МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Современные технологии программирования мобильных систем»

Тема «Автостоянка»

Исполнитель

студент 2 курса группы 9 Кульш Денис Александрович

(Ф.И.О.)

Руководитель работы ассистент Панченко О.Л

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

Курсовой проект защищен с оценкой

Председатель Пацей Н.В

(подпись)

Минск 2021

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc72784309)

[2.Анализ требований к программному средству и разработка функциональных требований 6](#_Toc72784310)

[3.Проектирование программного средства 10](#_Toc72784311)

[3.1 Администратор 13](#_Toc72784312)

[4.Создание (реализация) программного средства 16](#_Toc72784313)

[5.Тестирование, проверка работоспособности и анализ полученных результатов 22](#_Toc72784314)

[6.Руководство по установке и использованию 25](#_Toc72784315)

[6.1 Установка приложения 26](#_Toc72784316)

[ЗАКЛЮЧЕНИЕ 31](#_Toc72784317)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 33](#_Toc72784318)

[Приложение А 34](#_Toc72784319)

[Приложение Б 34](#_Toc72784320)

[Приложение В 36](#_Toc72784321)

[Приложение Г 36](#_Toc72784322)

# ВВЕДЕНИЕ

В данный момент все больше и больше людей переходят на онлайн приложения, и это открывает практически безграничные перспективы как для компаний, так и для обычных людей.

Многие ищут свободные места на парковках, и это занимает время, которое можно потратить на работу или отдых.

Удобнее всего это делать через специальное приложение, которое поможет найти свободное место за считаные секунды.

Программное средство «Автостоянка» - программное средство, где каждый может найти свободное место, не тратя свое время.

Приложение имеет 2 типа пользователей: администратор и пользователь. Они имеют разный функционал. Пользователи могут просмотреть свой кабинет пользователя, а также оставлять комментарии и просматривать их, которые хранятся в базе данных, и самое главное, парковаться. Администратор может осуществлять поиск пользователей и их удаление, просмотр всех комментариев и удаление, время парковки различных пользователей, а также просмотр всех доступных и недоступных мест с фильтрацией.

Также будет реализована регистрация и авторизация пользователей. Хранение всех пользователей будет осуществлено в базе данных, и при авторизации или регистрации будет непосредственное обращение к ней. Также вся информация о стоянке и местах будут хранится в базе данных.

1.Аналитический обзор литературы

Для выполнения этого курсового проекта не обошлось без использования дополнительной литературы. Таковой является информация на сайте <https://metanit.com/>. На этом сайте имеется много информации, нужной для разработки этого курсового проекта. Также использовалась информация с сайта <http://professorweb.ru/>.

Самый большое количество информации было получено с официального сайта <https://docs.microsoft.com/en-us/dotnet/csharp/>. Статьи с <https://habr.com/ru/> были также очень полезны.

В качестве аналога я выбрал сайт <https://www.parkouka.by/>, т.к. он максимально приближен к моей задумке. Краткий обзор этого сервиса представлен ниже.

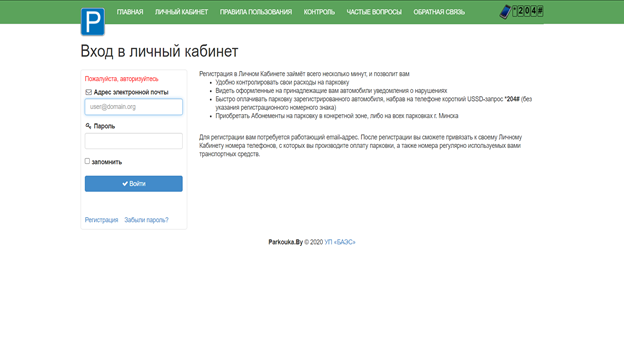


Рисунок1.1 – раздел «Личный кабинет»

На данной странице, мы видим вход и регистрацию для дальнейшего использования. В данный момент большинство ресурсов использует авторизацию пользователей для дальнейшего взаимодействия между сайтом и пользователем. В верхней части, находится навигационное меню, для упрощения перехода между страницами.

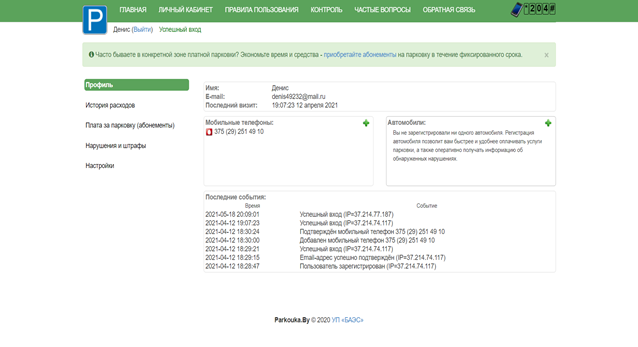


Рисунок 1.2 – Личный кабинет

Данный личный кабинет предоставляет информацию о пользователе и последних событиях.

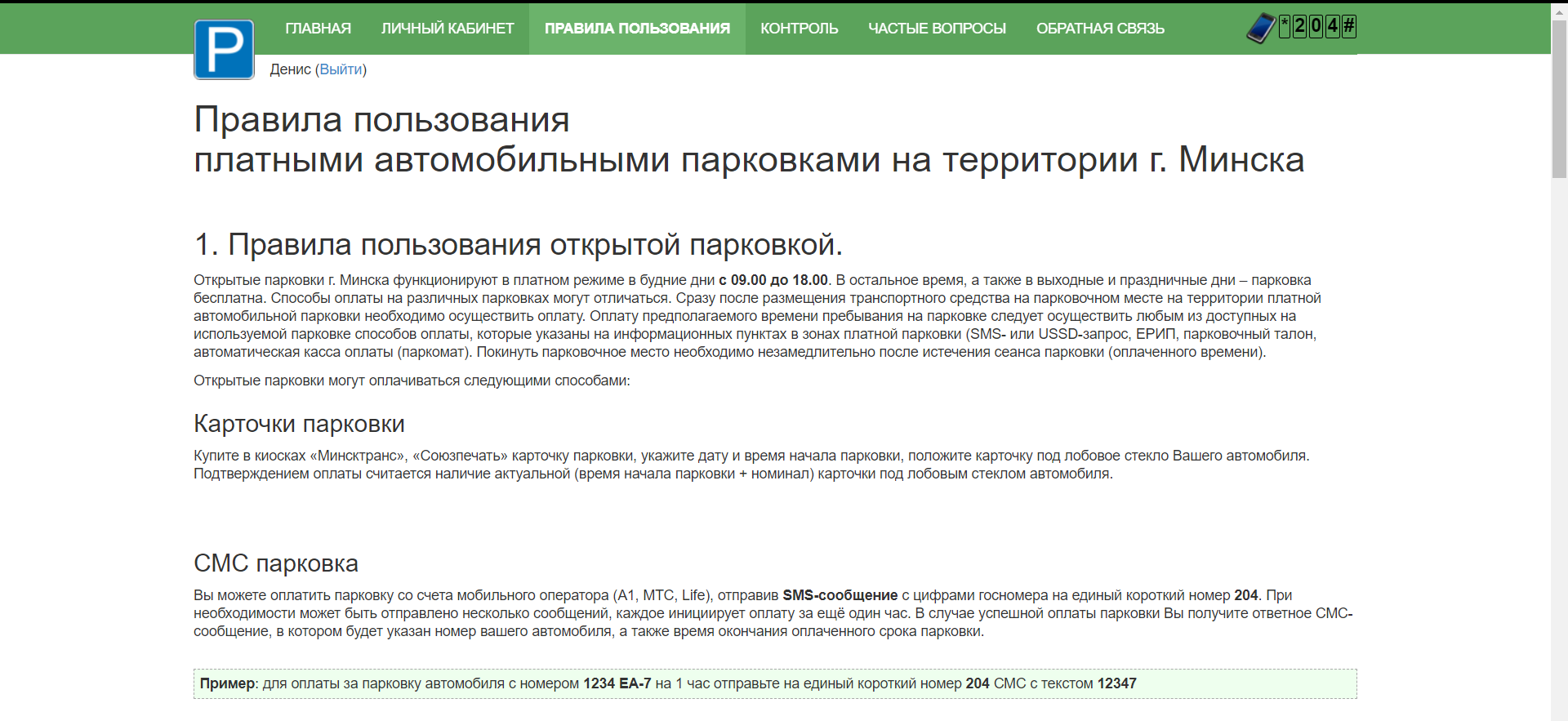


Рисунок 1.3 – Правила пользования

Данная страница показывает и рассказывает о правилах использования данным веб-сайтом и системой парковки в Беларуси.

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

Задачей курсового проекта является разработка приложения, предназначенного осуществления выбора стояночного места, а также ряда дополнительных операций: добавление и просмотр комментариев, личного кабинета, администрирование.

Программный продукт должен быть реализовать на объектно-ориентированном языке программирования C# с использованием технологии WPF. Windows Presentation Foundation (WPF) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Window Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

Также существует урезанная версия CLR, называющаяся WPF/E, она же известна как Silverlight.

База данных была разработана в Microsoft SQL Server.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире.

SQL Server характеризуется такими особенностями как:

* Производительность. SQL Server работает очень быстро.
* Надежность и безопасность. SQL Server предоставляет шифрование данных.
* Простота. С данной СУБД относительно легко работать и вести администрирование.

Для организации баз данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Основной используемый язык запросов —Transact-SQL — реализован на структурированном языке запросов (SQL) с расширениями.

Для осуществления связи между базой данных и приложением на C# необходим посредник, которым будет являться технология Entity Framework. Она основана на платформе .NET Framework и предназначена для работы с данными.

Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Эта технология предоставляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

* Database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных;
* Model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере;

Code first: разработчик создает класс модели данных, которые будут храниться в базе данных, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

В курсовом проекте для разработки был выбран второй подход.

Приложение необходимо реализовать с использованием шаблона(логики) MVVM и иными шаблонами с организацией взаимодействия с базой данных. В качестве клиентской части необходимо реализовать оконное приложение с использованием стандартных и иных библиотек.

Для разработанного продукта необходимо предусмотреть следующие задачи:

* Интуитивно понятный интерфейс;
* Поле для отображения информации пользователя;
* Механизм подбора стояночного места;
* Администрирование

По итогу работы над продуктом, его необходимо протестировать и отладить, проанализировать его возможности и, по необходимости, внести дополнения или улучшения проекта, с возможностью дальнейшего расширения проекта.

