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 мобильной систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Современные технологии программирования мобильных систем»

Тема «Бронирование железнодорожных билетов»

Исполнитель

студент 2 курса группы 8 Друщиц Максим Валерьевич

Руководитель работы ассистент Панченко О. Л.

Курсовой проект защищён с оценкой

Председатель Пацей Н.В.

Минск 2021

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc9543690)

[1. Аналитический обзор литературы 4](#_Toc9543691)

[2. Анализ требований к программному средству и разработка функциональных требований 8](#_Toc9543692)

[3. Проектирование программного средства 11](#_Toc9543693)

[4. Создание (реализация) программного средства 17](#_Toc9543694)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 22](#_Toc9543695)

[6. Руководство по использованию Пользователю 2](#_Toc9543696)9

[7. Руководство по использованию Администратору 3](#_Toc9543696)4

[ЗАКЛЮЧЕНИЕ 3](#_Toc9543697)7

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 3](#_Toc9543698)8

[Приложение А 3](#_Toc9543699)9

[Приложение Б 3](#_Toc9543700)9

[Приложение В](#_Toc9543701) 40

[Приложение Г](#_Toc9543702) 41

[Приложение Д](#_Toc9543703) 42

[Приложение Е](#_Toc9543704) 42

[Приложение Ж 43](#_Toc9543705)

# ВВЕДЕНИЕ

Покупка билетов на поезда/самолёты/автобусы постепенно все больше и больше перемещается из оффлайн в онлайн-пространство, и это открывает практически безграничные перспективы как для компаний, так и для обычных людей.

Многим людям гораздо удобнее и практичнее выбирать комфортные места и покупать билеты в онлайн приложениях или сайтах.

Удобнее всего это делать через специальные приложения, которые помогают компаниям предоставлять более качественные услуги, а покупателям – комфортнее ими пользоваться. С помощью разрабатываемого программное средство можно забронировать билеты на любой поезд компании.

Программное средство «Бронирование железнодорожных билетов» – это программное средство, где каждый желающий клиент компании может выбрать необходимое ему место на любой поезд, предоставляемый компанией.

Также это приложение позволяет отслеживать все забронированные билеты, как текущие, так и истёкшие.

Приложение имеет 2 типа пользователей: администратор и пользователь. Они имеют разные полномочия. Пользователи могут следить за списком забронированных билетов, и, непосредственно бронировать необходимые билеты. Администратор имеет полные права. Они могут удалять любых пользователей, просматривать информацию о пользователях и их забронированных билетах.

Также реализована авторизация всех пользователей. Это значит, что перед тем, как забронировать билет на поезд, нужно будет пройти полную регистрацию. Хранение всех пользователей будет осуществлено в базе данных, и при авторизации или регистрации будет непосредственное обращение к ней. Также вся информация о забронированных билетах будет хранится в базе данных.

Реализована возможность редактирования аккаунта.

В ходе использования этой программы, пользователи смогут получать информацию о интересующих их билетах на поезд.

# 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://rw.by, т.к. он максимально приближен к моей задумке. Краткий обзор этого сервиса представлен ниже.

Для посетителей Rw.by доступен [каталог](http://catalog.onliner.by/) предоставляемых билетов на поезда. Вот так выглядит страница с выбором билета (рисунок 1.1).

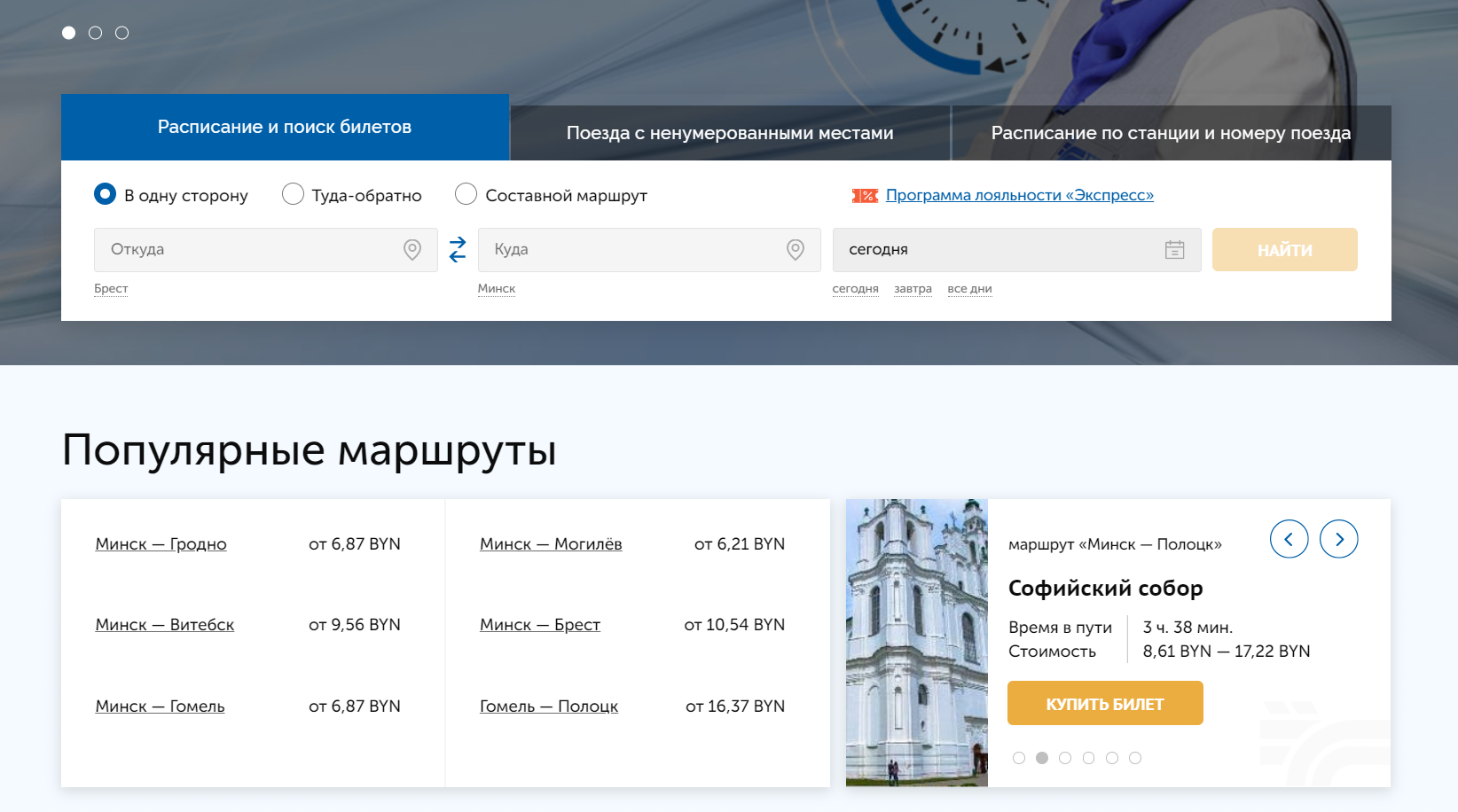


Рисунок 1.1 – раздел «Мобильные телефоны»

Во вкладке “Услуги пассажирам” отображается панель с выбором необходимого маршрута и даты отправления. Пользователя предлагается заполнить главные поля для покупки билета, выбор же поезда реализовано на другой странице, на которую пользователь попадает после нажатия кнопки “Найти” (рисунок 1.2).

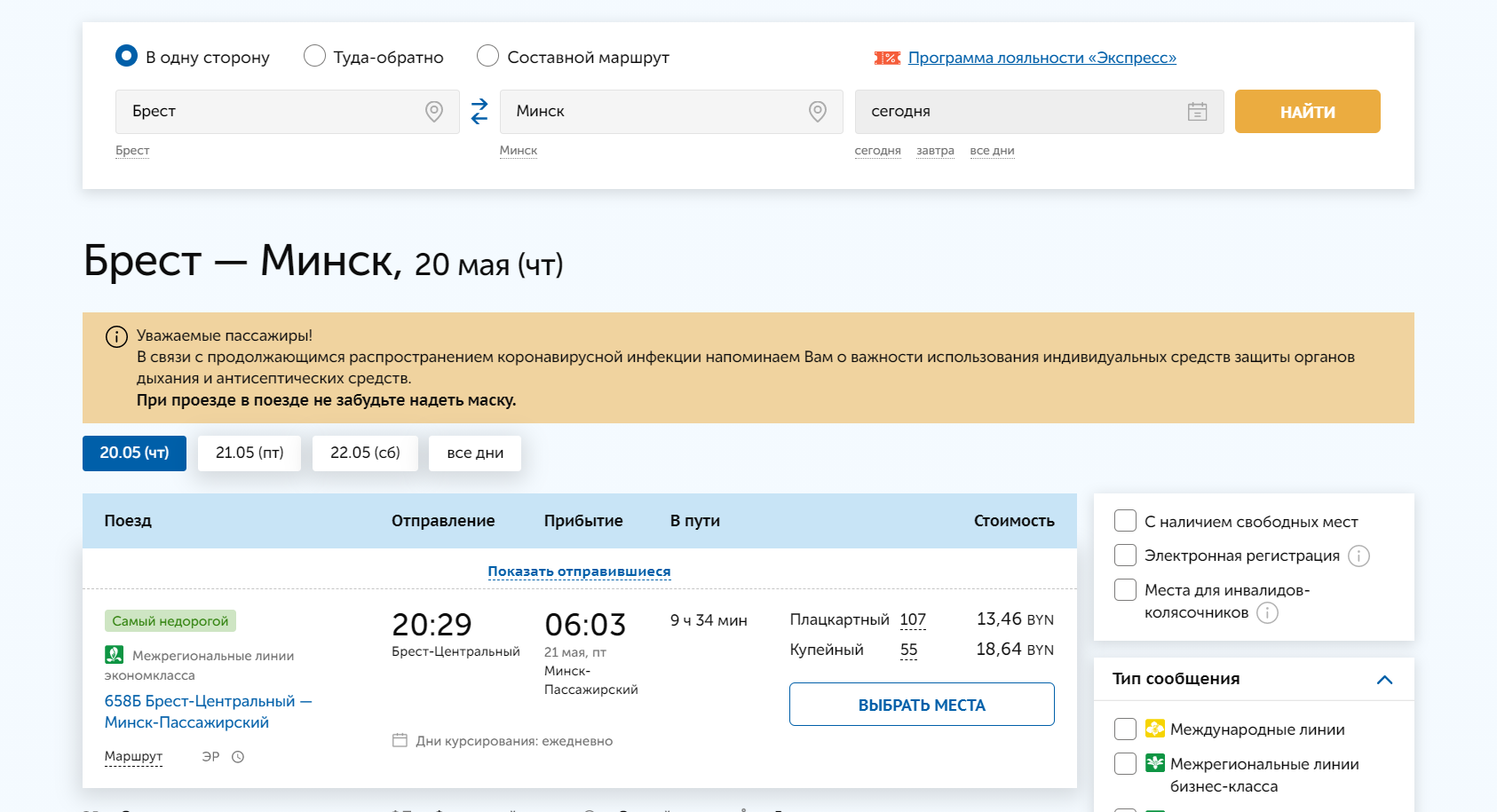


Рисунок 1.2 – выбор поезда

Тут пользователь может ознакомиться со всеми поездами, которые идут по заданному маршруту в выбранную дату. После выбора интересующего поезда пользователь попадает на следующую страницу, где уже непосредственно выбирает тип места, номер вагона и номер места

Если же вам нужно указать какой-то дополнительный параметр, например сразу оплатить постельное бельё, пользователь указывает такой параметр на этой странице. Дополнительные условия для поиска и всплывающие подсказки помогают в этом.

Здесь же пользователь делает выбирает типа места (плацкарт, купе, бизнес класс, эконом место). Далее пользователь выбирает номер вагона. В интерактивной схеме вагона пользователь ознакамливается с доступными местами. Выбирает понравившееся место и переходит в окно оплаты (рисунок 1.3).

