МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

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

Тема: Программное средство для бронирования номеров в отеле

Исполнитель

студент (ка) 2 курса группы 4 Копыл Ефим Игоревич

(Ф.И.О.)

Руководитель работы ассистент Северинчик Н.А.

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

Курсовой проект защищен с оценкой

Председатель Северинчик Н.А.

(подпись)

Минск 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_ \_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированное

программирование"

|  |  |
| --- | --- |
| Специальность: 1-40 01 01 Программное обеспечение информационных технологий  Студент: Копыл Е.И. | Группа: 4 |
| **Тема: Программное средство для бронирования номеров в отеле** | |

**1. Срок сдачи студентом законченной работы**: "29 мая 2020 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции администратора:
  + Поддерживать работу c базой данных;
  + Регистрировать бронь;
  + Оповещать клиента о выполнении брони (почта, sms);
  + Выполнять поисковые запросы, фильтрацию.
* Функции клиента:
  + Выполнять регистрацию и авторизацию;
  + Заполнять форму брони по заданным формам;
  + Просматривать информацию о статусе брони;
  + Оставлять отзывы.

**2.2.** При выполнении курсового проекта необходимо использовать принципы проектирования ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. При разработке использовать несколько наиболее подходящих шаблонов проектирования ПО.

**3. Содержание расчетно-пояснительной записки**

(перечень вопросов, подлежащих разработке)

* Введение
* Постановка задачи и обзор литературы (алгоритмы решения, обзор прототипов, актуальность задачи)
* Проектирование архитектуры проекта (структура модулей, классов).
* Разработка функциональной модели и модели данных ПС (выполняемые функции)
* Тестирование
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненного курсового проекта:**

* + Теоретическая часть курсового проекта должна быть представлена в формате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются частично в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить на диск, указанный преподавателем.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2020 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2020 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2020 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2020 |  |
| 5 | Кодирование программного средства | 23.04.2020 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2020 |  |
| 7 | Оформление пояснительной записки | 7.05.2020 |  |
| 9 | Сдача проекта | 29.05.2020 |  |

**5. Дата выдачи задания** 12.02.2020 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Н. А. Северинчик*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

Содержание

[ВВЕДЕНИЕ 6](#_Toc72747212)

[1 Аналитический обзор прототипов и литературных источников 7](#_Toc72747213)

[2 Анализ требований к программному средству и разработка функциональных требований 10](#_Toc72747214)

[3 Проектирование программного средства 12](#_Toc72747215)

[**3.1 Проектирование архитектуры приложения** 12](#_Toc72747216)

[3.2 Проектирование базы данных 14](#_Toc72747217)

[3.3 Проектирование доступа к базе данных 17](#_Toc72747218)

[3.4 Проектирование вариантов использования 17](#_Toc72747219)

[4 Реализация программного средства 20](#_Toc72747220)

[4.1 Реализация сущностей 20](#_Toc72747221)

[4.2 Реализация архитектуры MVVM 20](#_Toc72747222)

[**4.3 Реализация авторизации и регистрации** 23](#_Toc72747223)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 23](#_Toc72747224)

[6 Руководство по использованию 29](#_Toc72747225)

[6.1 Руководство по использованию администратором 31](#_Toc72747226)

[6.2 Руководство по использованию пользователем 33](#_Toc72747227)

[ЗАКЛЮЧЕНИЕ 37](#_Toc72747228)

[СПИСОК ЛИТЕРАТУРЫ 38](#_Toc72747229)

[ПРИЛОЖЕНИЕ А 39](#_Toc72747230)

[ПРИЛОЖЕНИЕ Б 40](#_Toc72747231)

[ПРИЛОЖЕНИЕ В 41](#_Toc72747232)

[ПРИЛОЖЕНИЕ Г 42](#_Toc72747233)

[ПРИЛОЖЕНИЕ Д 43](#_Toc72747234)

ВВЕДЕНИЕ

Данный курсовой проект посвящён разработке программного средства для бронирования номеров в отеле, основной целью которого является автоматизация работы отеля. В настоящее время существует малое количество аналогов, реализованных в качестве десктопных приложений. Большинство таких сервисов представлено в виде Web-сайтов.

Приложение «Hotel Diamond Plaza» - инструмент позволяющий вам ознакомиться со списком свободных номеров, без особых усилий заказать бронь на удобную для вас дату, посмотреть вашу бронь.

Также, большое внимание я уделил дизайну. Я постарался над создание удобного, простого, а, главное, понятного интерфейса. Гармоничное расположение элементов и их оформление в стиле минимализма прекрасно сочетаются со всем нужными полями заполнения для оформления брони номера в отеле.

В качестве интерфейса приклaдного программирования был выбран обширный API-интерфейс — Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом.

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

Хранение данных осуществляется в Microsoft SQL Server.

1 Аналитический обзор прототипов и литературных источников

Важным этапом в разработке программного продукта является аналитический сбор информации прототипов и различных источников информации с темой разработки.

На сегодняшний день приложения для бронирования номеров в отеле довольно распространены, но малое количество имеет интуитивно понятный интерфейс.

Рассмотрим наиболее популярные программы для бронирования номеров:

«Amon Hotels» - это ПО для бронирования номеров в отелях Amon по всему миру. Интерфейс «Amon Hotels» представлен на рисунке 1.1, 1.2