Программное средство предоставляет пользователю следующие функциональные возможности:

Пользователь:

* Просматривать свой личный кабинет;
* Просматривать комментарии;
* Оставлять комментарии;
* Парковать свой автомобиль с помощью механизма ПС;
* Выезжать из автостоянки с помощью механизма ПС;
* Просматривать время нахождения на парковке и итоговую стоимость;
* Регистрироваться и авторизоваться;

Администратор:

* Просмотр информацию о пользователе;
* Поиск пользователя;
* Удаление пользователя;
* Просмотр всех комментариев;
* Удаление комментариев;
* Просмотр времени стоянки всех пользователей;
* Просмотр статуса места;
* Фильтрация мест;

Наглядно это видно на UML-схеме, изображенной на рисунке 2.1 (Приложение A).

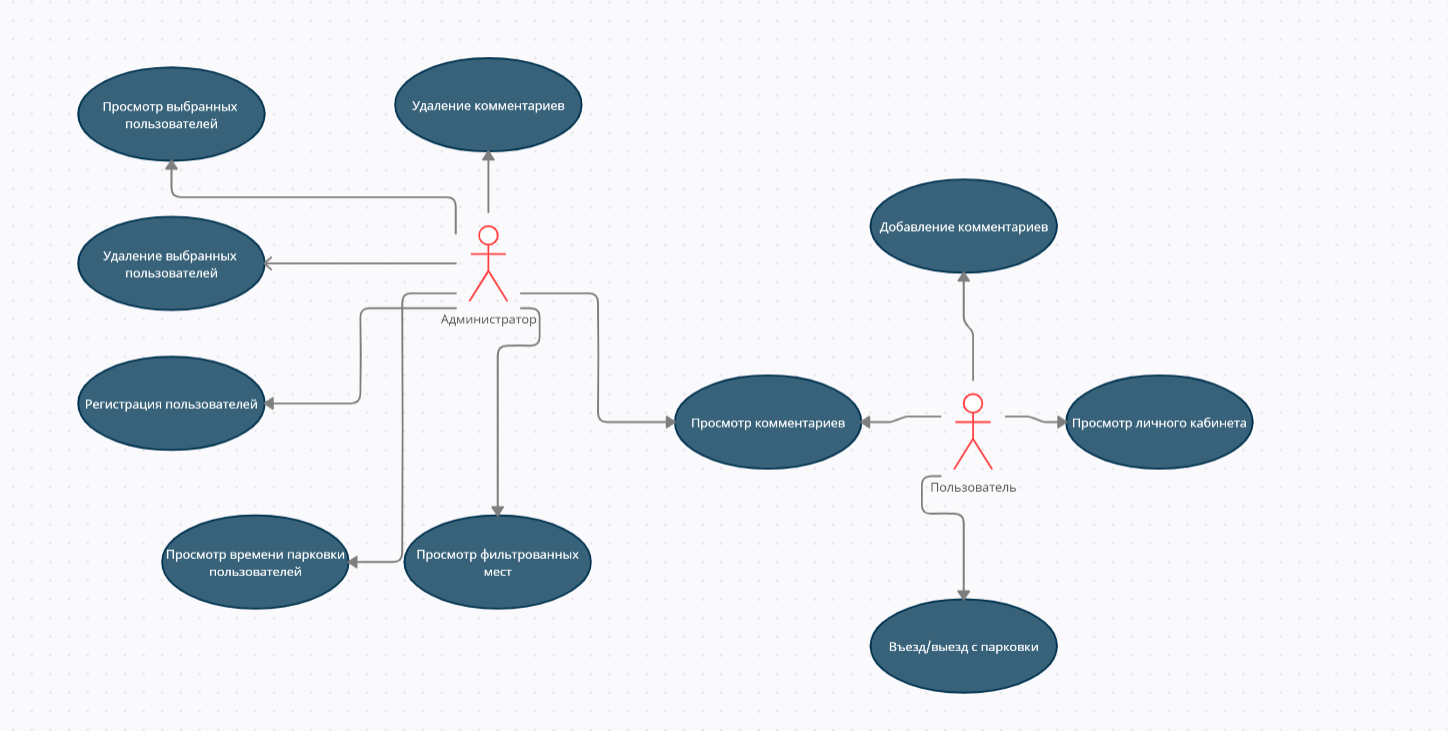


Рисунок 2.1 – «UML-диаграмма возможностей»

# 3.Проектирование программного средства

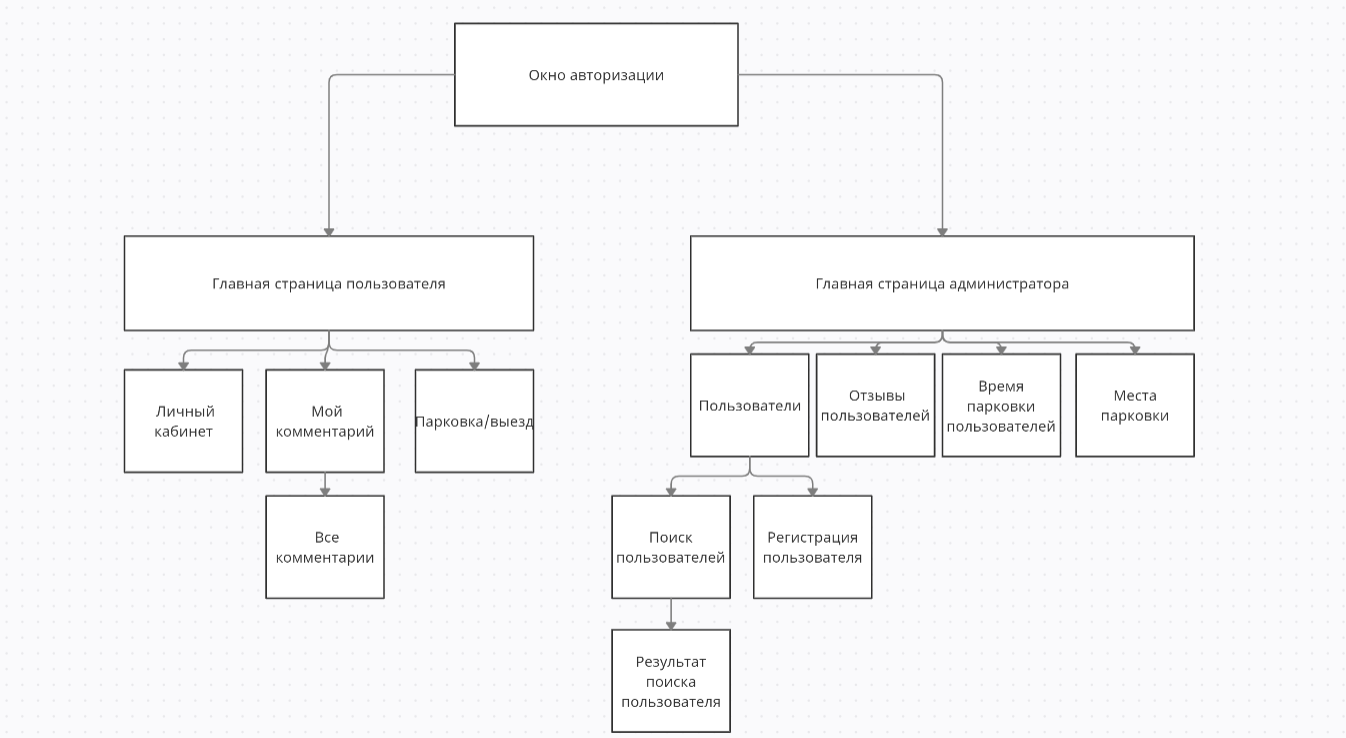


Рисунок 3.1 - «Авторизация и главные страницы»

На рисунке 3.1 представлена общая схема работы приложения.

При запуске приложения у нас появляется окно авторизации, которое требует ввода логина и пароля для начала работы.

В случае, если вас нету своей учётной записи, её можно создать, нажав на кнопку «Регистрация» в окне авторизации. Вы будете направлены в окно регистрации, где вам нужно будет заполнить все поля. После успешной регистрации, вам нужно будет вернутся в окно авторизации и ввести ваши данные, указанные при регистрации.

Далее указываем почту и пароль. После ввода логина и пароля в соответствующих полях, проверяется правильность ввода данных.

В случае неверного ввода данных, выведется соответствующее сообщение. В ином случае окно авторизации закроется и откроется главное окно приложения.

Страница «Личный кабинет» (рисунок 3.2).

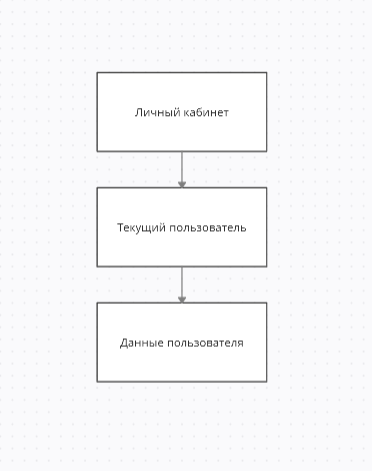


Рисунок 3.2 - «Личный кабинет»

На рисунке 3.2 показана схема страницы «Личный кабинет».

Это страница выводит информацию о пользователе, такие как:

* Id аккаунта
* Имя
* Фамилия
* Mail
* Телефон

На эту страницу можно попасть, нажав на элемент с соответствующей картинкой, находящейся в левом верхнем углу, первым элементом.



Рисунок 3.3 - «Комментарии для пользователей»

На этой странице пользователь может оставлять комментарии, которые ограничены 300 символами.

Также пользователь может перейти в другое окно, в котором он может увидеть комментарии, оставленные другими пользователями.

Пользователь видит:

* Имя пользователя
* Дату и время комментария
* Сам комментарий

Страница с комментариями стоит на второй позиции в навигационном меню сверху, после страницы «Личный кабинет».



Рисунок 3.4 - «Парковка/выезд»

Данная страница, представленная на схеме 3.4 является главной.

На этой странице пользователь может посмотреть загруженность данной парковки, а также цену в минуту, установленную на данной парковке.

При нажатии кнопки «обновить» пользователь может в реальном времени просмотреть эту загруженность, сказанной выше. При нажатии на кнопку «Припарковаться/выехать», пользователю нужно будет указать сектор, в котором он захочет парковаться, и механизм сам выберет место для парковки, и выведет на экран.