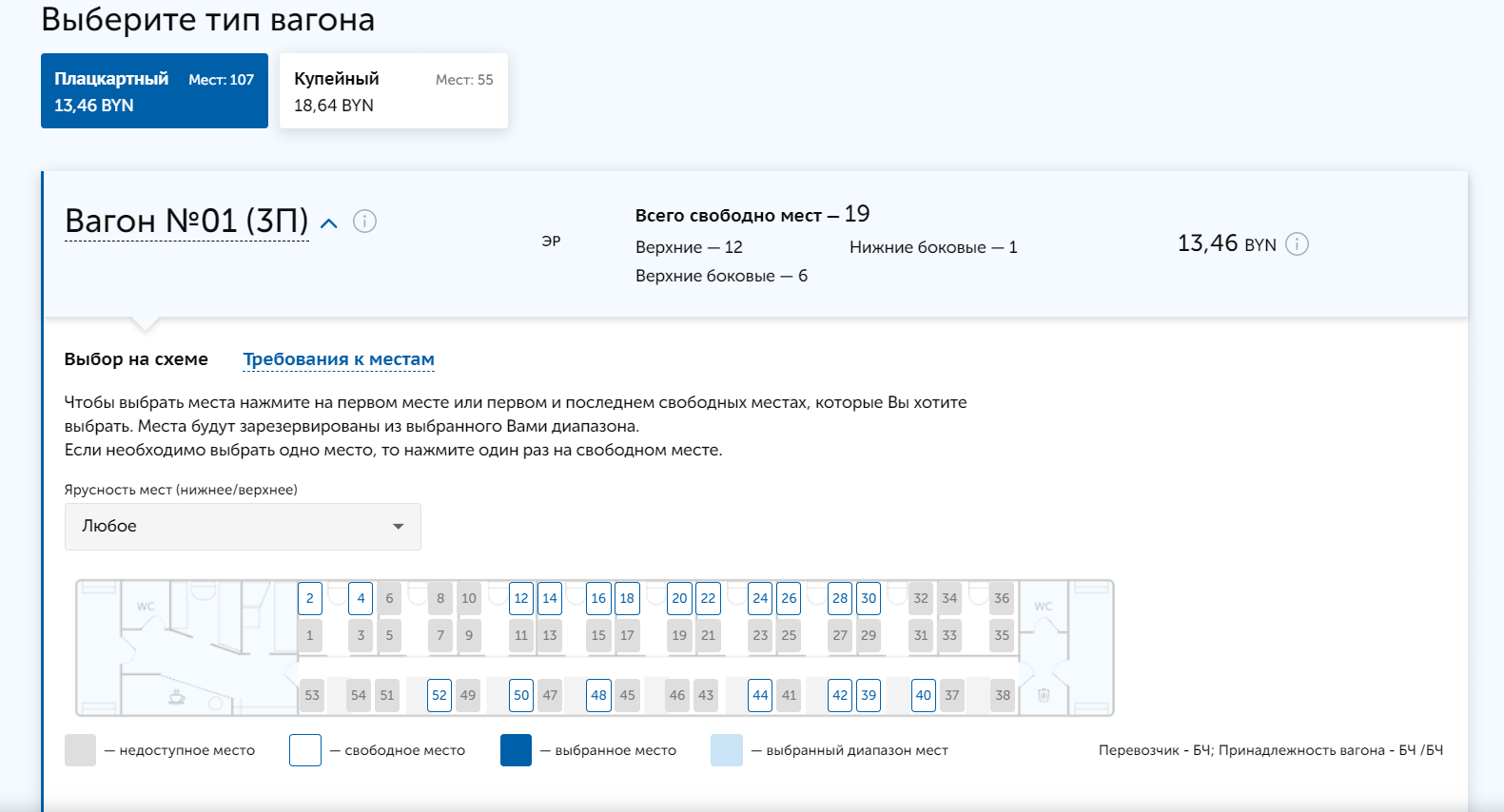


Рисунок 1.3 – Выбор параметров

После выполнения оплаты пользователь попадает на страницу выполненных заказов, где отображена вся информация о заказанном билете и где находятся все предыдущие билеты пользователя.

Я решил сделать проект на эту тему, так как не видел ещё desktop-версии подобного приложения для Windows.

Этот сайт является образцом для моего проекта (https://pass.rw.by/ru/).

# Анализ требований к программному средству и разработка функциональных требований

Задачей курсового проекта является разработка приложения, предназначенного просмотра информации доступных маршрутов и, непосредственно бронирования необходимого билета. Информацию о пользователе заполняется непосредственно с самого приложения и проверяется администратором. Бронирование билета осуществляется путём заполнения формы нового билета.

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 проектировании для разработки был выбран первый подход.

Приложение необходимо реализовать с использованием шаблона проектирования MVVM и с организацией взаимодействия с базой данных. В качестве клиентской части необходимо реализовать оконное приложение с использованием стандартных и иных библиотек. Приложение должно иметь возможность запускаться без использования интегрированных средств разработки.

Для разработанного продукта необходимо предусмотреть следующие задачи:

* Главное поле для отображения данных;
* Интуитивно понятный интерфейс;
* Механизм поиска пользователей по их логину;
* Возможность просмотра информации о маршруте.

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:

* Идентификация пользователей
* Добавление новой брони на билет
* Выбор типа места, вагона, номера места
* Вход в приложение, используя социальные сети
* Хранение и отображение забронированных билетов
* Управление своими забронированными билетами (редактирование, удаление)

Администратор может всё то же самое, и плюс ко всему:

* Проверка поданных пользователей
* Удаление пользователей
* Просмотр информации о пользователях и их забронированных билетах

Наглядно это видно на UML-схеме, изображённой на рисунке 2.1 (Приложение А).

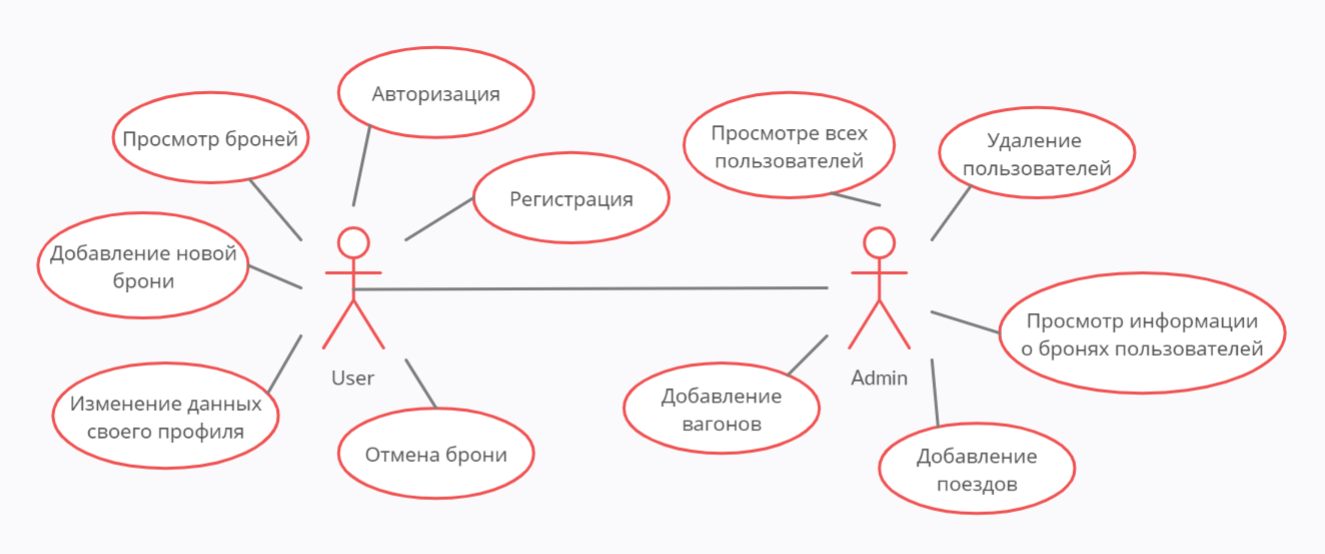


Рисунок 2.1 – «UML-диаграмма возможностей»

# Проектирование программного средства

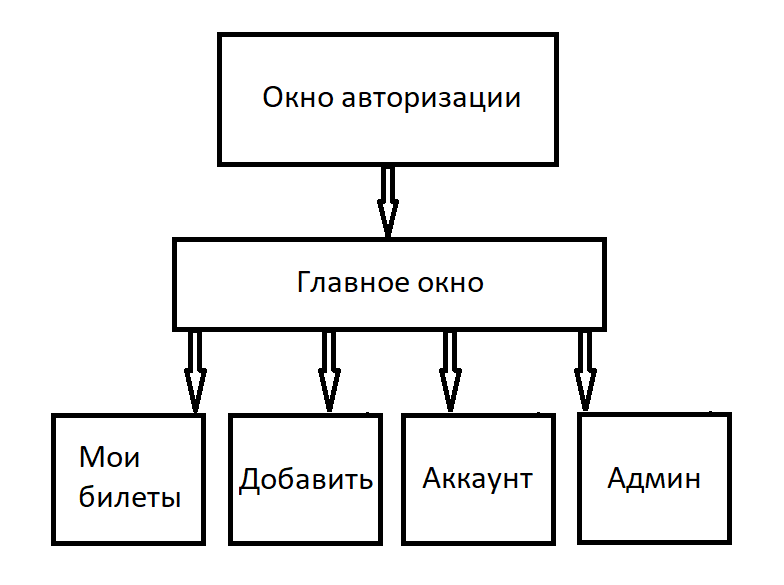


Рисунок 3.1 – «Авторизация и главное окно»

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.

Нажав на соответствующие кнопки в окне авторизации, у вас появится окно, где нужно будет ввести данные для входа в учётную запись, через которую производится вход.

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.

В главном окне работа идёт в страничном и оконных режимах.

Рассмотрим каждую страницу в отдельности.

Страница «Мои билеты» (рисунок 3.2).

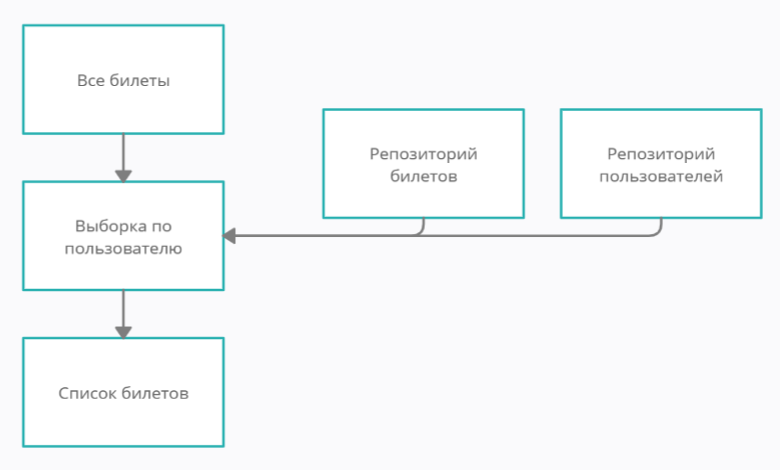


Рисунок 3.2 – «Мои билеты»

Эта страница содержит список всех броней билетов.

Также она содержит механизм сортировки билетов по определённым критериям.

Осуществить поиск можно по таким критериям:

* Название маршрута
* Дата бронирования билета

На основе критериев сортировки, формируется результирующий набор. При нажатии кнопку “подробнее” пользователь видит более подробную информацию о забронированном билете. При нажатии кнопку “удалить” пользователь имеет возможность отменить бронь выбранного билета.

Страница «Добавить» (рисунок 3.3).

