показана на рисунке 3.6.

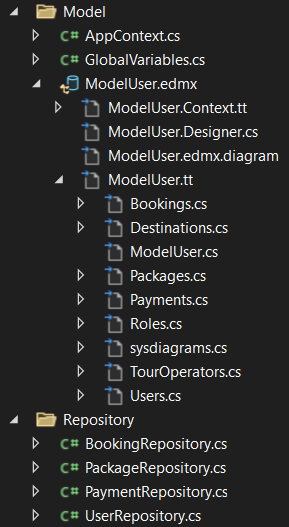
****

Рисунок 3.6 — Подробная структура папок «Model» и «Repository»

Здесь был использован Entity Framework, с помощью которого была создана модель данных на основе созданной базы данных. Описание файлов папки представлено в таблице 3.5.

Таблица 3.5 – Описание файлов папок «Model» и «Repository»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Model1.edmx | Файл модели Entity Framework, который содержит информацию об объектно-ориентированной модели базы данных, включая сущности, атрибуты и отношения между ними. |
| AppContext.cs | Файл модели, который предсоставляет контекс данных, используется для доступа к модели базы данных. |
| GlobalVariables.cs | Файл модели, который используется для хранения глобальной переменной идентификатора пользователя. |
| BookingRepository.cs | Файл репозитория для доступа к временным сущностям Bookings. Он содержит методы для создания, чтения, поиска объектов Booking. |
| PackageRepository.cs | Файл репозитория для доступа к сущностям Packages. Он содержит методы для создания, чтения, поиска объектов Packages. |
| PaymentRepository.cs | Файл репозитория для доступа к сущностям Payments. Он содержит методы для создания, чтения, поиска объектов Payments. |
| UserRepository.cs | Интерфейс для репозитория доступа к сущностям Users. Он содержит методы для создания, чтения и поиска объектов Users. |

Рассмотрим папку «View».Подробная структура указана на рисунке 3.4.

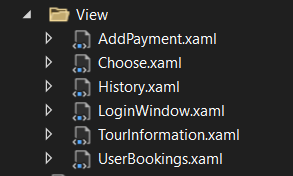


Рисунок 3.4 – Подробная структура папки «View»

Таблица 3.6 – Описание файлов папки «View»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| AddPayment.xaml | Окно, которое используется для оплаты бронирования. |
| Choose.xaml | Основное окно, хранящее в себе все путевки и их фильтрацию. |
| History.xaml | Окно, хранящее историю платежей пользователя. |
| UserBookings.xaml | Окно, содержащая все бронирования для данного пользователя и их статус. |
| TourInformation.xaml | Окно, предоставляющее более детальную информацию о путевке для пользователя |
| LoginWindow.xaml | Окно регистрации или авторизации. |

Структура клиентской части программного средства разделена на основные папки: Model для описания данных, View для файлов элементов интерфейса, Interface для интерфейсов репозитория, Repository для взаимодействия с данными.

## 3.3 Взаимоотношение между классами

Для визуализации взаимосвязей между классами в разрабатываемой программе применяется диаграмма UML. Это графическое представление, представленное в виде связанного графа с вершинами (классами) и ребрами (отношениями) между ними.

Для более детального отображения внутренней структуры программы и взаимосвязей между ее компонентами используется диаграмма классов. В данном приложении особое внимание уделено такому проектированию, где каждый класс выполняет свои функции и почти не зависит от других компонентов. Подробное представление диаграммы классов можно найти в [Приложении А](#_Приложение_А) к данному документу.

## 3.4 Проектирование последовательности проекта

Для наглядного отображения взаимодействия между объектами в различных сценариях использования системы, используется диаграмма последовательностей в UML. Она является эффективным средством визуализации порядка и способа взаимодействия компонентов системы при выполнении определенных функций или использовании сценариев.

Для обозначения времени в диаграмме применяется вертикальная штриховая линия, известная как «линия жизни», которая отражает период существования объекта в системе. Сами объекты представлены в виде прямоугольников, а сообщения, передаваемые между объектами для обмена информацией или выполнения определенных действий, изображаются в виде линий со стрелками, отражающими направление передачи данных или вызова методов между объектами.

Детальное изображение диаграммы последовательностей представлено в [Приложении Б](#_Приложение_Б) к настоящему документу.

## 3.5 Вывод по разделу

В данном разделе были рассмотрены основные компоненты, на которые основано программное средство. Модель базы данных, находящаяся в папке Model, является основой для работы структур частей клиентского приложения и приложения администратора. Обе части программного обеспечения разработаны с учётом данной модели, что обеспечивает эффективное взаимодействие с данными.

# 4 Реализация программного средства

## 4.1 Авторизация и регистрация

Класс «LoginWindow» представляет собой окно входа в приложении пользователя. Главная задача этого класса – регистрация или авторизация пользователя при входе в приложение.

Класс имеет две панели – панель регистрации и авторизации. Первоначально панель авторизации находится в активном состоянии. Для того, чтобы активировать панель регистрации, необходимо вызвать обработчик событий «SwitchStackPanel\_Click», который изменит свойство Visibility для обеих панелей.