При выезде с данной парковки, будет указано, сколько пользователь оставался на данной парковке и итоговая сумма.

# 3.1 Администратор

После авторизации, администратор попадает на отдельную страницу с разными вкладками на которых он может производить администрирование, представленных далее.

А также на главной странице будет указано имя аккаунта администратора.

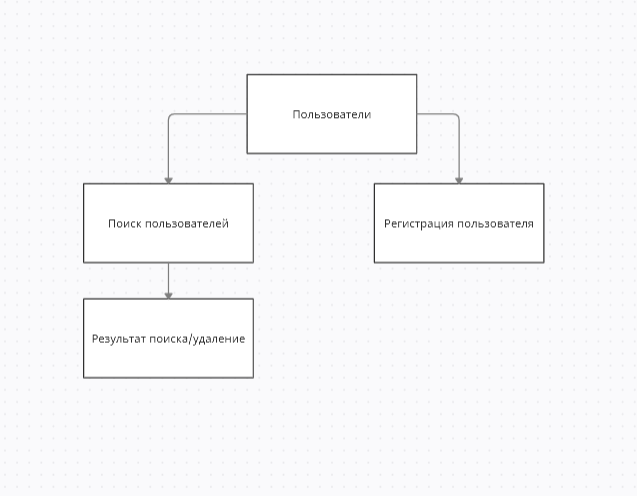


Рисунок 3.5 - «Пользователи»

На данных страницах, администратор может производить поиск, регистрацию, а также удаление пользователей.

Первым делом, после нажатия на вкладку с пользователями открывается новое окно с поиском или регистрацией. При выборе регистрации, администратор может заполнить форму и создать нового пользователя для дополнительного упрощения. При выборе поиска пользователя, администратор вводит данные почты аккаунта и ему показывает результат поиска с последующей возможностью удаления пользователя.

При удалении пользователя, происходит удаление всех оставленных комментариев.

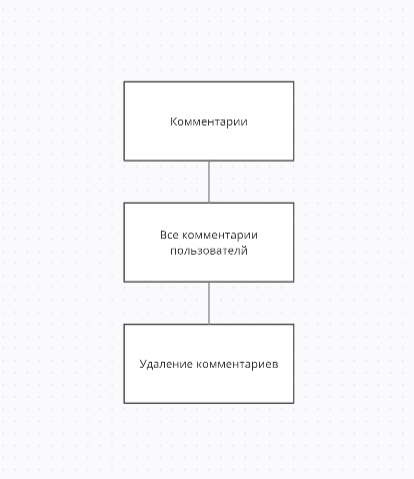


Рисунок 3.6 - «Комментарии»

На данной странице администратор может посмотреть все оставленные комментарии, которые отсортированы по дате, а также произвести их удаление с помощью нажатия соответствующей кнопки.



Рисунок 3.7 - «Время парковки пользователей»

На данной странице администратор может просматривать историю парковок всех пользователей, для дальнейших нужд.

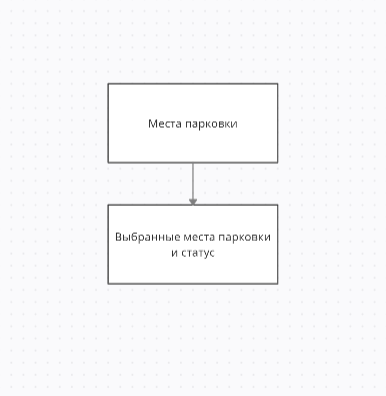


Рисунок 3.8 - «Места парковки»

При входе на данную страницу, администратор выбирает сектор, в котором он хочет посмотреть статус места.

Структура проекта представлена в приложении Б.

В этой таблице приведены основные логические составляющие пакеты, которые используются в работе приложения.

Таблица 3.7 – Описание структурных пакетов проекта

|  |  |
| --- | --- |
| Имя пакета | Описание |
| Model | Здесь описаны модели, с которыми происходит работа в приложении: |
| Database | Содержит классы и интерфейсы для работы с данными БД. Они являются связующим элементом между данными в БД и теми данными, с которыми работает приложение |
| Resources | Файлы с кортинками |
| View | Содержит представления, которые позволяют пользователю работать с приложением. Они описывают графическую составляющую приложения. |
| ViewModel | Содержит логику, которая позволяет получить данные при помощи View, обработать их, использую при этом Model, после чего передать в базу данных. |
| App.config | Файл конфигурации приложения |
| App.xaml | Основная задача данного файла состоит в определении ресурсов, общих для приложения |

А также во время разработки приложения, будут добавлены дополнительные пакеты для работы приложения.

# 4.Создание (реализация) программного средства

При создании приложения использовался паттерн Singleton, а также, для работы с базой данных. Описание представлена ниже.

Диаграмма классов представлена в приложении В.

Пакетная структура проекта показана на рисунке 4.1

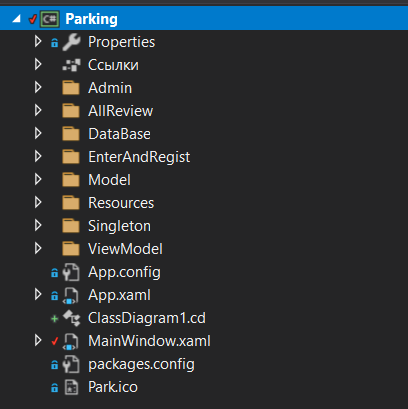


Рисунок 4.1 – «Структура пакетов проекта»

В Database находится структура модели базы данных приложения. При проектировании базы данных данного курсового проекта для взаимодействия с базой данных используется Entity Framework.

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

Для работы с данными нужно было построить базу данных представленную ниже:

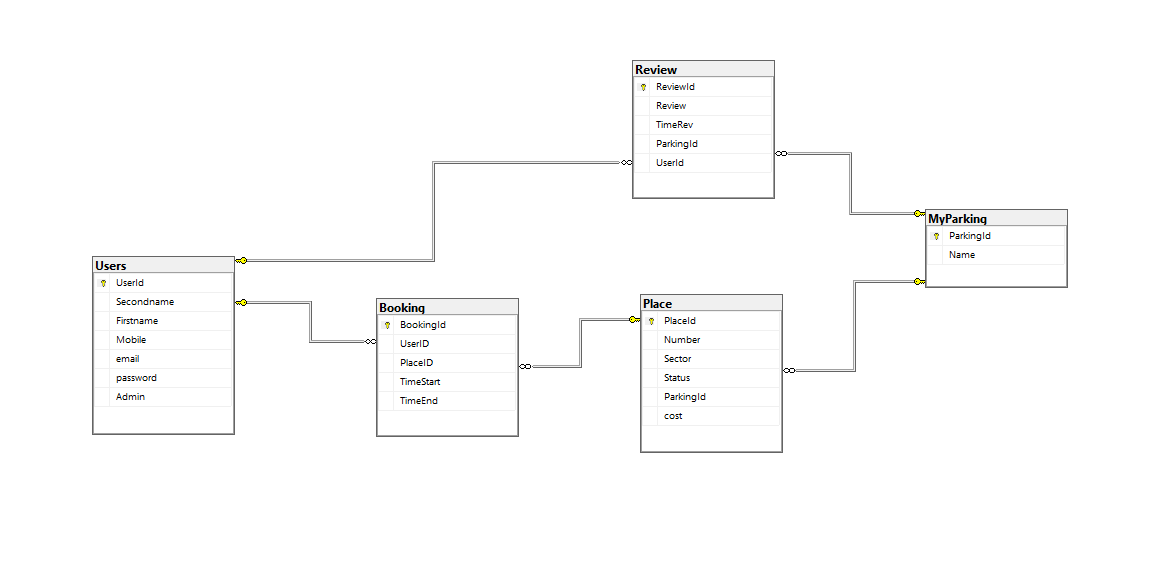


Рисунок 4.2 – «Диаграмма таблиц со связями в базе данных»

Описание таблицы Users (таблица 4.2)

Таблица 4.2 - «описание Users»

|  |  |
| --- | --- |
| UserId | Идентификатор |
| Secondname | Фамилия пользователя |
| Firstname | Имя пользователя |
| Mobile | Номер телефона пользователя |
| email | Почта пользователя |
| password | Пароль пользователя |
| Admin | Проверка админа |

Описание таблицы Booking (таблица 4.3)

Таблица 4.3 - «описание Booking»

|  |  |
| --- | --- |
| BookingId | Идентификатор |
| UserId | Идентификатор пользователя |
| PlaceId | Идентификатор места |
| TimeStart | Начало времени парковки |
| TimeEnd | Конец времени парковки |

Описание таблицы Place (таблица 4.4)

Таблица 4.4 - «описание Place»

|  |  |
| --- | --- |
| PlaceId | Идентификатор |
| Number | Номер места |
| Sector | Сектор места |
| Status | Статус места |
| ParkingId | Идентификатор парковки |
| Cost | Стоимость места/в минуту |

Описание таблицы MyParking (таблица 4.5)

Таблица 4.5 - «описание MyParking»

|  |  |
| --- | --- |
| ParkingId | Идентификатор |
| Name | Название парковки |

Описание таблицы Review (таблица 4.6)

Таблица 4.6 - «описание Review»

|  |  |
| --- | --- |
| ReviewId | Идентификатор |
| Review | Комментарий(отзыв) |
| TimeRev | Время комментария |
| ParkingId | Идентификатор парковки |
| UserId | Идентификатор пользователя |

Краткое описание назначение таблиц:

Таблица Users. Данная таблица предназначена для хранения данных о пользователях и админе. Заполнение информации происходит при регистрации пользователями или регистрации пользователя администратором.

Таблица Booking. Данная таблица предназначена для хранения данных о бронировании мест пользователями. Заполнение данных происходит при парковке и выезде из парковки.

Таблица Place. Данная таблица предназначена для хранения данных о местах парковки. Количество мест фиксированное количество и заполняется SQL-запросом при создании данной таблицы.

Таблица MyParking. Данная таблица предназначена для хранения данных о названии парковки. Данная таблица заполняется SQL-запросом при создании данной таблицы.

Таблица Review. Данная таблица предназначена для хранения комментариев пользователей. Заполняется при добавлении нового комментария.

Папка Admin:

Папка Admin хранит все окна для работы администратора представленная на рисунке 4.7.

