

Содержание

| | |
|---|----|
| ВВЕДЕНИЕ | 2 |
| 1. Аналитический обзор прототипов и литературных источников | 3 |
| 1.1 Аналитический обзор источников | 3 |
| 1.2 Обзор аналогов..... | 3 |
| 2. Анализ требований к программному средству и разработка функциональных требований | 6 |
| 2.1 Описание средств разработки..... | 6 |
| 2.2 Описание разрабатываемой функциональности приложения | 6 |
| 2.3 Спецификация функциональных требований..... | 7 |
| 3. Проектирование программного средства | 8 |
| 3.1 Общая структура проекта | 8 |
| 3.2 Архитектура приложения | 8 |
| 3.3 Схема работы приложения | 9 |
| 3.4 Проектирование базы данных | 10 |
| 4. Реализация программного средства..... | 13 |
| 4.1 Выполняемые функции | 13 |
| 4.2 Реализация общей структуры проекта..... | 13 |
| 4.2 Основные классы программного средства..... | 15 |
| 5. Тестирование, проверка работоспособности и анализ полученных результатов | 16 |
| 5.1 Тестирование регистрации и авторизации | 16 |
| 5.2 Тестирование поиска | 17 |
| 5.3 Тестирование ввода данных | 18 |
| 6. Методика использования программного средства..... | 19 |
| ЗАКЛЮЧЕНИЕ | 24 |
| СПИСОК ЛИТЕРАТУРЫ | 25 |
| ПРИЛОЖЕНИЕ А | 26 |
| ПРИЛОЖЕНИЕ Б | 27 |
| ПРИЛОЖЕНИЕ В | 32 |

ВВЕДЕНИЕ

В современном мире люди постоянно куда-то спешат: работа, семья, домашние заботы. Постоянное однообразие делает жизнь серой, обыденной. Дни бегут один за другим, кажется, что времени все меньше и меньше. Продуктивность, интерес, энтузиазм пропадают.

Человек должен работать, чтобы жить, а не жить, чтобы работать. Путешествия отличный способ сбежать от обыденности. Ведь в мире столько всего красивого и интересного.

Каждый выбирает свой способ путешествия: кто-то предпочитает отправиться навстречу неизвестности и спонтанности, кто-то любит планировать все сам, а кого-то привлекает полностью организованный тур.

Данное программное устройство подойдет тем, и тем, кто хочет приехать «на все готовое», и тем, кто предпочитает сделать все самому. Пользователю достаточно выбрать страну и отель, забронировать и отправиться в увлекательное путешествие. До выбора подходящего варианта пользователь может ознакомиться со всеми предложениями. Также есть возможность получать рассылку о новых предложениях и акциях.

Для успешной реализации курсового проекта необходимо:

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

Содержание данной пояснительной записки отражает все этапы выполнения моего курсового проекта.

1. Аналитический обзор прототипов и литературных источников

1.1 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.

1.2 Обзор аналогов

Для разработки нового программного продукта, предназначенного для управления туристическим агентством, необходимо провести анализ существующих программных решений в данной области. Путем изучения преимуществ и недостатков этих аналогов можно определить требования к разрабатываемому программному продукту, учитывающие опыт предыдущих разработок, а также внести улучшения и изменения.

В процессе работы были рассмотрены сайты туристических агентств.

Веб-сайт туристического агентства «Tez Tour» (<https://tourist.tez-tour.com/>).

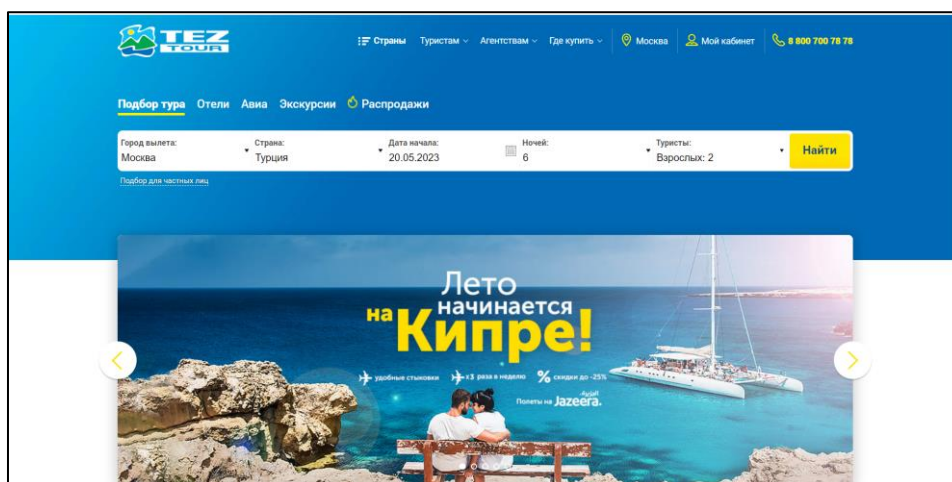


Рисунок 1.1 – Интерфейс «Tez Tour»

На сайте предоставляется информация о туристических путевках и

направлениях путешествий. Пользователи могут просматривать различные предложения по отдыху, включая описания туров, фотографии, цены и доступные даты. Есть возможность осуществить поиск по следующим критериям: место назначения, тип отдыха и длительность, показано на рисунке 1.2.1. Сайт предоставляет информацию о дополнительных услугах, таких как бронирование отелей, трансферы и страхование. Пользователи могут зарегистрироваться на сайте для получения персонализированных предложений и возможности оформления бронирования. Сайт также предлагает способы связи с операторами для получения консультации и помощи.

Веб-сайт туристического агентства «Booking» (<https://www.booking.com/index.ru.html>) представлен на рисунке 1.2.

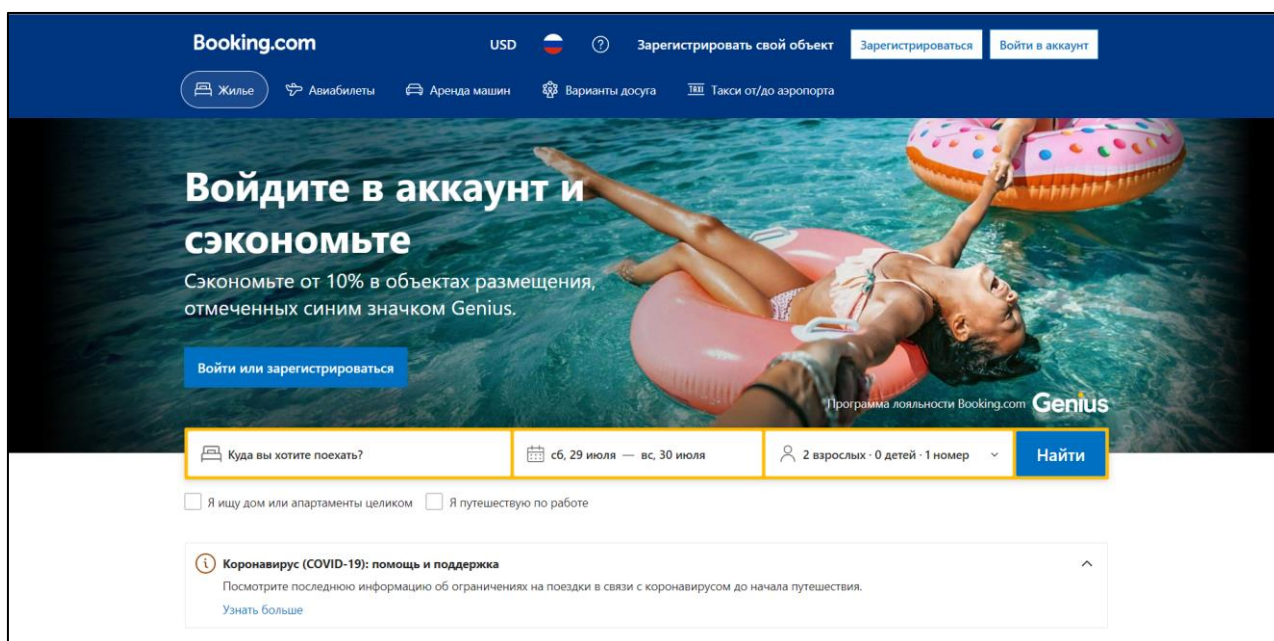


Рисунок 1.2 – Интерфейс «Booking»

На сайте предоставляется широкий выбор вариантов размещения по всему миру, с возможностью фильтрации по местоположению, цене, типу размещения и другим критериям. Пользователи могут просматривать информацию о каждом варианте размещения, включая фотографии, описания, отзывы и оценки других пользователей, показано на рисунке 1.3. Есть возможность осуществить онлайн-бронирование с выбором дат пребывания, количества гостей, вариантов номеров и дополнительных удобств. Booking.com предлагает различные способы оплаты и отмены бронирования. Пользователи могут использовать персональные аккаунты для сохранения предпочтений, получения специальных предложений и управления своими бронированиями.