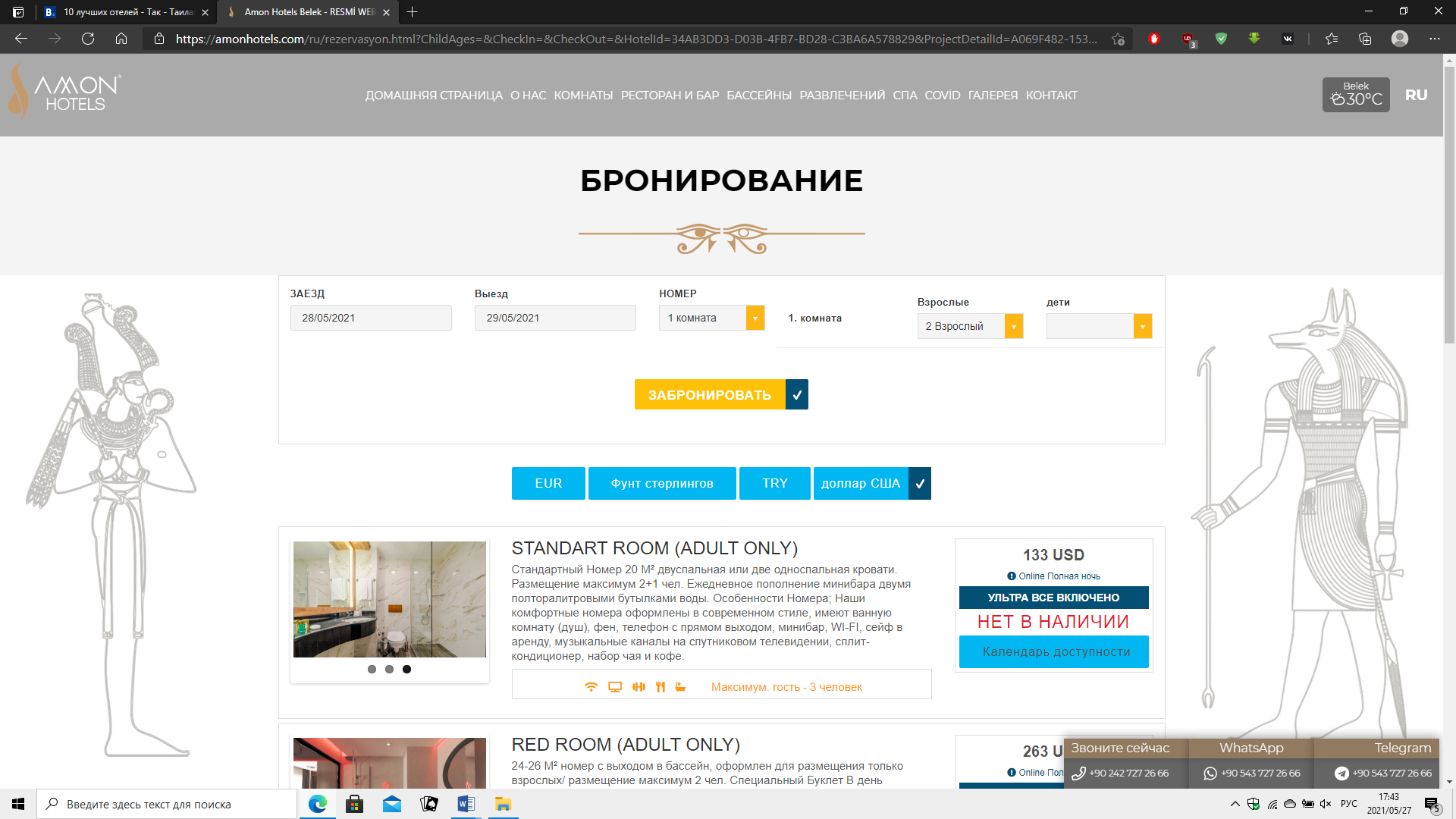


Рис. 1.1 – «Amon Hotels» бронирование номера

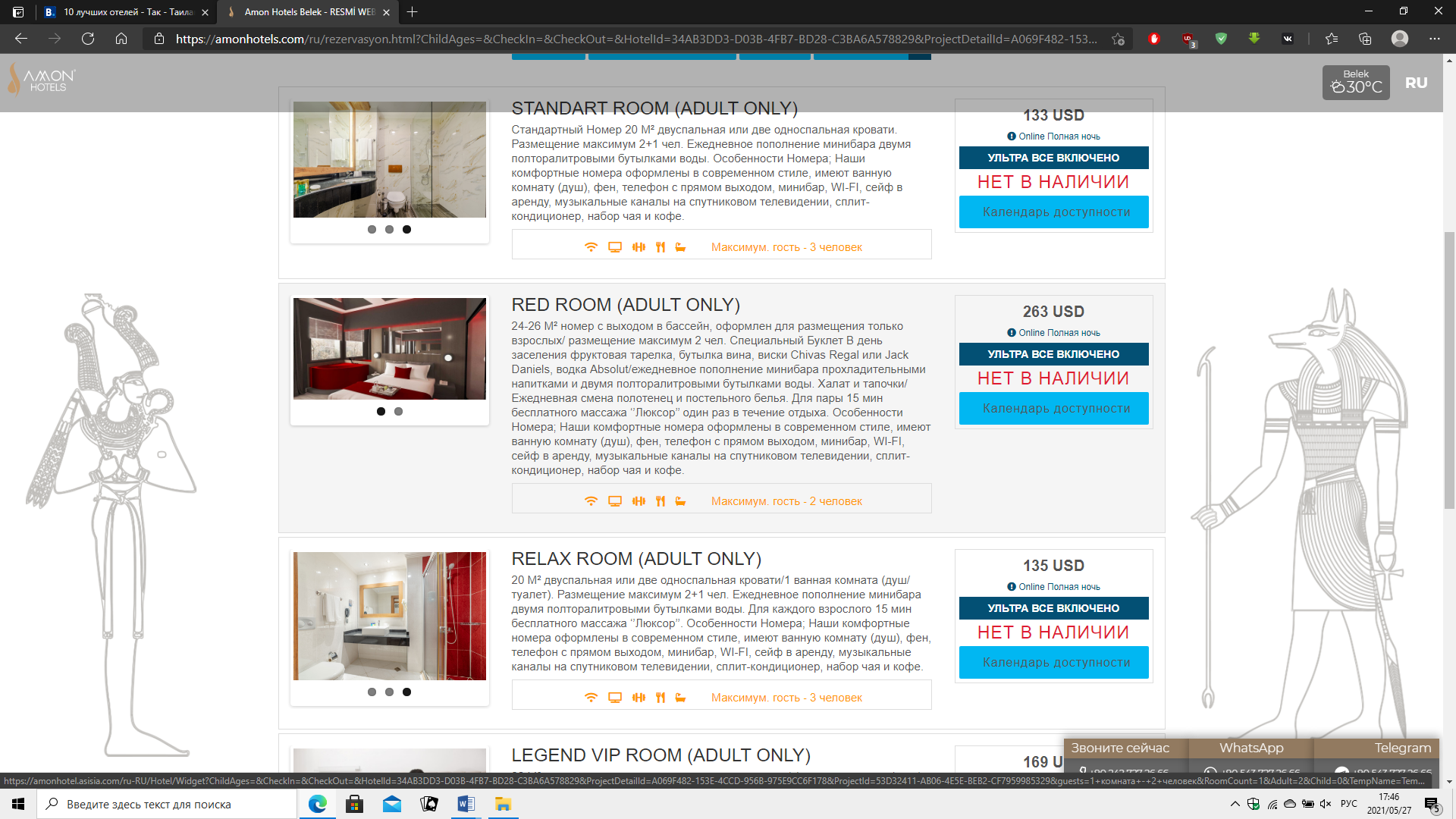


Рис. 1.2 – «Amon Hotels» информация о номере

Проанализировав «Amon Hotels», можно выделить основные минусы данного программного сред.

Основные минусы:

* Загруженный интерфейс;
* Отсутствует информация об отеле.

Основные плюсы:

* Неплохой UI;
* Подробное описание номера;
* Хороший выбор номероа.

