[ВВЕДЕНИЕ 4](#_Toc153541804)

[1. ОБЩАЯ ЧАСТЬ 5](#_Toc153541805)

[1.1. Цель разработки 5](#_Toc153541806)

[1.2. Средства разработки 5](#_Toc153541807)

[2. СПЕЦИАЛЬНАЯ ЧАСТЬ 7](#_Toc153541808)

[2.1. Постановка задачи 7](#_Toc153541809)

[2.1.1. Входные данные 7](#_Toc153541810)

[2.1.2. Выходные данные 7](#_Toc153541811)

[2.1.3. Подробные требования к проекту 8](#_Toc153541812)

[2.2. Внешняя спецификация 8](#_Toc153541813)

[2.2.1. Описание задачи 8](#_Toc153541814)

[2.2.2. Входные и выходные данные 13](#_Toc153541815)

[2.2.3. Методы 16](#_Toc153541816)

[2.2.4. Тесты 21](#_Toc153541817)

[2.2.5. Контроль целостности данных 21](#_Toc153541818)

[2.3. Проектирование 23](#_Toc153541819)

[2.3.1. Схема архитектуры приложения 23](#_Toc153541820)

[2.3.2. Логическая схема данных 24](#_Toc153541821)

[2.3.3. Физическая схема данных 25](#_Toc153541822)

[2.3.4. Структурная схема 29](#_Toc153541823)

[2.3.5. Функциональная схема 32](#_Toc153541824)

[2.3.6. Диаграмма классов 32](#_Toc153541825)

[2.3.7. Схема тестирования 35](#_Toc153541826)

[2.3.8. Схема пользовательского интерфейса 35](#_Toc153541827)

[2.4. Результат работы программы 35](#_Toc153541828)

[3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 39](#_Toc153541829)

[3.1. Инструментальные средства 39](#_Toc153541830)

[3.2. Отладка программы 39](#_Toc153541831)

[3.3. Защитное программирование 40](#_Toc153541832)

[3.4. Характеристики программы 43](#_Toc153541833)

[ЗАКЛЮЧЕНИЕ 44](#_Toc153541834)

[СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ 46](#_Toc153541835)

ПРИЛОЖЕНИЕ А. Техническое задание

ПРИЛОЖЕНИЕ Б. Текст программы

ПРИЛОЖЕНИЕ В. Сценарий и результаты тестовых испытаний

ПРИЛОЖЕНИЕ Г. Руководство пользователя

ПРИЛОЖЕНИЕ Д. Скрипт базы данных

# ВВЕДЕНИЕ

В современных реалиях все чаще можно увидеть цифровизацию привычных, бытовых действий, все это благодаря открытой быстро растущей области информационных технологий.

На рынке информационных систем для гостиниц существует много разнообразных программ, предоставляющих широкий набор функциональных возможностей. От базового создания и отслеживания бронирований до автоматизации сложных процессов управления гостиничным бизнесом. Важно подчеркнуть, что не все эти системы подходят для каждой гостиничной сферы и требуют тщательного выбора в соответствии с конкретными потребностями и особенностями управления гостиничными заявками.

Большинство пользователей сети Интернет, включая тех, кто активно временят в гостинице, проводят свое время за компьютером и телефоном. Их образ жизни обычно не предполагает трату излишне много времени на мелкие процессы, например, оформление заявок на обслуживание. В связи с этим, для многих из них будет полезным наличие приложения, которое позволяет подавать заявки на обслуживание удаленно.

В контексте данной курсовой работы требуется создать информационную систему, направленную на оптимизацию процесса приема запросов на обслуживание электронной техники в гостиничном бизнесе.

Разработанная информационная система учета запросов предоставит пользователям возможность подать на бронирование и обслуживание номера. Кроме того, данная система значительно упростит процесс учета запросов в гостиничном сервисе, так как все заявки будут организованы в структурированном формате.

# ОБЩАЯ ЧАСТЬ

## Цель разработки

Целью разработки информационной системы управления заявками в гостинице является оптимизация процесса обработки запросов от гостей и улучшение внутренней организации работы с заявками. Информационная система направлена на создание удобного и доступного интерфейса для гостей, что обеспечит легкость в подаче заявок бронирования и обслуживания номеров, а также на предоставление персоналу гостиницы эффективных средств взаимодействия с поступающими заявками, с целью оперативного и качественного обслуживания гостей.

## Средства разработки

Для проектирования, разработки и тестирования программного обеспечения, а также оформления документации по проекту использовались программные средства, представленные в таблице 1.

Таблица 1 - Программные средства разработки

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тип средства | Название средства | Назначений |
| 1 | 2 | 3 | 4 |
| 1 | Операционная система | Microsoft Windows 10 | Организация взаимодействия программ и пользователя |
| 2 | Интерфейс взаимодействия | Mircosoft SQL Server Management Studio 19 | Для взаимодействия СУБД |
| 3 | Построение схем БД | SqlDBM | Построение логической и физической модели базы данных |
| 4 | СУБД | Microsoft SQL Server | Создание и манипуляция базой данных, обеспечение безопасности, надежности хранения и целостности данных, предоставления средств администрирования БД. |
| 5 | Текстовый редактор | Microsoft Word  2019 | Разработка документации, формирование отчетных  документов по шаблону. |

В качестве средств вычислительной техники при разработке ПО использовался ноутбук Asus.  Характеристики представлены в таблице 2.

Таблица 2 - Технические характеристики устройства при разработке ПО

|  |  |  |
| --- | --- | --- |
| № | Тип средства | Название средства |
| 1 | 2 | 3 |
| Ноутбук Asus | | |
| 1 | Размер экрана: | 15 – «15,6» |
| 2 | Разрешение экрана: | 1366x768 |
| 3 | Линейка процессора: | Intel(R) Pentium(R) CPU N4200 |
| 4 | Количество ядер процессора: | 6 |
| 5 | Оперативная память: | 8 ГБ |
| 6 | Тип видеокарты: | Встроенная |
| 7 | Видеокарта: | NVIDIA GeForce 920MX |
| 8 | Конфигурация накопителей: | SSD |
| 9 | Общий объем всех накопителей: | 930 ГБ |

# СПЕЦИАЛЬНАЯ ЧАСТЬ

## Постановка задачи

Разработать информационную систему управления гостиницей «Star Maze Hotel». Приложение представляет собой WPF приложение дающая возможность сотрудникам управлять заявками бронирования, поступившими к ним, а клиентам создавать данные заявки брони.

## Входные данные

Входные данные представлены в следующем виде:

* Логин
* Пароль
* Фамилия
* Имя
* Отчество
* Номер
* Тип комнаты
* Услуга
* Стоимость
* Даты заезда/выезда
* Статус заявки
* Заявка
* Роль

## Выходные данные

Выходные данные представлены в следующем виде:

* Комната
* Дата заезда/выезда
* Описание номера
* Услуга
* Стоимость
* Логины пользователей
* Количество пользователей
* ФИО клиента
* Почта
* Номер телефона
* Роль
* Название гостиниц
* Адрес гостиницы
* Статус
* Номер заявки
* Дата создания

## Подробные требования к проекту

Подробные требования к проекту «информационная система управления гостиницей «Star Maze Hotel» представлены в Приложении A. Техническое задание.

## Внешняя спецификация

## Описание задачи

Главной задачей является разработка WPF приложения для упрощения процесса записи на бронирование и ведение данных о пользователях системы.

Программный продукт состоит из трех взаимосвязанных составляющих: база данных, API (Application programming interface) и приложение для пользователей.

При открытии приложения пользователю должно быть открыто окно авторизации. После ввода данных, пользователю открывается окно в зависимости от его роли.

Если роль пользователя «Клиент» открывается окно клиента. На данном окне должны быть расположены 4 вкладки.

На вкладке «Бронирование» должен находится список номеров, поля выбора даты заезда и даты выезда из выпадающего списка календаря, также присутствует выбор номера и услуги выбранного номера. После заполнения всех данных необходимо нажать на кнопку «Добавить», что произведет отправку заявки бронирования. В правом в верхнем углу расположена кнопка выхода из аккаунта.

На вкладке «Отель», расположена таблица с отображаемыми данными всех возможных открытых отелей. В правом в верхнем углу расположена кнопка выхода из аккаунта.

На вкладке «Услуги», расположена таблица с отображаемыми данными всех услуг данной гостиницы. Над таблицей расположено поле поиска по таблице, при вводе которого появится искомая услуга. В правом в верхнем углу расположена кнопка выхода из аккаунта.

На вкладке «Заселение», расположена таблица с отображаемыми данными занятых всех дат других пользователей. В правом в верхнем углу расположена кнопка выхода из аккаунта.