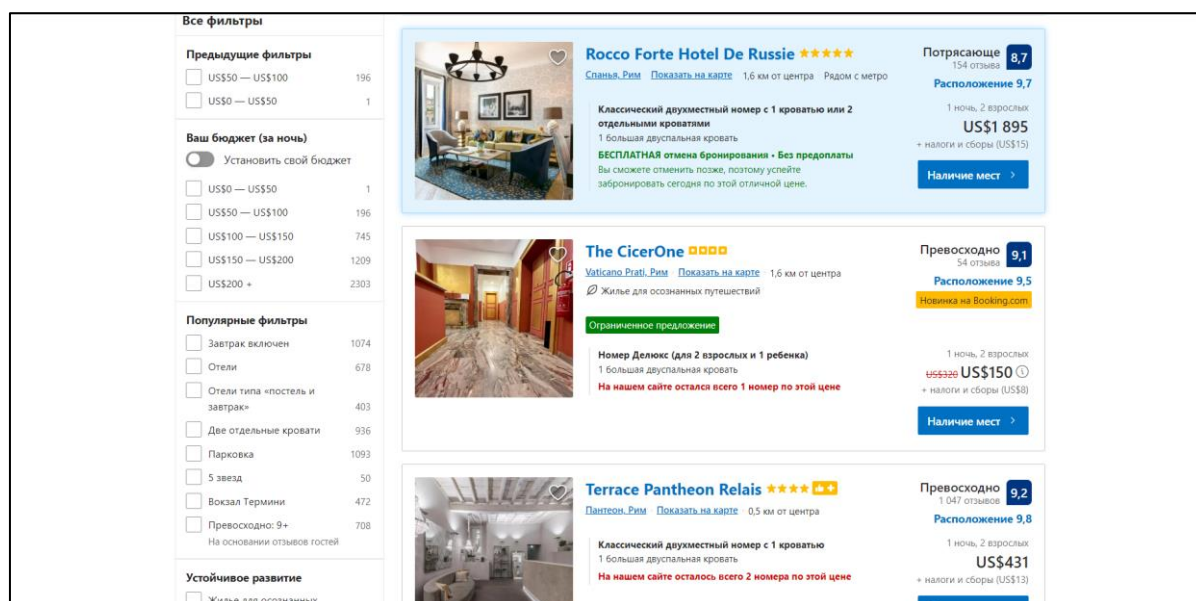


Рисунок 1.3 – Интерфейс «Booking»

Веб-сайт туристического агентства «TripAdvisor» (https://www.tripadvisor.com) представлен на рисунке 1.4.

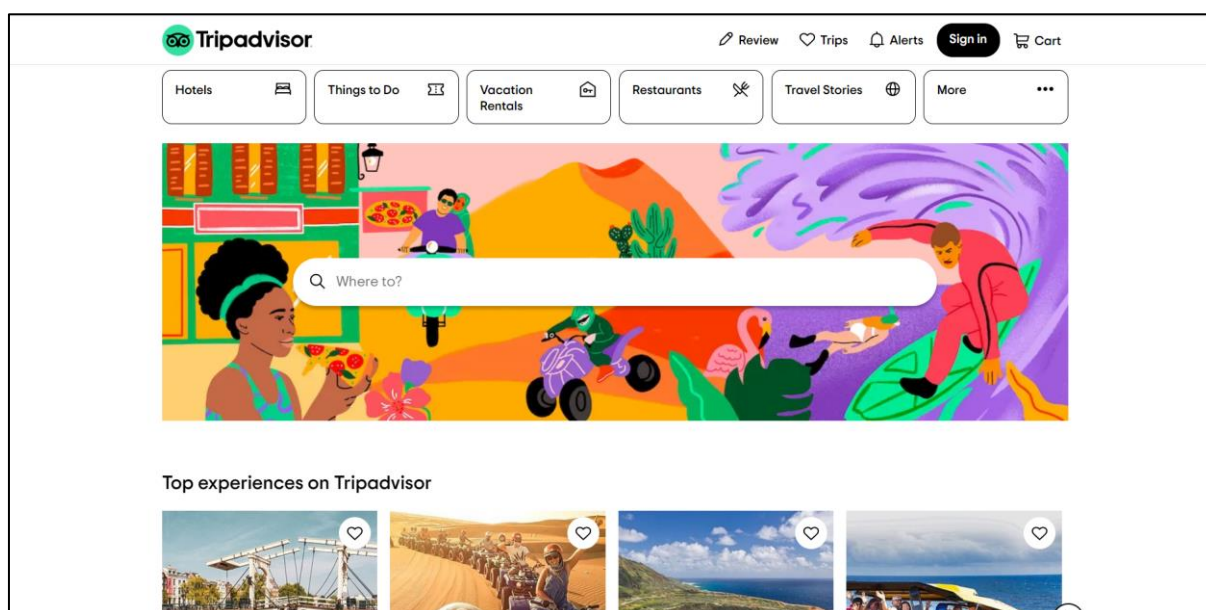


Рисунок 1.4 – Интерфейс «Booking»

Сайт имеет современный и интуитивно понятный дизайн. Интерфейс часто обновляется и улучшается для повышения удобства пользования. «TripAdvisor» позволяет пользователям оставлять отзывы и оценивать различные отели, рестораны и достопримечательности. Есть возможность поиска и бронирования отелей, ресторанов, авиабилетов, а также фильтрация результатов поиска по различным критериям, таким как цена, расстояние, тип размещения и др. «TripAdvisor» содержит путеводители и советы, предоставленные пользователями.

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 6.0;
- язык программирования C#;
- расширяемый язык разметки XAML;
- технология WPF;
- технология Entity Framework Core;
- технология Fluent API;

MS SQL Server.

2.2 Описание разрабатываемой функциональности приложения

Программное средство предоставляет пользователю следующие функциональные возможности:

- регистрация в системе;
- вход в приложение;
- просмотр отелей;
- бронирование отелей;
- написание отзыва об отеле;
- просмотр информации о брони;
- добавление отеля в избранное;
- фильтрация отелей по цене и рейтингу;
- хранение личной информации в базе данных;

- изменение личной информации.
- Функции администратора:
- просмотр всех пользователей;
 - добавление, редактирование и удаление отелей;
 - просмотр бронирований.

2.3 Спецификация функциональных требований

После проведения анализа были выявлены следующие функциональные требования:

- архитектура приложения должна соответствовать шаблонам проектирования, таким как MVVM, Command;
- необходимо создание базы данных для хранения информации приложения;
- должна производиться валидация вводимых пользователем данных;
- корректным образом должны быть обработаны 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 поиска отелей по следующим критериям: страна, даты проживания, количество гостей;
- возможность фильтрации по цене и рейтингу.

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

3. Проектирование программного средства

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи.

3.1 Общая структура проекта

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

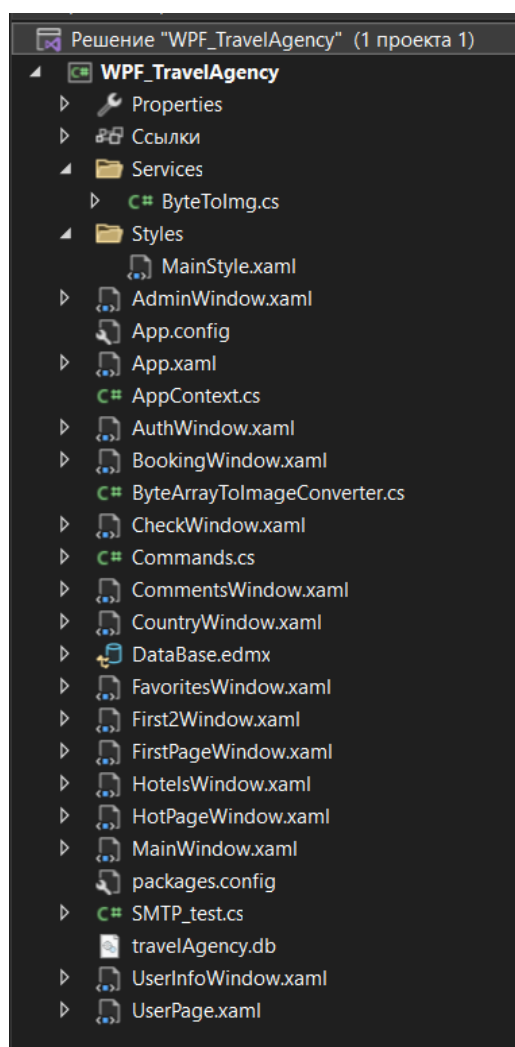


Рисунок 3.1 – Общая структура проекта

3.2 Архитектура приложения

Архитектура приложения определяет структуру и организацию его

компонентов. Она определяет, как приложение будет разбито на разные модули, как они будут взаимодействовать друг с другом и с внешними системами, а также как обеспечить его модульность, расширяемость, переносимость, масштабируемость, управляемость и безопасность.

В приложении используется архитектура, основанная на фреймворке Entity Framework (EF) - объектно-ориентированном фреймворке доступа к данным в .NET-приложениях. С помощью EF разработчики могут работать с данными через высокоуровневый объектно-ориентированный интерфейс, вместо явного написания SQL-запросов. Особенностью EF является его возможность ORM (Object-Relational Mapping), которая позволяет сопоставлять объекты приложения с таблицами в базе данных. Это упрощает работу с данными и облегчает процесс разработки приложения.

3.3 Схема работы приложения

Для описания маршрутов работы программного продукта была создана диаграмма навигации, которая показывает пути перехода между страницами приложения, представлена на рисунке 3.2.