Продукт «Booking.com» - cервис бронирования номеров в отеле

Интерфейс «Booking.com» представлен на рисунке 1.3, 1.4

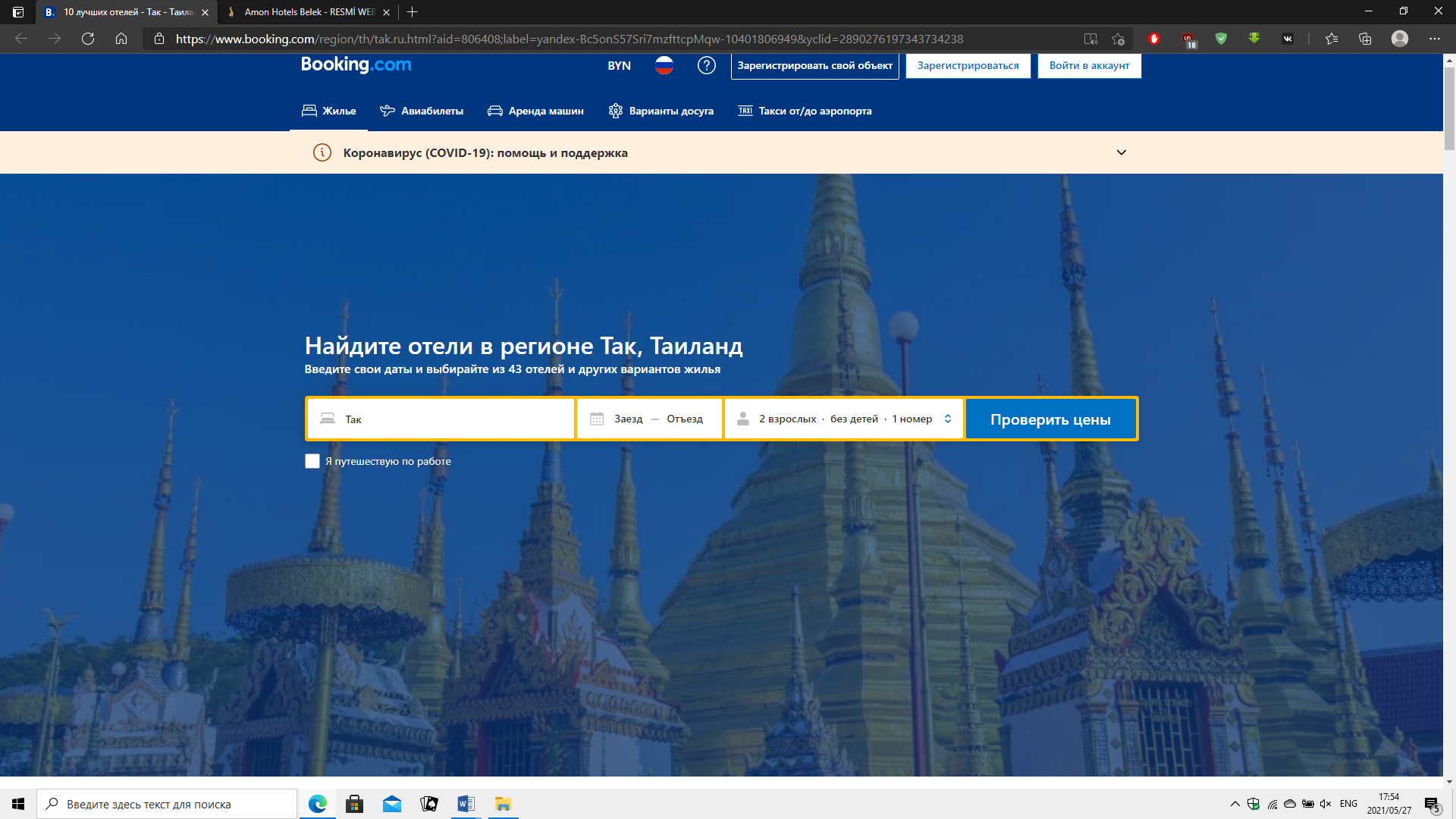


Рис. 1.3 – «Booking.com» главная страница

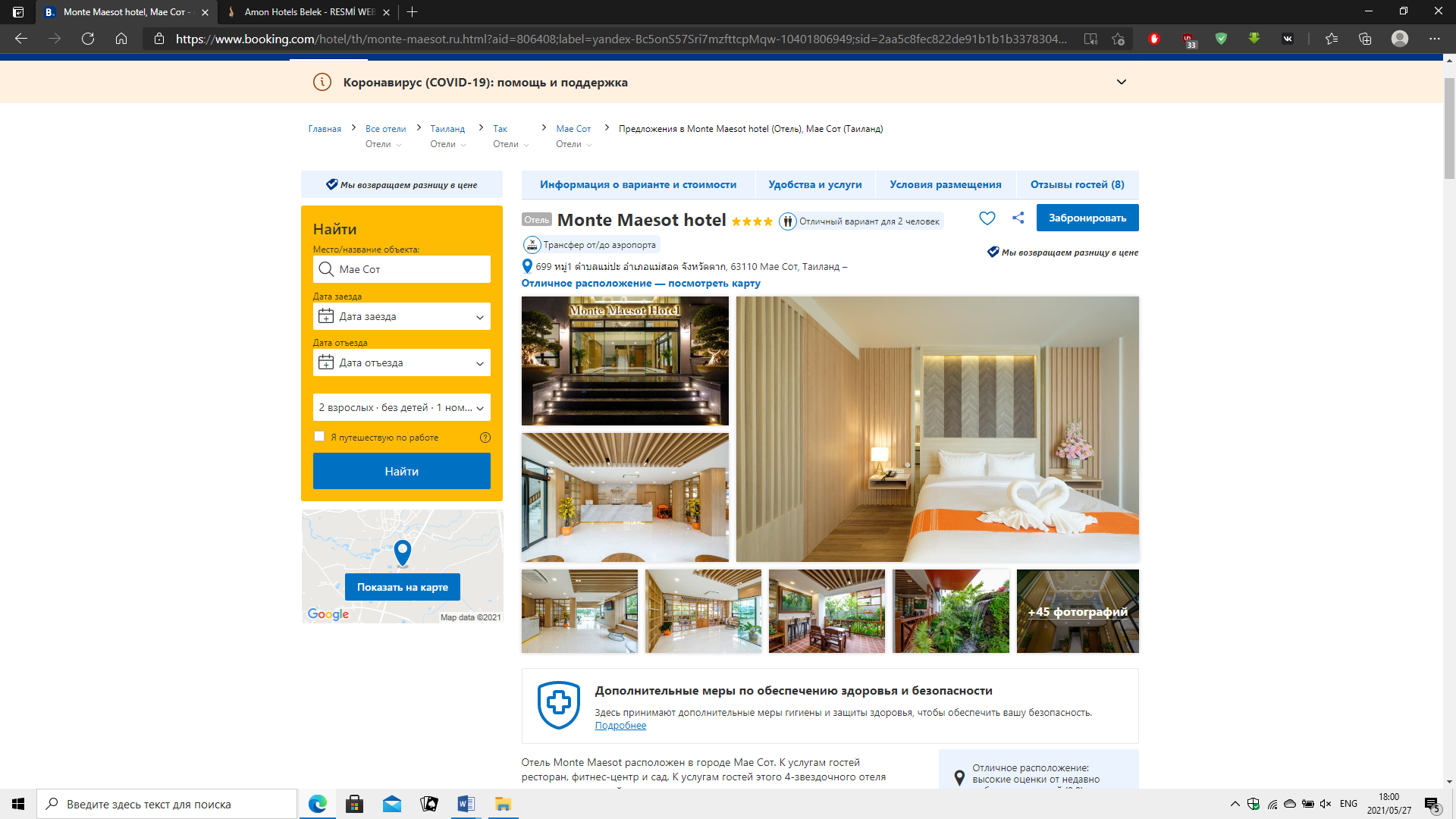


Рис. 1.3 – «Booking.com» главная страница отеля

Проанализировав ПО «Booking.com», можно выделить её основные минусы и плюсы.

Основные минусы:

* Сложный интерфейс;
* Огромное кол-во элементов в интерфейсе.

Основные плюсы:

* Большой выбор номеров и отелей;
* Простые и приятные цвета.

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.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

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 объектно-ориентированного программирования;
* 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. Пользователь данного программного решения должен иметь возможность:

* Регистрироваться в системе;
* Ознакомиться со свободными номерами;
* Получить полную информацию о номере;
* Производить поиск необходимого номера;
* Просматривать предложенные ему номера;
* Выбор брони с подходящим для него временем и номером;
* Забронировать номер;
* Просматривать свою бронь;
* Отменить бронь.

Администратор имеет возможность:

* вносить новую информацию о Номерах, Ценах номеров, Типах номеров;
* удалять информацию о Номерах и автоматически выселять гостей из номеров;

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 аутентификацию и входить в систему под соответствующим введенным данным;
* приложение должно предоставлять возможность поиска свободных номеров по следующим критериям: кол-во взрослых и детей и дата заезда и выезда.
* Приложение должно уведомлять пользователя об успешном заказе билетов путем отправки сообщения на почту.

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 сформированными списком.

3 Проектирование программного средства

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектировaния является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

## **3.1 Проектирование архитектуры приложения**

Архитектура программного обеспечения — совокупность вaжнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбрaнных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для частичного удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении частично используется архитектурный шаблон Model-View-ViewModel (MVVM). В данном приложении он был применен только для смены типа авторизации пользователя, в дальнейшем в приложении происходит взаимодействие между окнами.

Шаблон MVVM имеет три основных слоя: модель, которая представляет бизнес-логику приложения, представление пользовательского интерфейса, и представление-модель, в котором содержится вся логика построения графического интерфейса и ссылка на модель, поэтому он выступает в качестве модели для представления.

На рисунке 3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.



Рисунок 3.1 – Структура шаблона MVVM

Модель описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную 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.

Нередко модель реализует интерфейсы INotifyPropertyChanged, который позволяет уведомлять систему об изменениях свойств модели. Благодаря этому облегчается привязка к представлению, хотя взаимодействие между моделью и представлением отсутствует.

View определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Интерфейс и качество его реализации занимает важную роль в конечном результате, разработка эффективного и понятного интерфейса, является важной задачей. Поэтому для создания качественного интерфейса необходимо понять, как пользователь будет взаимодействовать с приложением. Для этого была составлена схема (рис. 3.2), на которой представлен принцип работы приложения с точки зрения пользователя и администратора.

ViewModel или 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.

ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели. Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд.

Для разработки приложения в качестве сервиса для хранения данных была выбрана СУБД Microsoft SQL Server. На диаграмме размещение на рисунке 3.2 показано графическое представление инфраструктуры приложения.