Если пользователь авторизовался под ролью «Администратор» ему открывается окно администратора. На данном окне должны быть расположены 2 вкладки.

На вкладке «Отель», расположена таблица с отображаемыми данными всех гостиниц. Внизу таблицы присутствуют кнопки «Добавление», «Изменение» и «Удаление», при нажатии на который происходит добавление новой гостиницы, изменение новой гостиницы или удаление гостиницы их списка. В правом в верхнем углу расположена кнопка выхода из аккаунта.

На вкладке «Услуги», расположена таблицы с отображаемыми данными комнаты и услуги, статуса. Внизу таблицы присутствуют кнопки «Добавление», «Изменение» и «Удаление», при нажатии на который происходит добавление новой услуги, статуса или комнаты, изменение новой услуги, статуса или комнаты, или удаление услуги, статуса или комнаты. В правом в верхнем углу расположена кнопка выхода из аккаунта.

Если пользователь авторизовался под ролью «Администратор БД» ему открывается окно администратора БД. На данном окне должны быть расположены 6 вкладки.

На вкладке «Услуги», «Бронирование», «Заселение», «Отель», «Услуги» расположена таблицы с услугой, статус, номер, тип номера, стоимостью, датами заселения/выезда, гостиницей. Внизу под каждой таблицей расположены кнопки «Добавление», «Изменение» и «Удаление», производящие новые данными, изменять старые данные и удалять не нужные данные. В правом в верхнем углу расположена кнопка выхода из аккаунта.

На вкладке «Пользователи» расположена таблицами отображаемая всех пользователей системы. Внизу под таблицей расположены кнопки «Добавление», «Изменение» и «Удаление», производящие новых сотрудников и присваивание им роли. В правом в верхнем углу расположена кнопка выхода из аккаунта.

На вкладке «График» расположена кнопки «Получить график», при нажатии на которую производится подсчет данных количества пользователей и выводит график. В правом в верхнем углу расположена кнопка выхода из аккаунта.

Если пользователь авторизовался под ролью «Управляющий» ему открывается окно управляющего. На котором демонстрируются отправленные заявки пользователей и присутствуют кнопки «Отказ», «Изменить» и «Подтвердить» необходимые после просмотра заявки бронирования клиента. В правом в верхнем углу расположена кнопка выхода из аккаунта.

В ходе разработки был проведен анализ предметной области, чтобы понять, как организация работает. В результате были выявлены процессы, которые можно автоматизировать. Они представлены на рисунках 1-5.