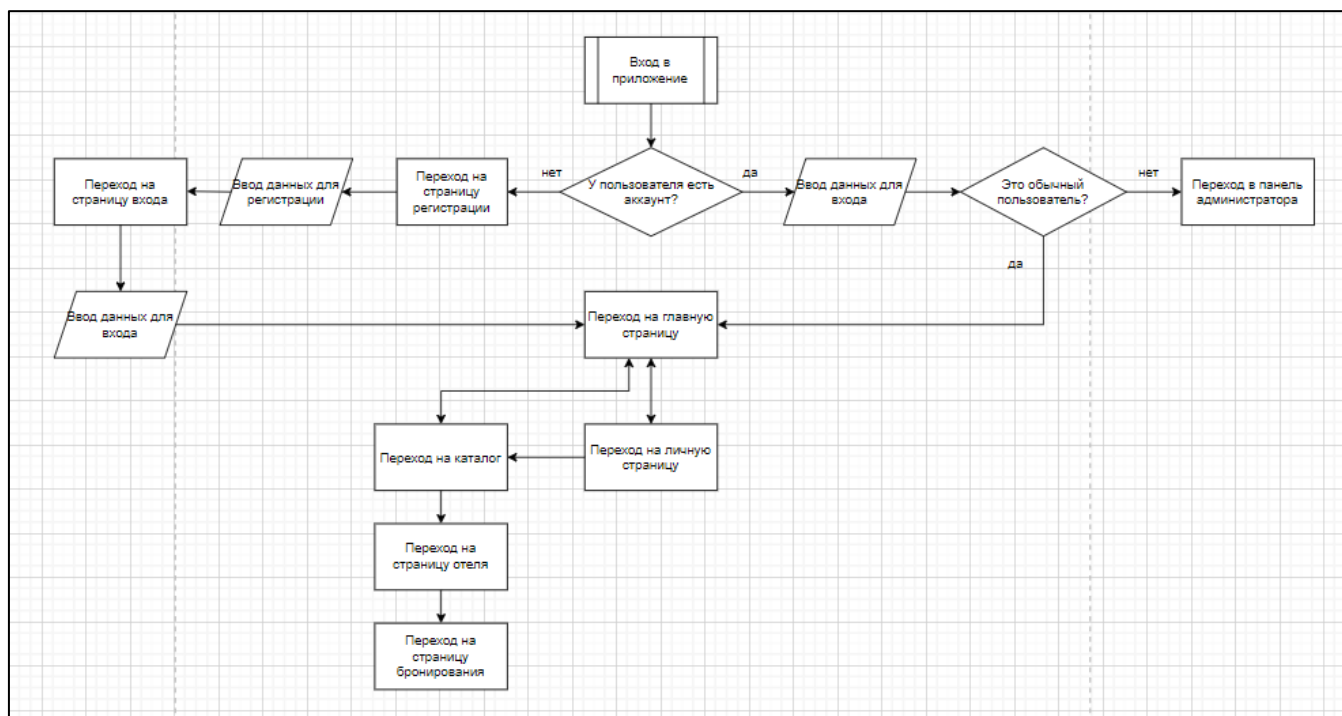


Рисунок 3.2 – Схема навигации по страницам приложения «travel AGency»

При запуске программы происходит появление окна авторизации, запрашивающее ввод учетных данных для продолжения взаимодействия приложением. После прохождения авторизации пользователь видит главную страницу с функцией поиска отеля. Компоненты приложения представлены

отдельными окнами, открывающимися по нажатию соответствующих элементов управления пользователем.

3.4 Проектирование базы данных

Для разработки приложения в качестве сервиса для хранения данных была выбрана СУБД Microsoft SQL Server. Логическая схема данных представлена на рисунке 3.3.

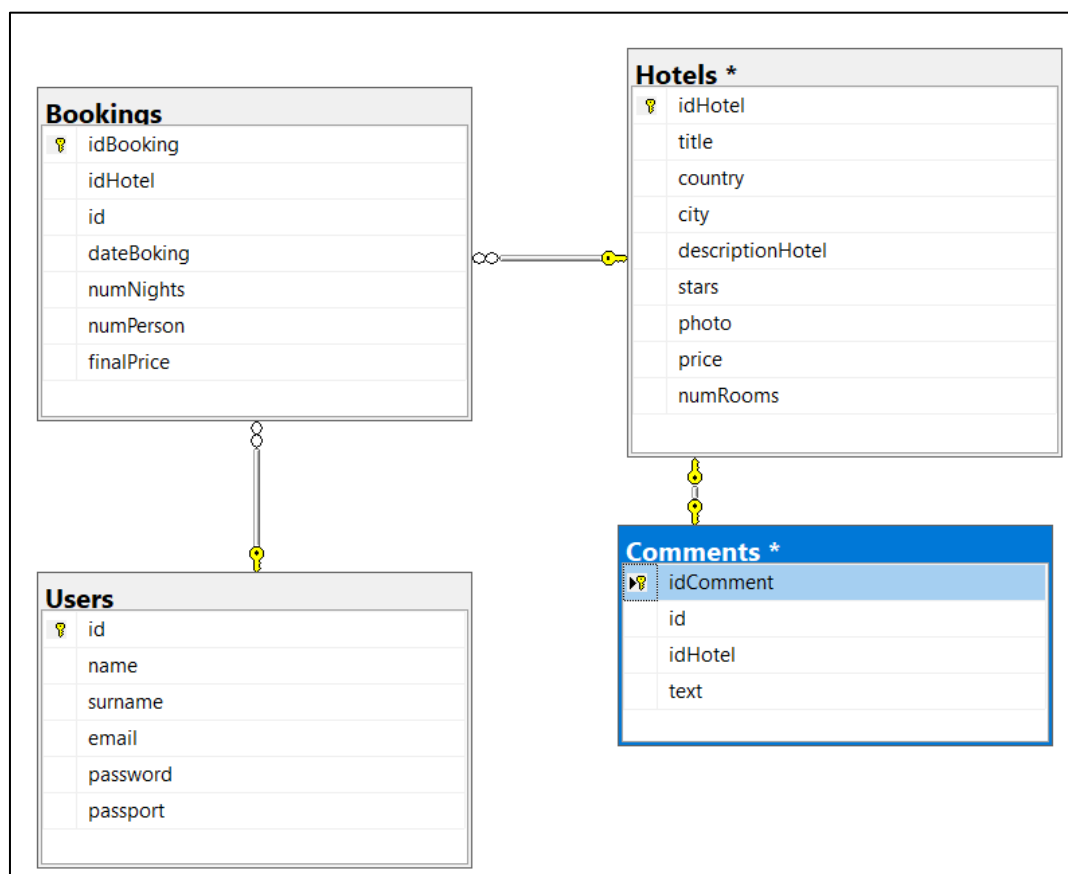


Рисунок 3.3 – Логическая схема данных

База данных состоит из 4 таблиц: Users, Hotels, Bookings, Comments. Они представлены ниже вместе со своими структурами.

На рисунке 3.4 представлена структура таблицы Users. В данной таблице хранятся имя и фамилия пользователя, адрес его электронной почты, данные паспорта и пароль. Первичным ключом является поле «id».


| | | |
|--|--------------|--------------------------|
|  id | int | <input type="checkbox"/> |
| name | nvarchar(50) | <input type="checkbox"/> |
| surname | nvarchar(50) | <input type="checkbox"/> |
| email | nvarchar(50) | <input type="checkbox"/> |
| password | nvarchar(20) | <input type="checkbox"/> |
| passport | nvarchar(20) | <input type="checkbox"/> |

Рисунок 3.4 – Структура таблицы Users

В таблице Hotels представленной на рисунке 3.5 хранятся следующие данные: название отеля, страна, адрес, описание отеля, рейтинг, фото, цена и количество комнат. Первичным ключом является поле idHotel.


| | | |
|---|---------------|--------------------------|
|  idHotel | int | <input type="checkbox"/> |
| title | nvarchar(50) | <input type="checkbox"/> |
| country | nvarchar(50) | <input type="checkbox"/> |
| city | nvarchar(50) | <input type="checkbox"/> |
| descriptionHotel | nvarchar(MAX) | <input type="checkbox"/> |
| stars | int | <input type="checkbox"/> |
| photo | image | <input type="checkbox"/> |
| price | int | <input type="checkbox"/> |
| numRooms | int | <input type="checkbox"/> |

Рисунок 3.5 – Структура таблицы Hotels

Для хранения информации о бронировании создана таблица Bookings, представленная на рисунке 3.6. Данные представляют собой: id отеля, id пользователя, с помощью которых связываются таблицы Users, Hotels и Bookings, дату бронирования, количество ночей, количество гостей и итоговую стоимость. Первичным ключом является idBooking.


| | | |
|---|------|--------------------------|
|  idBooking | int | <input type="checkbox"/> |
| idHotel | int | <input type="checkbox"/> |
| id | int | <input type="checkbox"/> |
| dateBoking | date | <input type="checkbox"/> |
| numNights | int | <input type="checkbox"/> |
| numPerson | int | <input type="checkbox"/> |
| finalPrice | int | <input type="checkbox"/> |

Рисунок 3.6 – Структура таблицы Bookings

В таблице Comments на рисунке 3.7 содержатся данные о комментарии (id пользователя, id отеля, текст комментария и первичный ключ iComment).


| | | | |
|---|-----------|----------------------|--------------------------|
|  | idComment | int | <input type="checkbox"/> |
| | id | int | <input type="checkbox"/> |
| | idHotel | int | <input type="checkbox"/> |
| | text | <u>nvarchar(MAX)</u> | <input type="checkbox"/> |

Рисунок 3.7– Структура таблицы Comments

Для установки логических связей между таблицами были разработаны наборы внешних ключей, необходимых для соединения информации в таблицах по определенным столбцам.

4. Реализация программного средства

После завершения этапа анализа и планирования разработки приложения, следующим шагом является непосредственная реализация программного решения в соответствии с предварительно определенными требованиями и шаблонами.