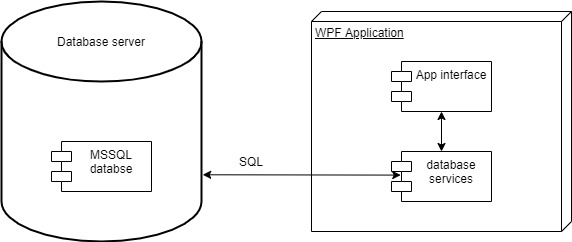


Рисунок 3.2 – Диаграмма размещения

3.2 Проектирование базы данных

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности базы данных.

Проектирование базы данных проводится в два этапа: концептуальное (инфологическое) и логическое (даталогическое) проектирование.

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня aбстракции. В результате этого этапа создаётся ER-модель. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Основными понятиями ER-модели являются: сущность, связь и атрибут

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация обычно является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибут сущности это любая детaль, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

В рамках этого этапа была создана ER-модель, которая включает 4 сущностей:

* администратор;
* номер в отеле;
* гость;
* комментарий пользователя.

Также в ER-модели были определены необходимые связи. Например, между сущностями администратор и номер в отеле была установлена связь один-ко-многим. Для каждой сущности были выделены атрибуты. Например, для Номер в отеле в качестве атрибутов были выделены такие характеристики, как идентификатор управляющего администратора, идентификатор номера в отеле, идентификатор гостя в этом номере.

Логическое (даталогическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели дaнных логическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Логическая модель базы данных представлена на рисунке 3.3

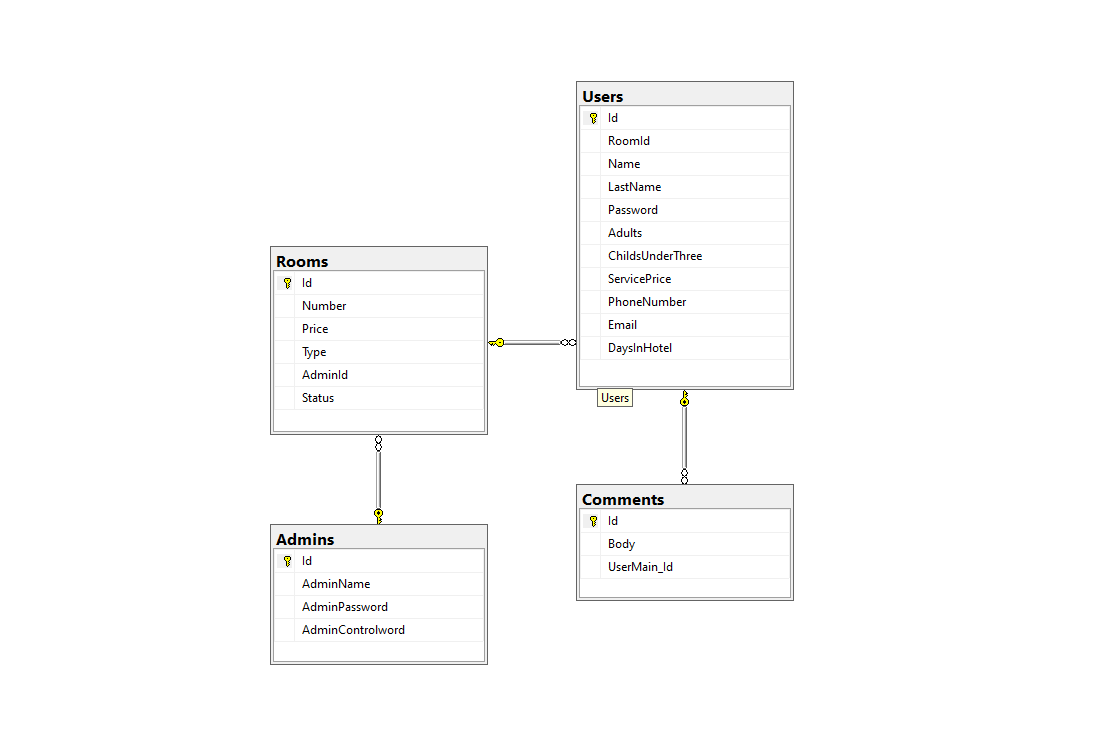


Рисунок 3.3 - Логическая модель базы данных

Всего в базе данных содержится 4 таблицы. В таблице Users хранятся все пользователи, зарегистрированные в приложении. Таблица Comments содержит информацию о отзывах, которые сделал гость отеля. В таблице Rooms содержится информация о номерах в отелях. Таблица Admins содержит информацию о всех подконтрольных номерах отеля и о всех администраторах отелей.

Таблица Users состоит из 11 столбцов:

* Id;
* RoomId;
* Name;
* LastName;
* Password;
* Adults;
* ChildsUnderThree;
* ServicePrice;
* PhoneNumber;
* Email;
* DaysInHotel.

В столбце RoomId хранится Id номера в котором поселился гость, Name – имя гостя, LastName – фамилия гостя, Password – пароль зарегистрированного пользователя, Adults – кол-во взрослых в номере (включая самого гостя), ChildUnderThree – кол-во детей до 3-ех лет, ServicePrice – сумма к оплате за услуги отеля, PhoneNumber – номер телефона гостя для возможной связи с ним, Email – адрес электронной почты гостя отеля, DaysInHotel – период проживания гостя в отеле в днях.

Таблица Rooms состоит из 6 столбцов:

* Id;
* Number;
* Price;
* Type;
* AdminId;
* Status.

В столбце Number хранится № номера в отеле, Price – цена номера за ночь, Type – типо комнаты (Default/VIP), AdminId – идентификатор управляющего администратора, Status – статус номера (свободен/занят).

Таблица Admins состоит из 4 столбцов:

* Id;
* AdminName;
* AdminPassword;
* AdminControlword.

В столбце AdminName хранится имя аккаунта администратора, в столбце AdminPassword – пароль аккаунта администратора, AdminControlword – ключевое слово аккаунта администратора.

Таблица Comments состоит из 3 столбцов:

* Id;
* Body;
* UserMain\_Id.

В столбце Body хранится текст отзыва гостя отеля, в столбце UserMain\_Id – идентификатор аккаунта гостя, оставившего отзыв.

3.3 Проектирование доступа к базе данных

Для доступа к базе данных используется Entity Framework Core 6. Этот подход предоставляет ряд существенных преимуществ: нам не нужно беспокоиться о коде доступа к данным и не нужно знать деталей работы СУБД SQL Server и синтаксиса языка запросов T-SQL, вместо этого мы работаем с таблицами базы данных как с классами C#, с полями этих таблиц - как со свойствами классов, а синтаксис SQL-запросов заменен на  [LINQ](https://professorweb.ru/my/LINQ/base/level1/info_linq.php). Entity Framework Core берет на себя обязанности по преобразованию кода C# в SQL-инструкции.

Существуют 3 подхода по проектированию базы данных:

* Database-First;
* Model-First;
* Code-First.

В своей работе я использовал подход Code-First, при данном подходе модель EDMX не используется, и мы вручную настраиваете классы C# объектной модели (данный подход поддерживает как генерацию сущностных классов из существующей базы данных, так и создание базы данных из созданной вручную модели объектов C#).

3.4 Проектирование вариантов использования

На рисунке 3.4 представлена диаграмма использования приложения для пользователя и администратора. Данная диаграмма представлена в приложении Д.