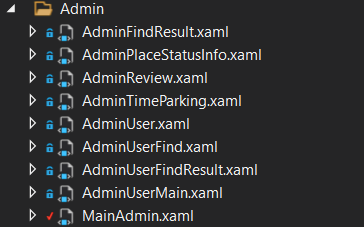


Рисунок 4.7 – «Папка Admin»

Таблица 4.8 - «Описание файлов Admin»

|  |  |
| --- | --- |
| **Название** | **Краткое описание** |
| AdminPlaceStatusInfo.xaml. | Данное окно выводит информацию о статусе мест выбранные в фильтре. |
| AdminReview.xaml | Данное окно выводит все комментарии пользователей. Администратор может удалить любой комментарий. |
| AdminTimeParking.xaml | Данное окно выводит информацию о времени парковки всех пользователей. |
| AdminUser.xaml | В данном окне администратор может заполнить поля и зарегистрировать нового пользователя. |
| AdminUserFind.xaml | В данном окне администратор указывает пользователя для поиска. |
| AdminUserFindResult.xaml | Данное окно выводит результат поиска пользователя и возможность удаления пользователя. |
| AdminUserMain.xaml | Страница для выбора регистрации либо поиска пользователя |
| MainAdmin.xaml | Главная окно администратора. |

Папка Allreview:

Папка Allreview хранит окно со всеми комментариями для вывода пользователям и класс для этих же комментариев, для передачи в ListBox представленные на рисунке 4.9.

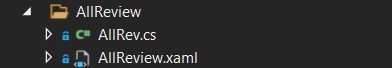


Рисунок 4.9 – «Папка Allreview»

|  |  |
| --- | --- |
| **Название** | **Краткое описание** |
| AllReview.xaml | Окно со всеми комментариями для вывода пользователям |
| AllRev.cs | Класс для этих же комментариев, для передачи в ListBox |

Папка Database:

Папка Database хранит все классы таблиц базы данных показанная на рисунке 4.10.

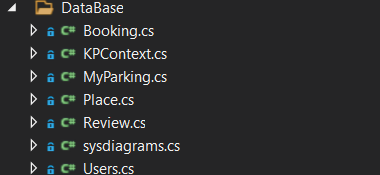


Рисунок 4.10 - «Папка Allreview»

Вся информация о таблицах базы данных была описана выше, но давились два класса KPContext.cs (определяет контекст данных, используемый для взаимодействия с базой данных) и sysdiagrams.cs (диаграмма).

Папка EnterAndRegist:

Папка EnterAndRegist содержит окно, которое появляется при запуске приложения. Представляет собой окно входа и регистрации. Рисунок 4.11.



Рисунок 4.11 - «Папка EnterAndRegist»

Папка Model:

Папка Model хранит две модели. Первая модель – модель для комментариев для вывода админу. Вторая модель – модель для вывода информации о времени парковки пользователей. Рисунок 4.12.

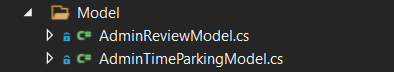


Рисунок 4.12 - «Model»

Папка Singleton:

Папка Singleton содержит паттерн Singleton, который при входе выводит информацию об админе. Рисунок 4.13.



Рисунок 4.12 - «Singleton»

Папка ViewModel:

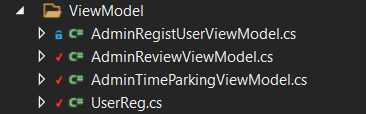


Рисунок 4.13 - «ViewModel»

Папка ViewModel содержит 4 модели связи для EnterAndRegist и Admin.

1.AdminRegistUserViewModel - представление модели, методы для регистрации пользователей и валидацию.

2.AdminReviewViewModel - представление модели. Содержит конструктор для вывода пользовательских комментариев и метод для удаления комментариев.

3. AdminTimeParkingViewModel - представление модели. Содержит конструктор для вывода информации о времени парковки пользователей.

4. UserReg – представление модели. Содержит методы для входа и регистрации, валидации, шифрование пароля.

Окно MainWindow:

Содержит 3 вкладки с личным кабинетом, комментариями и основным механизмом парковки.

# 5.Тестирование, проверка работоспособности и анализ полученных результатов

При тестировании данного приложения, были применены сценарии, которые могли бы привести к ошибке. В этой главе мы рассмотрим некоторые такие сценарии и посмотрим на их обработку.

В момент регистрации, возможна такая ситуация, в которой пользователь ничего не ввёл или ввёл неверные данные. Обработка данного сценария приведена на рисунке 5.1

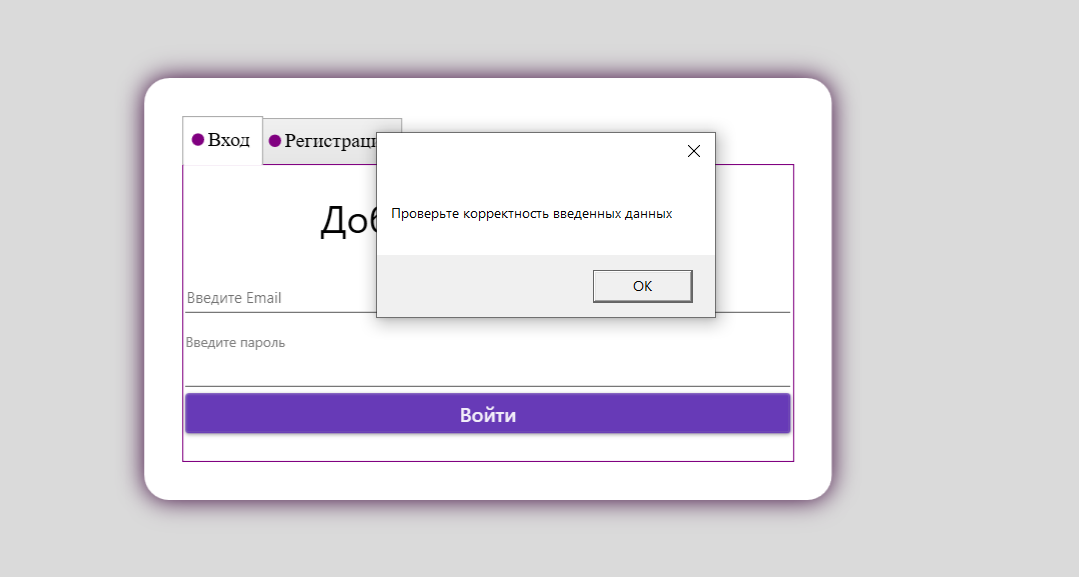


Рисунок 5.1 – «Ошибка корректности ввода»

Также возможен такие случаи:

* Пользователь хочет зарегистрироваться, при этом не указав никаких данных (рисунок 5.3)

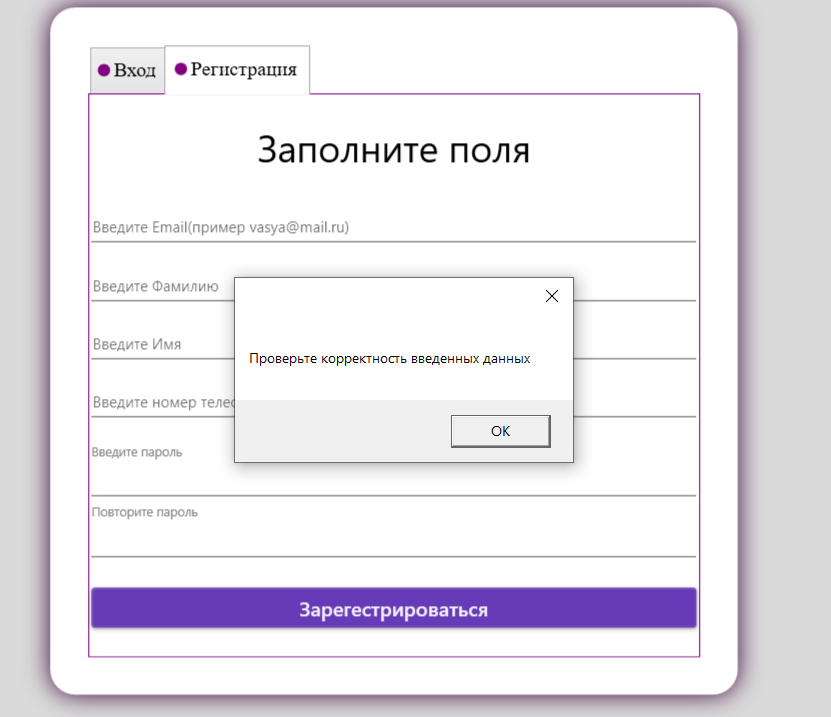


Рисунок 5.3 – «Отсутствие данных»

Как видно на рисунке 5.3, при нажатии кнопки регистрации, выведется сообщение об корректности данных.

При несовпадении паролей будет выводиться сообщение. Рисунок 5.4

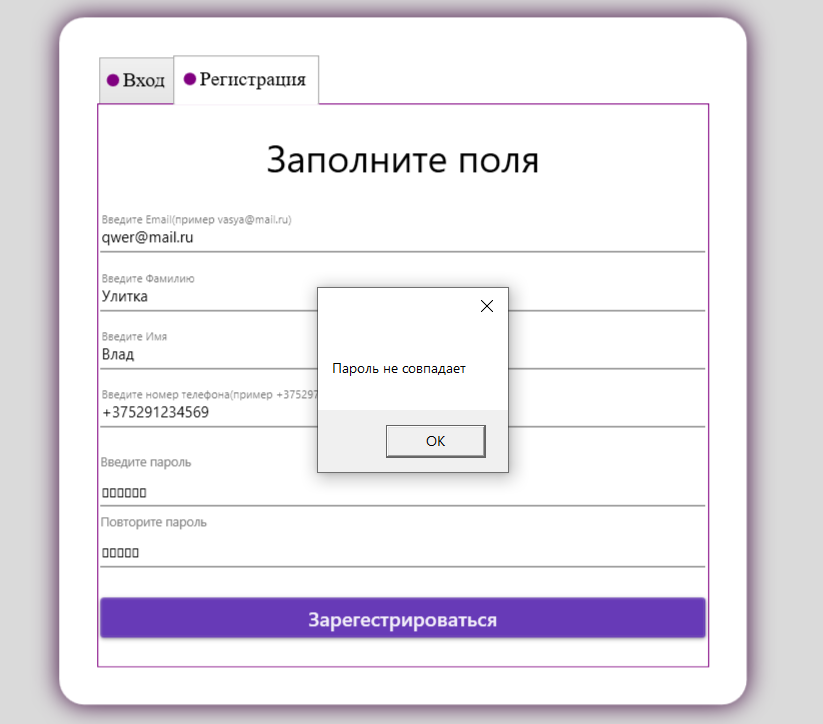


Рисунок 5.3 – «Несовпадение пароля»

Также возможен такой исход, при котором пользователь хочет зарегистрироваться при помощи такого mail/логина, который уже есть. Обработка такого рода сценариев показана на рисунке 5.5.