4.1 Выполняемые функции

Для описания функциональности программного продукта были созданы диаграммы на языке моделирования UML, которые визуализируют компоненты системы. Описание функциональных возможностей программного продукта, представленное в виде диаграмм вариантов использования, можно найти в приложении А. Для демонстрации общей структуры и иерархии классов системы, их коопераций, атрибутов, методов, интерфейсов и взаимосвязей между ними была разработана диаграмма классов, представленная на рисунке 4.1.

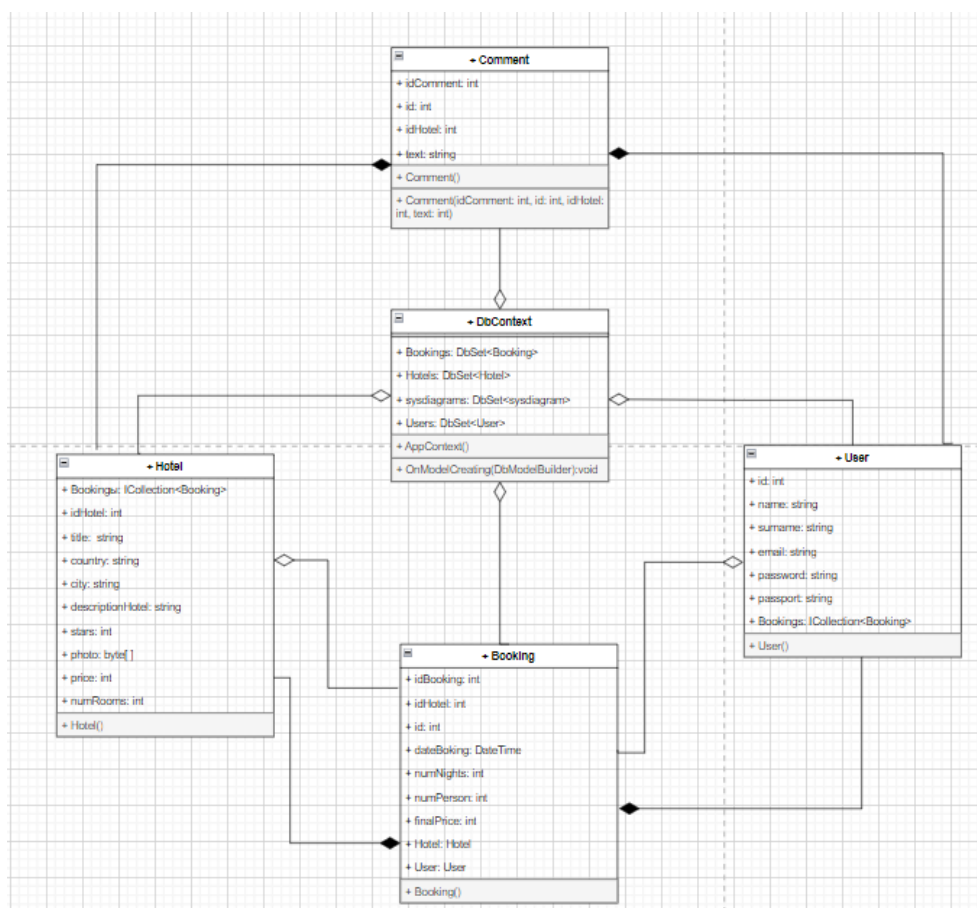


Рисунок 4.1 – Диаграмма классов программного продукта

4.2 Реализация общей структуры проекта

При разработке общей структуры программы был выбран подход к взаимодействию с базой данных. Для этой цели была использована технология

Entity Framework (EF), которая предоставляет возможности работы с объектами через Language Integrated Query (LINQ) в формате "LINQ to Entities" и с помощью "Entity SQL". Для создания таблиц в базе данных были определены классы с соответствующими свойствами, а затем были определены наборы сущностей типа DbSet в контексте данных.

Архитектура программного продукта "travel AGenсY" предусматривает использование двух сред управления информацией: клиентской и администраторской. Это позволяет предоставить различным пользователям удобные интерфейсы, адаптированные под их потребности и роли, которые определяются при авторизации. Данная структура обеспечивает удобную навигацию и взаимодействие с функциональными возможностями приложения для различных типов пользователей.

Описание окон программы расположено в таблице 4.1.

Таблица 4.1 – Общее описание окон проекта

| Название | Описание |
|----------------------|---|
| AdminWindow.xaml | Окно администратора, позволяющее администратору добавлять, удалять и редактировать данные об отелях. Просматривать информацию о пользователях и бронировании. |
| AuthWindow.xaml | Окно авторизации пользователя. |
| BookingWindow.xaml | Окно бронирования отеля. |
| CheckWindow.xaml | Окно, содержащее информацию о брони. |
| CommentsWindow.xaml | Окно для написания отзыва. |
| CountryWindow.xaml | Окно, содержащее каталог отелей. |
| FirstPageWindow.xaml | Главное окно, содержащее переходы в личный кабинет и в каталог отелей. |
| HotPageWindow.xaml | Окно, содержащее подробную информацию об отеле, кнопку бронирования. |
| MainWindow.xaml | Окно для регистрации пользователя. |
| UserInfoWindow.xaml | Окно, содержащее информацию о пользователе с кнопками изменения, удаления пользователя, а также просмотра информации о брони |

Административные окна в программе предназначены для редактирования информации о текущих записях в базе данных. Они обеспечивают удобный интерфейс для изменения данных без необходимости написания прямых запросов к таблицам или строкам в базе данных. Это позволяет администраторам легко управлять и обновлять информацию, хранящуюся в системе.

4.2 Основные классы программного средства

Для выполнения технических задач программного средства должны быть реализованы следующие функции и соответствующие им классы и методы:

- регистрация;
- авторизация;
- поиск отеля;
- бронирование;
- фильтрация;
- заполнение и просмотр отзывов.

Все это описано в таблице 4.2.

Таблица 4.2 – Общее описание возможностей проекта

| Название | Описание |
|------------------------------|---|
| Регистрация | Пользователь вводит данные, они проверяются на валидность. В случае успешного выполнения открывается окно авторизации, пользователь добавляется в базу данных. |
| Авторизация | Пользователь вводит данные для входа, если такие данные присутствуют в базе, пользователь попадает на главную страницу. |
| Поиск отеля | В блоке для поиска пользователь заполняет поля, и после нажатия кнопки «Искать» попадает в каталог отелей, где может просматривать варианты и использовать фильтры. |
| Бронирование | При выборе отеля пользователь нажимает кнопку «Забронировать», появляется форма, в которую передаются данные пользователя из базы, при подтверждении брони, данные о ней записываются в базу и отправляются пользователю. |
| Просмотр личного кабинета | В личном кабинете хранится информация о пользователе и его брони. Пользователь может редактировать данные. |
| Написание и просмотр отзывов | При нажатии кнопки «Написать отзыв» появляется окно с TextBox, куда пользователь пишет комментарий. Также он может просмотреть свои и чужие отзывы. |

Для работы с базой данных MS SQL Server создан класс AppContext. В любом приложении, работающем с БД через Entity Framework Core, необходимо использовать контекст (класс производный от DbContext) и набор данных DbSet, через который можно взаимодействовать с таблицами из БД.

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 ситуации, в которых поведение программы является неправильным, нежелательным или несоответствующим требованиям.

Задачи тестирования:

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

Для обеспечения корректности работы программы обрабатываются различные ошибки, возникающие в процессе работы. Данное программное средство использует подключение к базе данных, следовательно, неправильно введенные данные или же их отсутствие может повлечь за собой неработоспособность приложения.

5.1 Тестирование регистрации и авторизации

При регистрации, если пользователь ввел некорректные данные, выводится сообщение об ошибке рисунок 5.1.

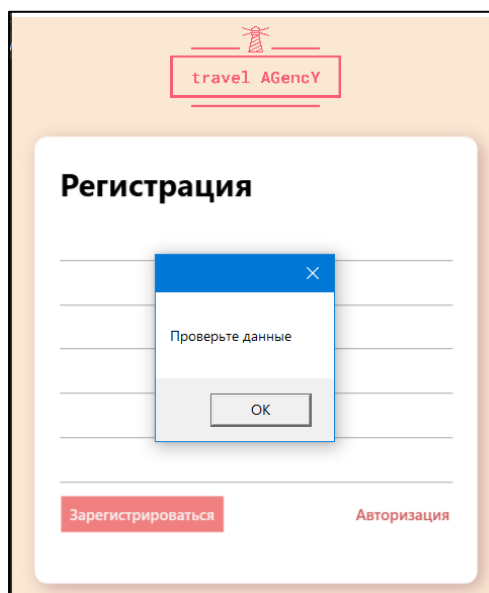


Рисунок 5.1 — Обработка некорректных данных

Такая же обработка находится в авторизации, при введении некорректной

почты или пароля, который не совпадает с тем, что был при регистрации рисунок 5.2.