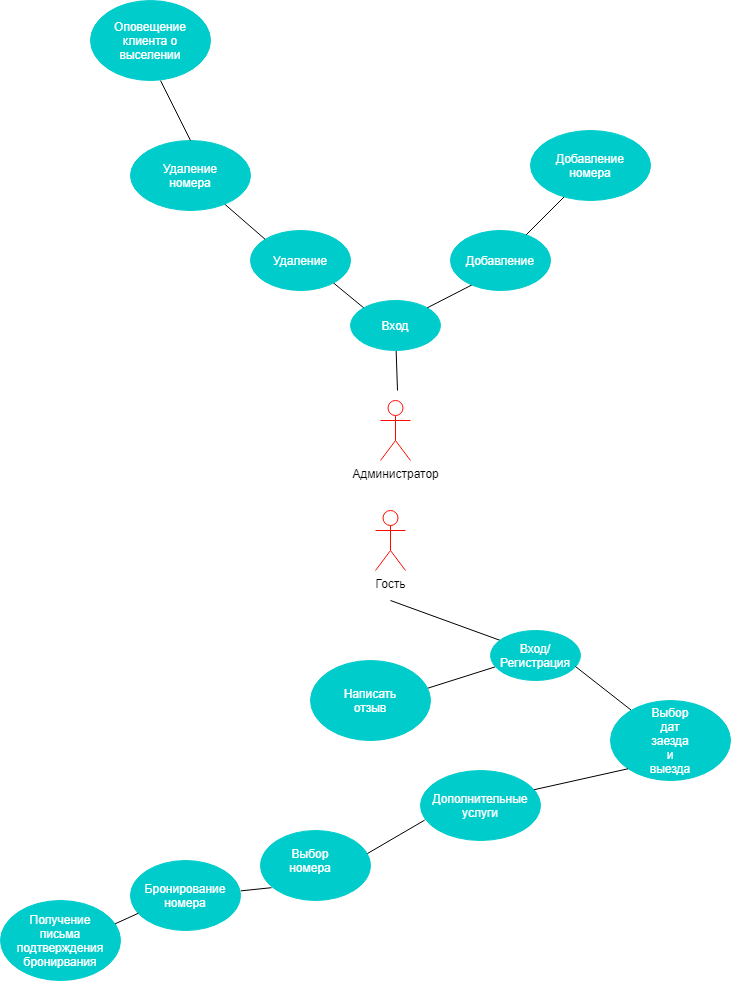


Рисунок 3.4 - Диаграмма вариантов использования для пользователя и администратора

После входа в приложение пользователю откроется окно заполнения данных для брони, где пользователь может ознакомиться со всеми возможными дополнительными услугами, выбрать даты проживания, а также оставить отзыв. Далее следует перейти в окно окончательного оформления брони, где, указав номер телефона, адрес электронной почты и номер, в котором решили остановиться, вы получите письмо с информацией о своей брони.

При авторизации пользователя с ролью «Администратор» ему будет доступна панель администратора со всеми соответствующим функционалом.

Для администратора была создана отдельная ветвь окон, при помощи которых возможно добавление и удаление данных о номерах отеля.

4 Реализация программного средства

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

4.1 Реализация сущностей

В соответствии с требованиями в качестве хранилища данных программного средства должна быть база данных, поэтому первым шагом в реализации программы является выбор технологии, позволяющей это осуществить. Выбор остановился на ORM технологии Entity Framework. Она предоставляет три подхода по проектированию базы данных. В данном программном решении был использован подход Code-First. При данном подходе модель EDMX не используется. Создание базы данных происходит из созданной вручную модели объектов C#. Созданные модели объектов совпадают с сущностями, которые были сформированы раннее в разделе 3.2.

Диаграмма классов UML для сущностных классов представлена на рисунке

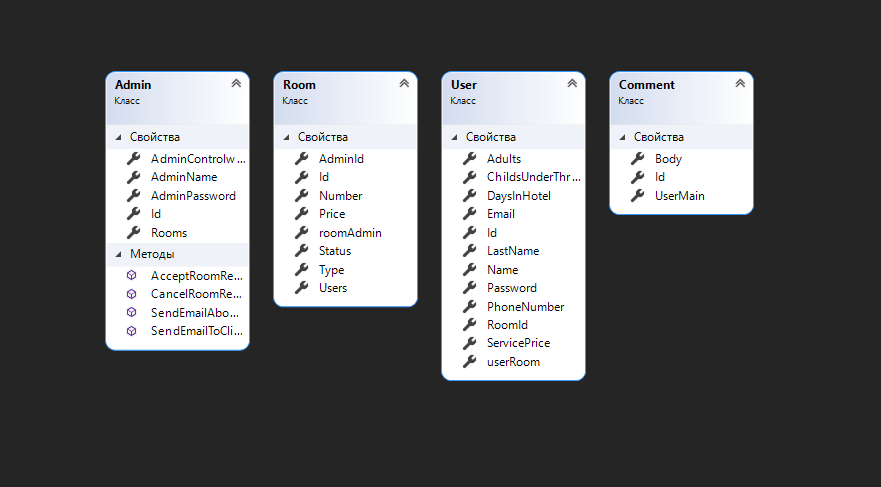


Рисунок 4.1 − Диаграмма классов

4.2 Реализация архитектуры MVVM

Проект построен на архитектуре MVVM, поэтому файлы проекта разделены на 3 логических модуля: Model, View и ViewModel.

Разделение проекта на логические модули представлено на рисунке 4.2.

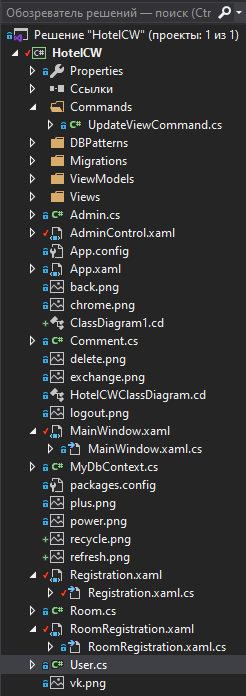


Рисунок 4.2 – Логические модули проекта

В папке DBPatterns находятся классы, реализующие паттерны работы с базой данных, такие как Repository и Unit of work.

В папке проекта содержатся сущностные классы, которые используются для создания БД. Описание сущностных классов представлено в разделе 3.2.

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. . Данный класс представлен в листинге 4.1

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

using HotelCW.ViewModels;

namespace HotelCW.Commands

{

class UpdateViewCommand : ICommand

{

private MainViewModel viewModel;

public UpdateViewCommand(MainViewModel viewModel)

{

this.viewModel = viewModel;

}

public event EventHandler CanExecuteChanged;

public bool CanExecute(object parameter)

{

return true;

}

public void Execute(object parameter)

{

if (parameter.ToString() == "User")

{

viewModel.SelectedViewModel = new UserViewModel();

}

else if (parameter.ToString() == "Admin")

{

viewModel.SelectedViewModel = new AdminViewModel();

}

}

}

}

Листинг 4.1 – Класс Command

Класс реализует два метода:

* CanExecute: определяет, может ли команда выполняться;
* Execute: собственно, выполняет логику команды.

В папке ViewModels находятся все классы ViewModel для каждого представления. Диаграмма классов UML для классов модуля ViewModel представлена на рисунке 4.1. Также данная диаграмма представлена в приложении Г.

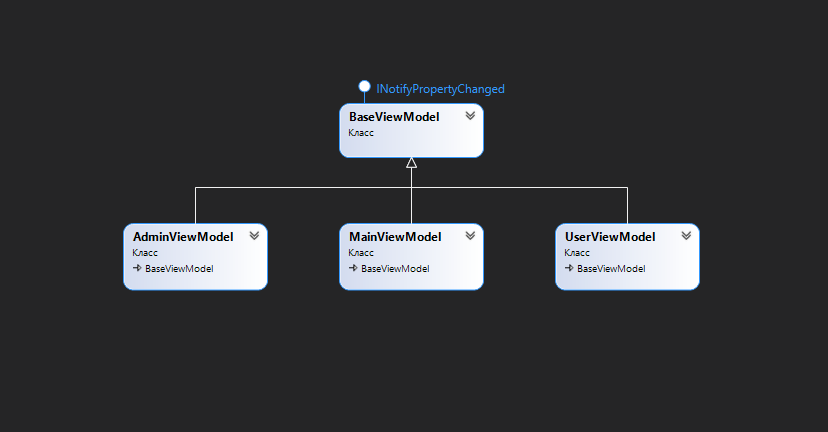


Рисунок 4.1 – Диаграмма классов модуля ViewModel

На диаграмме 4.1 видно, что все классы ViewModel наследуются от базового класса ViewModel – BaseViewModel.

В папке Views хранятся все используемые представления.