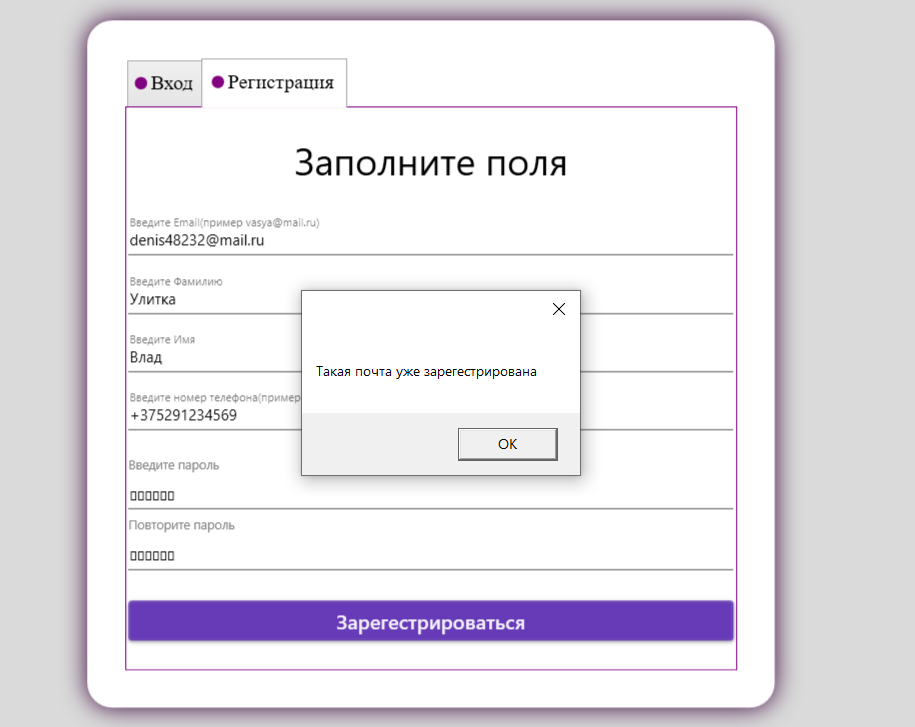


Рисунок 5.5 – «Регистрация пользователя с занятой почтой»

А также, если существует такой же телефон. Рисунок 5.6

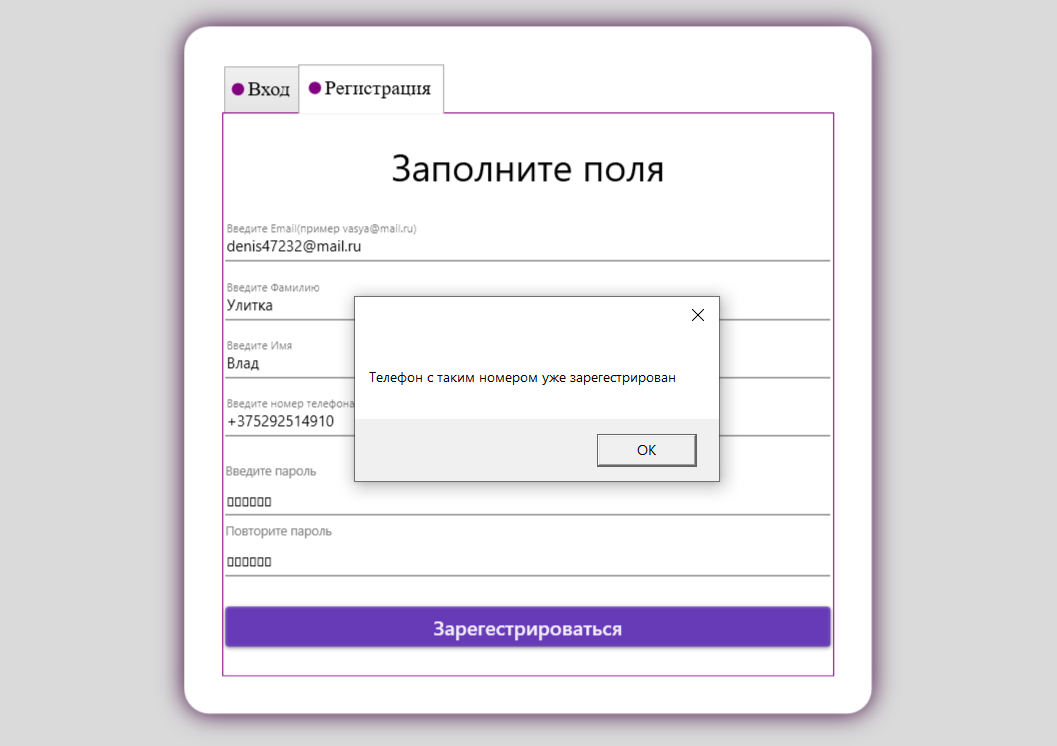


Рисунок 5.5 – «Регистрация пользователя с занятым телефоном»

При написании комментария, если комментарий не пройдет валидацию, то выведется сообщение. Пользователь может ввести слишком много символов. Рисунок 5.6.

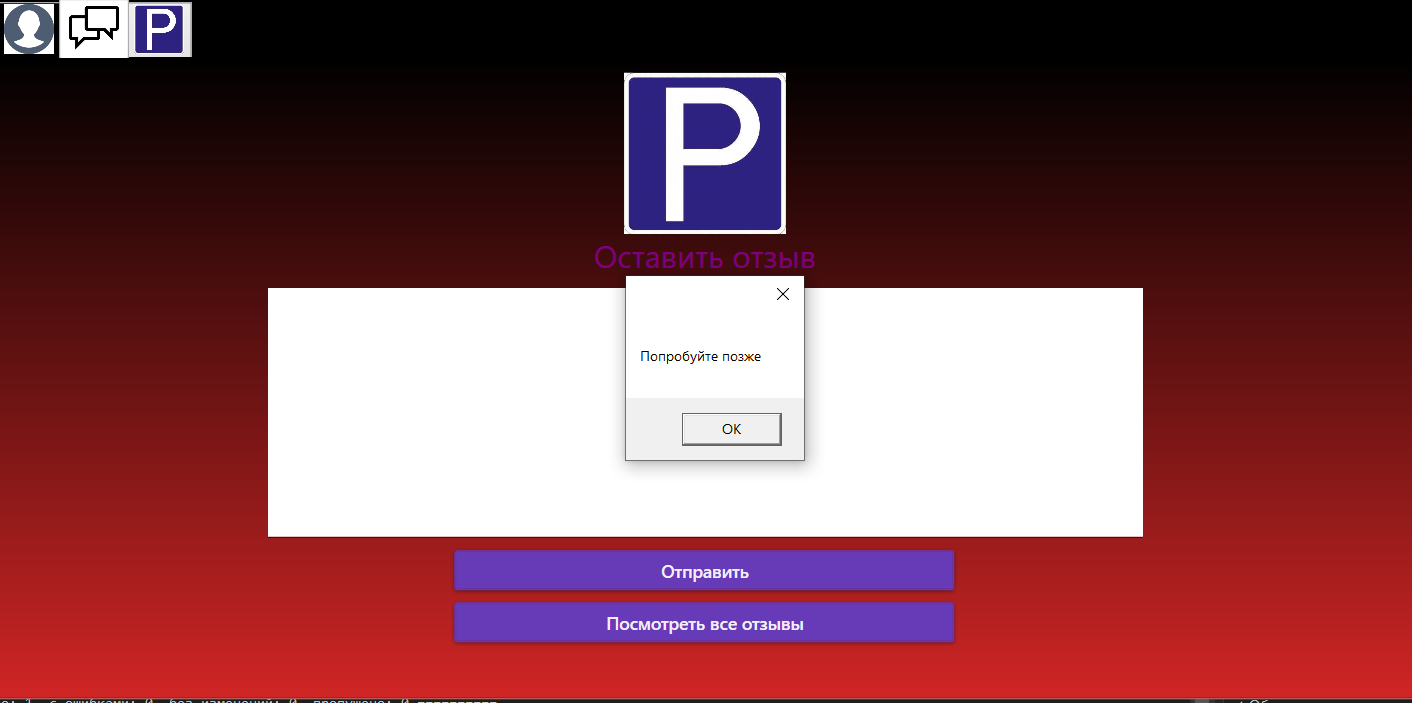


Рисунок 5.6 – «Выход за пределы»

# 6.Руководство по установке и использованию

## 6.1 Установка приложения

Установите MicrosoftSQLServer по ссылке:. [SQL Server Downloads | Microsoft](https://www.microsoft.com/en-us/sql-server/sql-server-downloads)(рисунок 6.1)