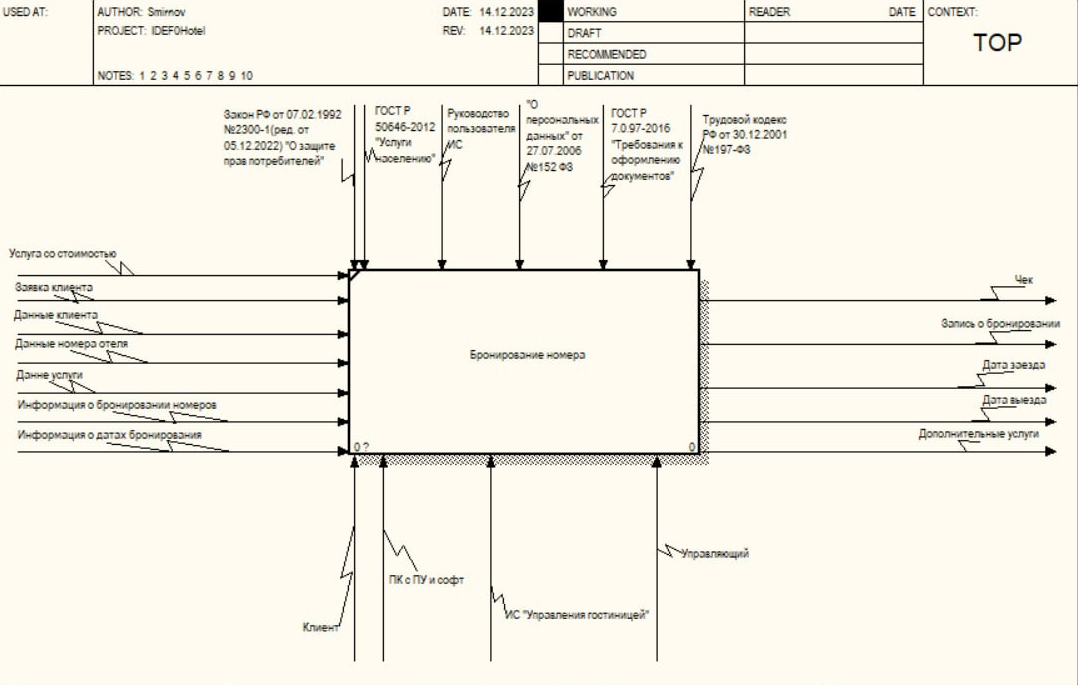


Рисунок 1 - Диаграмма бизнес процессов уровень A0 для бронирования

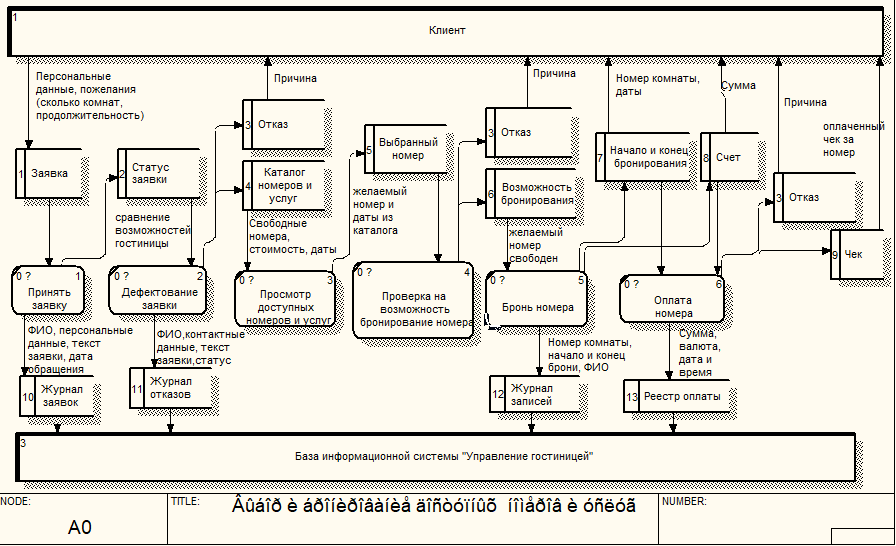


Рисунок 2 – Диаграмма DVD клиент

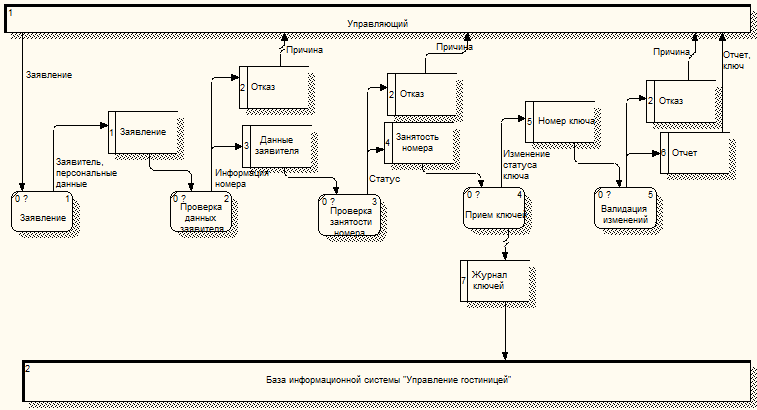


Рисунок 3 - Диаграмма DVD управляющего

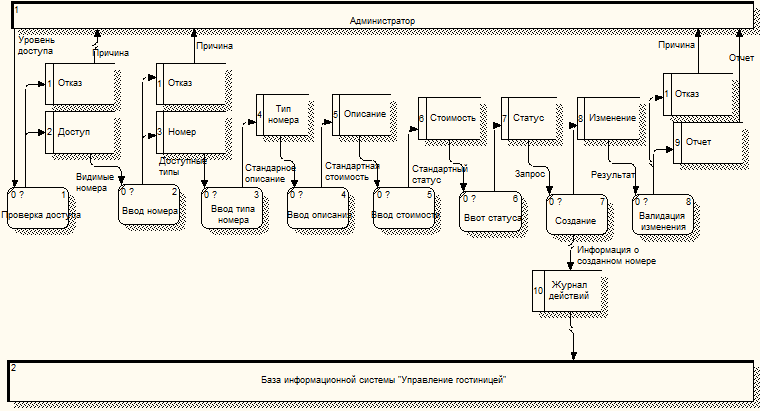


Рисунок 4 - Диаграмма DVD администратора

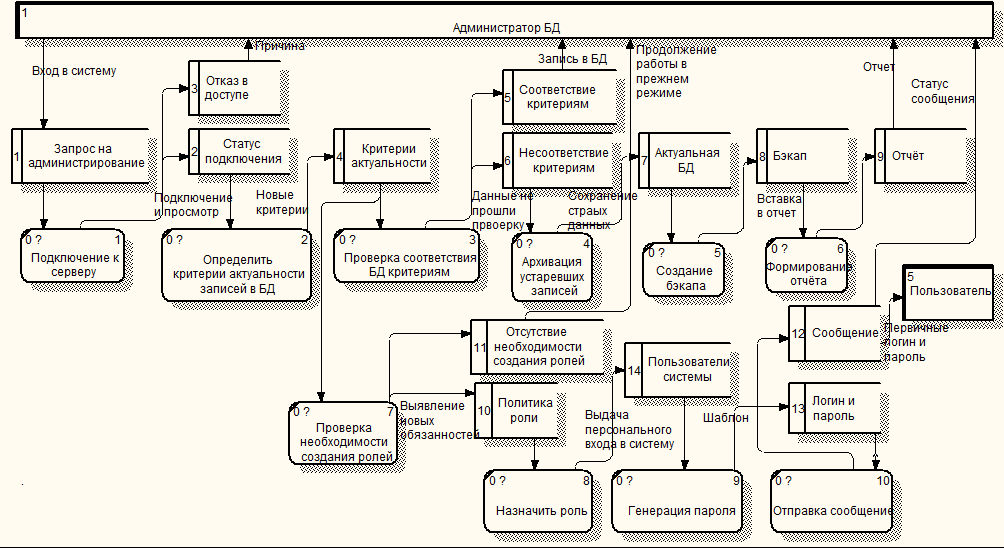


Рисунок 5 – Диаграмма DVD администратора БД

## Входные и выходные данные

В таблицах под номерами 3-4 представлены входные и выходные данные приложения.

Таблица 3 - Входные данные приложения

| Поле | Ограничение | Тип | Форма ввода | Описание |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Логин | {3,50}, только буквы латинского алфавита | Строка | Поле ввода | Поле ввода для логина пользователя при авторизации |
| Пароль | {6,25}, Только буквы латинского алфавита | Строка | Поле ввода | Поле ввода для пароля пользователя при авторизации |
| Имя | {5,255}, только буквы латинского алфавита | Строка | Поле ввода | Поле ввода для имени пользователя при регистрации |
| Фамилия | {5,255}, только буквы латинского алфавита | Строка | Поле ввода | Поле ввода для фамилии пользователя при регистрации |
| Отчество | {5,255}, только буквы латинского алфавита | Строка | Поле ввода | Поле ввода для отчества пользователя при регистрации |
| Номер | Число, не отрицательное | Строка | Поле ввода | Поле ввода для присвоения номера при создании |
| Тип комнаты | Ограничения не предоставляются | Строка | Выпадающий список | Выпадающий список для выбора типа комнаты при создании |
| Услуга | Ограничения не предоставляются | Строка | Выпадающий список | Выпадающий список для выбора услуги при создании |
| Стоимость | Ограничения не предоставляются | Строка | Выпадающий список | Выпадающий список для выбора стоимости при создании |
| Даты заезда/выезда | Ограничения не предоставляются | Строка | Выпадающий список | Выпадающий список для выбора даты заезда/выезда при создании |
| Статус | Ограничения не предоставляются | Строка | Флажок | ComboBox для выбора статуса при изменения |
| Заявка | Ограничения не предоставляются | Строка | Поле ввода | Поле ввода для добавления/изменеия заяки |
| Роль | Ограничения не предоставляются | Строка | Выпадающий список | Выпадающий список для выбора роли при изменении роли пользователя |

Таблица 4 - Выходные данные приложения

| Поле | Ограничение | Тип | Описание |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Список заявок пользователя | Не предусмотрено | Лист | Элемент представляет собой список заметок с полями: комнатой, датой заезда/выезда, описанием номера, услугой, стоимостью, статусом брони, логином клиента |
| Список чек-аута брони | Не предусмотрено | Лист | Элемент представляет собой список заявок с полями: ФИО клиента, Стоимость, Даты заезда/выезда |
| График пользователей в системе | Не предусмотрено | Фотография | Фотография, на которой изображен график с Логины пользователей, Количество пользователей и цены |
| Список заселения | Не предусмотрено | Строка | Элемент представляет собой список всех заселений с полями: Статус, Дата заезда/выезда |
| Список гостиниц | Не предусмотрено | Строка | Элемент представляет собой список всех гостиниц с полями: Название гостиниц, Адрес гостиницы |
| Список заявок пользователя | Не предусмотрено | Строка | Элемент представляет собой список заявок пользователей с полями: Номер заявки, дата создания, статус заявки) |

## Методы

При разработке приложения использовалось среда разработки Visual Studio. Visual studio использовалось для написания API и компьютерного приложения на WPF использую язык C#.

Данные приложения были написаны с использованием основных парадигм объектно-ориентированного программирования, таких как, инкапсуляция (будет использоваться для ограничения доступа одних компонентов программы к другим); наследование (необходимо для повторного использования кода и способствует независимому расширению программного обеспечения и интерфейсов); полиморфизм (для обработки различных типов данных).

Ниже на рисунке 6 приведён пример инкапсуляции.

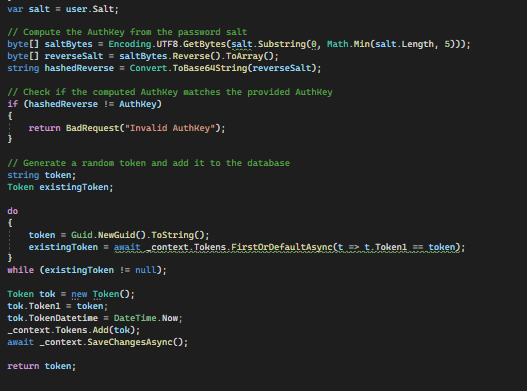


Рисунок 6 - Пример инкапсуляции

Следующий рисунок 7 представляет наследование.

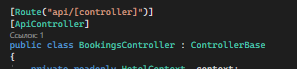


Рисунок 7 - Пример наследования

В создании API использовался паттерн MVC (Model-view-controller). Из представления (View) пользователь осуществляет вызов методов контроллера. Из контроллера осуществляется обновление модели в соответствии с действиями пользователями. Затем данная модель передаётся в представление, в свою очередь представление может осуществлять запрос данных из модели.

На рисунке 8 представлена модель.

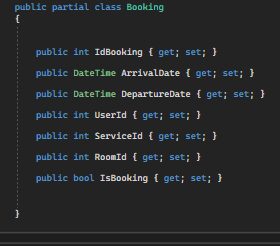


Рисунок 8 - Пример модели из парадигмы MVC

На рисунке 9 представлен контроллер.

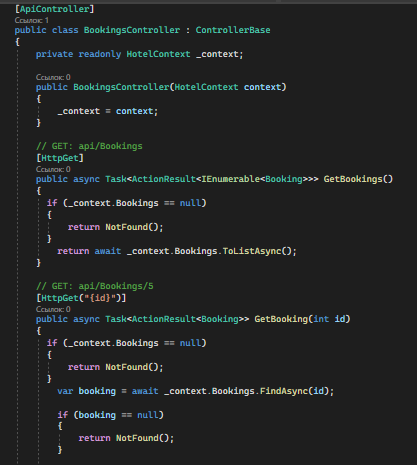


Рисунок 9 - Пример контроллера из парадигмы MVC

Рефакторинг, фокусируется на улучшении структуры и читаемости кода без изменения его внешнего поведения. Целью рефакторинга является упрощение кода, устранение дублирования, повышение модульности и поддерживаемости. Рефакторинг часто проводится для улучшения понимания кода другими разработчиками, облегчения добавления новых функций и исправления ошибок.

На рисунках 10 – 11 представлен код до и после рефакторинга.