**4.3 Реализация авторизации и регистрации**

Для того, чтобы пользоваться приложением, необходимо для начала зарегистрироваться в системе. В приложении А представлен код реализующий регистрацию.

Если пользователь уже зарегистрирован, то следует войти в систему. В приложении Б представлен код реализующий вход в систему.

В базе данных хранятся захешированные пароли пользователей. Хеширование было произведено при помощи метода GetHashCode(), который позволяет зашифровывать данные.

5 Тестирование, проверка работоспособности и анализ полученных результатов

В курсовом проекте задействуется обработка исключений, таким образом, что пользователь будет уведомлен о неудачном выполнении операции. Присутствуют различные всплывающие окна, выводящие сообщение об ошибке. На рисунках 5.1-5.3 приведены примеры обработки ошибок при неверно введенных данных в окне регистрации в приложение.

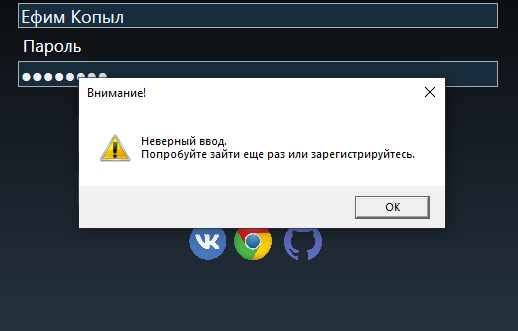


Рисунок 5.1 – Пользователь ввел неверные данные для авторизации

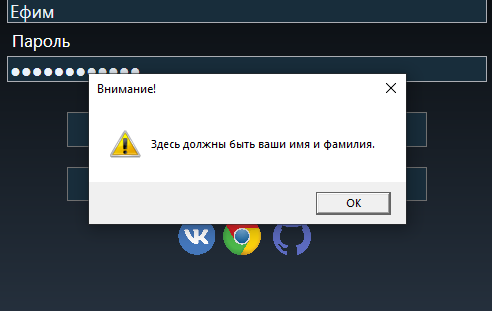


Рисунок 5.2 – Введено только имя гостя для регистрации

(необходимо ввести имя и фамилию)

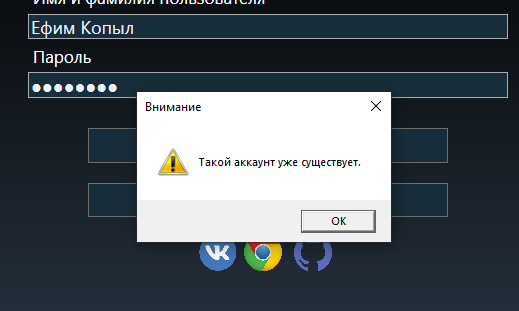


Рисунок 5.3 – Пользователь с таким аккаунтом уже зарегистрирован

Также присутствует уведомление о других ошибках: например, неправильный формат данных или просьба выполнить какое-то действие, которое пользователь забыл. Примеры представлены на рисунке 5.4

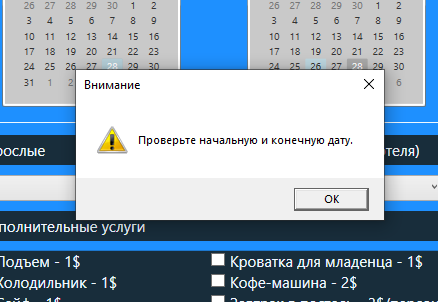


Рисунок 5.4 – Просьба заполнить все поля для бронирования номера

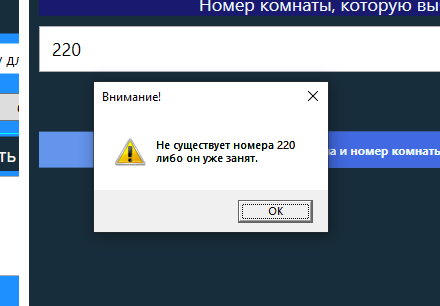


Рисунок 5.5 – Ошибка при неверно указанном номере гостем в окне

окончательной регистрации

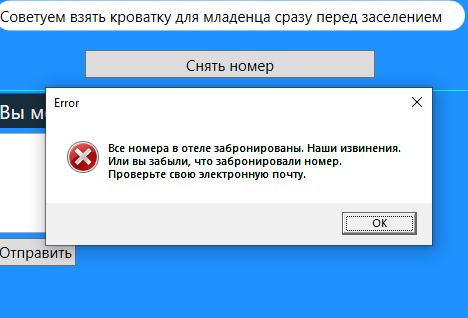


Рисунок 5.6 – Уведомление гостя в случае повторного бронирования с уже забронированным номером

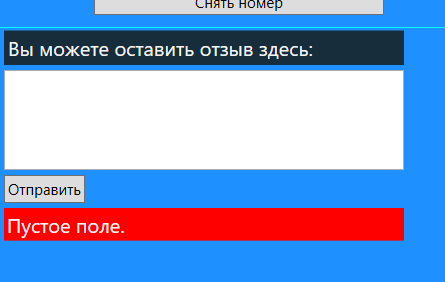


Рисунок 5.7 – Ошибка при попытке отправить пустой отзыв

Кроме этого, были продуманы уведомления о других ошибках со стороны администратора: например, неправильный формат данных или просьба выполнить какое-то действие, которое администратор забыл. Примеры представлены на рисунке 5.8

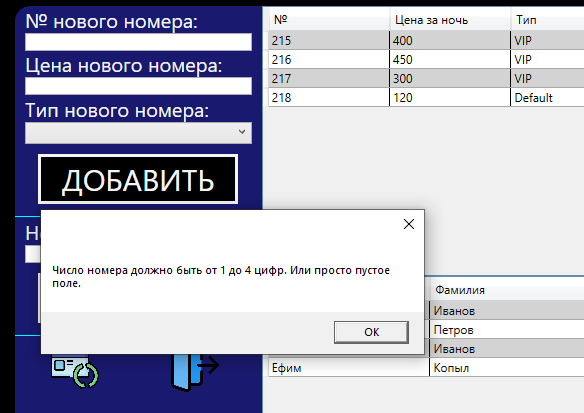


Рисунок 5.8 – Просьба корректно ввести № номера

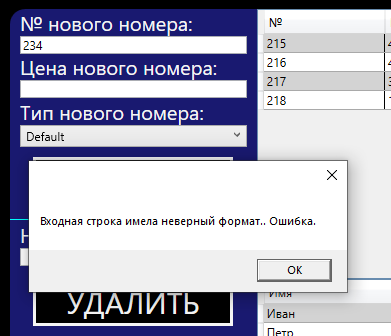


Рисунок 5.9 – Ошибка при отсутствии цены номера

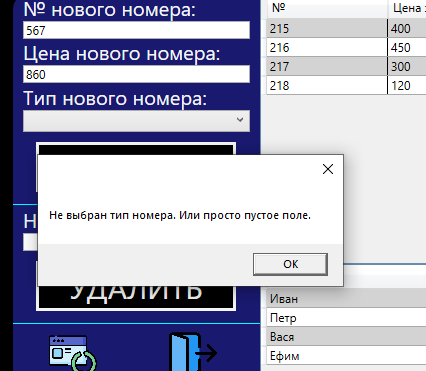


Рисунок 5.10 – Просьба корректно ввести тип номера

6 Руководство по использованию

Для того, чтобы пользоваться приложением, необходимо для начала зарегистрироваться в системе. Окно регистрации с уведомлением показано на рисунке 6.1

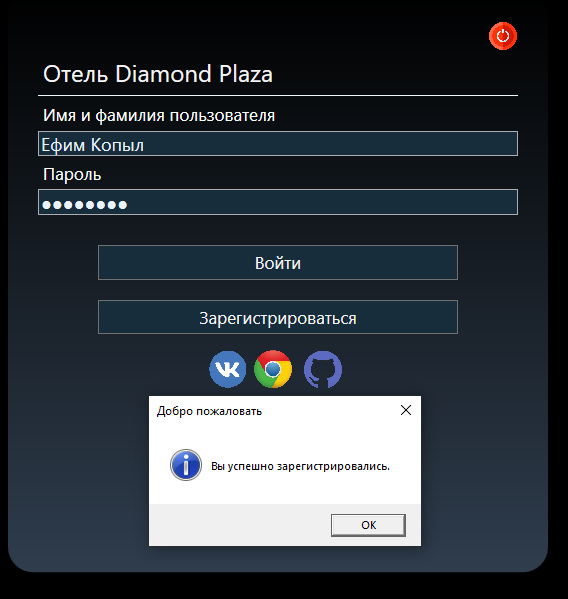


Рисунок 6.1 – Окно регистрации

Если у пользователя уже есть аккаунт, то нужно войти в систему. Окно входа показано на рисунке 6.2.