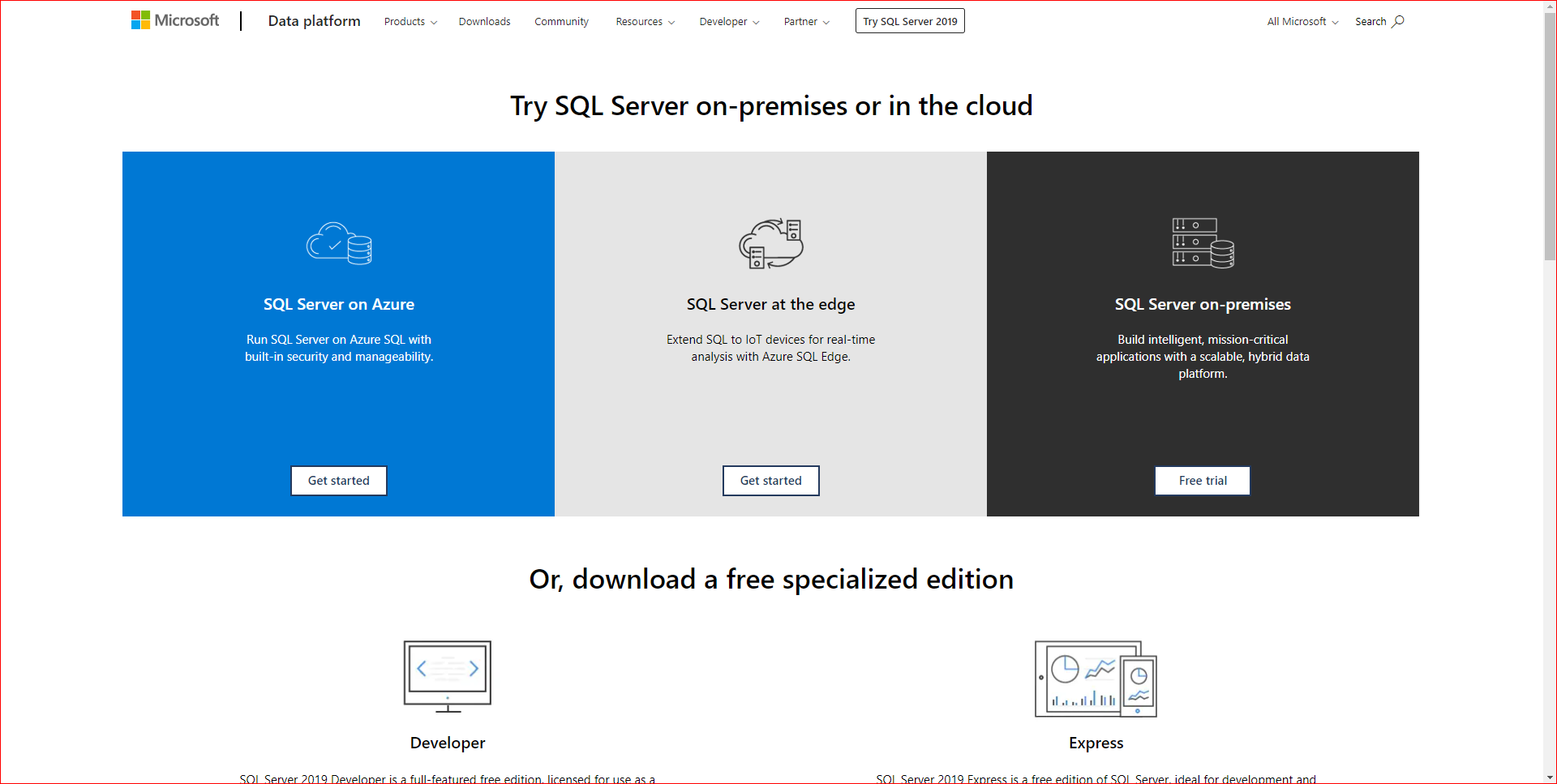


Рисунок 6.1 – «Установка MicrosoftSQLServer»

Для установки программного средства скачайте архив с GitHub <https://github.com/Dingo-K/KP.git>.

Распакуйте его в выбранную вами папку. После распаковки запустите файл CreateDatabase.sql (рисунок 6.2)

В MicrofotSQLServer нажмите Execute для создания базы данных приложения.

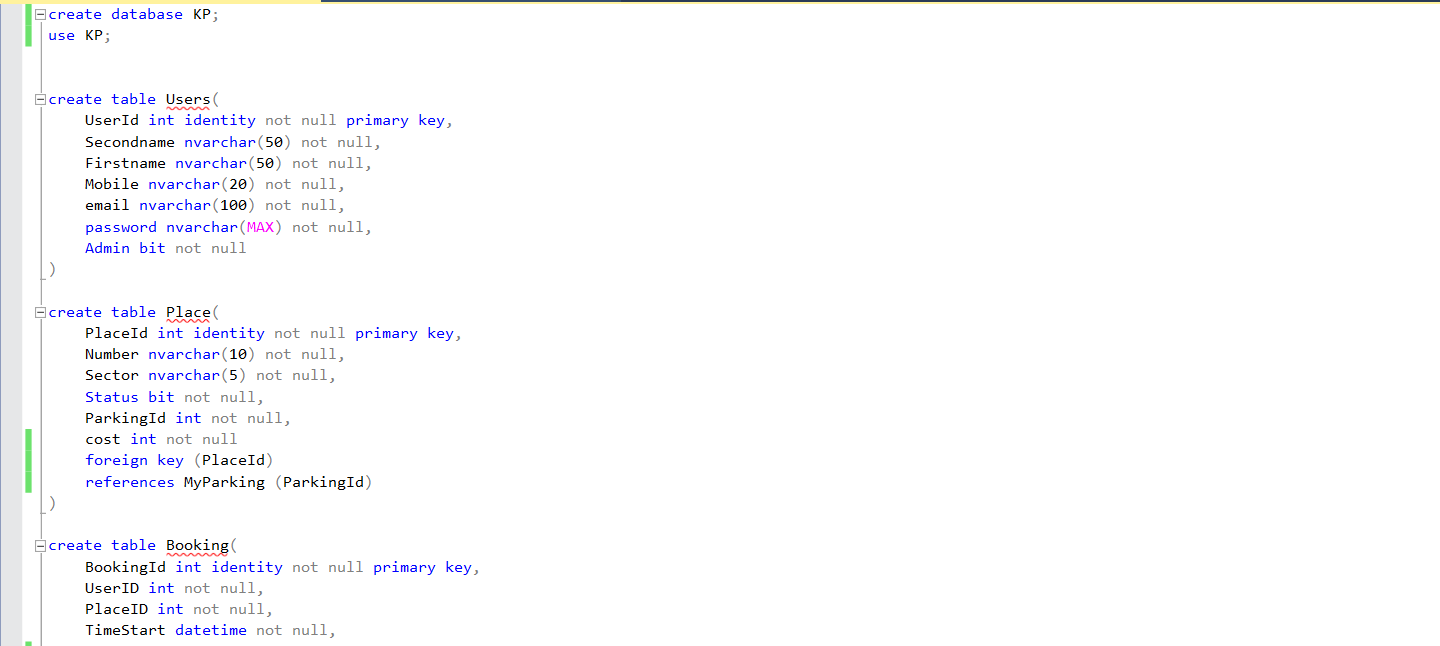


Рисунок 6.2 – «Создание базы данных»

При запуске приложения у нас появляется окно авторизации, которое требует ввода логина и пароля для начала работы.

В случае, если вас нету своей учётной записи, её можно создать, нажав на кнопку «Регистрация». Вы будете направлены в окно регистрации, где вам нужно будет заполнить все поля. После успешной регистрации, вам нужно будет вернутся в окно авторизации и ввести ваши данные, указанные при регистрации.

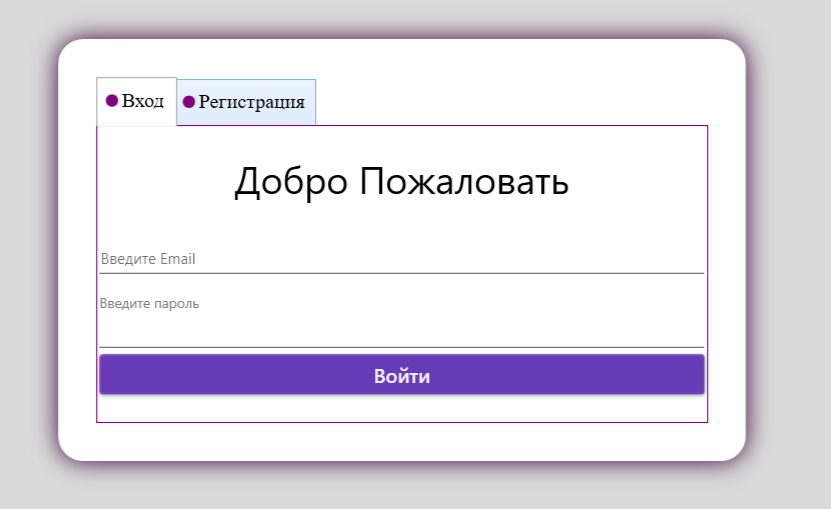


Рисунок 6.1 – «Окно авторизации»

После успешного входа, мы попадаем в главное окно, которое изображено на рисунке 6.2.

В этом окне мы можем просмотреть информацию о пользователе

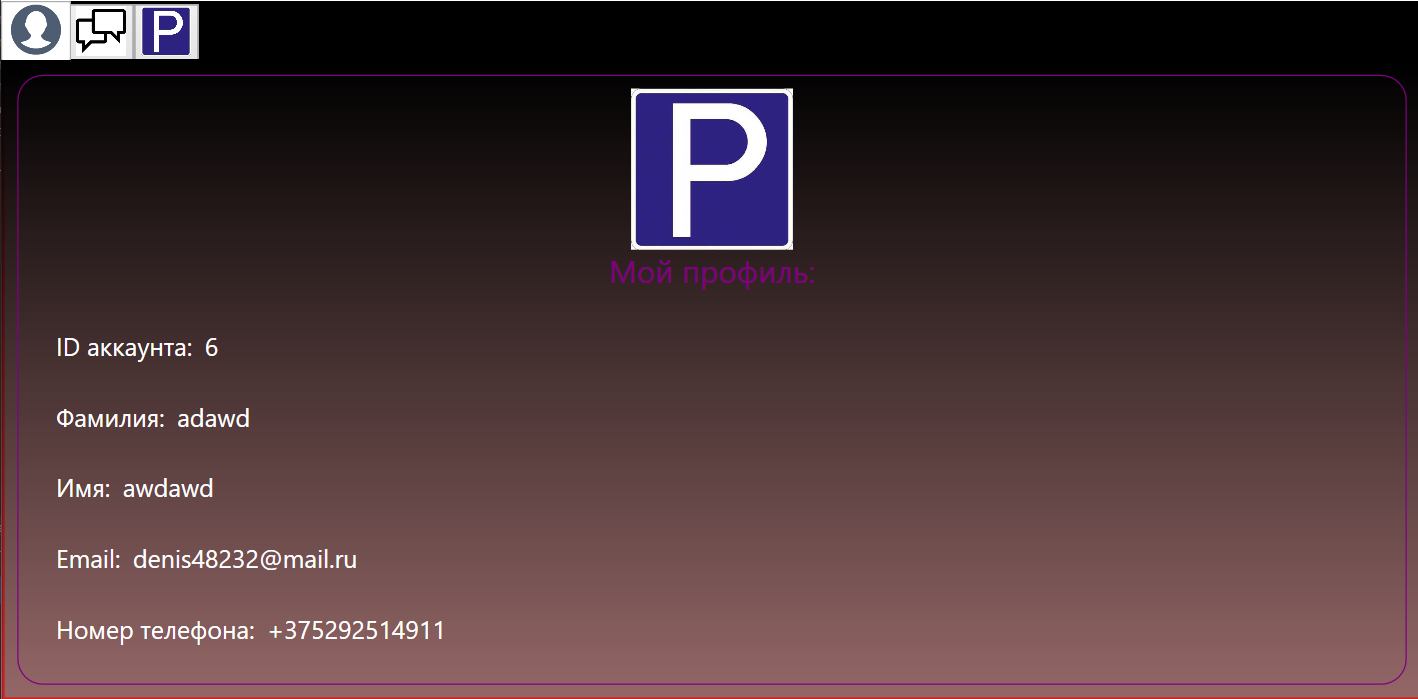


Рисунок 6.2 – «Личный кабинет»

Вторая вкладка – это комментарии, она появляется при нажатии на вторую вкладку. Рисунок 6.3.