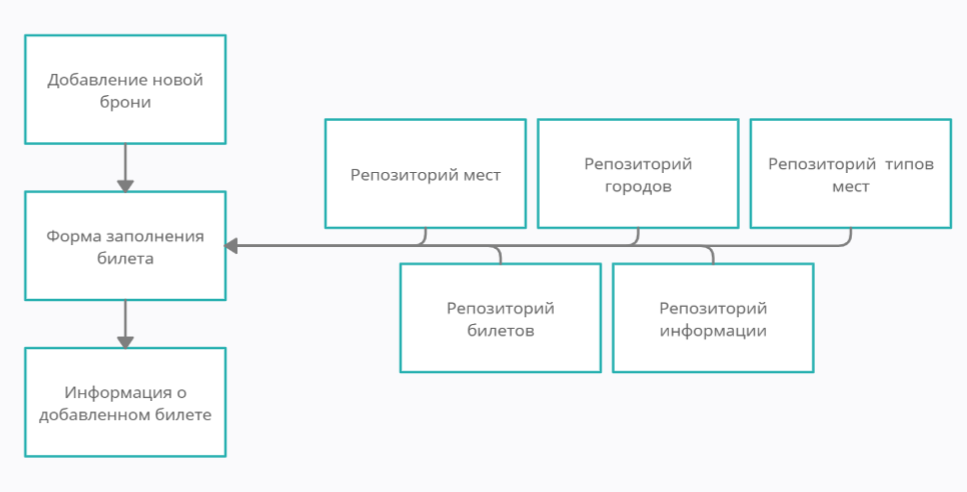


Рисунок 3.3 – «Добавить»

На рисунке 3.3 показана схема работы страницы «Добавить».

На этой странице осуществляется просмотр информации о доступных маршрутах и, непосредственно, бронирование конкретного билета.

Переход только что зарегистрированного пользователя возможен только после завершения полной регистрации в окне “Аккаунт”.

Добавление брони осуществляется в самом окне. Здесь пользователь выбирает из предложенных город отправления и город прибытия. После чего сразу выводится цена билета по заданному маршруту. Далее пользователь выбирает дату отправления и время отправления из предложенных. Выбирает тип места (Эконом, бизнес класс, плацкарт), номер вагона и номер места из доступных. Все данных предоставляются в выпадающих списках, реализованных при помощи ComboBox. После выбора всех параметров билета пользователь может нажать кнопу “Добавить”.

После добавление новой брони, открывается окно, в котором выводится вся информация о выбранном билете.

Добавление билета невозможно, пока пользователь не заполнит все поля, об этом ему сообщает строка подсказок внизу страницы.

На странице есть кнопка “Очистить”, чтобы пользователю не пришлось менять данные, если он передумал бронировать конкретный маршрут, а задать его заново.

Страница «Аккаунт» (рисунок 3.4).

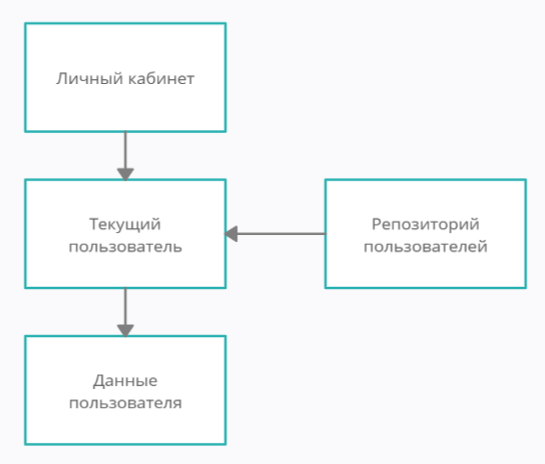


Рисунок 3.4 – «Аккаунт»

На рисунке 3.4 показана схема страницы «Личный кабинет».

На этой странице можно редактировать свои данные, такие как:

* Имя
* Фамилия
* Отчество
* Mail
* Телефон
* Дата рождения
* Номер паспорта
* Пол

На этой странице есть 2 кнопки: «Сохранить», «Удалить».

При нажатии на 1-ю кнопку происходит обновление информации о текущем пользователе в базе данных.

При на нажатии на 2-ю кнопку – удаление всех броней текущего пользователя, последующее удаление самого профиля и перенаправление на окно авторизации.

Заполнения данных на этой странице является обязательным перед тем как пользователю станет доступно бронирование билета.

Страница «Админ» (рисунок 3.3).

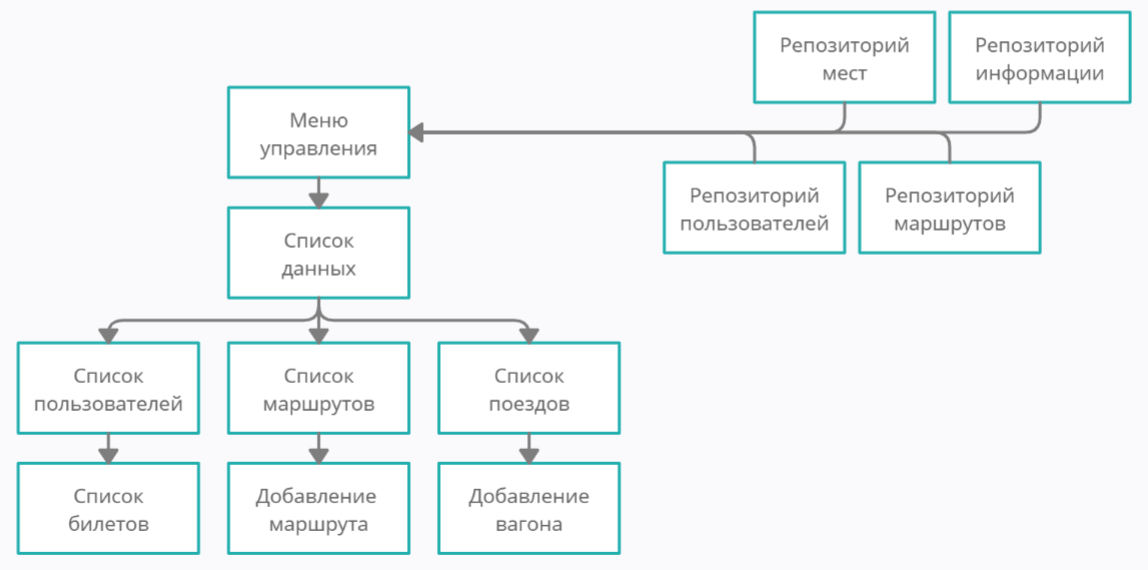


Рисунок 3.5 – «Админ»

В этом окне у нас производится администрирование нашего приложения.

Всего есть 2 типа привилегий:

* Пользователь
* Администратор

Пользователь не имеет доступ к этой странице. При попытке зайти в это меню, у него будет выведено сообщение о недостатке прав для доступа к этой странице.

Администратор видит всех зарегистрированных в приложении пользователей. Может просматривать информацию о пользователе, а также его бронях. Администратор может удалить любого пользователя и все его брони на текущий момент. Поддерживает работу с поездами и вагонами (местами).

Осуществлена сортировка по Именам, Фамилиям, и email адресам пользователей. На нижней панель администратор может выполнять поиск пользователя по адресу его электронной почты, она же есть логин пользователя.

Структура проекта представлена в приложении В.

В таблице 3.7 описана структура проекта по пакетам классов.

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

Таблица 3.7 – Описание структурных пакетов проекта

|  |  |
| --- | --- |
| Имя пакета | Описание |
| DB | Здесь описаны модели, с которыми происходит вся работа в приложении:   * Город * Место * Тип места * Билет * Пользователь * Маршрут |
| Behaviors | Класс для обработки введённых данных в поле “пароль” |
| Commands | Содержит классы и интерфейсы для работы с данными БД. Они являются связующим элементом между данными в БД и теми данными, с которыми работает приложение |
| Styles | Файлы стилей |
| View | Содержит все 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 |

Структуры данных, а также их зависимости друг от друга представлены в приложении Б.

Как показано в приложении Б, в вершине иерархии стоят класс Ticket, который характеризует брони билета соответственно. Класс City характеризует регион, из которого отправляется и в который пребывает поезд. Класс Seat характеризует места в поезде, в неё же хранится информация о доступных и занятых местах. Класс Seat\_type характеризует тип места в поезде (Эконом класс, бизнес класс, Плацкарт). Класс Seat связан с классом Seat\_type по полю seat\_id и id соответственно. Класс Voyage представляет собой маршрут следования поезда. User, характеризует пользователя с любым уровнем доступа. Ticket связан с User по полю client\_id и id соответственно, с Voyage по voyage\_id и id, с Seat по полям seat\_id и id. Voyage связан с City по полям id\_city\_\_\_of\_departure, id\_city\_\_\_of\_arrival и id соответственно. Диаграмма классов показана в приложении Г.

# Создание (реализация) программного средства

При создании приложения использовался паттерн проектирование MVVM. Он заключается разделении представления от бизнес логики. Это достигается за счёт ввода новой логической конструкции ViewModel. ViewModel связывает представление и бизнес логику приложения.

Пакетная структура проекта показана на рисунке 4.1

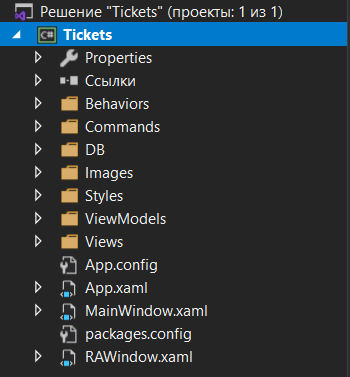


Рисунок 4.1 – «Структура пакетов проекта»

В DB 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.

Нередко все сущности (модель) наследует интерфейс INotifyPropertyChanged или INotifyCollectionChanged, которые позволяют отлавливать изменения и правильно на их реагировать и сохранять.

Для того, чтобы реализовать интерфейс INotifyPropertyChanged, нужно объявить свойство PropertyChanged и метод OnPropertyChanged().

Для того, чтобы начать отслеживание свойства, необходимо вызывать метод OnPropertyChanged(“Имя свойства”) как показано в приложении Д.

Для удобной работы с данными, используется паттерн Repository. Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой. На рисунке 4.2 показана структура.

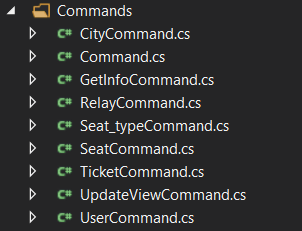


Рисунок 4.2 – «Паттерн Repository»

Интерфейсы IUserInterface, ITicketInterface, ISeatInterface, ISeatTypeInterface, IGetInfoCommand имплицируются в соответствующих классах и указывают: какие методы (функционал) должен иметь этот репозиторий. Репозитории показаны в приложении Е.

Интерфейсы репозиториев также показаны в приложении Е.

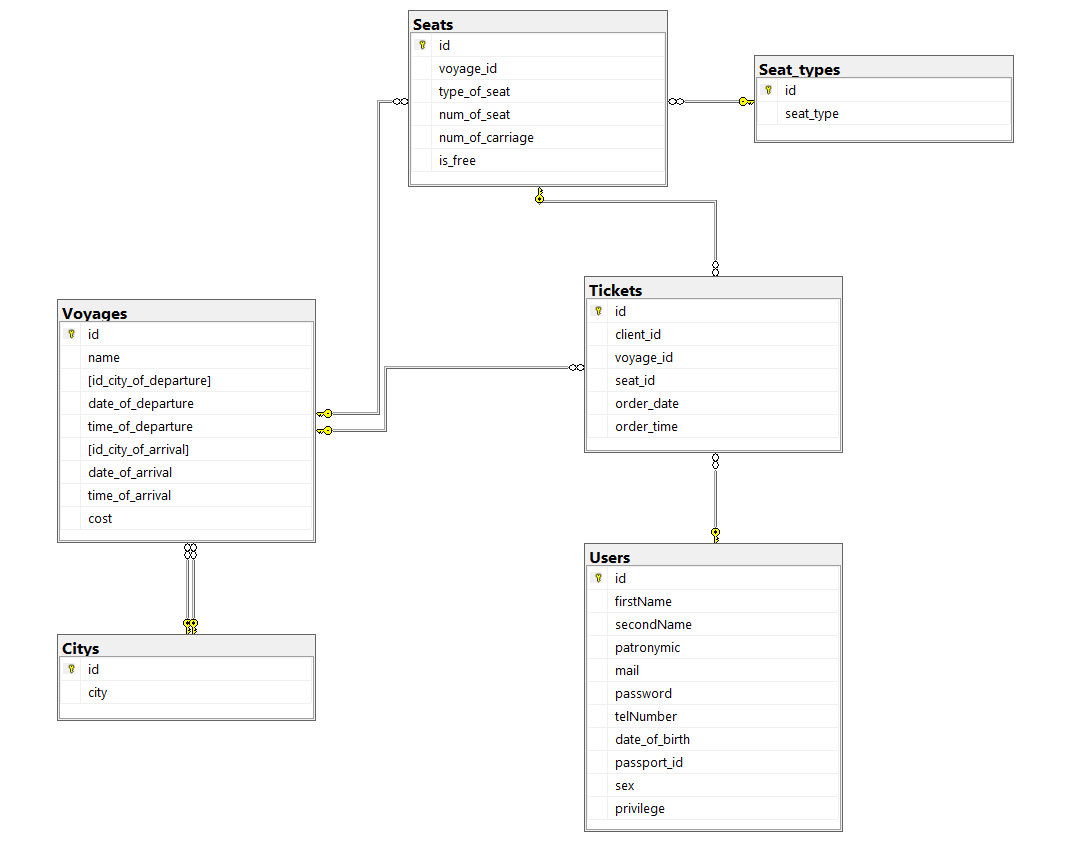
В проекте используется паттерн MVVM, а это значит, что должны присутствовать ViewModels, которые связывают View и Model. Во ViewModel информация обрабатываться и выводится/записываться в нужном виде. Структура пакета ViewModel представлена в приложении К.