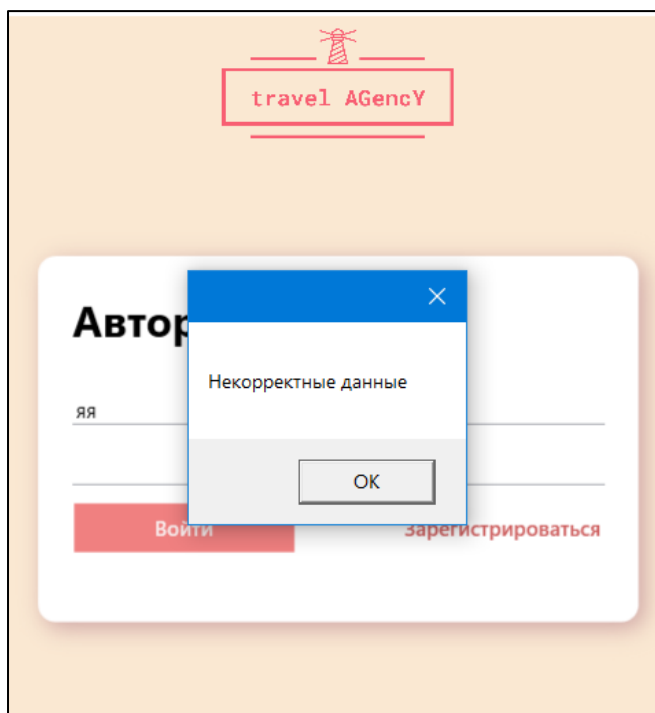


Рисунок 5.2 — Обработка некорректных данных

5.2 Тестирование поиска

Если пользователь не заполнил форму поиска или выбрал некорректную дату, появляется сообщение о том, что поля незаполнены пример на рисунке 5.3.

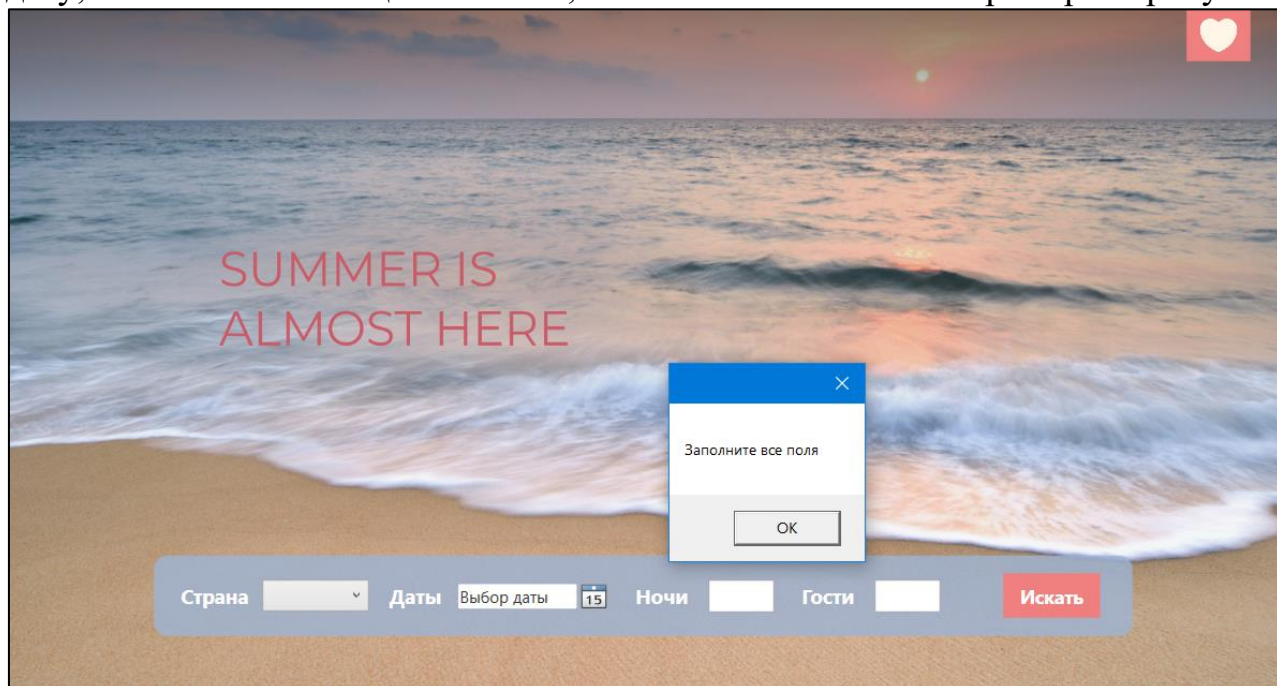


Рисунок 5.2 — Обработка некорректных данных

5.3 Тестирование ввода данных

Если пользователь или администратор не ввел, или ввел некорректные данные в форму брони, добавления или редактирования отеля, приложение выдаст предупреждение.

Таким образом, можно сделать вывод о важности этапа тестирования в жизненном цикле приложения. Позитивные тестовые случаи помогают проверить правильное функционирование программы, в то время как негативные позволяют выявить неправильное поведение или ошибки программы при непредвиденных или некорректных входных данных.

6. Методика использования программного средства

При запуске приложения открывается окно регистрации, представленное на рисунке 6.1.

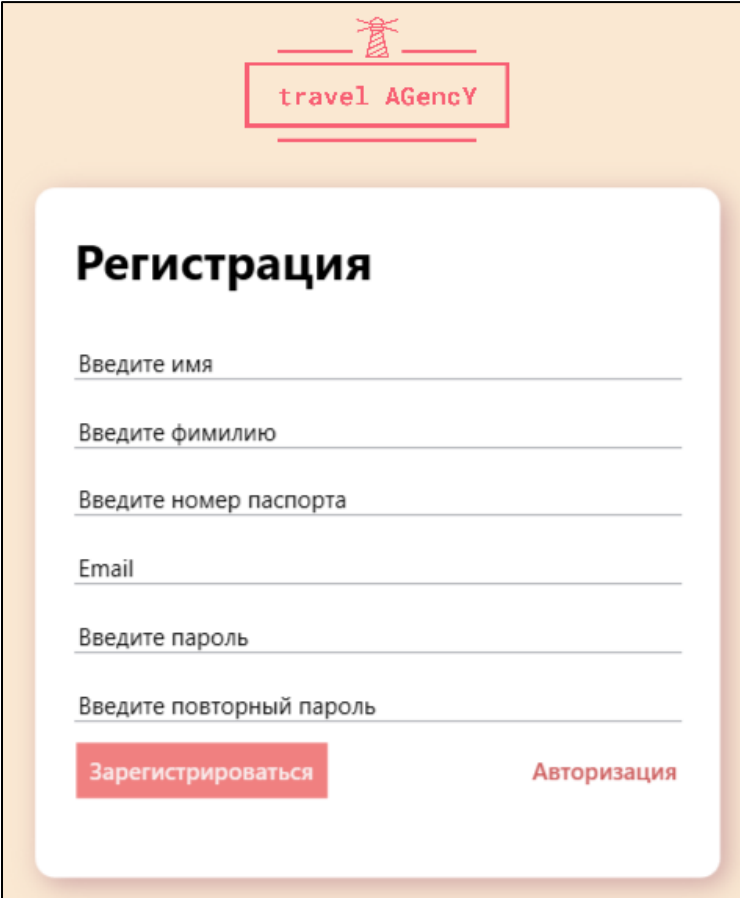
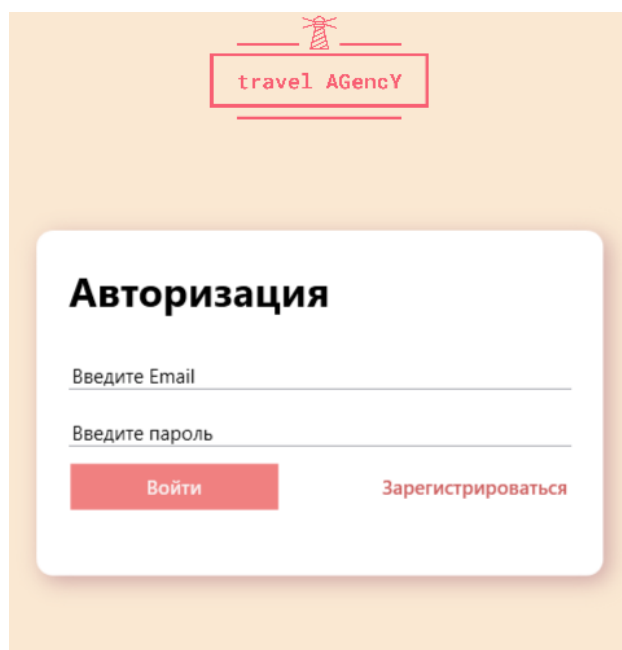


Рисунок 6.1 – Окно регистрации

Окно регистрации требует ввода данных. После успешной регистрации пользователя перебрасывает в окно авторизации, где он должен ввести данные для входа в приложение.

Окно авторизации представлено на рисунке 6.2. При корректном вводе пользователя перебрасывает на главную страницу.