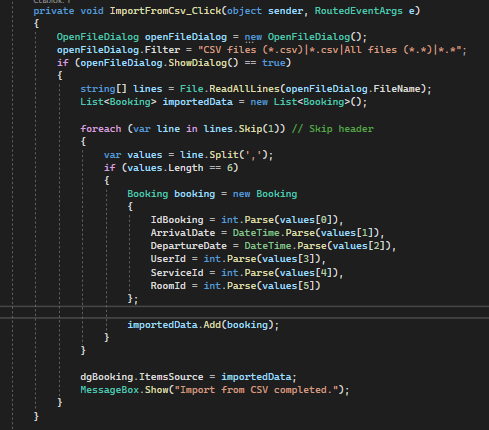


Рисунок 10 – Код до рефакторинга

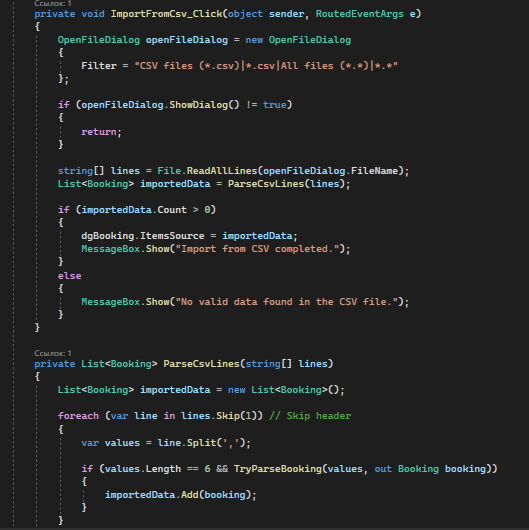


Рисунок 11 – Код после рефакторинга

Оптимизация означает изменение кода или алгоритма с целью улучшения производительности, эффективности или использования ресурсов. Она может включать в себя оптимизацию времени выполнения, использование меньшего объема памяти, улучшение скорости работы алгоритмов и т.д. Оптимизация может требовать компромиссов, таких как увеличение сложности кода или увеличение времени разработки для достижения желаемой производительности или эффективности.

На рисунке 12 – 13 представлен код до, и после оптимизация.



Рисунок 12 – Код до оптимизации

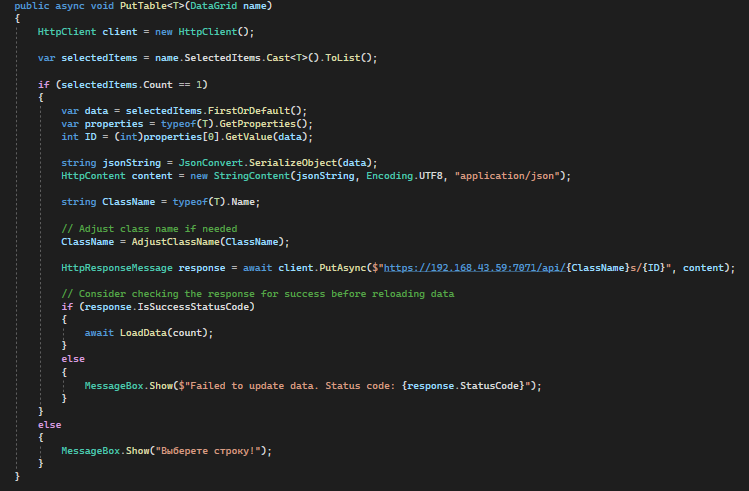


Рисунок 13 – Код после оптимизации

## Тесты

Для тестирования программы будет использоваться ручное тестирование по тестовым сценариям, также будет применено нагрузочное тестирование на разработанное API.

Сценарий и результаты тестовых испытаний представлены в Приложении В.

## Контроль целостности данных

В ходе разработки информационной системы были применены методы валидации входных данных, которые обеспечивают целостность этих данных. Этот подход включает контроль целостности данных, который определяет ситуации и действия приложения при выполнении задач, связанных с сохранением, изменением или удалением данных.

Таблица 5 - Контроль целостности данных

| № | Ситуация | Аномалия | Реакция программы |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 1 | Регистрация пользователя | Введена некорректная фамилия | Программа после попытки регистрации сообщает о провале регистрации. |
| Введено некорректное имя | Программа после попытки регистрации сообщает о провале регистрации. |
| Введено некорректное отчество | Программа после попытки регистрации сообщает о провале регистрации. |
| Введен некорректный логин | Программа после попытки регистрации сообщает о провале регистрации. |
| Введен некорректный пароль | Программа после попытки регистрации сообщает о провале регистрации. |
| 2 | Добавление и изменение номеров | Введен некорректный формат поля «Номер» | Программа после попытки добавления или изменения данных сообщает о провале добавления или обновления номера. |
| Выход за пределы ограничения в названии типа комнаты. | Программа после попытки добавления или изменения данных сообщает о провале добавления или обновления номера. |
| Введен некорректный формат поля «Стоимость» | Программа после попытки добавления или изменения данных сообщает о провале добавления или обновления номера. |
| Выход за пределы ограничения в названии услуги. | Программа после попытки добавления или изменения данных сообщает о провале добавления или обновления номера. |
| 3 | Добавление и изменение сотрудников | Выход за пределы ограничения в названии имени | Программа после попытки добавления или изменения данных сообщает о провале добавления или обновления сотрудника. |
| Поле осталось пустое | Программа после попытки добавления или изменения данных сообщает о провале добавления или обновления сотрудника. |
| Введен некорректный формат поля «Отчество» | Программа после попытки добавления или изменения данных сообщает о провале добавления или обновления сотрудника. |
| Выход за пределы ограничения в логине. | Программа после попытки добавления или изменения данных сообщает о провале добавления или обновления сотрудника. |
| Выход за пределы ограничения в пароле. | Программа после попытки добавления или изменения данных сообщает о провале добавления или обновления сотрудника. |
| Роль не выбрана | Программа после попытки добавления или изменения данных сообщает о провале добавления или обновления сотрудника. |

## Проектирование

## Схема архитектуры приложения

Приложение взаимодействует с сервером, на котором размещена база данных, посредством API. Запросы и ответы выполняются в соответствии с заранее согласованным протоколом и форматом данных между приложением и сервером.

Приложение имеет возможность отправлять запросы на чтение, изменение или удаление данных в базе данных через API. Сервер обрабатывает эти запросы и возвращает ответы с соответствующей информацией в формате, который понятен приложению. Такой механизм позволяет приложению эффективно получать и обрабатывать данные из базы данных, избегая прямого обращения к ней. Ниже на рисунке 14 представлена схема связи базы данных и приложения через API.

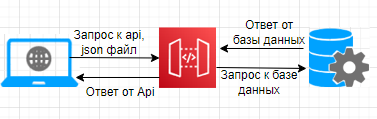


Рисунок 14 - Схема архитектуры приложения

## Логическая схема данных

В базе данных находится 9 таблиц это: услуга, статус, чек-аут, заселение, гостиница, роль, бронирование, гость, сотрудник. Таблицы между собой имеют связь один ко многим.

Ниже на рисунке 15 представлена логическая схема данных разработанная под приложение.



Рисунок 15 - Логическая схема базы данных

## Физическая схема данных

В физической схеме данных показаны какие типы данных имеют поля в таблицах. Также на ней стрелками показано какие таблицы имеют связь между собой.

Ниже на рисунке 16 представлена физическая схема базы данных

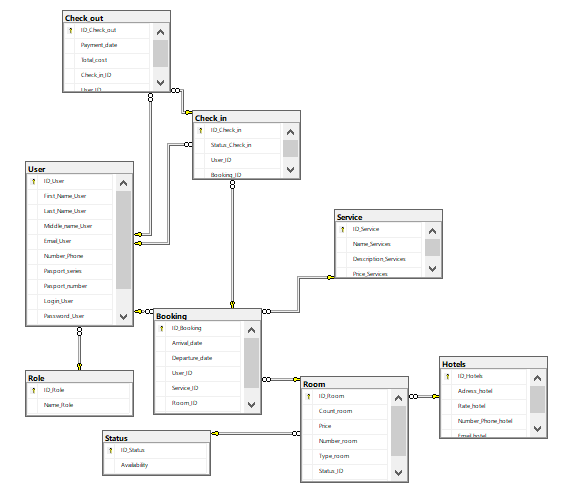


Рисунок 16 - Физическая схема базы данных

Словарь данных представлен в таблице ниже:

Таблица 6 - Словарь данных

| Ключ | Наименование | Тип данных | Обязательность заполнения | Описание |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Role – таблица с ролями | | | | |
| PK | ID\_Role | Int | Not null  identity(1,1) | Уникальный идентификатор роли |
|  | Name\_Role | Varchar(50) | Not null | Название роли |
| User – таблица с пользователями | | | | |
| PK | ID\_User | Int | Not null identity(1,1) | Уникальный идентификатор пользователя |
|  | First\_Name\_User | Varchar(30) | Not null | Фамилия пользователя |
|  | Last\_Name\_User | Varchar(30) | Not null | Имя пользователя |
|  | Middle\_name\_User | Varchar(30) | Not null | Отчество пользователя |
|  | Email\_User | Varchar(60) | Null | Почта пользователя |
|  | Number\_Phone | Varchar(30) | Null | Номер телефона пользователя |
|  | Pasport\_series | Varchar(4) | Null | Серия паспорта пользователя |
|  | Pasport\_number | Varchar(6) | Null | Номер паспорта пользователя |
|  | Login\_User | Varchar(50) | Not null | Логин пользователя |
|  | Password\_User | Varchar(100) | Not null | Пароль пользователя |
|  | Salt | Varchar(256) | Not null | Соль |
| FK | Role\_ID | int | Not null | Внешний ключ таблицы роль |
| Status – таблица со статусами | | | | |
| PK | ID\_Status | Int | Not null identity(1,1) | Уникальный идентификатор статуса |
|  | Availability | Varchar(30) | Not null | Доступность |
| Hotels – таблица с гостиницей | | | | |
| PK | ID\_Hotels | Int | Not null identity(1,1) | Уникальный идентификатор гостиницы |
|  | Adress\_hotel | Varchar(max) | Not null | Адрес гостиницы |
|  | Rate\_hotel | Int | Not null | Рейтинг гостиницы |
|  | Number\_Phone\_hotel | Varchar(20) | Not null | Телефон гостиницы |
|  | Email\_hotel | Varchar(60) | Not null | Почта гостиницы |
| Room – таблица номеров | | | | |
| PK | ID\_Room | Int | Not null identity(1,1) | Уникальный идентификатор номера |
|  | Count\_room | Int | Not null | Количество комнат |
|  | Price | Int | Not null | Цена комнаты |
|  | Number\_room | Int | Not null | Номер комнаты |
|  | Type\_room | Varchar(60) | Not null | Тип комнаты |
| FK | Status\_ID | Int | Not null | Внешний ключ таблицы статус |
| FK | Hotels\_ID | Int | Not null | Внешний ключ таблицы гостиница |
| Service – таблица с услугами | | | | |
| PK | ID\_Service | Int | Not null identity(1,1) | Уникальный идентификатор услуги |
|  | Name\_Services | Varchar(50) | Not null | Название услуги |
|  | Description\_Services | Varchar(max) | Not null | Описание услуги |
|  | Price\_Services | Int | Not null | Стоимость услуги |
| Check\_in – таблица с заселением | | | | |
| PK | ID\_Check\_in | Int | Not null identity(1,1) | Уникальный идентификатор заселения |
| FK | User\_ID | Int | Not null | Внешний ключ таблицы пользователя |
|  | Status\_Check\_in | Varchar(20) | Not null | Статус заселения |
| FK | Booking\_ID | Int | Not null | Внешний ключ таблицы бронирования |
| Check\_out – таблица с чек-аутом | | | | |
| PK | ID\_Check\_out | Int | Not null identity(1,1) | Уникальный идентификатор таблицы чек-аута |
|  | Payment\_date | Date | Not null | Дата оплаты |
|  | Total\_cost | Int | Not null | Общая стоимость |
| FK | Check\_in\_ID | Int | Not null | Внешний ключ таблицы |
| FK | User\_ID | Int | Not null | Внешний ключ таблицы |
| Tokens – таблицами с токеном | | | | |
| PK | token\_id | Int | Not null identity(1,1) | Уникальный идентификатор таблицы токен |
|  | token | Varchar(200) | Not null | Токен |
|  | token\_datetime | DateTime2 | Not null | Дата и время токена |
| Booking – таблица с бронированием | | | | |
| PK | ID\_Booking | Int | Not null identity(1,1) | Уникальный идентификатор таблицы бронирования |
|  | Arrival\_date | Date | Not null | Дата прибытия |
|  | Departure\_date | Date | Not null | Дата выезда |
| FK | User\_ID | Int | Not null | Внешний ключ таблицы пользователь |
| FK | Service\_ID | Int | Not null | Внешний ключ таблицы услуги |
| FK | Room\_ID | Int | Not null | Внешний ключ таблицы комнаты |

## Структурная схема

В данной структурной схеме, также известной как диаграмма классов, представлена графическая модель, показывающая связи, зависимости и отношения между классами в программном проекте. Классы, как объекты программирования, могут иметь свойства, методы и поля.

Диаграмма классов приложения:

На рисунке 17 представлена схема приложения роли «Клиент».

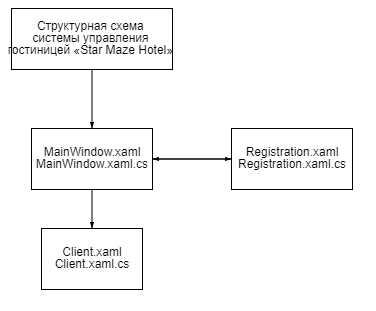


Рисунок 17 - Структурная схема приложения клиента

На рисунке 18 представлена схема приложения роли сотрудников.

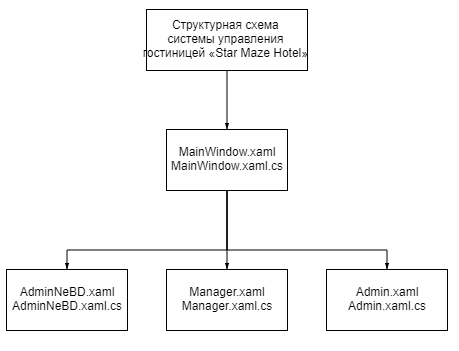


Рисунок 18 - Структурная схема приложения сотрудников

Ниже представлены таблицы с названием модулей и описание, в котором кратко описано за что отвечает каждый модуль.

Таблица 7 Модули приложения клиента.

| Номер | Название модуля | Описание модуля |
| --- | --- | --- |
| 1 | 2 | 3 |
| 1 | MainWindow.xaml  MainWindow.xaml.cs | Данный модуль содержит логику и разметку окна авторизации, отвечает за вход пользователя в систему. |
| 2 | Registration.xaml  Registration.xaml.cs | Данный модуль содержит логику и разметку окна регистрации, отвечает за добавление пользователя в систему. |
| 3 | Client.xaml  Client.xaml.cs | Данный модуль содержит логику и разметку  окна заявок авторизованного клиента. |

Таблица 8 Модули приложения сотрудников.

| Номер | Название модуля | Описание модуля |
| --- | --- | --- |
| 1 | 2 | 3 |
| 1 | MainWindow.xaml  MainWindow.xaml.cs | Данный модуль содержит логику и разметку окна авторизации, отвечает за вход пользователя в систему. |
| 2 | AdminNeBD.xaml  AdminNeBD.xaml.cs | Данный модуль содержит логику и разметку окна администратора, отвечает за функционал пользователя с ролью «Администратор». |
| 3 | Manager.xaml  Manager.xaml.cs | Данный модуль содержит логику и разметку окна управляющего, отвечает за функционал пользователя с ролью «Управляющий». |
| 4 | Admin.xaml  Admin.xaml.cs | Данный модуль содержит логику и разметку окна администратора БД, отвечает за функционал пользователя с ролью «Администратор БД». |

## Функциональная схема

На функциональной схеме представлены функции, находящие на окнах приложений. Данная схема позволяет легче прослеживать связи между компонентами приложения и оценивать работу системы в целом.