В приложении И показана структура пакета View, в котором хранятся файлы представлений на языке разметки xaml, а также соответствующие им файлы на языке C#. В этом пакете находятся все «окна» и «страницы», которые мы видим, используя приложение. Назначение каждого файла описано в таблице ниже (таблица 4.4)

Таблица 4.4 – «Описание файлов View»

|  |  |
| --- | --- |
| **Название** | **Краткое описание** |
| AddTicket.xaml (UserControl) | Страница бронирования билета, в котором заполняются поля дальнейшего заказа. |
| Admin.xaml (UserControl) | Страница администратора для просмотра информации о пользователях и управления ими |
| Account.xaml (UserControl) | Страница профиля пользователя где он может редактировать свои данные |
| AllAnnouncement.xaml (страница) | Страница, отображающая все актуальные объявления |
| DialogWindowView.xaml (Window) | Окно, в котором нужно выбрать вариант ответа «Да» или «Нет» на заданный там вопрос |
| LoginView.xaml(UserControl) | Страница, где пользователь заполняет поля для входа в систему |
| RegistrathionView.xaml(UserControl) | Страница, где пользователь заполняет поля для первичной регистрации |
| MainWindow.xaml (Window) | Главное окно приложения |
| InfoWindow.xaml (Window) | Окно для закрепления объявления или его быстрого просмотра |
| RAWindow.xaml (Window) | Окно запуска приложения где пользователь регистрируется либо авторизуется |
| MyTicketsWindow.xaml (Window) | Основное окно для просмотра информации о забронированных билетах |

На рисунке 4.5 изображена диаграмма таблиц со связями в базе данных.

Рисунок 4.5 – «Диаграмма таблиц»

Описание таблицы Tickets (таблица 4.6)

Таблица 4.6 – «описание Voyages»

|  |  |
| --- | --- |
| id | Идентификатор |
| name | Название маршрута |
| city\_of\_departure | Id города отправления |
| date\_of\_departure | Дата отправления |
| time\_of\_departure | Время отправления |
| city\_of\_arrival | Id города прибытия |
| date\_of\_ arrival | Дата прибытия |
| time\_of\_ arrival | Время прибытия |
| cost | Цена |

Таблица Voyages редактируется Администратором приложения через созданное программное средство – не работая с базой данных напрямую.

Описание таблицы Tickets (таблица 4.7)

Таблица 4.7 – «описание Tickets»

|  |  |
| --- | --- |
| id | Идентификатор |
| client\_id | Id клиента |
| voyage\_id | Id маршрута |
| seat\_id | Id места |
| order\_date | Дата заказа |
| order\_time | Время заказа |

Таблица Tickets заполняется пользователями внутри приложения по мере добавления новых броней на билеты.

Описание Seats (таблица 4.8)

Таблица 4.8 – «описание Seats»

|  |  |
| --- | --- |
| Id | Идентификатор |
| voyage\_id | Id маршрута |
| type\_of\_seat | Тип места |
| num\_of\_seat | Номер места |
| num\_of\_carriage | Номер вагона |
| If\_free | Свободно/занято место |

Таблица Seats заполняется Администратором приложения внутри программного средства в разделе “Администратор”.

Описание таблицы Citys (таблица 4.9)

Таблица 4.9 – «описание Citys»

|  |  |
| --- | --- |
| id | Идентификатор |
| city | Название города |

Описание таблицы Seat\_types (таблица 4.10)

Таблица 4.10 – «описание Seat\_types»

|  |  |
| --- | --- |
| id | Идентификатор |
| seat\_type | Тип места |

Таблицы Citys и Seat\_types являются “вшитыми” в базу данных, то есть заполняется при установке приложения и далее редактируются только через редактор базы данных. Хотя такое редактирование в принципе не было запланировано при создании программного средства.

Описание Users (таблица 4.11)

Таблица 4.11 – «описание Users»

|  |  |
| --- | --- |
| Id | Идентификатор |
| firstName | Имя |
| secondName | Фамилия |
| pathonymic | Отчество |
| password | Хеш пароля |
| telNumber | Номер телефона |
| Mail | Email адрес |
| date\_of\_birth | Дата рождения |
| privilege | Привилегии |
| Sex | Пол |
| passport\_id | Номер паспорта |

Таблица Users редактируется авторизованным пользователем внутри приложения, но редактируется как самим пользователем, удаление аккаунта, так и Администратором приложения – удаление любого зарегистрированного пользователя.

# 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 после нажатия на кнопку “Войти” приведена на рисунке 5.1

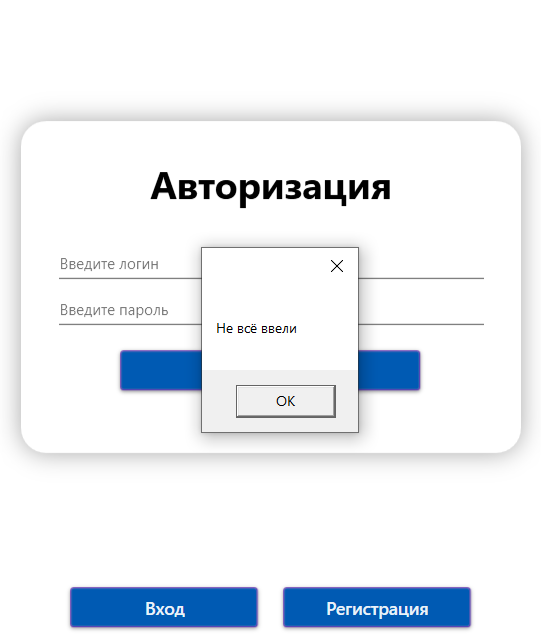


Рисунок 5.1 – «Ошибка корректности ввода»

При неверных данных – рисунок 5.2

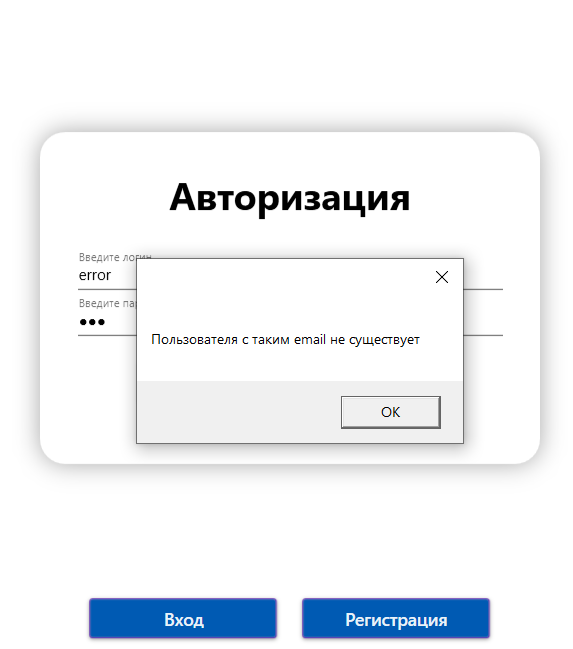


Рисунок 5.2 – «Сценарий, с неверными данными»

Также возможен такие случаи:

* Пользователь хочет зарегистрироваться, при этом не указав никаких данных (рисунок 5.3)
* Пользователь, когда осуществляет ввод паролей, указывает несовпадающие пароли (рисунок 5.4)

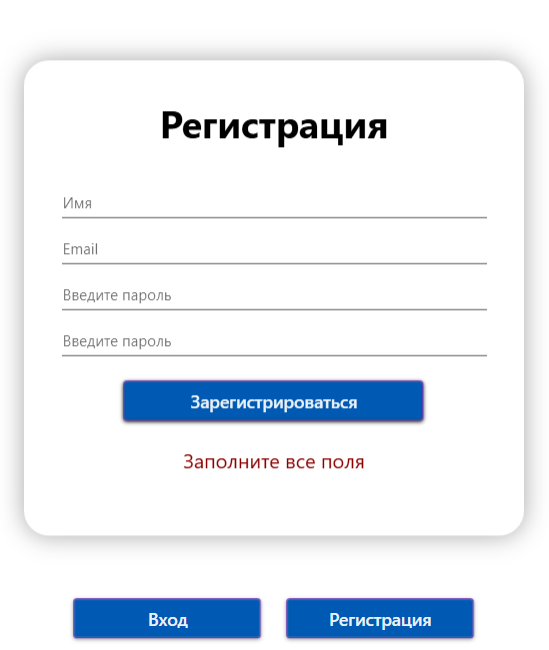


Рисунок 5.3 – «Отсутствие данных»

Как видно на рисунке 5.3, кнопка регистрации просто не будет активна, если не введены данные, необходимые для регистрации.

При несовпадении паролей будет точно такая ситуация – неактивная кнопка «Регистрация»

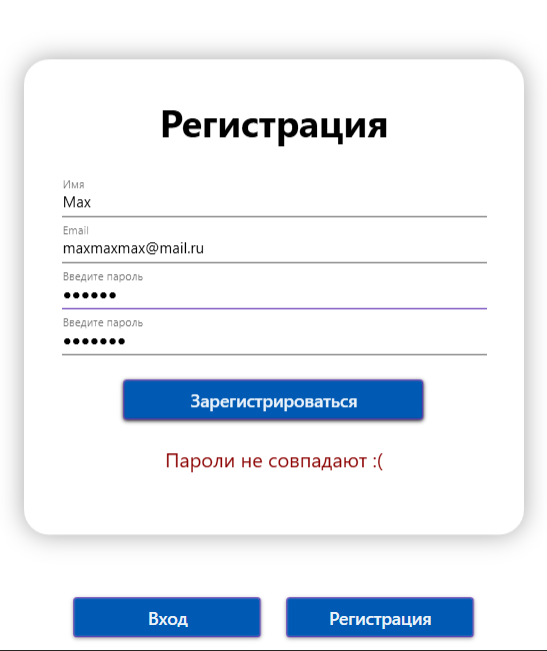


Рисунок 5.4 – «сценарий, при несовпадении паролей»

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.