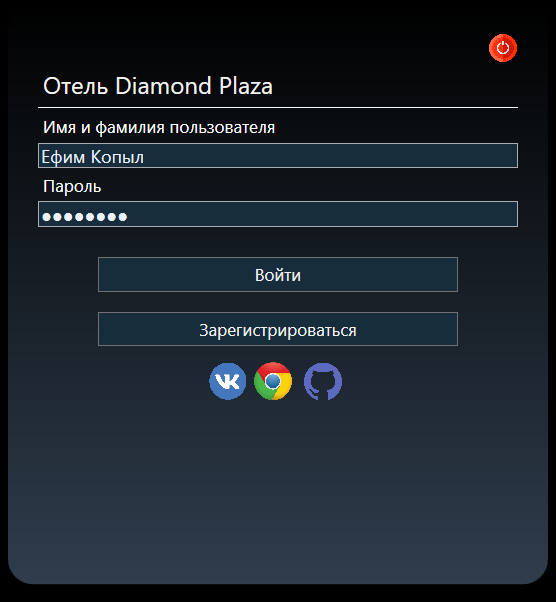


Рисунок 6.2 – Окно входа

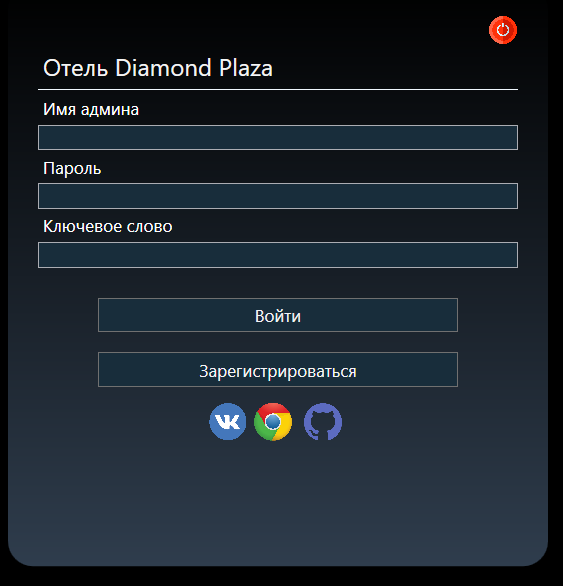


Рисунок 6.3 – Окно входа администратора

6.1 Руководство по использованию администратором

После входа в систему администратору откроется главное окно, в котором ему станут доступна панель работы администратора, где находятся все необходимые элементы управления.

Рабочая область администратора и описанные элемент интерфейса представлены на рисунке 6.4

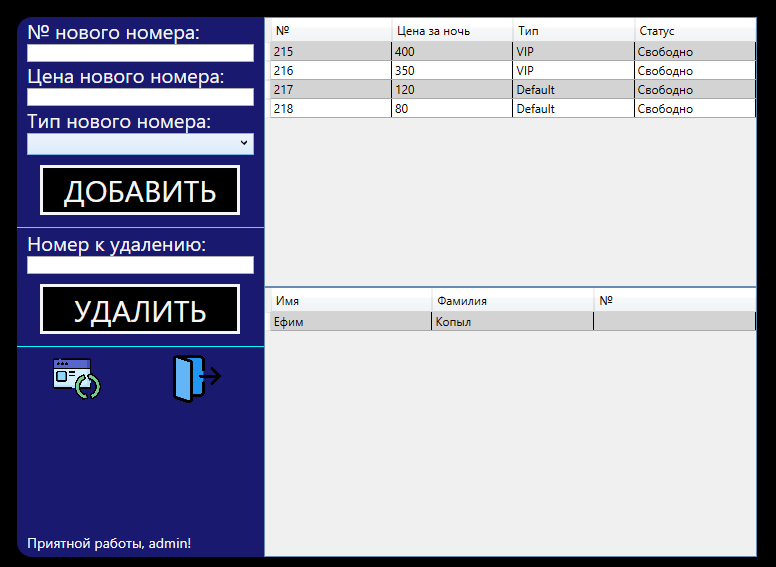


Рисунок 6.4 – Окно администратора

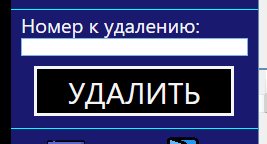


Рисунок 6.5 – Выбор номера для удаления

При помощи кнопок «Добавить администратор может добавить новый объект в базу данных. Представлено на рисунках 6.6

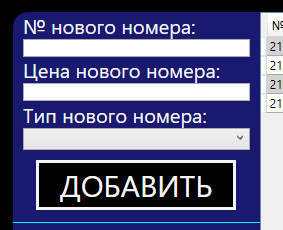


Рисунок 6.6 – Область добавления номера

При нажатии на иконку выхода из аккаунта пользователь может вернуться к окну авторизации.

6.2 Руководство по использованию пользователем

После входа в систему пользователю откроется главная панель. Необходимо заполнить поля формы, после чего выбрать интересующий номер. Представлено на рисунке 6.7.

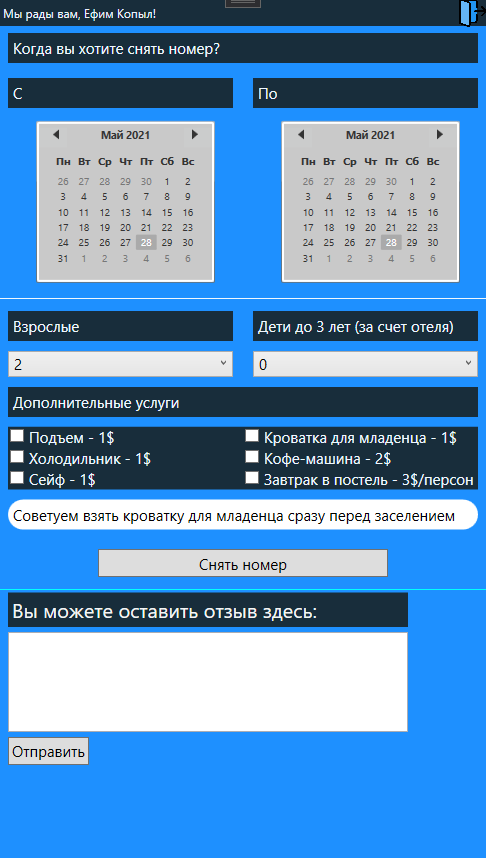


Рисунок 6.7 – Главная панель

Отзыв можно оставить в нижней секции панели. Представлено на рисунке 6.10.

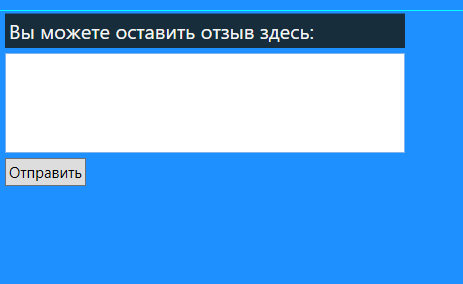


Рисунок 6.10 – Отзыв об услугах отеля

Выбор номера осуществляется при помощи комбобокса внутри которых доступные номера с описанием. После успешного заказа пользователю на почту прихоидт уведомление. Представлено на рисунке 6.11.