На рисунке 19-20 представлена функциональная схема приложение

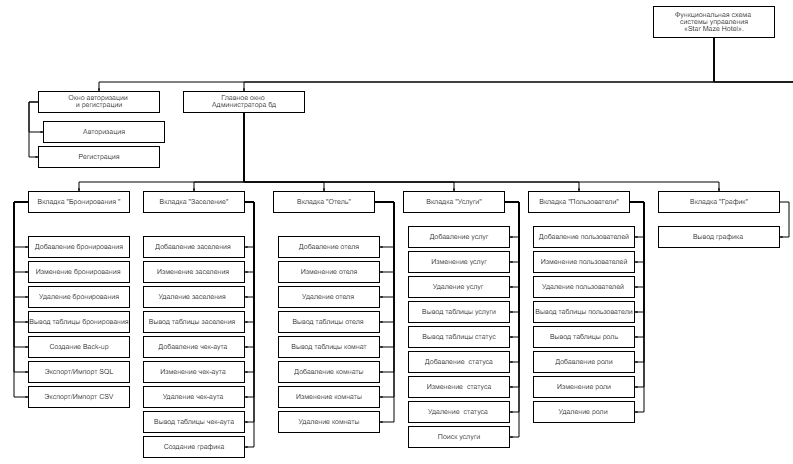


Рисунок 19 - Функциональная схема приложения для персонала первая часть

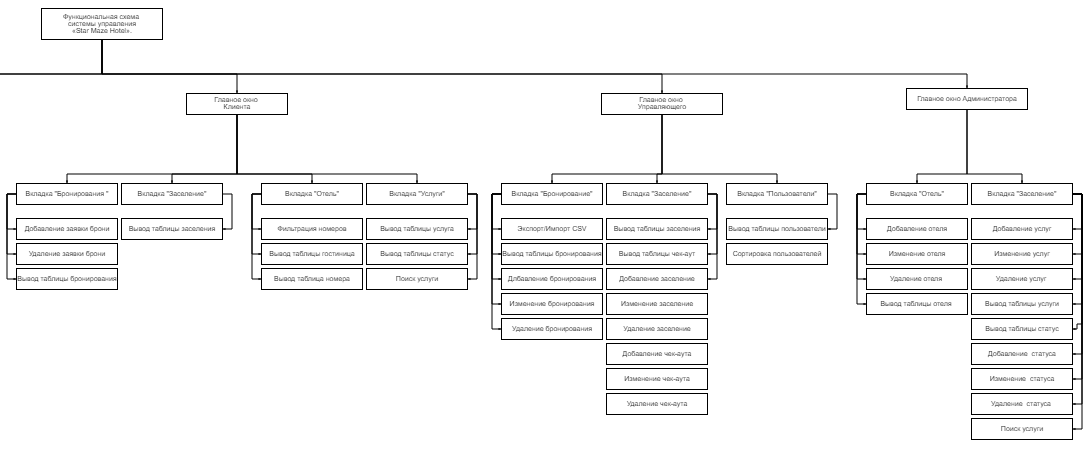


Рисунок 20 - Функциональная схема приложения вторая часть

## Диаграмма классов

В таблице представлено описание классов проекта.

Таблица 9 - Классы и методы приложения персонала.

| № | Наименование класса | Тип класса | Описание | |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | |
|  | MainWindow.cs | public | Данный класс отвечает за авторизацию пользователя в системе. |
|  | Registration.cs | public | Данный класс отвечает за регистрацию |
|  | CheckOut.cs | Public interface | Данный класс отвечает за функционал пользователя с роль «Технический специалист» |
|  | Room.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Settings.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Booking.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Client.cs | Public | Данный класс отвечает за функционал пользователя с роль «Клиент» |
|  | Manager.cs | Public | Данный класс отвечает за функционал пользователя с роль «Управляющий» |
|  | User.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | CheckIn.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Token.cs | Public | Данный класс, предназначенный для представления и управления токенами в системе. |
|  | Resources.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Status.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Role.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Hotel.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Service.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | DataBaseClass.cs | Public | Данный класс отвечает за подключение базы данных |
|  | AdminNeBD.cs | Public | Данный класс отвечает за функционал пользователя с роль «Администратор БД» |
|  | Admin.cs | Public | Данный класс отвечает за функционал пользователя с роль «Администратор» |

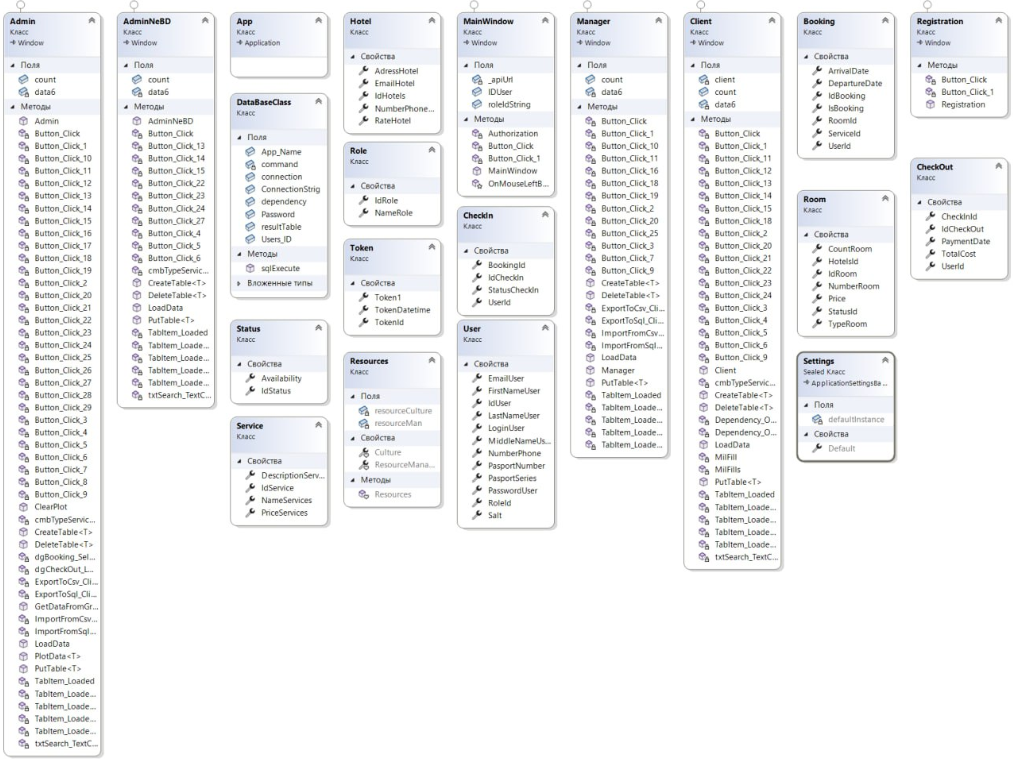


Рисунок 21 - Диаграмма классов приложения

## Схема тестирования

Схема тестирования показана в приложении В Сценарий и результаты тестовых испытаний.

## Схема пользовательского интерфейса

Ниже представлена схема пользовательского интерфейса, которая показывает, как осуществляется переход между окнами приложений.

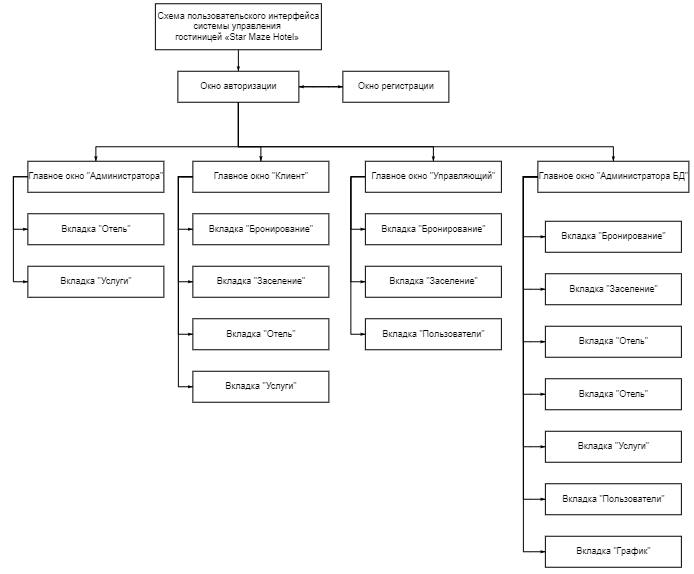


Рисунок 22 - Схема пользовательского интерфейса

## Результат работы программы

В результате выполнения поставленной задачи, было разработано решение, которое является информационной системой управления гостиницей «Star Maze Hotel».

На рисунках 23-27 представлен результат работы программы. Больше информации о результат работы программы описан в приложении Г «Руководство пользователя».

Ниже представлены фотографии приложения для пользователя:

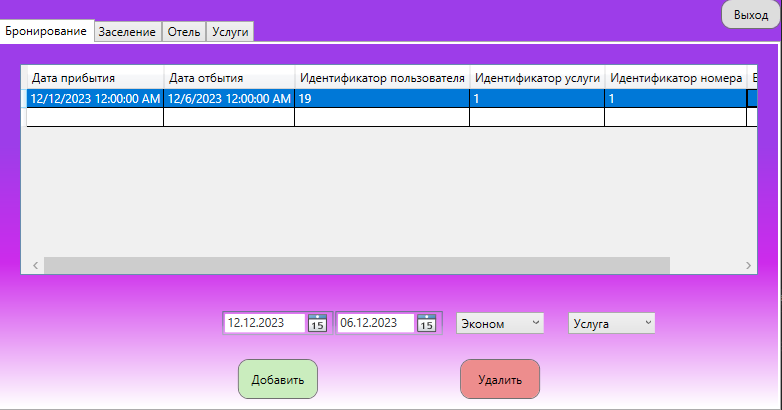


Рисунок 23 – Заявки бронирования пользователя

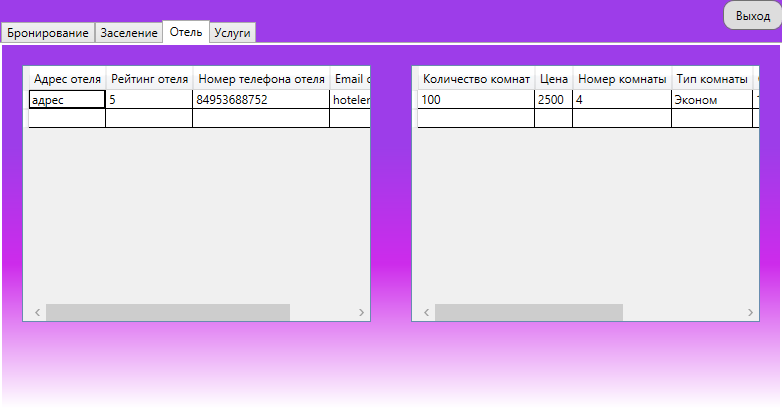


Рисунок 24 – Отображения данных вкладки «Отель»