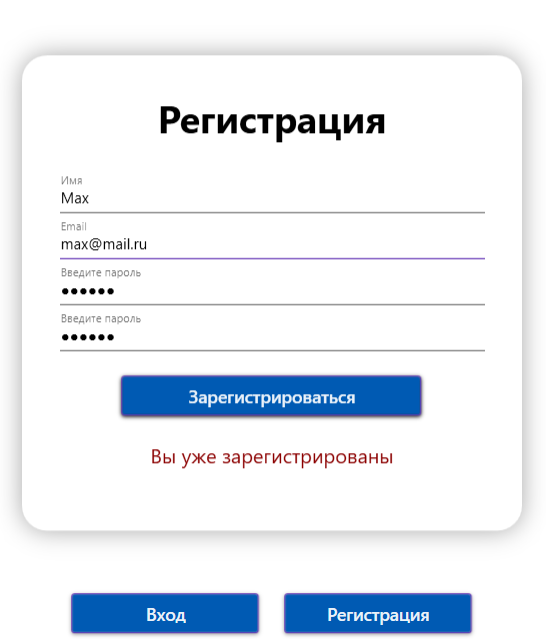


Рисунок 5.5 – «Регистрация пользователя с занятым логином»

Также может возникать такая ситуация, когда пользователь, который не имеет достаточные привилегии, хочет зайти в меню управления. Ему не даст туда зайти ограничение по привилегиям. Пример показан на рисунке 5.6

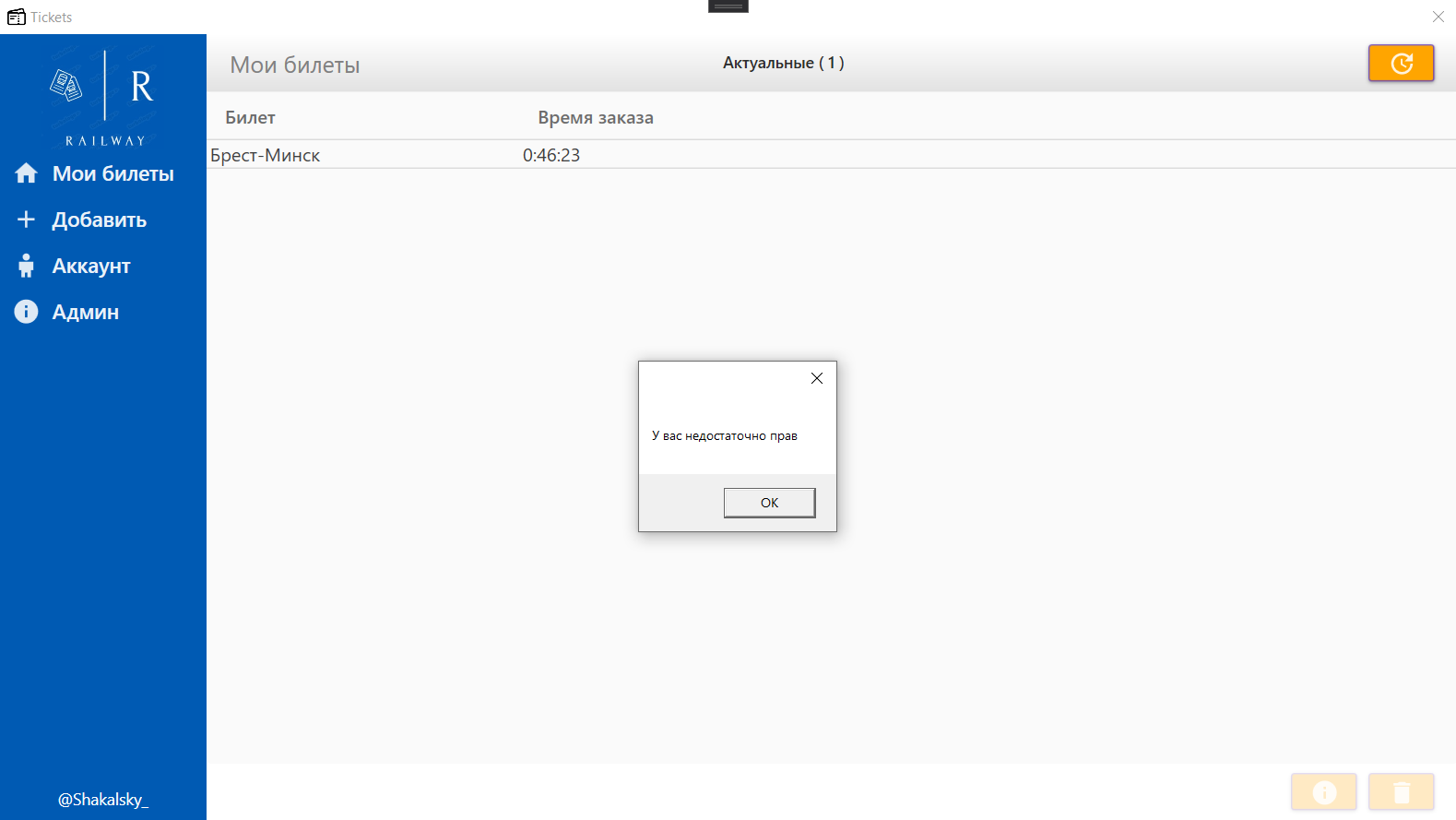


Рисунок 5.6 – «ограничение привилегий»

При разработке программного средства задумывалось, что оно будет иметь одного администратора, который регистрируется в базе данных пользователей при первом запуске приложения. То есть смена роли, при желании, может быть изменена только вручную при помощи редактора базы данных.

# Руководство по использованию пользователю

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.

Нажав на соответствующие кнопки в окне авторизации, у вас появится окно, где нужно будет ввести данные для входа в учётную запись, через которую производится вход.

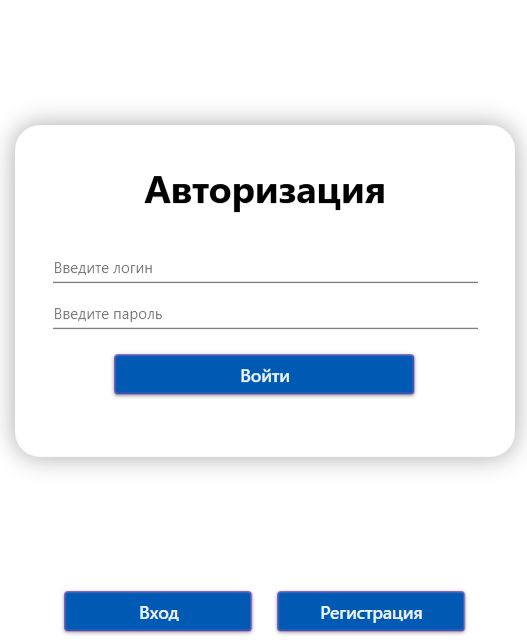


Рисунок 6.1 – «Окно авторизации»

После успешного входа, мы попадаем в главное окно c текущими бронями пользователя, которое изображено на рисунке 6.2.

В этом окне мы может по нескольким критериям пользователь может отсортировать свои брони. Эти критерии находятся вверху страницы.

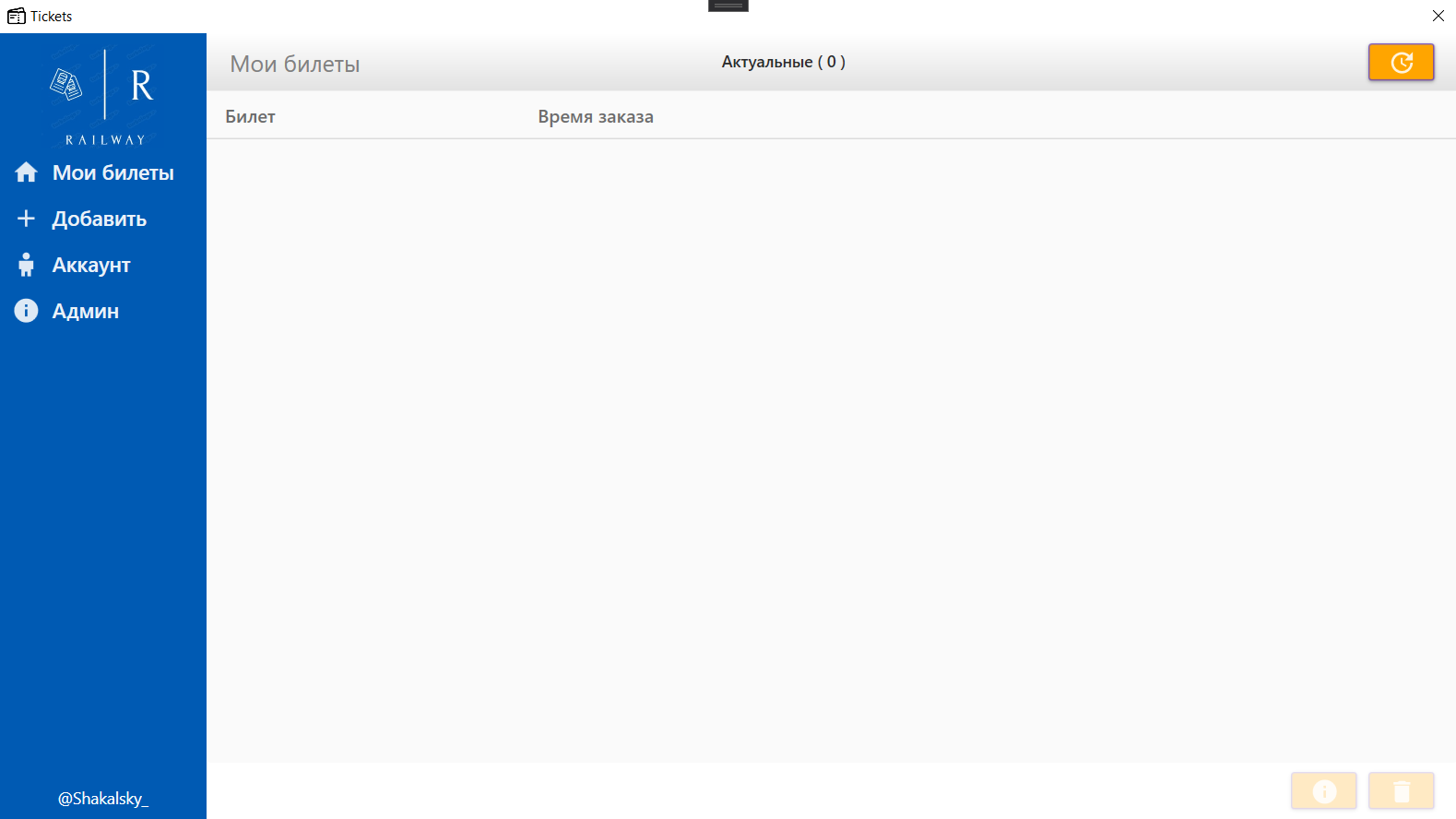


Рисунок 6.2 – «Мои билеты»

На странице размещено три кнопки: “Обновить”,” Информация” и ”Удалить”.

После добавления нового билета от отобразится в данном окне.

Если по какой-то причине забронированный билет не отображается у пользователя на странице билеты, он может обновить таблицу броней.

Кнопка “Информация” показывает в новом окне подробную информацию о выбранном билете.

При нажатии на кнопку “Удалить” пользователь может отменить бронь выбранного билета.

Ниже показан рисунок 6.3, на котором показана страница «Добавить».