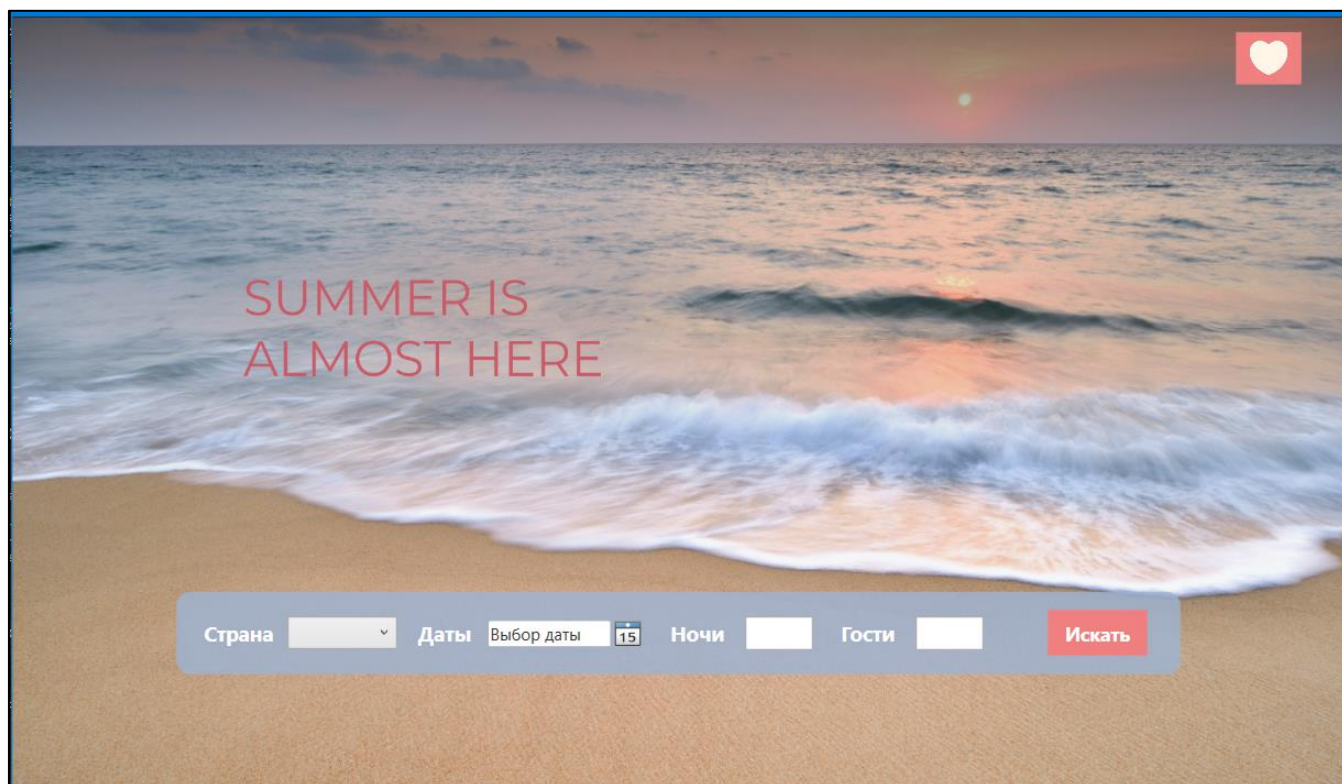


The image shows a login/registration window for a website called "travel AGENCY". At the top, there is a logo with a red windmill icon and the text "travel AGENCY" in a red box. Below the logo, the title "Авторизация" (Authorization) is displayed in bold black text. There are two input fields: "Введите Email" (Enter Email) and "Введите пароль" (Enter password). Below these fields are two buttons: a red button labeled "Войти" (Login) and a red button labeled "Зарегистрироваться" (Register).

Рисунок 6.1 – Окно регистрации

Главное окно изображено на рисунке 6.2.

Пользователь может в строке поиска задать нужные параметры и перейти к каталогу отелей.



The image shows the main window of the travel agency website. The background is a scenic view of a beach at sunset with waves crashing onto the shore. The text "SUMMER IS ALMOST HERE" is overlaid in red. In the top right corner, there is a red heart icon. At the bottom, there is a search bar with the following elements: "Страна" (Country) with a dropdown arrow, "Даты" (Dates) with a "Выбор даты" (Choose date) button and a calendar icon showing "15", "Ночи" (Nights) with an input field, "Гости" (Guests) with an input field, and a red "Искать" (Search) button.

Рисунок 6.2 – Главное окно

Также пользователь может просмотреть информацию о себе в личном

кабинете рисунок 6.3, нажав на кнопку в правом верхнем углу.

Рисунок 6.3 – Главное окно пользователя

При нажатии кнопки «Искать» пользователь попадает в каталог отелей, в котором может производить фильтрацию и выбор отеля, рисунок 6.4.

Рисунок 6.4 – Окно каталога

При выборе понравившегося отеля, пользователь может просмотреть подробную информацию, отзывы, оставить отзыв и забронировать отель, рисунок 6.5.

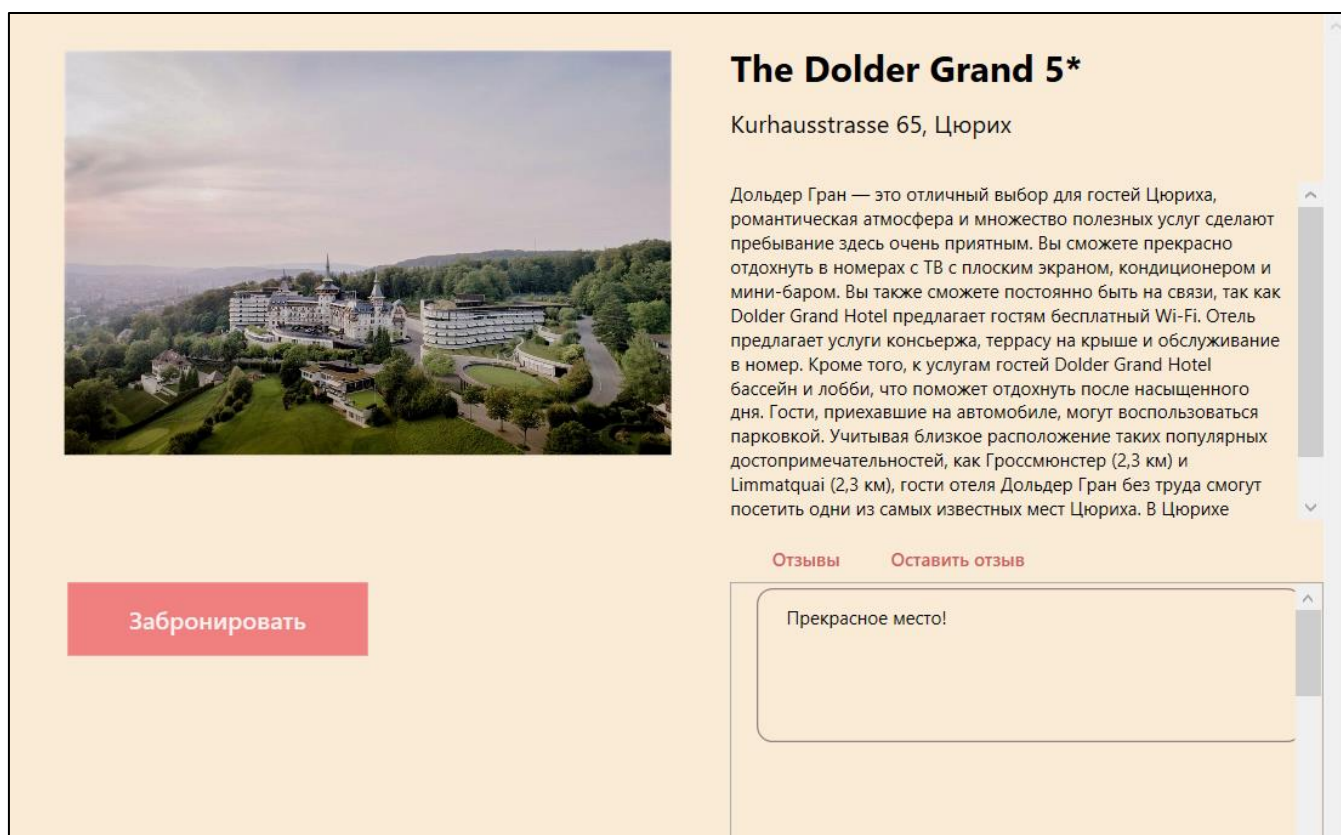


Рисунок 6.5 – Страница отеля

Чтобы забронировать отель нужно нажать кнопку «Забронировать», подтвердить все данные и нажать кнопку еще раз (рисунок 6.6).