В случае, когда пользователь хочет авторизоваться на сервисе, ему необходимо ввести в поля свои логин и пароль. В случае если поля будут пустыми или таких данных не окажется в базе данных – пользователю будут выводиться сообщения с исключением, которое произошло.

При вызове обработчика «LoginButton\_Click» будет осуществлена проверка на нахождение соответствия указанных данных. Для этого используется метод «AuthenticateUser», который изначально сверит нахождение соответствия логина, далее пароля и в конце – роли пользователя. В случае, если его роль не «Пользователь», то метод вернет значение false. Если же все данные сойдутся, идентификатор пользователя будет занесен в класс «GlobalVariables» и откроется окно «Choose».

При регистрации алгоритм входа в систему отличается. Изначально пользователь должен ввести новый логин, пароль, имя, фамилию и электронный адрес. Все эти поля проверяются на валидность, используя условия и методы «IsAllLetters» и «IsValidEmail». Далее вызывается метод «RegisterUser», который проверяет наличие такого же логина в базе данных и возвращает false в случае нахождения соответствия. Если же соответствие не найдено, то создается новый пользователь, идентификатор которого также заносится в класс «GlobalVariables» и открывается окно «Choose».

Все основные методы класса «LoginWindow» представлены в [приложении В](#_Приложение_В).

## 4.2 Оплата бронирования

Класс «AddPayment» предоставляет возможность пользователю произвести оплату бронирования, которое он совершил ранее.

Первоначально пользователю открывается окно, хранящее в себе детальную информацию о бронировании, которое он выбрал. Вверху ему предоставляется навигационное меню, состоящее из 4 обработчиков событий – «MainButton\_Click», «HistoryButton\_Click», «BookingButton\_Click», «LeftButton\_Click». Каждый из этих обработчиков перенаправит пользователя в другое окно. Помимо этого, пользователю доступен еще один обработчик, «Payment\_Click». При его вызове программа обратится к репозиторию «PaymentRepository» для использования метода «MakePayment», который отвечает за создание нового экземпляра «Payments».

Внутри данного метода происходит проверка на наличие идентификатора бронирования, переданного во входных параметрах. В случае, если данный идентификатор будет не найден, то пользователю покажется сообщение о том, что данное бронирование не найдено. Если же бронирование будет определено, произойдет вторая проверка на статус данного бронирования. Если оно имеет статус оплачено, что пользователь также получит сообщение об этом. В противном случае будет создан новый экземпляр Payments.

Все основные методы класса «AddPayment» представлены в [приложении В](#_Приложение_В).

## 4.3 Отмена бронирования

Класс «InfoBookings» предоставляет возможность администратору произвести отмену бронирования, совершенного пользователем.

Для этого администратору необходимо выбрать строку из панели DataGrid с названием «iform». В случае вызова обработчика «CancelButton\_Click» без выбранной строки, администратор поулчит сообщение о том, что необходимо выбрать какое-то бронирование. Если же строка выбран и вызван обработчик, то сразу же происходит обращение к методу «CancelBooking», находящемуся в репозитории «BookingRepository».

Далее происходит проверка на статус выбранного бронирования. В случае, если оно уже оплачено, администратор получит уведомление об этом. В противном случае указанное бронирование будет удалено.

Все основные методы класса «InfoBookings» представлены в [приложении В](#_Приложение_В).

## 4.4 Вывод по разделу

В данном разделе была рассмотрена реализация некоторых возможностей проекта. В их число входят: оплата бронирования (класс «AddPayment»), авторизация и регистрация (класс «LoginWindow»), отмена бронирования (класс «InfoBookings»). Все они помогают пользователям корректно и удобно взаимодействовать с приложением и интерфейсом.

# 5 Тестирование, проверка работоспособности и анализ полученных результатов

В данном программном средстве все исключительные ситуации обработаны с помощью вывода информации в диалоговое окно. Исключения могут возникнуть при некорректных данных входа или регистрации, а также при проблемах с базой данных.

Авторизация проходит путем проверки нахождения указанных учетных данных в базе данных. В случае, если оба входных параметра будут пусты, то появится диалоговое окно, представленное на рисунке 5.1.

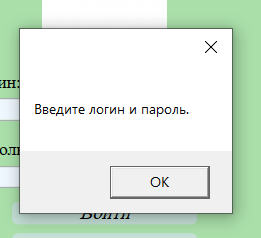


Рисунок 5.1 – Демонстрация исключения при отсутствии введенных данных

В случае, если будут введены неверные данные для логина и пароля, появится диалоговое окно, представленное на рисунке 5.2.

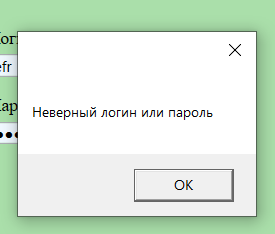


Рисунок 5.2 – Демонстрация исключения при неверно введенных данных

Также в приложении присутствует валидация регистрации. Она включает в себя проверка на уже существующего пользователя, заполнение всех полей, минимальную длину пароля, нахождение в строке только буквенных значений, проверка формата электронного адреса. Исключительная ситуация при нахождении существующего пользователя, показана на рисунке 5.3.