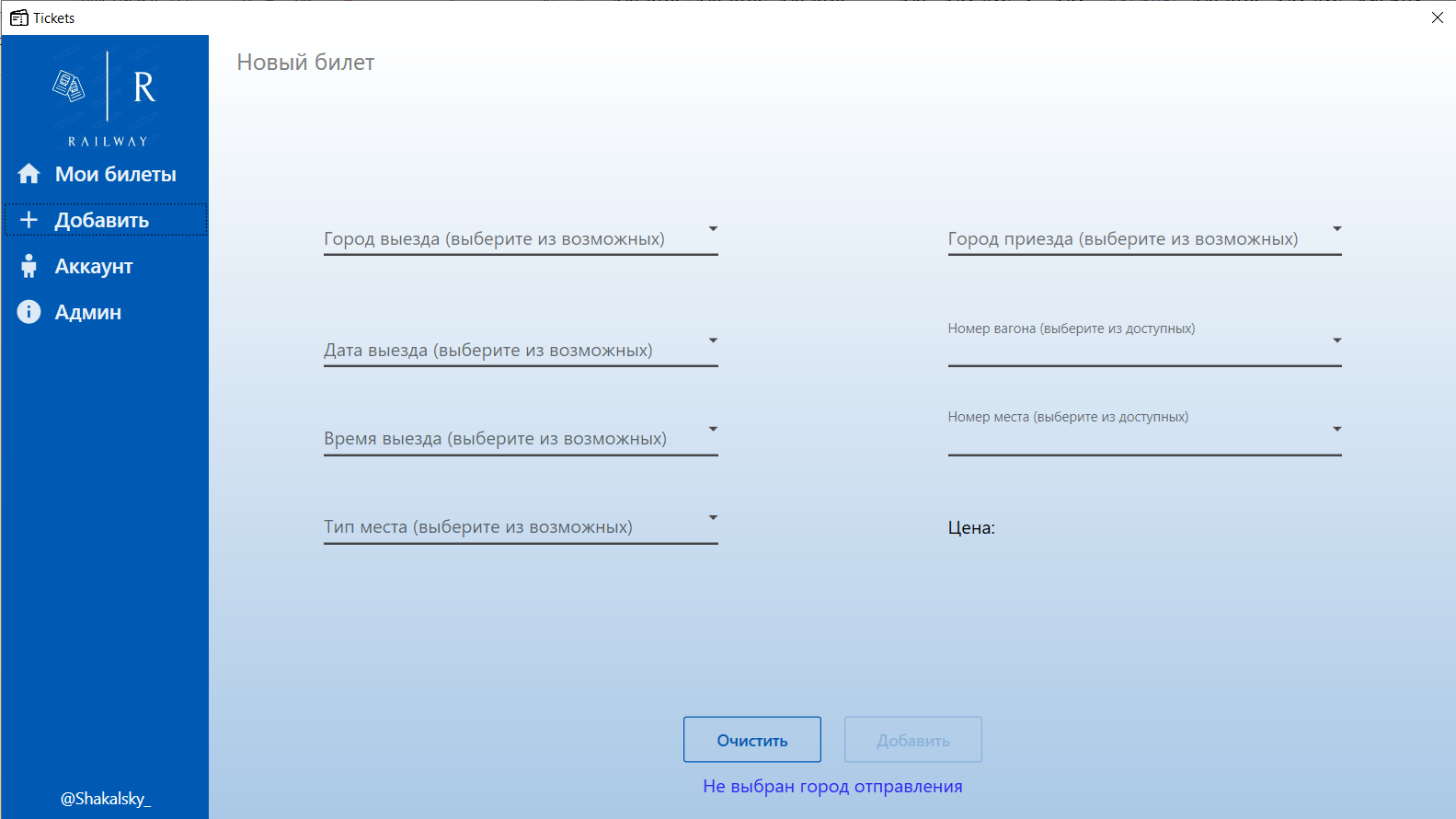


Рисунок 6.3 – «Страница Добавить»

По мере заполнения ключевых полей информация, данные в других полях подстраивается под выбор пользователя (Приложение Д). После выбора точки отправления и прибытия пользователю сообщается цена билета. Подсказки внизу страницу помогают новым пользователем с добавлением брони.

Ниже показана окно, появляющееся после успешного добавления брони на билет (рисунок 6.4)

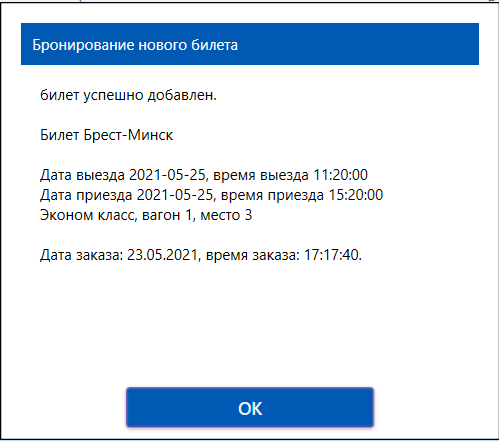


Рисунок 6.4 – «Информации о новой брони»

Также на странице «Мои» есть кнопка “Удалить”, нажав на которую будет сброшена текущая информация о новой брони.

Для добавления новой брони пользователю необходимо завершить полную регистрацию в приложении, о чём его оповещает окно подсказка (рисунок 6.5)

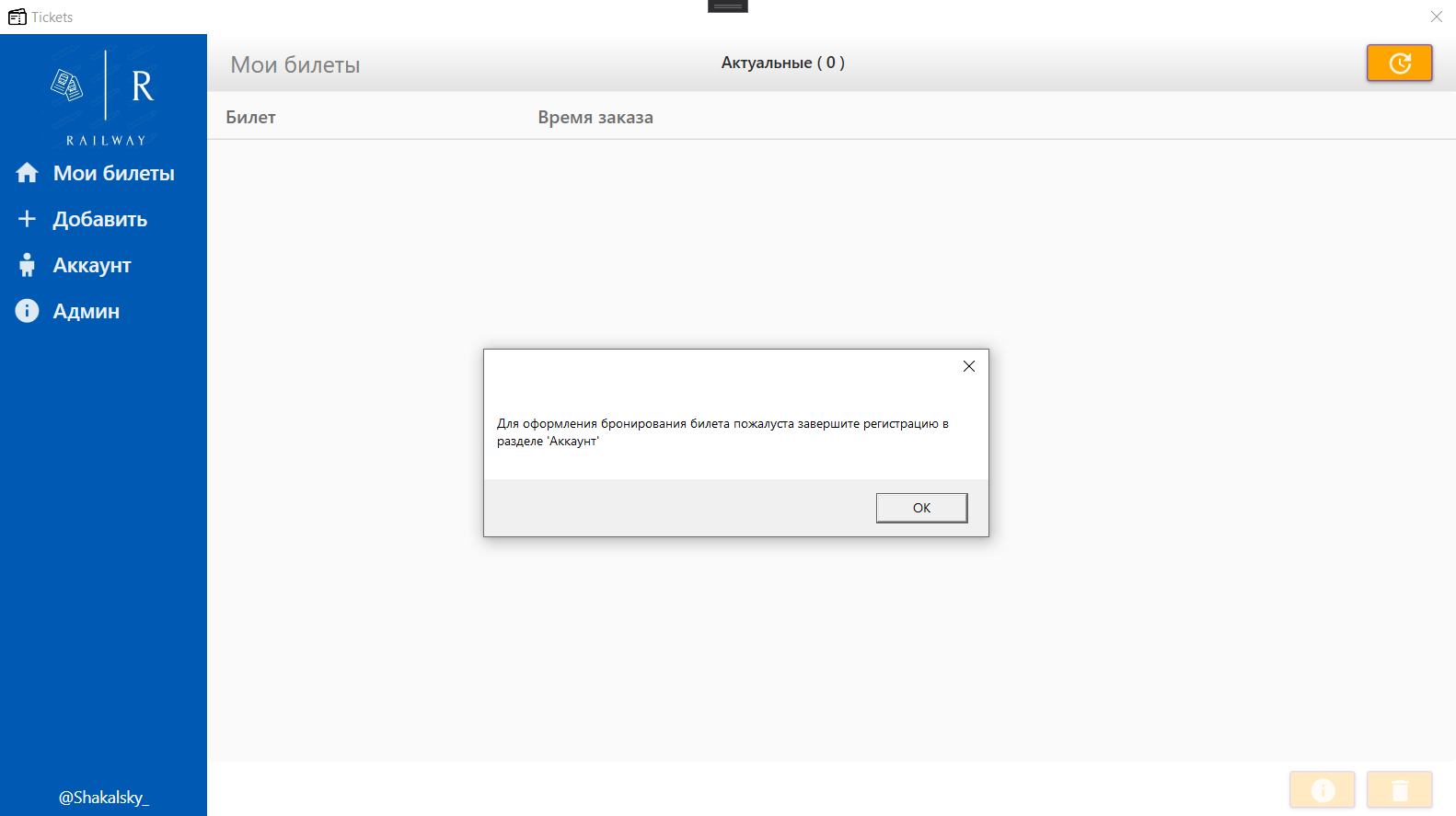


Рисунок 6.5 – «Подсказка пользователю»

При переходе на страницу «Аккаунт» мы попадаем на страницу, в которой отображены наши личные данные, введённые в процессе первичной регистрации, которые можем изменять. Доступные для заполнения поля: Имя, Фамилия, Отчество, Номер мобильного телефона, Дата рождения и Номер паспорта.

Ниже, на рисунке 6.6, представлена страница «Аккаунт».

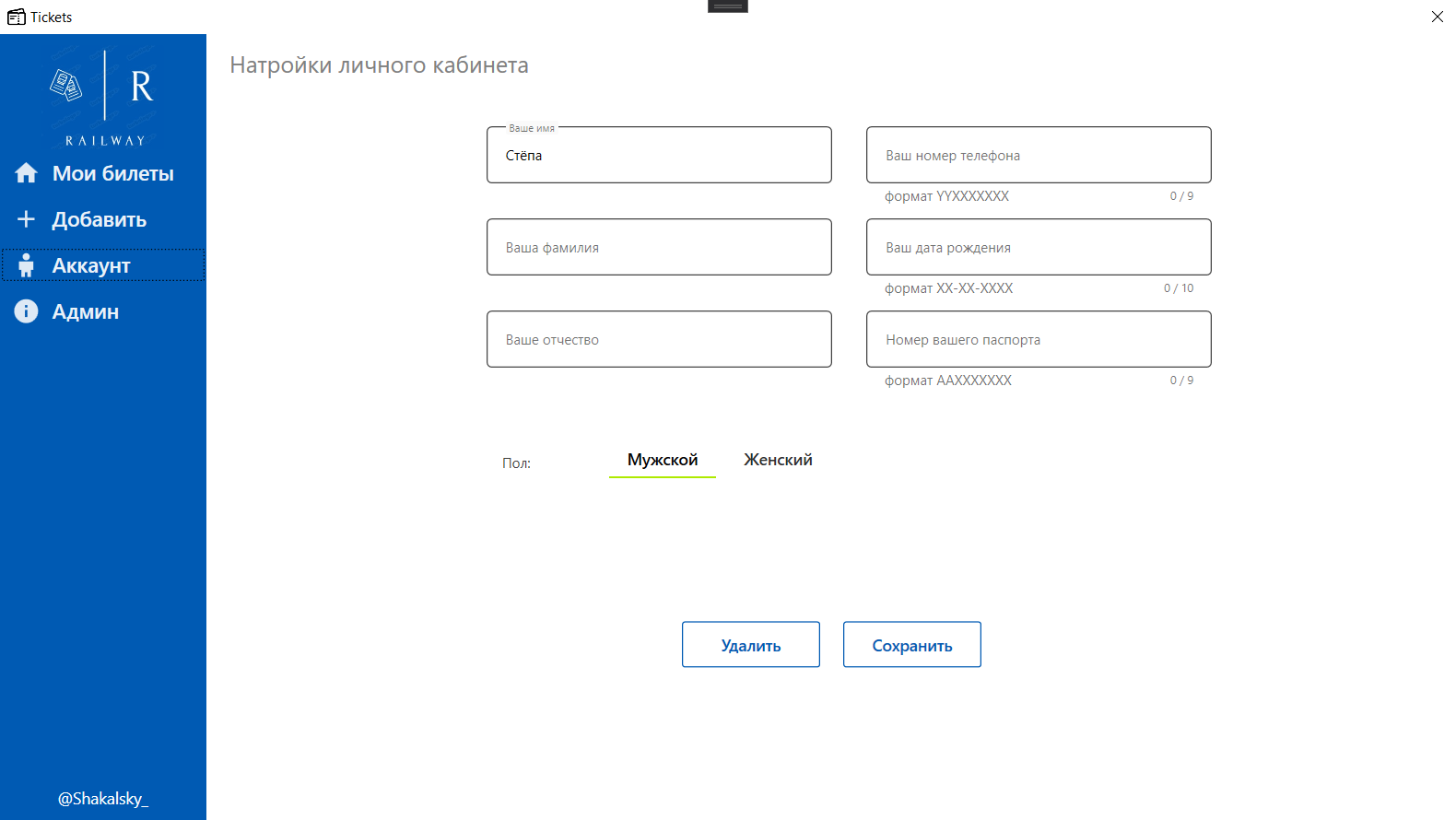


Рисунок 6.6 – «Страница Аккаунт»

# Руководство по использованию Администратору

После авторизации в приложении на роли Администратора, пользователю становится доступной вкладка меню на главной странице “Админ”. В открывшемся окне Администратору доступно 3 вкладки: Пользователи, Маршруты и Вагоны.