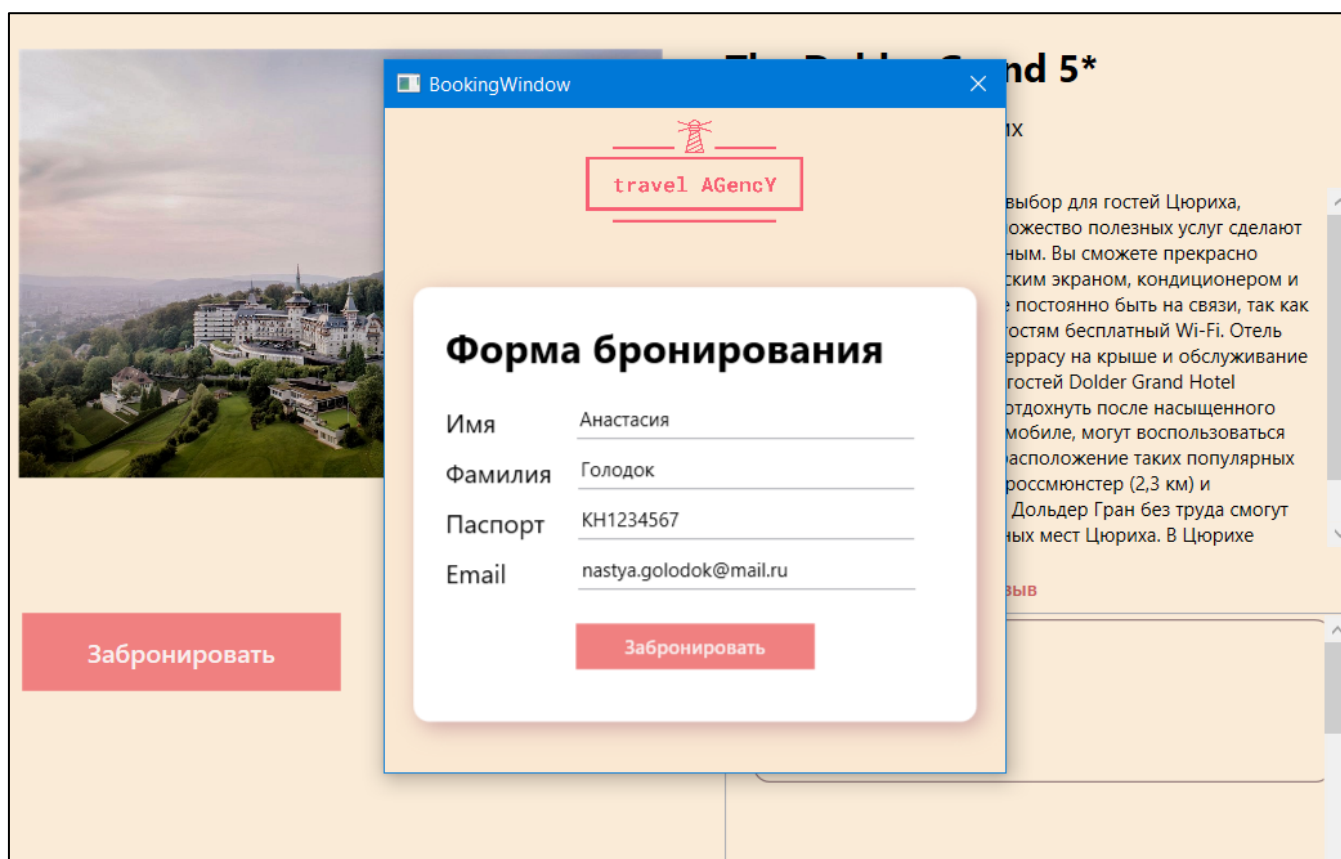


Рисунок 6.6 – Добавление новой задачи

Для того, чтобы просмотреть информацию о брони нужно зайти в личный кабинет (рисунок 6.7).

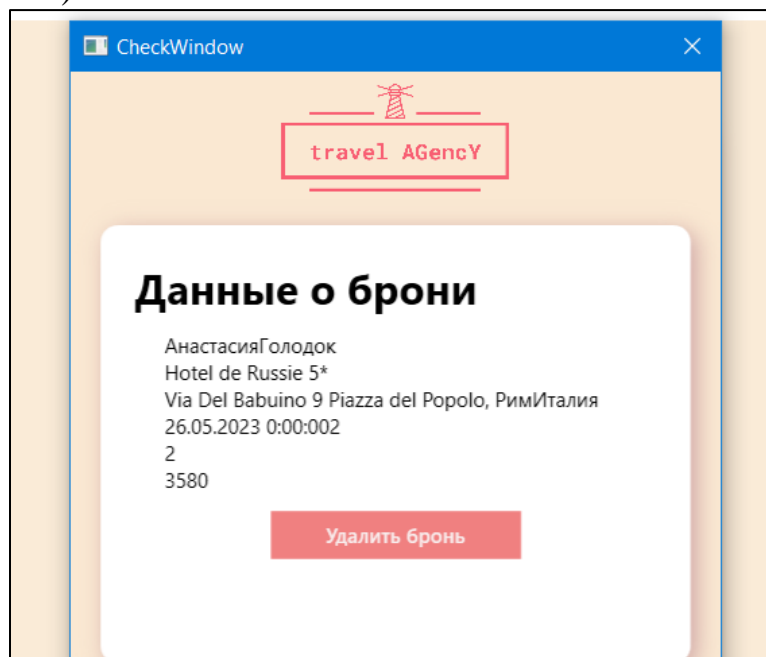


Рисунок 6.7 – Окно брони

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта было разработано программное средство «travel AGENCY» на языке C# с использованием технологий Entity Framework Core, WPF.

При разработке были выполнены все пункты из списка предполагаемого основного функционала приложения:

- создана база данных;
- реализована возможность пользователю зарегистрироваться или войти в существующую учетную запись;
- реализована возможность авторизованным пользователям просмотра и бронирования отеля;
- реализована возможность просмотра и редактирования личной страницы пользователя с отображением информации о бронировании;
- реализована возможность оставлять отзывы;

Дополнительные функции для администратора:

- возможность просмотра всех пользователей и броней;
- реализована функция добавления, редактирования и удаления отелей;

Разработанное программное средство реагирует на ошибочный ввод данных выводя при этом соответствующее сообщение об ошибке. Интерфейс удобен и понятен.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

СПИСОК ЛИТЕРАТУРЫ

1. Пацей, Н.В. Курс лекций по языку программирования C# / Н.В. Пацей. – Минск: БГТУ, 2021. – 175 с. Дата доступа: 5.05.2023
2. WPF tutorial [Электронный ресурс] /Режим доступа: <https://www.wpf-tutorial.com/>. Дата доступа: 16.05.2023
3. Metanit [Электронный ресурс]. – Руководство по WPF. – Режим доступа: <https://metanit.com/sharp/wpf/>. Дата доступа: 16.05.2023
4. Professor WEB [Электронный ресурс]. – Подстановка шрифтов. – Режим доступа: https://professorweb.ru/my/WPF/UI_WPF/level6/6_4.php. Дата доступа: 10.05.2023
5. Microsoft Visual Studio [Электронный ресурс] – https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio – Дата доступа 23.04.2023

ПРИЛОЖЕНИЕ А

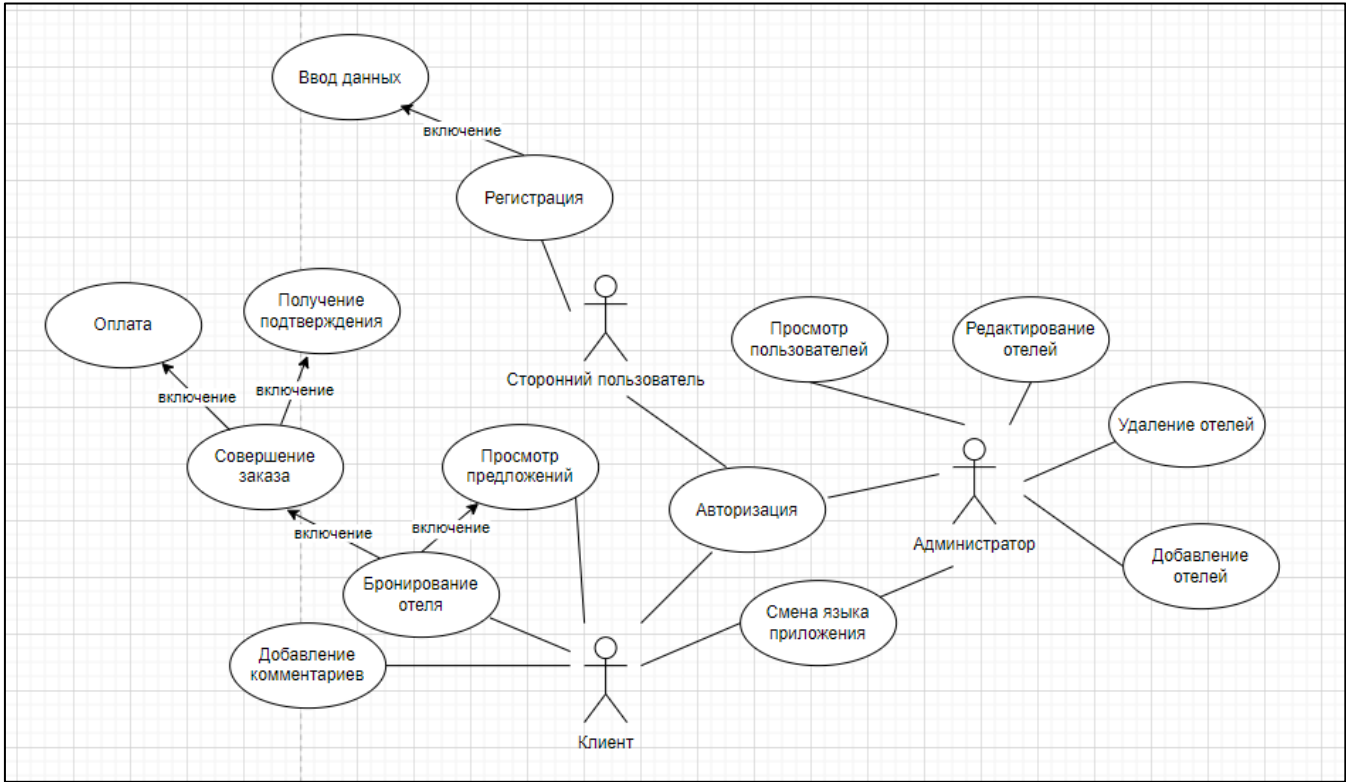


Диаграмма вариантов использования

ПРИЛОЖЕНИЕ Б

```

using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity.Migrations;
using System.Data.SQLite;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace WPF_TravelAgency
{
    /// <summary>
    /// Логика взаимодействия для AdminWindow.xaml
    /// </summary>
    public partial class AdminWindow : Window
    {
        private AppContext db = new AppContext();
        bool photoLoaded = false;

        public AdminWindow()
        {
            InitializeComponent();
            // db = new AppContext();
            Show();
        }

        private OpenFileDialog openFileDialog;
        private Uri fileUri;

        private byte[] B64Encode()
        {
            byte[] imageBytes;
            using (FileStream stream = new FileStream(openFileDialog.FileName,
FileMode.Open, FileAccess.Read))
            {
                using (BinaryReader reader = new BinaryReader(stream))
                {
                    imageBytes = reader.ReadBytes((int)stream.Length);
                }
            }

            return imageBytes;
        }

        private void Button_Main_Click(object sender, RoutedEventArgs e)
        {
            FirstPageWindow firstWindow = new FirstPageWindow();
            firstWindow.Show();
            Close();
        }
    }
}