Ниже представлены фотографии приложения для сотрудников:

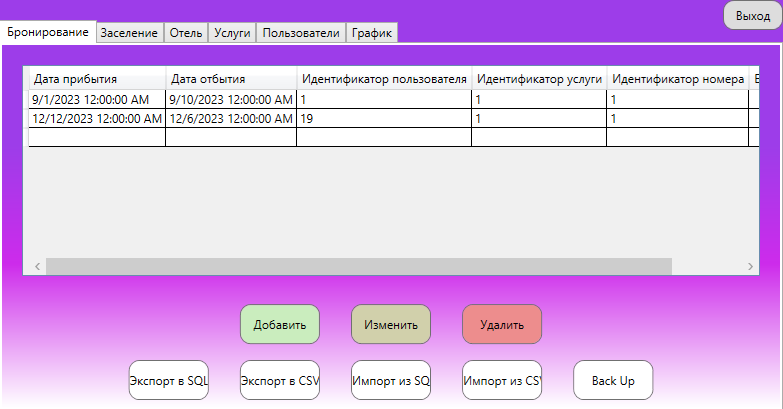


Рисунок 25 – Вкладка бронирования вместе с новой заявкой

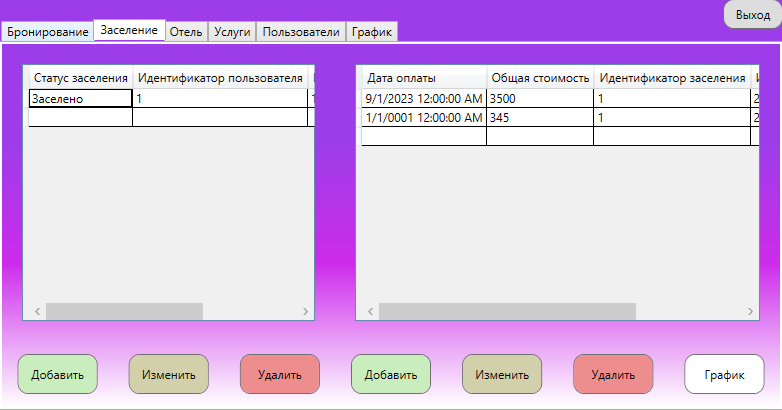


Рисунок 26 – Вкладка заселения

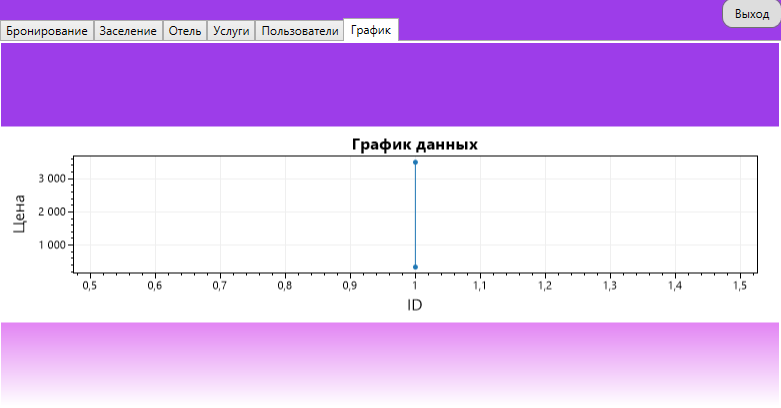


Рисунок 27 - График

# ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## Инструментальные средства

В качестве ядра для базы данных было выбрано ядро MSSQL. Для создания функций, таблиц и представлений была использована среда SQL Server Management Studio. MSSQL является одной из популярных систем управления базами данных. Так же для разработки рассматривались такие СУБД как Postgres и MySQL. Postgres это бесплатная и открытая СУБД, которая поддерживает широкий спектр функций и возможностей, аналогичных MSSQL. Но главным минусом является сложная настройка и администрирование. MySQL это еще одна бесплатная и открытая СУБД, которая является наиболее популярной СУБД в мире. Она поддерживает широкий спектр функций и возможностей, но не так много, как MSSQL или PostgreSQL. Большим минусом данной СУБД является то что она не поддерживает некоторые функции, доступные в MSSQL.

За инструментальные средства разработки была взята программа Microsoft Visual Studio 2019 для написания API и WPF приложения. Она была выбрана за счет удобства в использование данной среды для написания API. Для этого используется фреймворк ASP.NET, данный фрейворк позволяет строить API с использованием MVC. Архитектура MVC дает большое преимущество над другими средами разработки, так как может сама генерировать стандартные запросы к API.

## Отладка программы

Отладка приложения и API была осуществлена с помощью встроенных средств отладки в IDE Visual Studio. Отладка использовалась при включенном приложении и приостанавливалась, доходя до точки остановки.

В ходе работы над проектом было распространено несколько ошибок. Первая ошибка была проблемой подключения приложения к API, вызвана неправильным IP-адресом или его отключением из-за различных точек доступа интернета. Также к этому типу проблем можно отнести проблему с запуском API в связи с неправильным IP в файле launchSettings.json

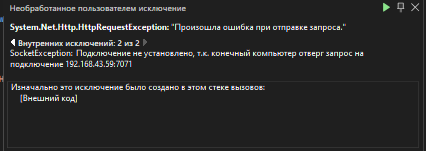


Рисунок 28 - Ошибка при отправке запроса

Второй распространенной проблемой было не правильный запрос, к методу, который вызывал ошибку. Ошибка исправляется проверкой запроса или изменением модели который помещается в запрос.

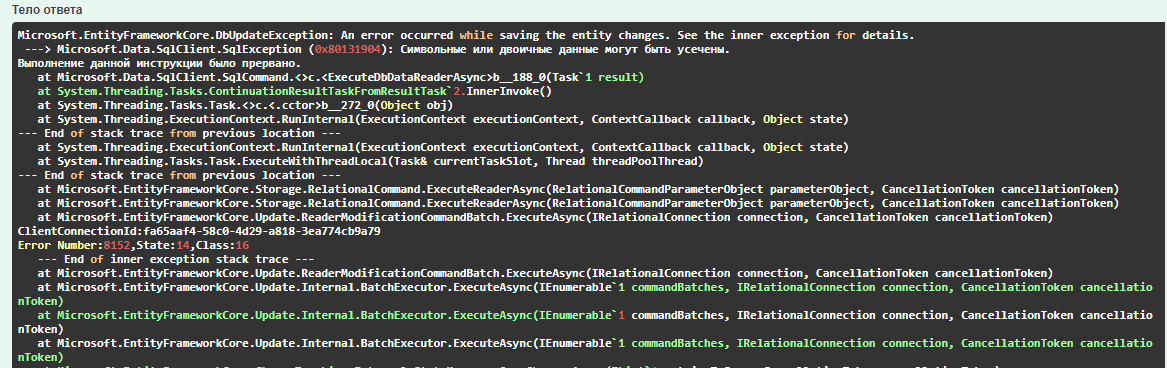


Рисунок 29 - Ошибка не правильного обращение к методу

## Защитное программирование

Механизм try\_catch в программе и API помогает обнаруживать ошибки во время выполнения кода и предпринимать соответствующие действия для их обработки, что повышает надежность и безопасность системы.

В случае возникновения ошибок, механизм try\_catch позволяет изолировать проблемный участок кода и обработать ошибку локально, не прерывая выполнение всего скрипта или приложения, что улучшает производительность и снижает риски сбоев.