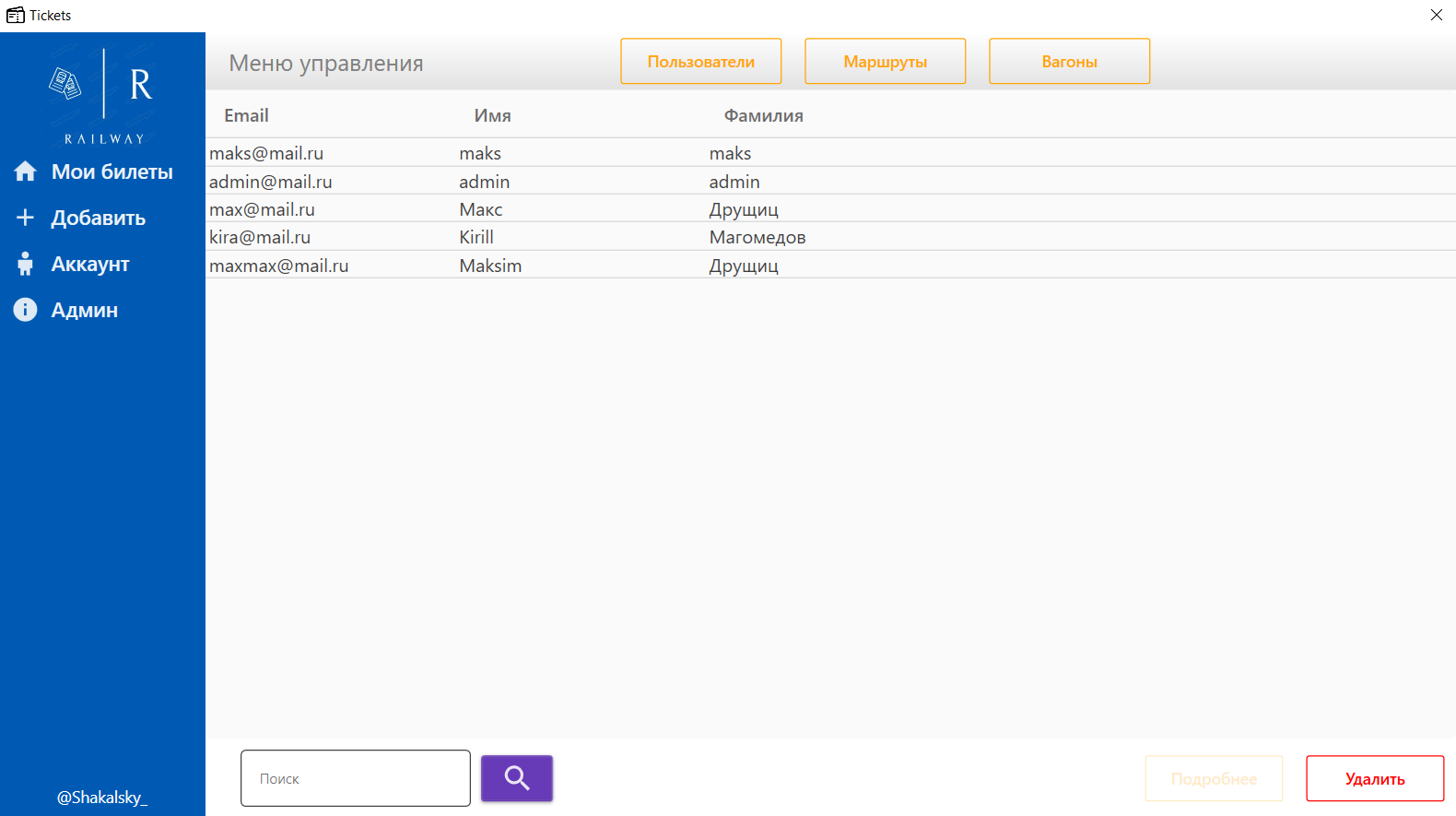
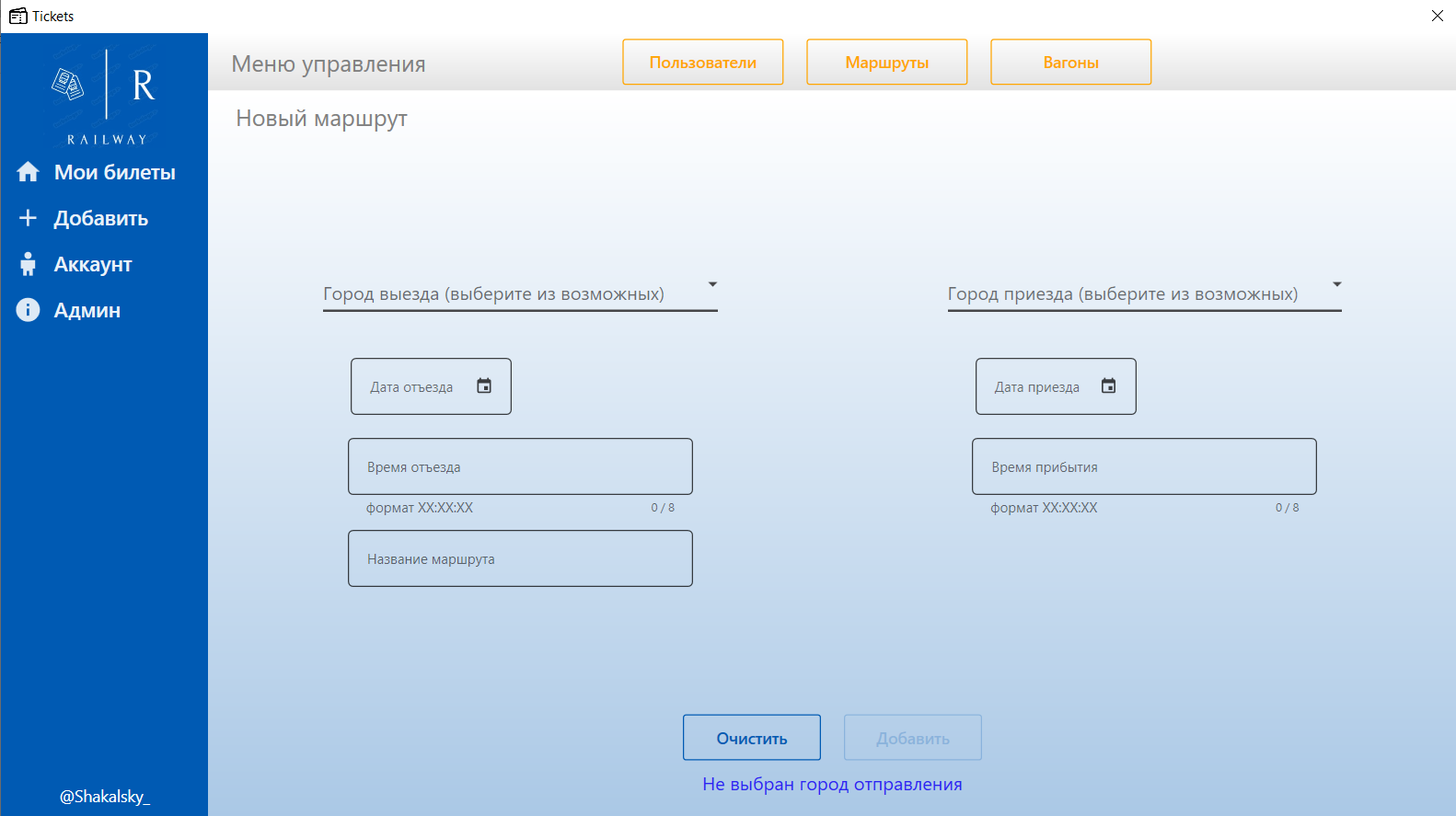
На рисунке 7.1 показана страница Пользователи.

Рисунок 7.1 – «Меню Администратора Пользователи»

На этой странице можно совершать действия над пользователями, просматривать данные конкретных пользователей, информацию об их забронированных билетах. Так же Администратор может удалять пользователей, и, соответственно, их брони на билеты.

На рисунке 7.2 показана страница Маршруты.

Рисунок 7.2 – «Меню Администратора Маршруты»

На странице Маршруты Администратор может добавить новые поезда по ужу существующим маршрутам. Добавление нового маршрута есть добавление новой даты и времени отправления по существующим путям движения поездов.

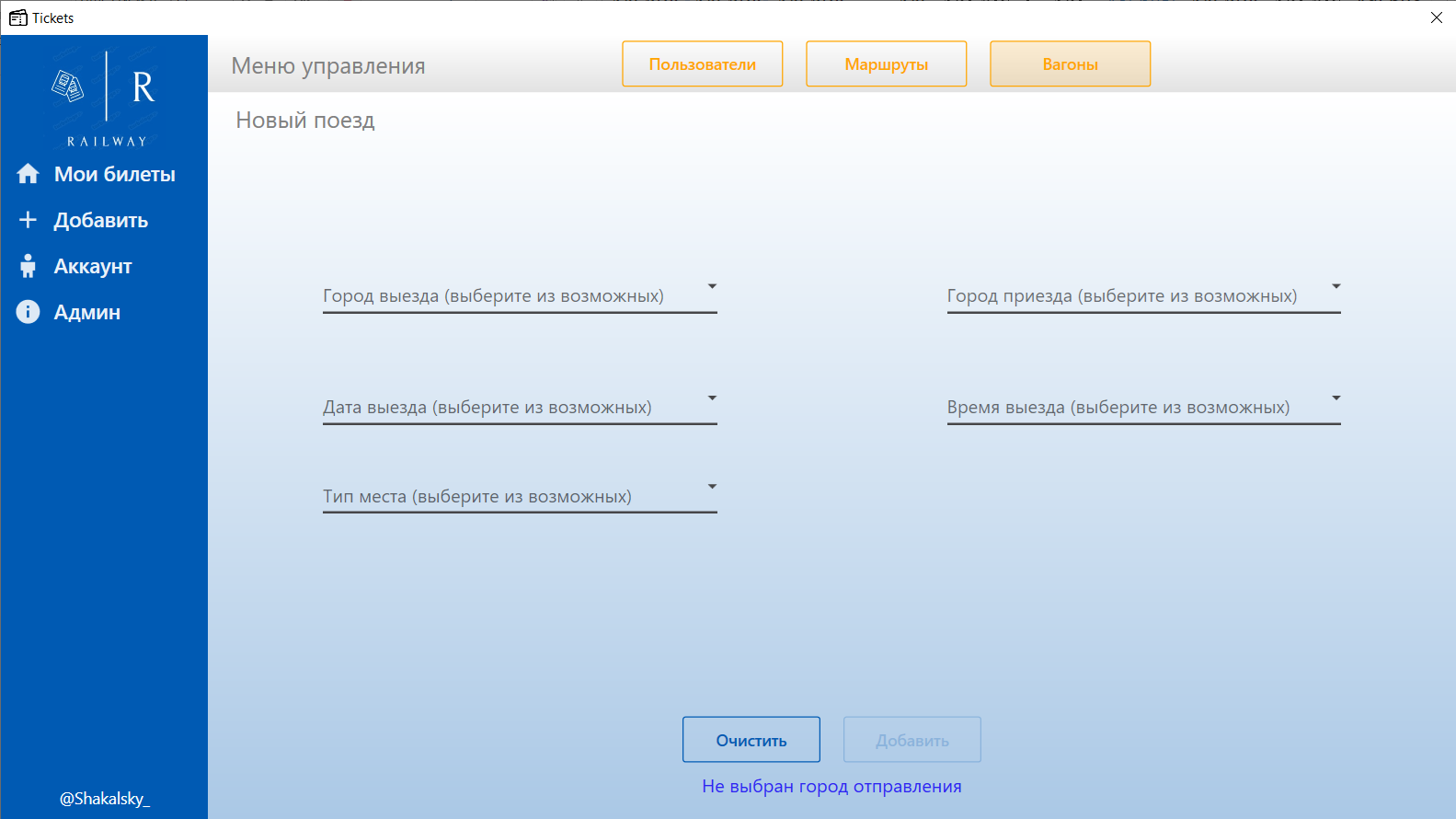
На рисунке 7.3 показана страница Вагоны.