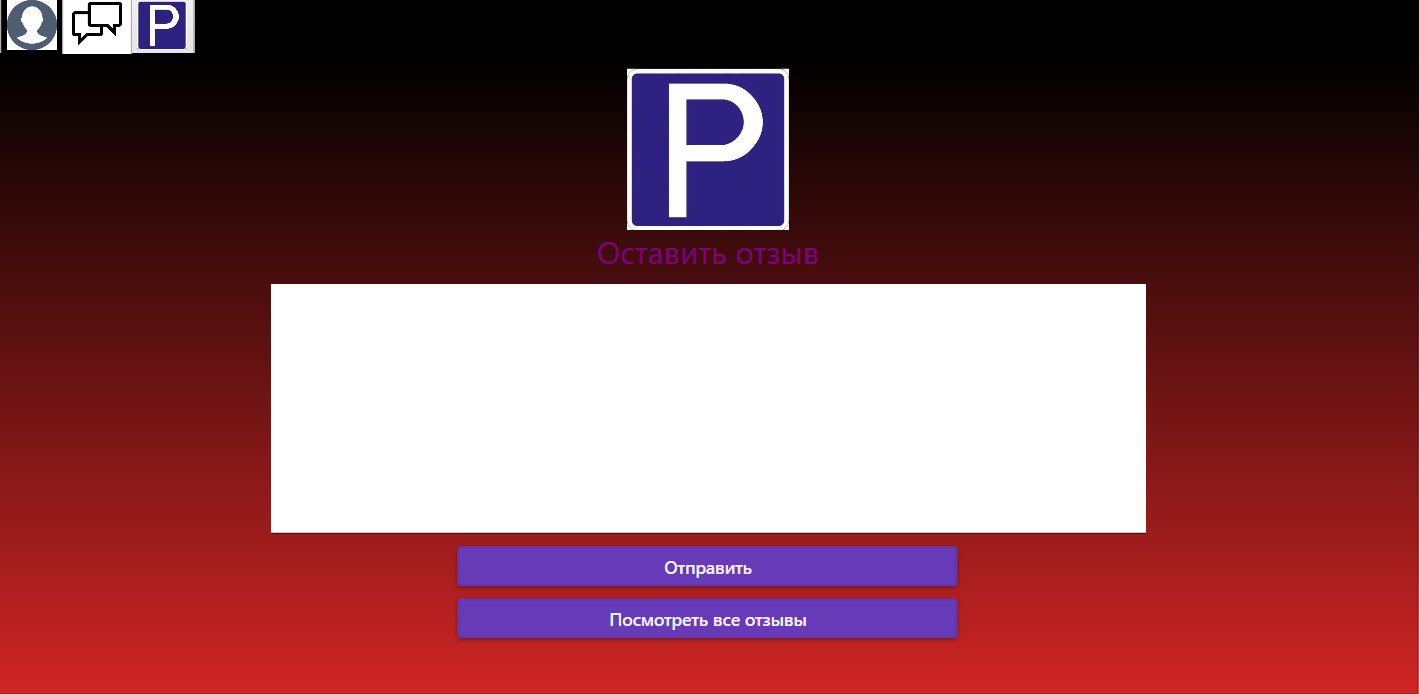


Рисунок 6.3 – «Отзывы»

На данной странице пользователь может отправить комментарий, а также посмотреть все отзывы.

Следующая третья вкладка – это главная страница, она появляется при нажатии на вторую вкладку. Рисунок 6.4.



Рисунок 6.4 – «Главная страница»

При нажатии на кнопку обновить, мы можем увидеть загруженность парковки в данный момент.

При нажатии на кнопку парковки и выбора сектора, приложение начинает подбирать автоматически место для более удобного использования. Т. К. Пользователю будет намного удобнее.

Далее представлено главная страница администратора. Рисунок 6.5

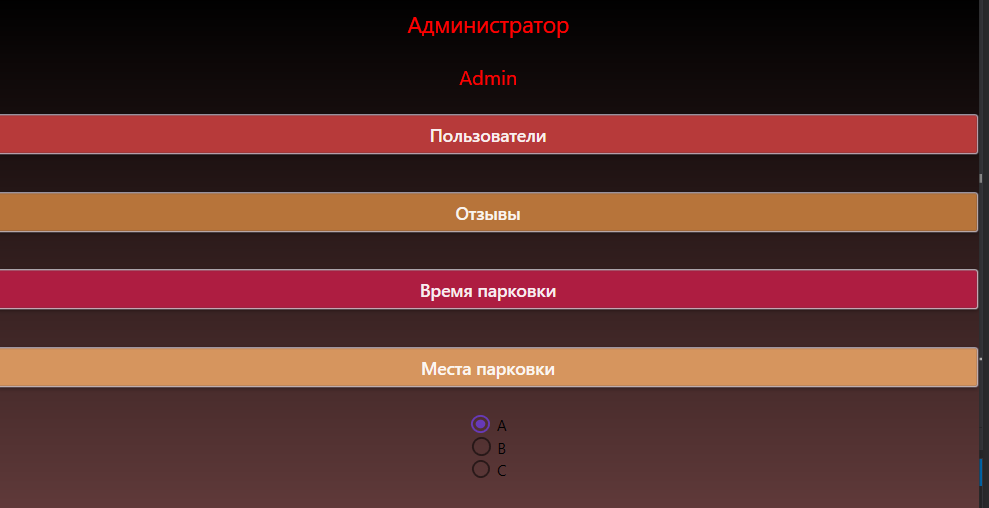


Рисунок 6.5 – «Главная страница»

На этой странице можно совершать действия над пользователями, отзывами и просматривать остальную информацию.

# ЗАКЛЮЧЕНИЕ

В данном курсовом проекте было разработано программное средство «Площадка объявлений» при помощи языка программирования C#, API-интерфейса Windows Presentation Foundation, технологии Entity Framework и базы данных Microsoft SQL Server. При выполнении курсового проекта использовались принципы и приемы ООП.

Данное программное средство имеет небольшой функционал, т. к. для приложения автостоянки не требуется большой и нагруженный.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

Пользователь:

* Просматривать свой личный кабинет;
* Просматривать комментарии;
* Оставлять комментарии;
* Парковать свой автомобиль с помощью механизма ПС;
* Выезжать из автостоянки с помощью механизма ПС;
* Просматривать время нахождения на парковке и итоговую стоимость;
* Регистрироваться и авторизоваться;

Администратор:

* Просмотр информацию о пользователе;
* Поиск пользователя;
* Удаление пользователя;
* Просмотр всех комментариев;
* Удаление комментариев;
* Просмотр времени стоянки всех пользователей;
* Просмотр статуса места;
* Фильтрация мест;

Приложение хорошо работает с базой данных. В ходе тестирования, не было выявлено нарушений в работе приложения с базой данных.

Для разработки дизайна использовалась библиотека MaterialDesign, которая имеет огромное количество красивых и удобных элементов управления.

Приложение было успешно протестировано, что означает пригодность этого приложения для своей цели.

Для хранения исходного кода и удобства контроля версий проекта использовался крупнейший веб-сервис для совместной разработки – GitHub.

Ссылка на проект: (<https://github.com/Dingo-K/KP.git>.)

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

[1]. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2018. – 175 с.

[2]. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с. [3]. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: http://msdn.microsoft.com/library/

[4]. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com

[5]. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: https://professorweb.ru

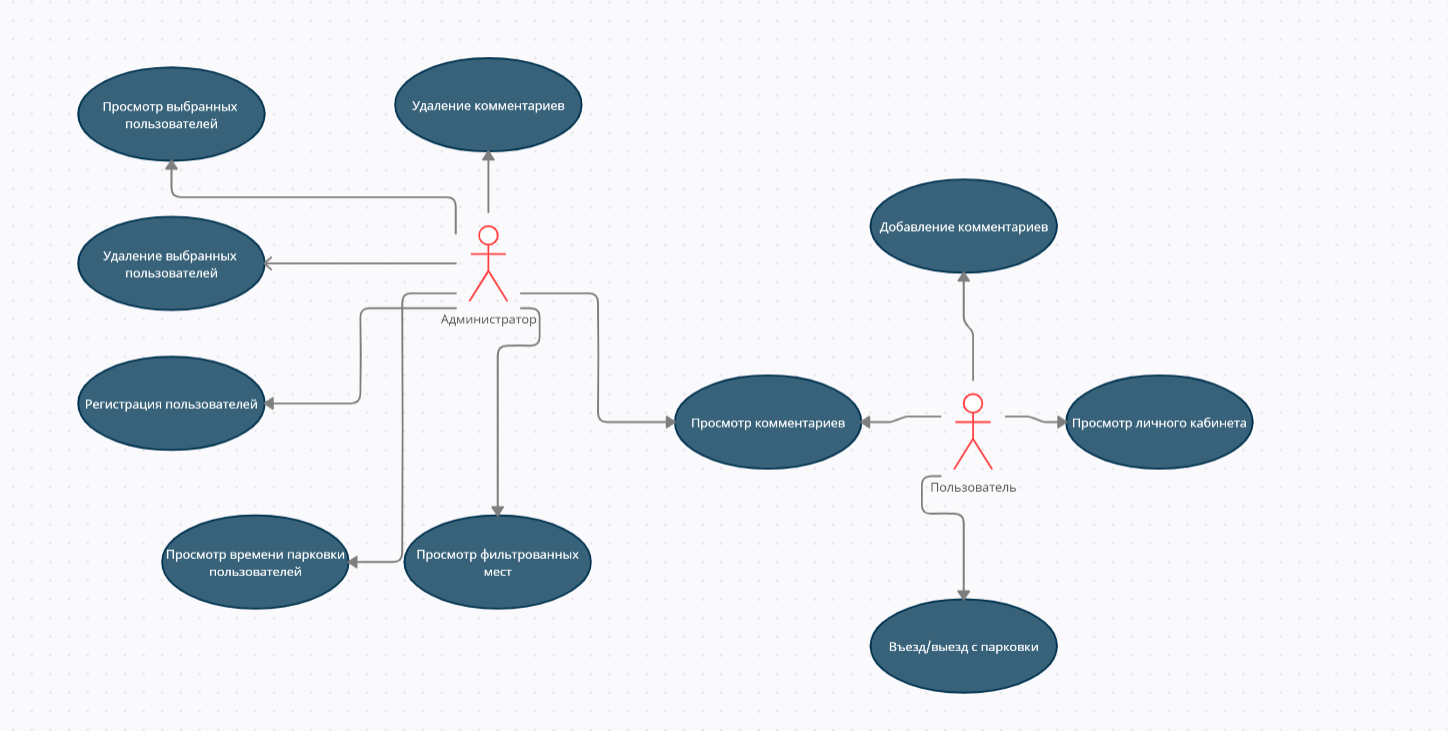
[6]. StackOverflow [Электронный ресурс] / Режим доступа: <https://stackoverflow.com>