Ниже представлен пример его использования:

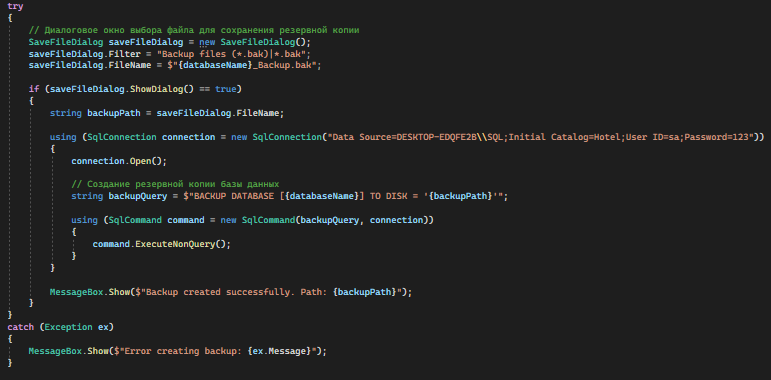


Рисунок 30 - Блок try catch

В ходе разработки программы было принято решение что для некоторых полей нужно сделать валидацию данных. Валидация была выполнена на стороне API для функции регистрации, для полей почты, пароля и имени пользователя. Ниже представлен скриншот с валидацией.

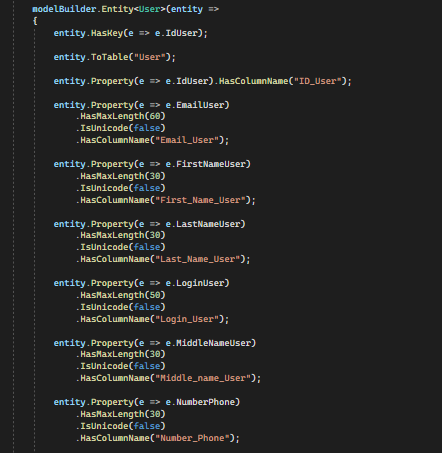


Рисунок 31 - Валидация данных

Так же для безопасности использовалось хеширования пароля пользователя с помощью соли.

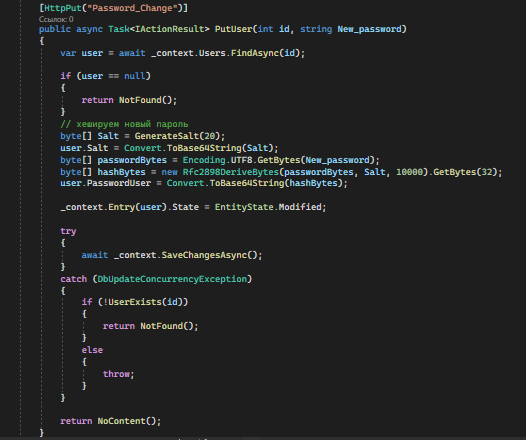


Рисунок 32 - Методы хеширования пароля

Так же для безопасности был использован токен, который генерируется при авторизации пользователя. В токене шифруется номер учетной записи пользователя и его роль. После чего у каждого запроса в API есть проверка на то какие роли имеют к ней доступ.

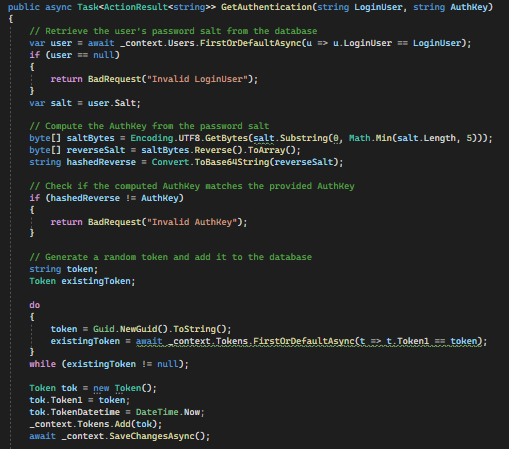


Рисунок 33 - Методы для работы с токеном

## Характеристики программы

Приложение гостиницы может быть запущено на операционных системах Windows 10 и выше.

Характеристики программы представлены в приложении Б «Текст программы» в таблице 1 «Модули».

# ЗАКЛЮЧЕНИЕ

За время работы на курсовом проекте была разработана информационная система управления гостиницей «Star Maze Hotel».

В ходе проекта были проведены работы по анализу данных, проектированию приложения включавшее в себя построение диаграмм и таблиц необходимых при разработке ПО. Диаграммы, которые были построены в течении работы можно отнести, логическую и физическую модель данных, диаграмму классов, диаграмму пользовательского интерфейса. А также были выделены входные и выходные данные программы.

В начальном этапе разработке был проведен анализ по программным средствам, которые будут нужны для реализации функционала приложения. К данным средствам относиться СУБД MSSQL используемая для написания базы данных. Для разработки API и WPF приложения использовалась среда Microsoft Visual Studio 2022, с использованием фреймворка ASP.NET.

В процессе разработки приложения был реализован полным функционал в соответствие с поставленными требованиями. В результате разработки приложение имеет такие возможности как: авторизация и регистрация пользователя, создание заявки бронирования, создание, редактирование и удаление записей «Сотрудник» и «Номер», подтверждение брони, отмена и изменения заявки бронирования. Реализованы фильтрация, поиск и сортировка заявок и номеров. Также на основе заселение формируется график стоимости и пользователей.

В ходе тестирования приложения применялись различные методы, такие как ручное тестирование и нагрузочное тестирование. Проведенные тесты показали и дали возможность устранения недостатков производительности, сообщили о функциональных ошибках.

В ходе выполнения курсового проекта были закреплены практические знания и навыки, полученные ранее. В процессе работы с SQL MSSQL были закреплены навыки создания таблиц, добавления и удаления записей, а также настройки отношений между таблицами, в разработки API с использованием языка программирования C# и использования объектно-ориентированного фреймворка ASP.NET Core, включая использование модели представления-контроллер.

Разработанное приложение было создано с целью удовлетворения всех поставленных требований и готово к использованию на практике. Оно выполняет все необходимые функции, что гарантирует максимальное удобство для пользователя. Каждая деталь приложения была разработана с учетом оптимальной производительности и надежности, что позволит достичь максимальной эффективности в работе.

В ходе разработки проекта было выявлено что данный проект имеет большой вектор развития и доработок.

# СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

1. ГОСТ 19404- 79 ЕСПД. Пояснительная записка. ПЕРЕИЗДАНИЕ Января 2010 г.
2. ГОСТ 7.80-2000 СИБИД. Библиографическая запись. Заголовок. Общие требования и правила составления.
3. ГОСТ Р 7.0.5-2008 БИБЛИОГРАФИЧЕСКАЯ ССЫЛКА. Общие требования и правила составления.
4. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.
5. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.
6. ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.
7. Стиллмен Э., Грин Дж. Изучаем C#. 3-е издание. Питер СПб 2014, 816с.
8. Роберт Мартин, Мика Мартин. Принципы, паттерны и методики гибкой разработки на языке C#. 715с.
9. Адитья Бхаргава. Грокаем алгоритмы. Издательский дом «Питер», 290с.
10. Сайт Developer Android [Электронный ресурс] URL: https://developer.android.com/studio/intro (дата обращения 15.04.2023).
11. Официальный сайт Microsoft [Электронный ресурс] <URL:https://docs.microsoft.com/ru-RU/sql/ssms/download-sql-server> management-studio-ssms?view=aps-pdw-2016 (Дата обращения 18.03.2023)
12. Сайт Metonit Java [Электронный ресурс] URL: https://metanit.com/java/ 2016 (Дата обращения 16.04.2023)
13. Сайт Metonit C# API [Электронный ресурс] URL: https://metanit.com/sharp/aspnet6/2.11.php (Дата обращения 18.03.2023)
14. Филипп Шабунин. Android для разработчико. 2017 год, 464с.
15. Сайт форум MSDN [Электронный ресурс] URL: <https://social.msdn.microsoft.com/Forums/en-US/home> (дата обращения 21.03.2023).
16. Сайт форум Stack Overflow [Электронный ресурс] URL: <https://stackoverflow.com/> (дата обращения 23.04.2023).
17. Уоррен Г. Марков. Microsoft SQL Server 2016. Полное руководство. Издательство: Питер, 2017 год, 832с.
18. Сайт Metonit MsSQL [Электронный ресурс] URL: https://metanit.com/sql/sqlserver/1.1.php (Дата обращения 12.03.2023)
19. Сайт TreatFace NeoLoad — ПО нагрузочного тестирования сетевых приложений [Электронный ресурс] URL: http://www.treatface.ru/brands/neotys/neoload-po-nagruzochnogo-testirovaniya-setevykh-prilozheniy/ (Дата обращения 29.04.2023)
20. Сайт NeoLoad NeoLoad Documentation [Электронный ресурс] URL: https://documentation.tricentis.com/neoload/8.2/en/WebHelp/#996.htm (Дата обращения 30.04.2023)