```

```

private void Button_Photo_Click(object sender, RoutedEventArgs e)
{
    openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "Image|*.jpg;*.jpeg;*.png;";
    if (openFileDialog.ShowDialog() == true)
    {
        fileUri = new Uri(openFileDialog.FileName);
        photoLoaded = true;
    }
}

private void Show()
{
    using (var transaction = db.Database.BeginTransaction())
    {
        try
        {
            var query = from hotel in db.Hotels
                        select new
                        {
                            hotel.idHotel,
                            hotel.title,
                            hotel.country,
                            hotel.city,
                            hotel.descriptionHotel,
                            hotel.stars,
                            hotel.photo,
                            hotel.price,
                            hotel.numRooms
                        };

            tableHotels.ItemsSource = query.ToList();

            var queryUs = from user in db.Users
                          select new
                          {
                              user.id,
                              user.name,
                              user.surname,
                              user.passport,
                              user.email,
                              user.password
                          };

            tableUser.ItemsSource = queryUs.ToList();

            var queryBook = from booking in db.Bookings
                             select new
                             {
                                 booking.idBooking,
                                 booking.idHotel,
                                 booking.id,
                                 booking.dateBoking,
                                 booking.numNights,
                                 booking.numPerson,
                                 booking.finalPrice
                             };

            tableBooking.ItemsSource = queryBook.ToList();
        }
        catch (Exception ex)
        {
            transaction.Rollback();
            MessageBox.Show(ex.Message);
        }
    }
}

```

```

    }
}

private void Button_Save_Click(object sender, RoutedEventArgs e)
{
    string nameHot = Title.Text;
    string country = Country.Text;
    string address = City.Text;
    TextRange txtRich = new TextRange(Description.Document.ContentStart,
Description.Document.ContentEnd);
    string description = txtRich.Text;
    int stars = int.Parse(Stars.Text);
    byte[] photo = B64Encode();
    int price = int.Parse(Price.Text);
    int numRooms = int.Parse(NumRooms.Text);
    if (nameHot != String.Empty || country != String.Empty || address !=
String.Empty || description != String.Empty)
    {
        MessageBox.Show("YcnewHo");

        Hotel hotel = new Hotel
        {
            title = nameHot,
            country = country,
            city = address,
            descriptionHotel = description,
            stars = stars,
            photo = photo,
            price = price,
            numRooms = numRooms
        };
        db.Hotels.Add(hotel);
        db.SaveChanges();
        Show();
    }
}

private void Button_Del_Click(object sender, RoutedEventArgs e)
{
    try
    {
        if (tableHotels.SelectedItems.Count > 0)
        {
            for (int i = 0; i < tableHotels.SelectedItems.Count; i++)
            {
                int id =
(int)tableHotels.SelectedItems[i].GetType().GetProperty("idHotel").GetValue(tableHotels.
SelectedItems[i]);
                Hotel delHotel = db.Hotels.Where(temp => temp.idHotel ==
id).FirstOrDefault();

                if (delHotel != null)
                {
                    db.Hotels.Remove(delHotel);
                }
            }
            db.SaveChanges();
            Show();
        }
    }
    catch (Exception ex)
    {

```



```

        MessageBox.Show(ex.Message);
    }
}

private void Button_Upd_Click(object sender, RoutedEventArgs e)
{
    try
    {
        //Button_Photo_Click(sender, e);
        int id =
(int)tableHotels.SelectedItems[0].GetType().GetProperty("idHotel").GetValue(tableHotels.
SelectedItems[0]);
        Hotel hotel = db.Hotels.Where(temp => temp.idHotel ==
id).FirstOrDefault();

        TextRange txtRich = new TextRange(Description.Document.ContentStart,
Description.Document.ContentEnd);
        string description = txtRich.Text;

        /*if (fileUri == null)
        {
            /**string base64String = Convert.ToBase64String(hotel.photo);
            string dataUri = $"data:image;base64,{base64String}";
            fileUri = new Uri(dataUri);**/

            hotel.photo = B64Encode();
        }*/

        hotel.title = Title.Text;
        hotel.city = City.Text;
        hotel.descriptionHotel = txtRich.Text;
        hotel.price = int.Parse(Price.Text);
        hotel.stars = int.Parse(Stars.Text);
        hotel.numRooms = int.Parse(NumRooms.Text);
        if (photoLoaded)
        {
            hotel.photo = B64Encode();
        }
        else
        {
            hotel.photo = hotel.photo;
        }

        db.SaveChanges();
        Show();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void tableHotels_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
    if (tableHotels.SelectedItem != null)
    {
        TextRange txtRich = new TextRange(Description.Document.ContentStart,
Description.Document.ContentEnd);
        string description = txtRich.Text;
        int id =
(int)tableHotels.SelectedItems[0].GetType().GetProperty("idHotel").GetValue(tableHotels.
SelectedItems[0]);

```

```

        Hotel selectedItem = db.Hotels.Where(temp => temp.idHotel ==
id).FirstOrDefault();

        string selectTitle = selectedItem.title;
        string selectCity = selectedItem.city;
        string selectCountry = selectedItem.country;
        string selectDesc = selectedItem.descriptionHotel;
        string selectStar = Convert.ToString(selectedItem.stars);
        string selectPrice = Convert.ToString(selectedItem.price);
        string selectNumRoom = Convert.ToString(selectedItem.numRooms);

        Title.Text = selectTitle;
        City.Text = selectCity;
        Country.Text = selectCountry;
        txtRich.Text = selectDesc;

        Stars.Text = selectStar;
        Price.Text = selectPrice;
        NumRooms.Text = selectNumRoom;
    }
}

private void Button_Clear_Click(object sender, RoutedEventArgs e)
{
    TextRange txtRich = new TextRange(Description.Document.ContentStart,
Description.Document.ContentEnd);
    string description = txtRich.Text;

    Title.Text = string.Empty;
    City.Text = string.Empty;
    Country.Text = string.Empty;
    txtRich.Text = string.Empty;
    Stars.SelectedItem = null;
    Price.Text = string.Empty;
    NumRooms.Text = string.Empty;
}
}
}

```

Листинг – Реализация администратора

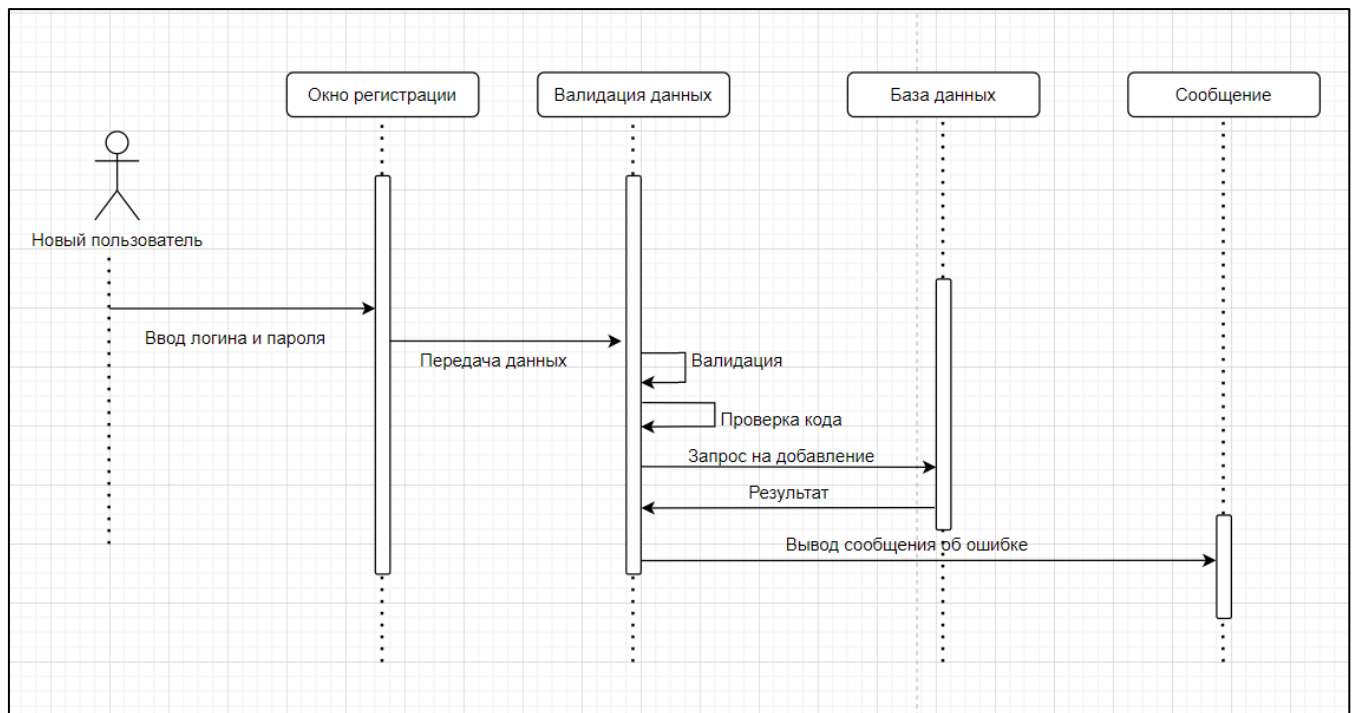
ПРИЛОЖЕНИЕ В

Диаграмма последовательности