Рисунок 7.3 – «Меню Администратора Вагоны»

На этой странице Администратор может добавлять вагоны любого из трёх типов (Эконом класс, Бизнес класс, Плацкарт) по существующим маршрутам.

Таким образов реализован пункт функционала Администратора “работа с базой данных”. Администратор поддерживает работу с пользователями. Добавляет новые поезда по существующим маршрутам и добавляет новые вагоны, а, соответственно, и новые места по добавленным ранее поездам.

# ЗАКЛЮЧЕНИЕ

В данном курсовом проекте было разработано программное средство «Бронирование железнодорожных билетов» 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.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* Идентификация пользователей;
* Регистрация пользователя;
* Редактирование профиля пользователя;
* Хранение и отображение добавленных броней;
* Добавления новой брони по заданной форме;
* Управление своими бронями на билеты (удаление);

Администратор может всё то же, что и модератор, и плюс ко всему:

* Удаление пользователей;
* Просмотр информации профиля и добавленных броней любого пользователя;
* Добавление новых поездов по заданных маршрутам;
* Добавление разных типов вагонов в заданные ранее поезда.

Проектирование осуществлялось по паттерну проектирования MVVM.

Для удобной работы с данными, был реализован паттерн Repository.

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/](https://github.com/VladKhramykh/MarketPlace_WPF) maks2000-del/wpf\_mvvm\_Tickets\_desktop).

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/

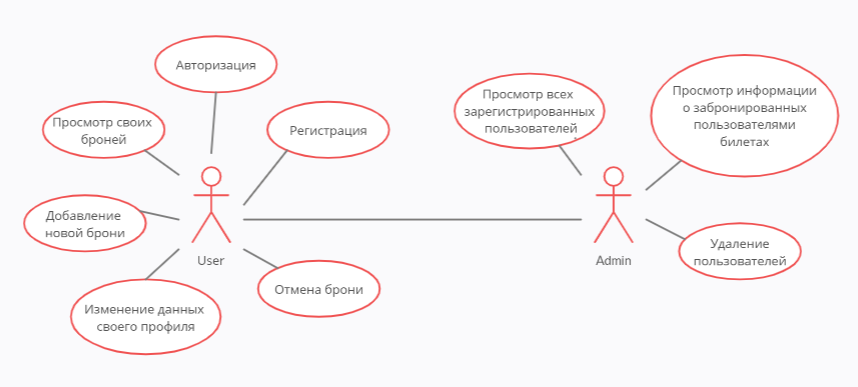
[4]. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com

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

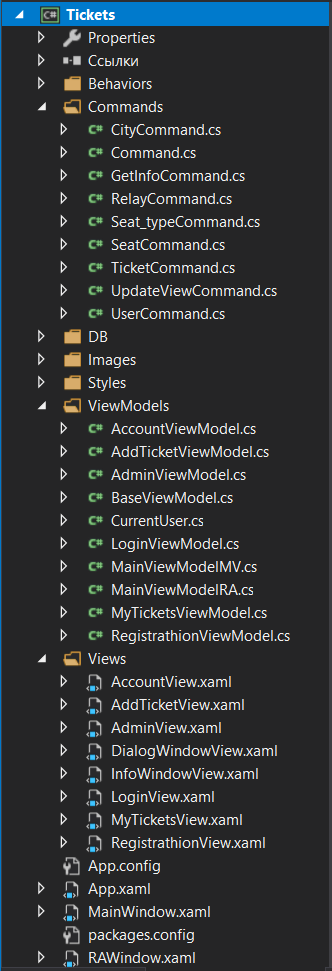
[6]. StackOverflow [Электронный ресурс] / Режим доступа: <https://stackoverflow.com>

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

Приложение А

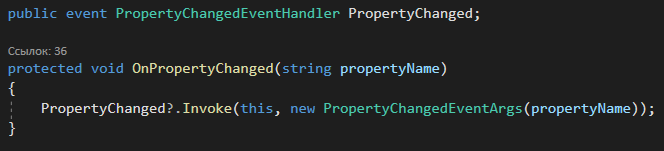


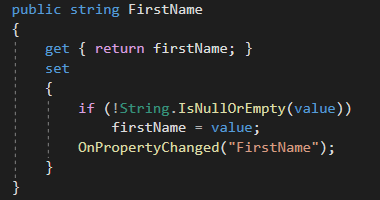
Приложение Б



Приложение В

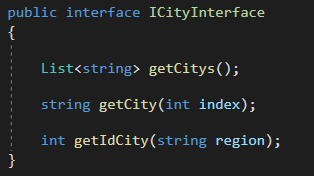
Листинг 1 – «INotifyPropertyChanged»



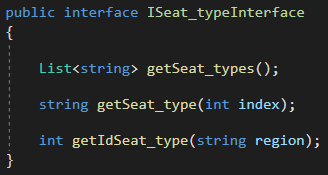


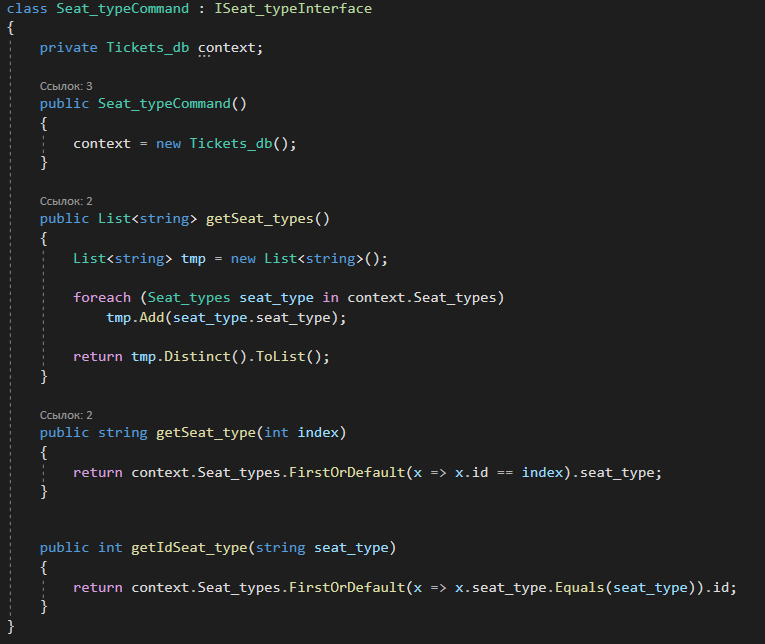
Приложение Г

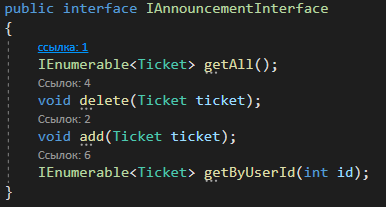
Листинг 2 – «Repositories»

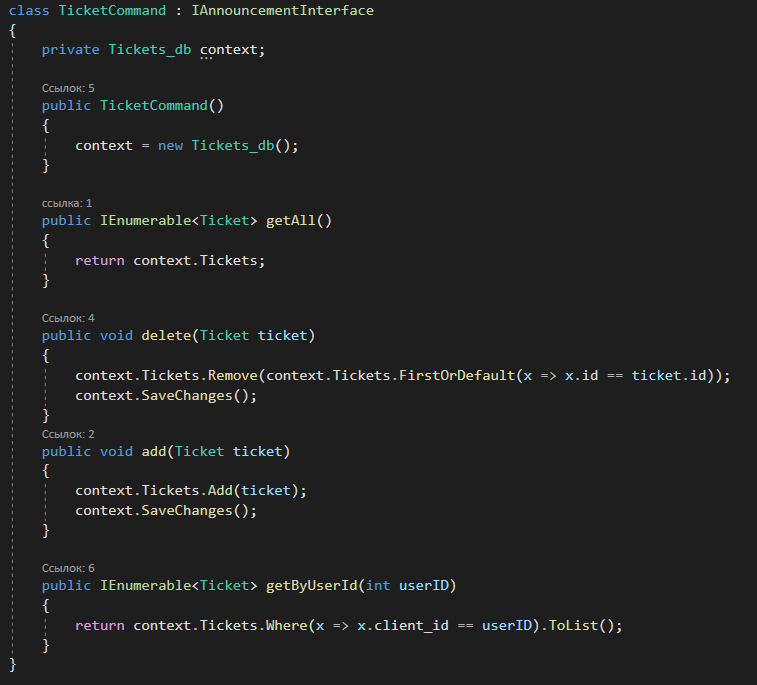


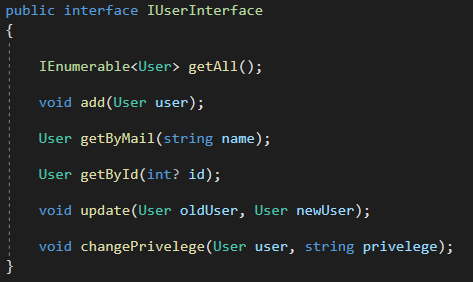


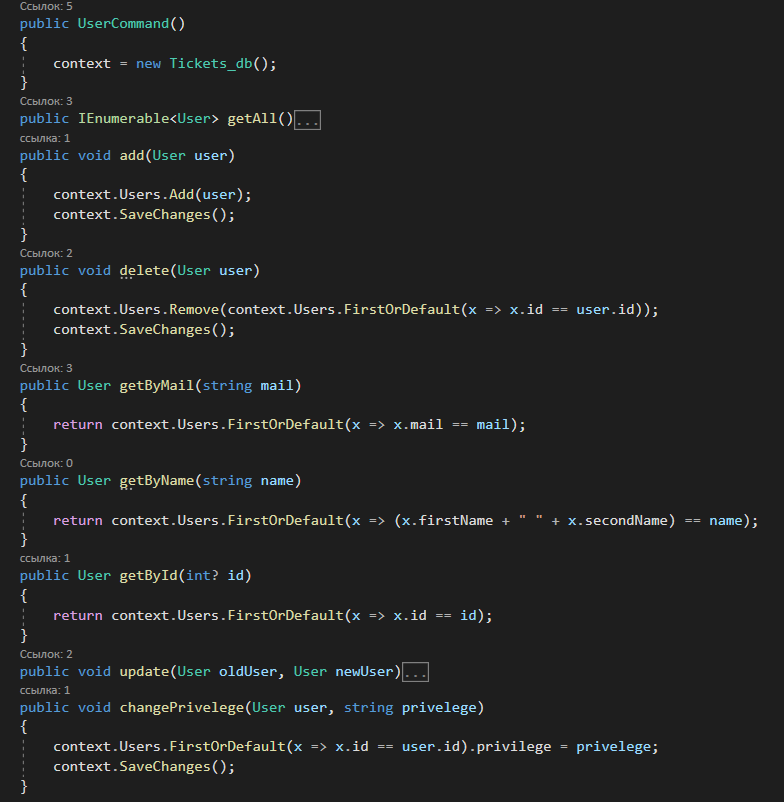


****

****

****

****

****

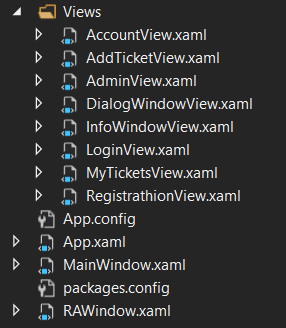
Приложение Д

Листинг 3 – «Структура метод подстраивания данных в ComboBox в зависимости от выбранных ранее»



Приложение Е

Листинг 4 – «Структура пакета View»



Приложение Ж

Листинг 5 – «Структура пакета ViewModel»