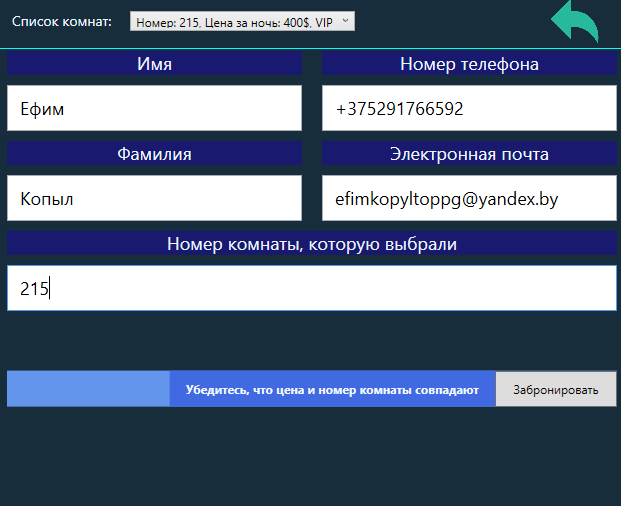


Рисунок 6.11 – Выбор номера

Письмо, приходящее пользователь на электронную почту. Показано на рисунке 6.12

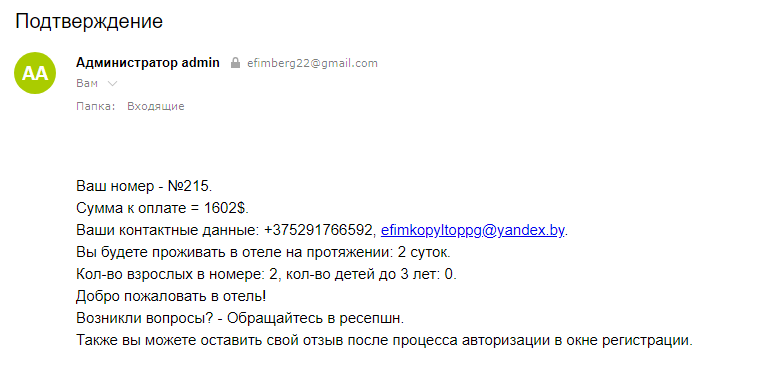


Рисунок 6.12 – Письмо подтверждения

При нажатии на иконку выхода из аккаунта пользователь может вернуться к окну входа.

ЗАКЛЮЧЕНИЕ

В итоге выполнения данного курсового проекта было разработано программное средство «Бронирование номеров в отеле». При разработке были выполнены все пункты из указанного списка предполагаемого основного функционала приложения.

В программном средстве были реализованы следующие функции:

* регистрация пользователя в системе;
* вход пользователя в систему;
* просмотр данных о номере, статусе брони;
* добавление и удаление данных о номерах в отеле;
* поиск необходимого номера;
* бронирование номера;
* уведомление пользователей о успешном бронировании номера;
* сортировка номеров по цене и по алфавиту.

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. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: <https://metanit.com> . Дата доступа: 20.04.2020

2. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> Дата доступа: 13.04.2020

3. Форум для программистов или разработчиков [Электронный ресурс] – <https://stackoverflow.com/> – Дата доступа: 5.05.2020

ПРИЛОЖЕНИЕ А

Листинг кода регистрации

try

{

string[] x = txtUsername.Text.Split();

User newUser = new User()

{

Name = x[0],

LastName = x[1],

Password = txtPassword.Password.Trim().GetHashCode().ToString(),

RoomId = null

};

using (var context = new MyDbContext())

{

foreach (User user in context.Users)

{

if (newUser.Name == user.Name && newUser.LastName == user.LastName && newUser.Password == user.Password)

{

MessageBox.Show("Такой аккаунт уже существует.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

}

UserRepository userRepository = new UserRepository(context);

userRepository.Create(newUser);

userRepository.Save();

}

MessageBox.Show("Вы успешно зарегистрировались.", "Добро пожаловать", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

MessageBox.Show("Здесь должны быть ваши имя и фамилия.", "Внимание!", MessageBoxButton.OK, MessageBoxImage.Warning);

}

ПРИЛОЖЕНИЕ Б

Листинг кода входа в систему

try

{

string[] x = txtUsername.Text.Split();

int check = 0;

using (var context = new MyDbContext())

{

foreach (User u in context.Users)

{

if (x[0] == u.Name && x[1] == u.LastName && txtPassword.Password.Trim().GetHashCode().ToString() == u.Password)

{

Registration registration = new Registration(u);

registration.Show();

//MessageBox.Show($"Добро пожаловать в отель 'Diamond Plaza', {u.Name} {u.LastName}!");

var parent = Window.GetWindow(this);

parent.Close();

txtUsername.Clear();

txtPassword.Clear();

}

else if (x[0] != u.Name || x[1] != u.LastName || txtPassword.Password != u.Password)

{

check++;

}

}

if (check == context.Users.Count())

{

MessageBox.Show("Неверный ввод. \nПопробуйте зайти еще раз или зарегистрируйтесь.", "Внимание!", MessageBoxButton.OK, MessageBoxImage.Warning);

txtUsername.Clear();

txtPassword.Clear();

}

}

}

catch (Exception ex)

{

MessageBox.Show($"{ex.Message}. Ошибка.", "Упс", MessageBoxButton.OK, MessageBoxImage.Error);

}

ПРИЛОЖЕНИЕ В

Листинг кода отправки сообщения на почту пользователя

public void SendEmailToClient(User client)

{

MailAddress fromMailAddress = new MailAddress("efimberg22@gmail.com", "Администратор " + AdminName);

MailAddress toAddress = new MailAddress(client.Email,"Уважаемый, " + client.Name + " " + client.LastName);

using (MailMessage mailMessage = new MailMessage(fromMailAddress, toAddress))

using (SmtpClient smtpClient = new SmtpClient())

{

mailMessage.Subject = "Подтверждение";

mailMessage.Body = $"Ваш номер - №{client.userRoom.Number}." +

$"\nСумма к оплате = {client.ServicePrice.ToString()}$." +

$"\nВаши контактные данные: {client.PhoneNumber.ToString()}, { client.Email.ToString()}." +

$"\nВы будете проживать в отеле на протяжении: {client.DaysInHotel.ToString()} суток." +

$"\nКол-во взрослых в номере: {client.Adults.ToString()}, кол-во детей до 3 лет: {client.ChildsUnderThree.ToString()}." +

$"\nДобро пожаловать в отель!\nВозникли вопросы? - Обращайтесь в ресепшн.\nТакже вы можете оставить свой отзыв после процесса авторизации в окне регистрации.";

smtpClient.Host = "smtp.gmail.com";

smtpClient.Port = 587;

smtpClient.EnableSsl = true;

smtpClient.DeliveryMethod = SmtpDeliveryMethod.Network;

smtpClient.UseDefaultCredentials = false;

smtpClient.Credentials = new NetworkCredential(fromMailAddress.Address, "LLW-XNG-Nny-3Gw");

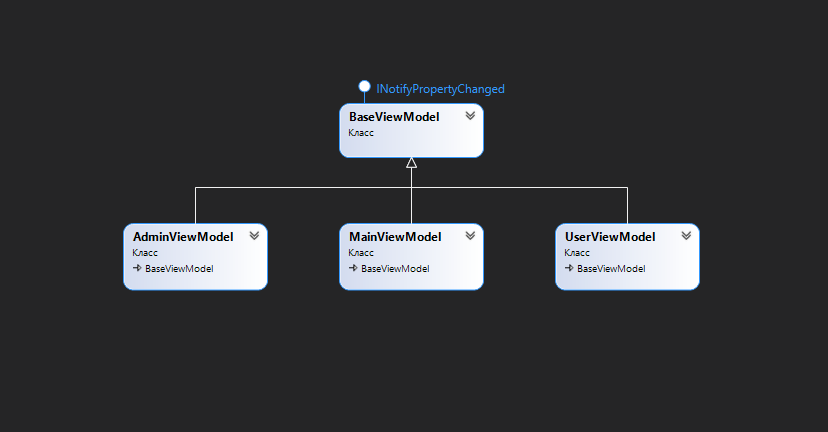
smtpClient.Send(mailMessage);

}

}

ПРИЛОЖЕНИЕ Г

Диаграмма UML классов модуля ViewModel



ПРИЛОЖЕНИЕ Д