[7]. Хабр [Электронный ресурс] / Режим доступа: <https://habr.com>

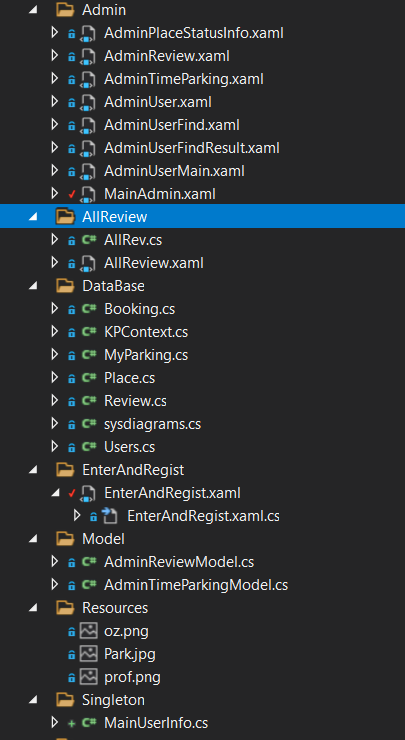
[8]. VkNet – Вконтакте API для C# [Электронные данные] / Режим доступа: <https://vknet.github.io/vk/>

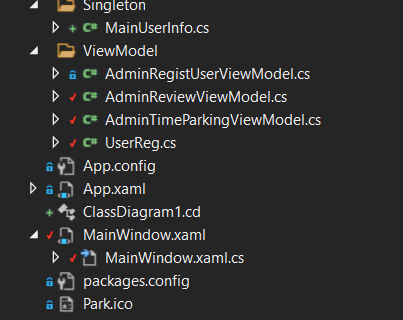
[9]. InstaSharp – API для взаимодействия с Instagram [электронный ресурс] / Режим доступа: <https://github.com/instasharp>

Приложение А

****

Приложение Б

****

****

Приложение В

****

Приложение Г

Синглтон для админа.

class MainUserInfo

{

public Info info { get; set; }

public void Show(string id, string firstname, string secondname, string email, string phone)

{

info = Info.Getinstance(id,firstname,secondname,email,phone);

}

}

class Info

{

public string UserId { get; private set; }

public string Firstname { get; private set; }

public string Secondname { get; private set; }

public string Email { get; private set; }

public string Phone { get; private set; }

public static Info instance;

protected Info(string id,string firstname,string secondname,string email,string phone)

{

UserId = id;

Firstname = firstname;

Secondname = secondname;

Email = email;

Phone = phone;

}

public static Info Getinstance(string id, string firstname, string secondname, string email, string phone)

{

if(instance == null)

{

instance = new Info(id, firstname, secondname, email, phone);

}

return instance;

}

}

**Приложение Д**

Регистрация и вход

private string firstname;

public string Firstname

{

get

{

return firstname;

}

set

{

firstname = value.Trim();

RaisePropertiesChanged("Firstname");

}

}

private string secondname;

public string Secondname

{

get

{

return secondname;

}

set

{

secondname = value.Trim();

RaisePropertiesChanged("Secondname");

}

}

private string mobile;

public string Mobile

{

get

{

return mobile;

}

set

{

mobile = value.Trim();

RaisePropertiesChanged("Mobile");

}

}

private string \_email;

public string Email

{

get

{

return \_email;

}

set

{

\_email = value.Trim();

RaisePropertiesChanged("Email");

}

}

private string \_password;

public string Password

{

get

{

return \_password;

}

set

{

\_password = value.Trim();

RaisePropertiesChanged("Password");

}

}

private string passwordAgn;

public string PasswordAgn

{

get

{

return passwordAgn;

}

set

{

passwordAgn = value.Trim();

RaisePropertiesChanged("PasswordAgn");

}

}

private string emailforEnter;

public string EmailforEnter

{

get

{

return emailforEnter;

}

set

{

emailforEnter = value.Trim();

RaisePropertiesChanged("EmailforEnter");

}

}

private string passwordforEnter;

public string PasswrdforEnter

{

get

{

return passwordforEnter;

}

set

{

passwordforEnter = value.Trim();

}

}

public void Enter()

{

using(KPContext kp = new KPContext())

{

int checkingenter = 0;

var forenter = kp.Database.SqlQuery<Users>($"select \* from Users where Users.email = '{emailforEnter}'");

foreach(var check in forenter)

{

if(check.email == emailforEnter && check.password == PasswrdforEnter && check.Admin == true)

{

checkingenter++;

MainAdmin mainAdmin = new MainAdmin();

MainUserInfo mainUserInfo = new MainUserInfo();

mainUserInfo.Show(check.UserId.ToString(), check.Firstname, check.Secondname, check.email, check.Mobile);

mainAdmin.Showing(mainUserInfo.info.Firstname);

mainAdmin.Show();

foreach (System.Windows.Window window in System.Windows.Application.Current.Windows)

{

if (window.DataContext == this)

{

window.Close();

}

}

}

if (check.email == emailforEnter && check.password == GetHashPassword(PasswrdforEnter) && check.Admin == false)

{

checkingenter++;

MainWindow main = new MainWindow();

main.UserInfo(check.UserId, check.Firstname, check.Secondname, check.email, check.Mobile);

main.Show();

foreach (System.Windows.Window window in System.Windows.Application.Current.Windows)

{

if (window.DataContext == this)

{

window.Close();

}

}

}

}

if (checkingenter == 0)

{

MessageBox.Show("Проверьте корректность введенных данных");

}

}

}

public void Regist()

{

if (ValidReg() == true)

{

using (KPContext kp = new KPContext())

{

Users user = new Users()

{

Secondname = secondname,

Firstname = firstname,

email = \_email,

Mobile = mobile,

password = GetHashPassword(\_password),

Admin = false

};

kp.Users.Add(user);

kp.SaveChanges();

}

MessageBox.Show("Регистрация прошла успешно");

Clear();

}

}

public ICommand regist => new DelegateCommand(Regist);

public ICommand enter => new DelegateCommand(Enter);

public bool ValidReg()

{

try

{

char[] Firstnamech = firstname.ToCharArray();

char[] Secondnamech = secondname.ToCharArray();

string regular\_mobile = "[+]{1}[0-9]{12}";

string regular\_email = "[@]{1}";

if (Regex.IsMatch(\_email, regular\_email) == false || \_email.Length > 100 || \_email.Length < 1)

{

throw new Exception("Email введен не корректно");

}

if (Regex.IsMatch(mobile, regular\_mobile, RegexOptions.IgnoreCase) == false)

{

throw new Exception("Номер телефона введен не корректно");

}

foreach (char i in Firstnamech)

{

if (char.IsLetter(i) == false || Firstnamech.Length > 50 || Firstnamech.Length < 1)

{

throw new Exception("Имя введено не корректно");

}

}

foreach (char i in Secondnamech)

{

if (char.IsLetter(i) == false || Secondnamech.Length > 50 || Secondnamech.Length < 1)

{

throw new Exception("Фамилия введена не корректно");

}

}

if(\_password != passwordAgn)

{

throw new Exception("Пароль не совпадает");

}

if(\_password.Length > 50 || \_password.Length < 6)

{

throw new Exception("Пароль введен не корректно");

}

if(Firstname.Length < 1 || Firstname.Length > 50)

{

throw new Exception("Имя введено не корректно");

}

if(Secondname.Length > 50 || Secondname.Length < 1)

{

throw new Exception("Фамилия введена не корректно");

}

using (KPContext kp = new KPContext())

{

var users1 = kp.Database.SqlQuery<Users>($"select \* from Users where Users.Mobile = '{mobile}'");

foreach (var usercheck in users1)

{

if (usercheck.Mobile != null)

{

MessageBox.Show("Телефон с таким номером уже зарегестрирован");

return false;

}

}

users1 = kp.Database.SqlQuery<Users>($"select \* from Users where Users.email = '{\_email}'");

foreach (var usercheck in users1)

{

if (usercheck.email != null)

{

MessageBox.Show("Такая почта уже зарегестрирована");

return false;

}

}

}

return true;

}

catch (Exception ex)

{

if(ex.Message == "Ссылка на объект не указывает на экземпляр объекта.")

{

MessageBox.Show("Проверьте корректность введенных данных");

return false;

}

MessageBox.Show(ex.Message);

return false;

}

}

private string GetHashPassword(string s)

{

//переводим строку в байт-массим

byte[] bytes = Encoding.Unicode.GetBytes(s);

//создаем объект для получения средст шифрования

MD5CryptoServiceProvider CSP =

new MD5CryptoServiceProvider();

//вычисляем хеш-представление в байтах

byte[] byteHash = CSP.ComputeHash(bytes);

string hash = string.Empty;

//формируем одну цельную строку из массива

foreach (byte b in byteHash)

{

hash += string.Format("{0:x2}", b);

}

return hash;

}