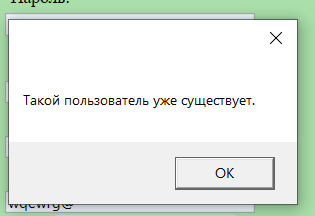


Рисунок 5.3 – Демонстрация исключения при нахождении существующего пользователя

Исключительная ситуация при наличии хотя бы одного пустого поля в форме регистрации, представлена на рисунке 5.4.

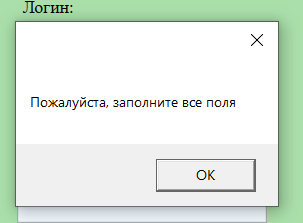


Рисунок 5.4 – Демонстрация исключения при пустых поляк в форме регистрации

Исключительная ситуация при введении пароля, состоящего из менее, чем 6 символов, представлена на рисунке 5.5.

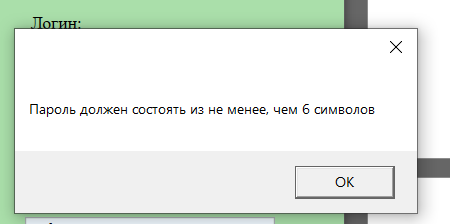


Рисунок 5.5 – Демонстрация исключения при неверном формате пароля

Исключительная ситуация при введении имени пользователя, состоящего из дополнительных символов помимо букв, представлена на рисунке 5.6.

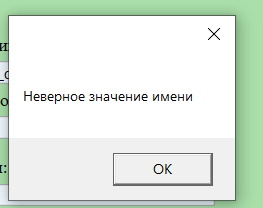


Рисунок 5.6 – Демонстрация исключения при неверном формате имени пользователя

Исключительная ситуация при введении неверного формата электронного адреса, представлена на рисунке 5.7.

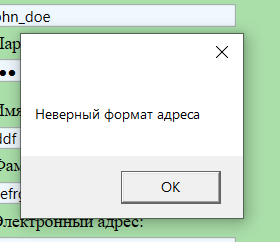


Рисунок 5.7 – Демонстрация исключения при неверном формате электронного адреса

В случае, когда пользователь пытается оплатить бронирование, которое он уже оплачивал ранее, появляется диалоговое окно, показанное на рисунке 5.8.

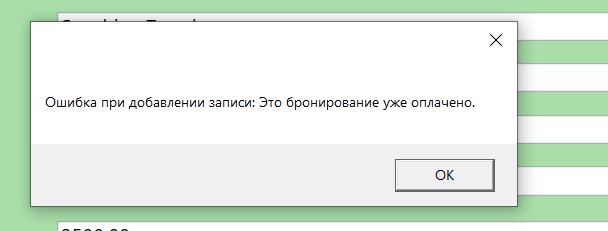


Рисунок 5.8 – Исключительная ситуация при попытке повторной оплаты бронирования

В случае, когда пользователь пытается открыть информацию о предлагаемом туре, не выбрав его, появляется диалоговое окно, показанное на рисунке 5.9.

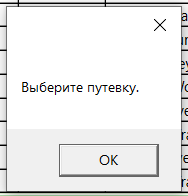


Рисунок 5.9 – Исключительная ситуация при попытке просмотра информации

В случае, когда администратор пытается удалить собственную учетную запись, появляется диалоговое окно, показанное на рисунке 5.10.

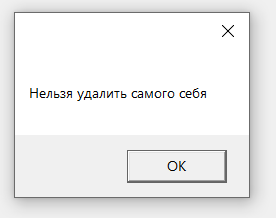


Рисунок 5.10 – Исключительная ситуация при попытке удаления текущей учетной записи

# 6 Руководство по использованию

Для того, чтобы воспользоваться приложением, необходимо зарегистрироваться в системе. Окно регистрации показано на рисунке 6.1:

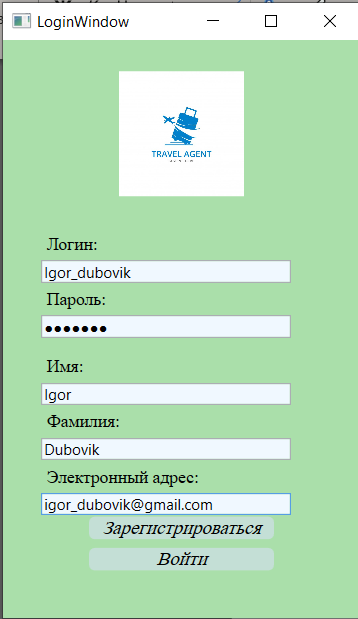


Рисунок 6.1 – Окно регистрации пользователя

Если же вы уже создавали свою учетную запись ранее, то введите действующий логин и пароль от аккаунта. Окно авторизации показано на рисунке 6.2:

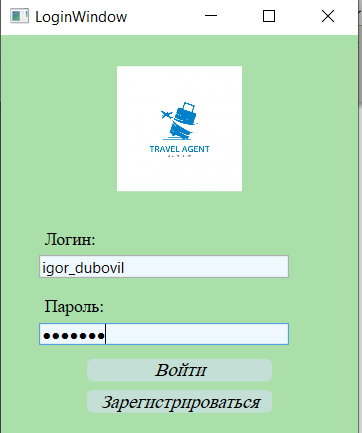


Рисунок 6.2 – Окно авторизации пользователя

После любого из этих действий, пользователю откроется главное окно Choose.xaml, которое предоставляет возможности просмотра доступных путевок, а также их фильтрация по параметрам. Данное окно представлено на рисунке 6.3:

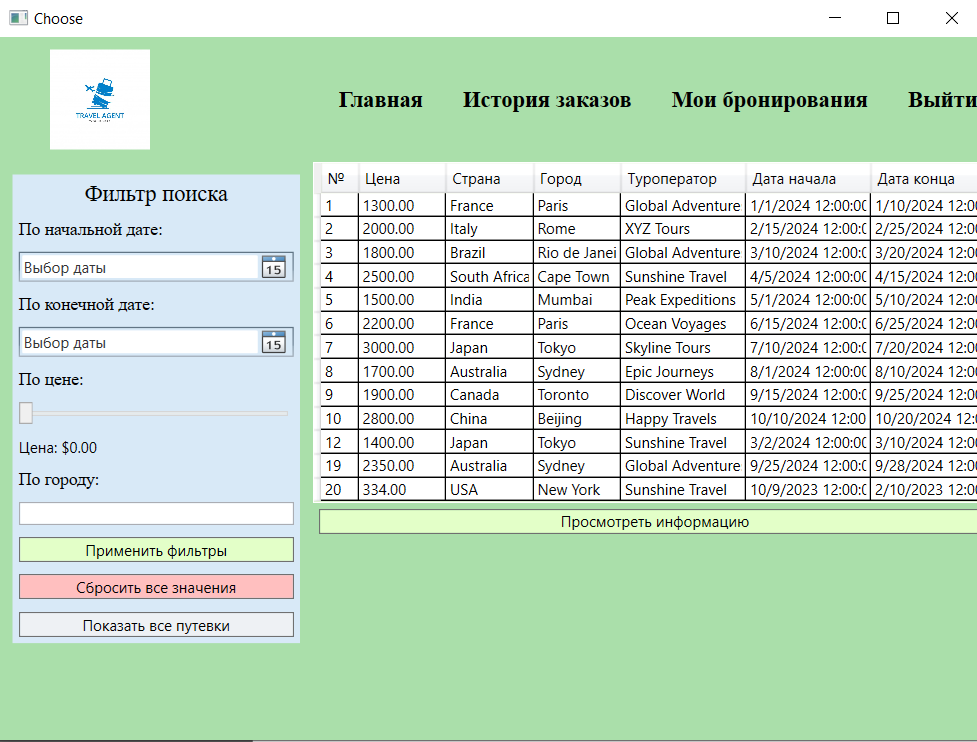


Рисунок 6.3 – Главное окно приложения

После этого пользователь может забронировать себе путевку. Для этого необходимо выбрать одну путевку из списка и нажать на кнопку «Посмотреть информацию». Пользователь будет перенесен в новое окно, которое предоставит ему более широкую информацию о данной путевки. Для бронирования путевки необходимо нажать на кнопку «Забронировать». Окно бронирования представлено на рисунке 6.4:

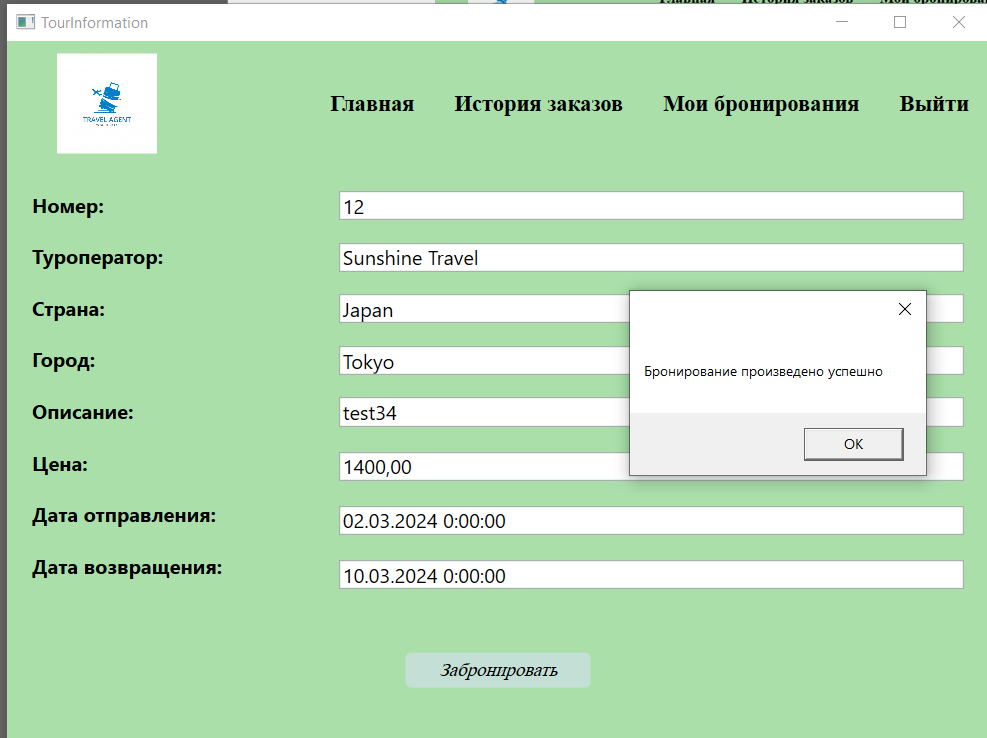


Рисунок 6.4 – Окно бронирования

После этого, данное бронирование будет внесено в окно «Мои бронирования», хранящее всю историю бронирований данного пользователя. Для оплаты бронирования необходимо нажать на кнопку «Оплатить». Окно истории бронирований представлено на рисунке 6.5:

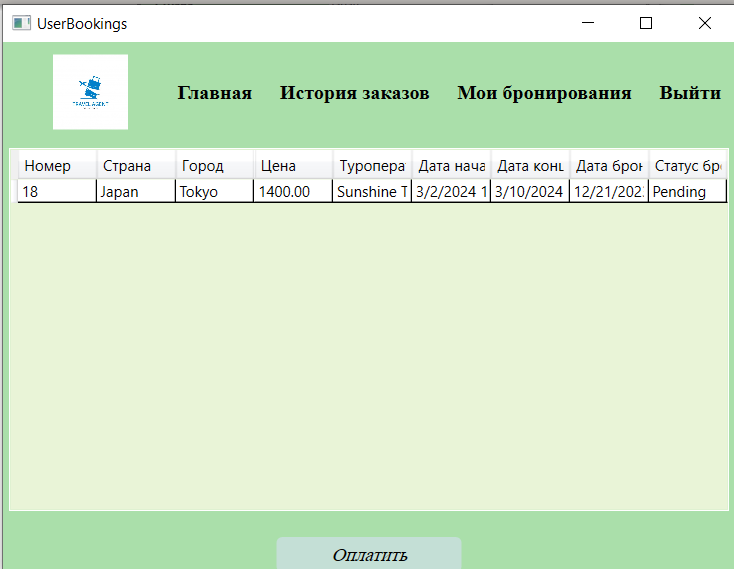


Рисунок 6.5 – Окно бронирований

После нажатия по кнопке «Оплатить», пользователь будет перенесен в окно подтверждения оплаты. Для подтверждения необходимо снова нажать на кнопку «Оплата». Окно подтверждения оплаты показано на рисунке 6.6:

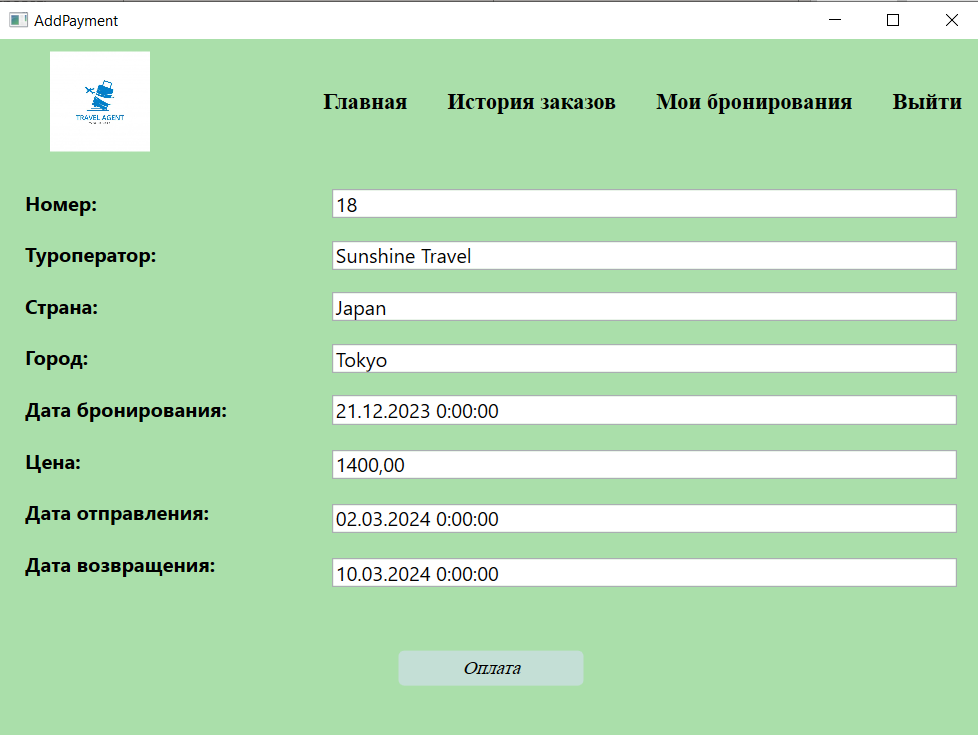


Рисунок 6.6 – Окно подтверждения оплаты

В итоге данная оплата также будет сохранена в окне истории заказов «История заказов», продемонстрировано на рисунке 6.7:

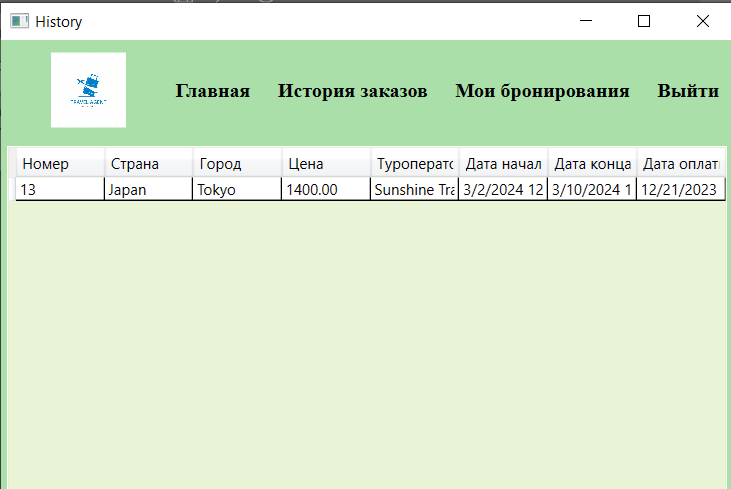


Рисунок 6.7 – Окно истории заказов

В итоге пользователь забронирует себе путевку и оплатит ее.

# Заключение

В ходе выполнения проекта были представлены различные аспекты разработки программного продукта для туристического сервиса продажи путевок. Рассмотрен анализ требований, проектирование архитектуры, выбор используемых технологий и инструментов, а также создание функциональных требований для приложения. Основное внимание уделено созданию удобного интерфейса для пользователей, а также обработке и обеспечению функциональности при авторизации, регистрации пользователей и вводе личных данных.

Для разработки приложения были выбраны современные технологии, такие как Microsoft Visual Studio 2022, .NET Framework, язык программирования C#, технология WPF, а также SQL Server Manager 2019 для управления базой данных. Использование паттерна Repository способствует разделению данных, бизнес-логики и визуализации, что делает систему более гибкой и легкой в сопровождении.

В разделе тестирования были рассмотрены все сценарии, при которых может произойти исключение. Они включают в себя авторизацию, регистрацию, ввода данных пользователей и администраторов. Это позволяет обеспечить корректную работу приложения и предотвратить возможные проблемы в процессе использования.

Руководство по использованию очень четко и поэтапно продемонстрировало основную задачу пользователей при посещении приложения – бронирование и оплата выбранной путевки.

Все представленные материалы отражают важные этапы разработки, уделяют внимание функциональности, удобству использования и безопасности приложения, что является ключевыми моментами в успешной разработке программного обеспечения для туристического сервиса.

# Список использованных источников

1. Microsoft Visual Studio [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio – Дата доступа 28.09.2023
2. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 15.10.2023
3. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 30.09.2023
4. Руководство по XAML // [Электронный ресурс]. – Режим доступа: https://www.tutorialspoint.com/xaml/index.htm– Дата доступа: 10.10.2023
5. Информационный сайт: Metanit [Электронный ресурс]. Режим доступа: <https://metanit.com/sharp/wpf> . Дата доступа: 29.09.2023.

# Приложение А

Диаграмма классов клиентского приложения:

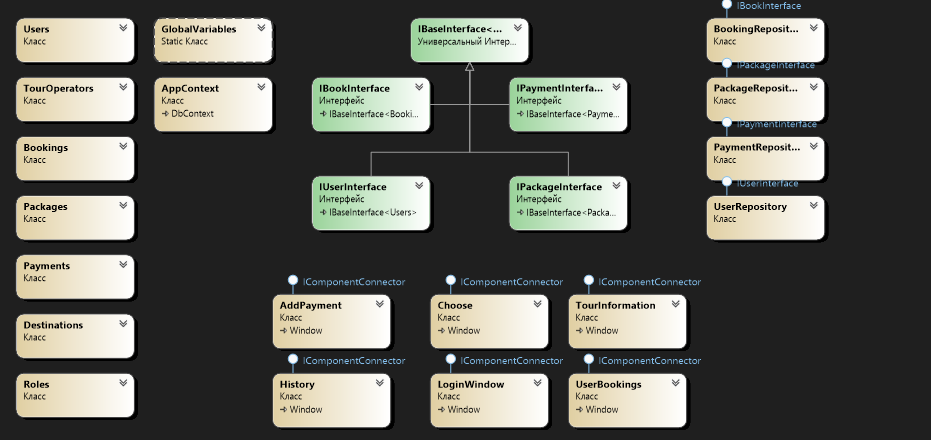
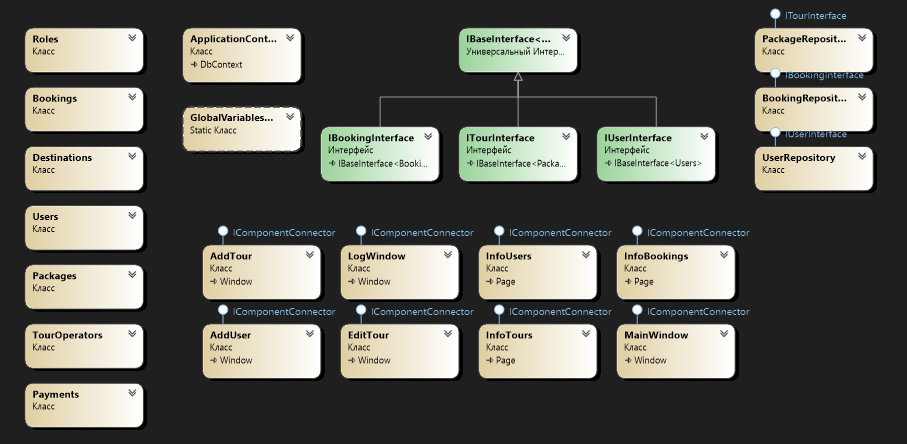
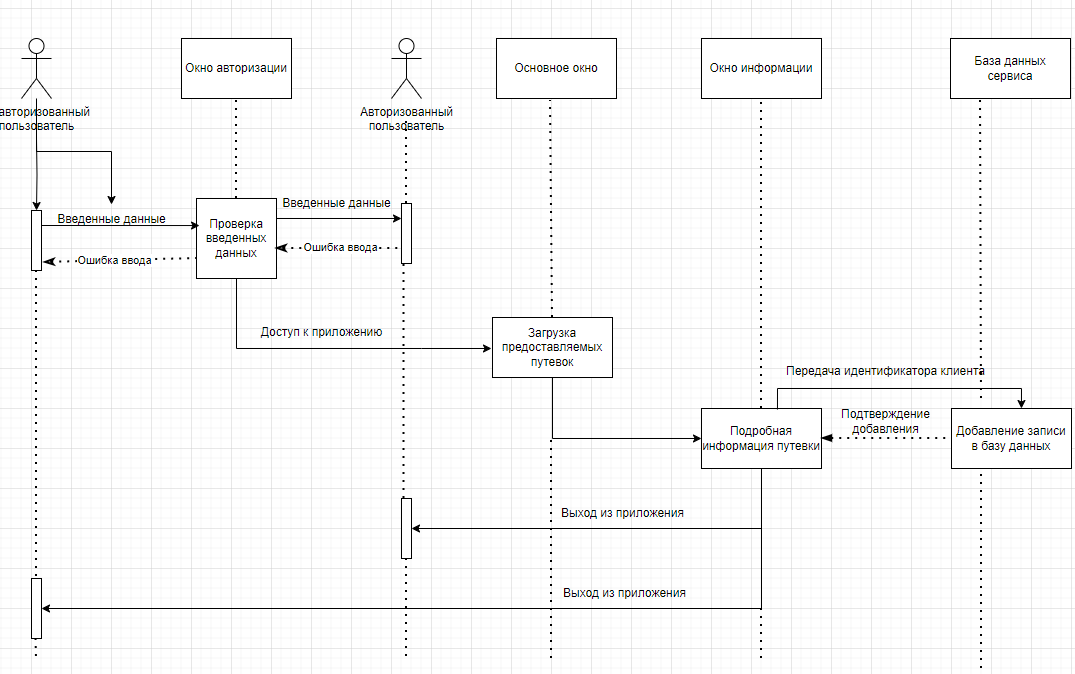


Диаграмма классов приложения администратора:



# Приложение Б

Диаграмма последовательности бронирования путевки:



# Приложение В

Листинг – Код основных методов класса LoginWindow

|  |
| --- |
| private void SwitchStackPanel\_Click(object sender, RoutedEventArgs e)  {  // Переключение видимости StackPanel при нажатии кнопки  stackPanelLogin.Visibility = stackPanelLogin.Visibility == Visibility.Visible ? Visibility.Collapsed : Visibility.Visible;  stackPanelRegistration.Visibility = stackPanelRegistration.Visibility == Visibility.Visible ? Visibility.Collapsed : Visibility.Visible;  }  private void RegButton\_Click(object sender, RoutedEventArgs e)  {  string username = txtUsername2.Text;  string password = txtPassword2.Password;  string firstname = txtFirstName.Text;  string lastname = txtLastName.Text;  string address = txtAddress.Text;  if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password) || string.IsNullOrEmpty(firstname) || string.IsNullOrEmpty(lastname) || string.IsNullOrEmpty(address))  {  MessageBox.Show("Пожалуйста, заполните все поля");  }  else if (!IsAllLetters(firstname))  {  MessageBox.Show("Неверное значение имени");  }  else if (!IsAllLetters(lastname))  {  MessageBox.Show("Неверное значение фамилии");  }  else if (!IsValidEmail(address))  {  MessageBox.Show("Неверный формат адреса");  }  else if (password.Length < 5)  {  MessageBox.Show("Пароль должен состоять из не менее, чем 6 символов");  }  else  {  if (RegisterUser(username, password, firstname, lastname, address))  {  Choose choose = new Choose();  choose.Show();  this.Close();  }  else  {  MessageBox.Show("Такой пользователь уже существует.");  }  }  }  private void LoginButton\_Click(object sender, RoutedEventArgs e)  {  string username = txtUsername.Text;  string password = txtPassword.Password;  if (string.IsNullOrWhiteSpace(username) && string.IsNullOrWhiteSpace(password))  {  // Проверка на пустые поля  MessageBox.Show("Введите логин и пароль.");  return;  }  else if (string.IsNullOrWhiteSpace(username))  {  // Проверка на пустые поля  MessageBox.Show("Введите логин.");  return;  }  else if (string.IsNullOrWhiteSpace(password))  {  // Проверка на пустые поля  MessageBox.Show("Введите пароль.");  return;  }  else  {  if (AuthenticateUser(username, password))  {  Choose choose = new Choose();  choose.Show();  this.Close();  }  else  {  MessageBox.Show("Неверный логин или пароль");  }  }  }  public bool AuthenticateUser(string username, string password)  {  using (var dbContext = new Model.AppContext())  {  var user = dbContext.Users.FirstOrDefault(u => u.login\_user == username);  if (user != null && user.password\_user == password)  {  // Проверяем роль пользователя  if (user.Roles != null && user.Roles.role\_name == "Пользователь")  {  // Пользователь авторизован и имеет роль "Пользователь"  GlobalVariables.ClientID = user.users\_id;  return true;  }  }  return false;  }  }  private bool RegisterUser(string username, string password, string firstname, string lastname, string address)  {  UserRepository userRepository = new UserRepository();  return userRepository.AddUser(username, password, firstname, lastname, address);  }  public bool IsAllLetters(string input)  {  return input.All(char.IsLetter);  }  private bool IsValidEmail(string email)  {  return email.Contains("@");  } |

Листинг – Код основных методов класса AddPayment

|  |
| --- |
| private void MainButton\_Click(object sender, RoutedEventArgs e)  {  Choose choose = new Choose();  choose.Show();  this.Close();  }  private void HistoryButton\_Click(object sender, RoutedEventArgs e)  {  History history = new History();  history.Show();  this.Close();  }  private void BookingButton\_Click(object sender, RoutedEventArgs e)  {  UserBookings bookings = new UserBookings();  bookings.Show();  this.Close();  }  private void LeftButton\_Click(object sender, RoutedEventArgs e)  {  LoginWindow log = new LoginWindow();  log.Show();  this.Close();  }  private void Payment\_Click(object sender, RoutedEventArgs e)  {  decimal.TryParse(txtPrice.Text, out decimal price);  int.TryParse(txtBookNumber.Text, out int booking\_number);  try  {  PaymentRepository paymentRepository = new PaymentRepository();  paymentRepository.MakePayment(booking\_number, price);  MessageBox.Show("Оплата произведена успешно");  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при добавлении записи: {ex.Message}");  }  }  public void MakePayment(int bookingId, decimal amount)  {  var bookingToUpdate = dbContext.Bookings.FirstOrDefault(b => b.booking\_id == bookingId);  if (bookingToUpdate != null)  {  if (bookingToUpdate.payment\_status == "Paid")  {  throw new InvalidOperationException("Это бронирование уже оплачено.");  }  Payments newPayment = new Payments  {  booking\_id = bookingId,  payment\_date = DateTime.Now,  amount = amount  };  bookingToUpdate.payment\_status = "Paid";  dbContext.Payments.Add(newPayment);  dbContext.SaveChanges();  }  else  {  throw new InvalidOperationException("Бронирование не найдено.");  }  } |

Листинг – Код основных методов класса InfoBookings

|  |
| --- |
| private string GetValueOrDefault(dynamic obj, string propertyName)  {  try  {  return obj.GetType().GetProperty(propertyName)?.GetValue(obj)?.ToString();  }  catch  {  return string.Empty;  }  }  private void CancelButton\_Click(object sender, RoutedEventArgs e)  {  if (iform.SelectedItem != null)  {  // Получаем выбранный объект из DataGrid  dynamic selectedData = iform.SelectedItem;  int book\_id = selectedData.booking\_id;  try  {  using (var dbContext = new Model.ApplicationContext())  {  // Создаем экземпляр класса, содержащего метод CancelBooking  var bookingRepository = new BookingRepository();  // Вызываем метод CancelBooking  bookingRepository.CancelBooking(book\_id);  MessageBox.Show("Бронирование успешно отменено");  }  }  catch (Exception ex)  {  MessageBox.Show($"Произошла ошибка: {ex.Message}");  }  }  else  {  MessageBox.Show("Выберите бронирование.");  }  }  public void CancelBooking(int bookingId)  {  var bookingToCancel = dbContext.Bookings.FirstOrDefault(b => b.booking\_id == bookingId && b.payment\_status == "Pending");  if (bookingToCancel != null)  {  dbContext.Bookings.Remove(bookingToCancel);  dbContext.SaveChanges();  }  else  {  throw new InvalidOperationException("Бронирование с указанным идентификатором уже оплачено или не найдено");  }  } |