МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет *компьютерных наук*

*Приложение по продаже ж/д билетов “RailWayTickets”*

*Курсовой проект*

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*В.В. Василега, 3 курс, д/о*

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*К.С. Чернышов, 3 курс, д/о*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*А.В. Нужных, преподаватель*

Воронеж 2020

Оглавление

[**Введение**](#_heading=h.t33rj8ddah6y) **3**

[**1. Постановка задачи**](#_heading=h.jyvj9zwxmci8) **3**

[**2. Анализ предметной области**](#_heading=h.tgf0zoakg5o) **4**

[**3. Диаграммы**](#_heading=h.trbej0prl2kd) **5**

[Use Case](#_heading=h.tm8xgdkaq3yf) 5

[Deployment Diagram](#_heading=h.nc97u5a7qwnb) 6

[ER-Diagram](#_heading=h.rfe924cqk4tl) 6

[State Diagrams](#_heading=h.e98a1juh4h2d) 7

[Поиск рейса и оформление билета](#_heading=h.c64avgx5fh7q) 7

[Авторизация/Регистрация](#_heading=h.o74z2fdz9fzs) 7

[Работа с билетами](#_heading=h.kqz44f6kpegb) 8

[Class Diagram](#_heading=h.nx978r8m3u2g) 9

[State Diagrams](#_heading=h.nbbvltohfadl) 10

[Sequence Diagrams](#_heading=h.1z5oc9f2qu0q) 13

[Авторизация](#_heading=h.12neo04mz8hp) 13

[Поиск рейсов и оформление билетов](#_heading=h.5i0ns6161fcc) 14

[Отмена бронирования клиентом](#_heading=h.35cjs793o2w4) 15

[Работа менеджера с билетами](#_heading=h.nre6wdk0gmuu) 16

[**4. Сценарии воронок конверсии**](#_heading=h.vcxvebailol9) **16**

[**Тестирование**](#_heading=h.eb3mt3ak6yu4) **16**

[**Заключение**](#_heading=h.4cco07hkx2sb) **18**

[**Отчет о проделанной работе**](#_heading=h.twttk26i5cjs) **19**

# Введение

Данный курсовой проект описывает приложение, облегчающее использование железнодорожных перевозок обычным человеком. Это оно делает за счёт того, что позволяет дистанционно просматривать расписания ж/д рейсов, искать их по основным критериям(маршрут, время отправления, т.д.), и бронировать билеты на них.

# 1. Постановка задачи

Цель курсовой работы: реализовать web-приложение, которое отвечает следующим требованиям:

1. Стабильная работа через большинство интернет-браузеров
2. Интуитивно понятный пользовательский интерфейс;
3. Возможность выполнения основных задач приложения:
   1. Просмотр расписания ж.д. рейсов;
   2. Поиск рейсов по критериям;
   3. Бронирование билетов на рейсы;
   4. Учет купленных/забронированных билетов.
4. MVC архитектура приложения с разделением на front-end и back-end;
5. Отсутствие нагромождений;
6. Отсутствие броских цветов;

Для достижения данной цели были выделены следующие задачи:

1. Разработка Front-end части приложения, которую можно открыть в виде сайта в интернет-браузере;
2. Разработка Back-end части приложения, развернутой на удаленном сервере приложений;
3. Создание связи между Front-end и Back-end частями приложения;
4. Разработка базы данных, расположенной на удаленном сервере.

# 2. Анализ предметной области

Для реализации серверной части приложения был выбран язык программирования Python. Выбор был сделан в связи с большим распространением данного ЯП, наличием большого числа библиотек и удобством создания архитектуры сервера.

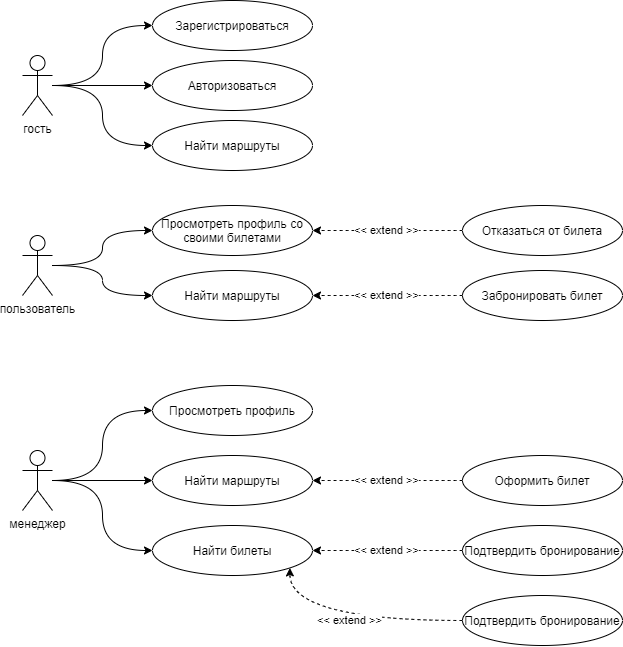
Веб-фреймворк Flask был выбран в качестве основы реализуемого проекта. Основание – легковесность и гибкость, данный фреймворк не имеет жесткой структуры и позволяет выбирать модули под конкретные задачи и устанавливать их по мере необходимости.

В данном проекте используется фреймворк Flask. Для организации работы базы данных был выбрана библиотека SQLAlchemy. В качестве СУБД была выбрана MySQL. СУБД MySQL была использована вследствие ее высокой производительности и простоте развертывания, а также способности MySQL и Flask взаимодействовать с помощью SQLAlchemy, что позволяет использовать подход ORM при разработке приложения.

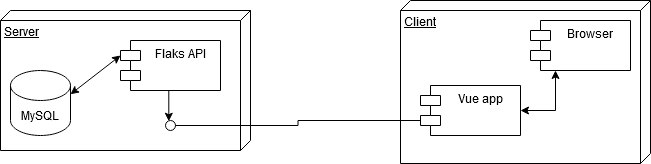
Для реализации клиентской части приложения был выбран фреймворк Vue.js. Данный фреймворк позволяет создавать одностраничное веб-приложение, используя компонентный подход. Выбор был сделан в связи с простотой использования, а также наличия библиотеки Bootstrap-Vue, которая предоставляет множество готовых компонентов. Для взаимодействия с сервером, использовалась библиотека axios.

# 3. Диаграммы

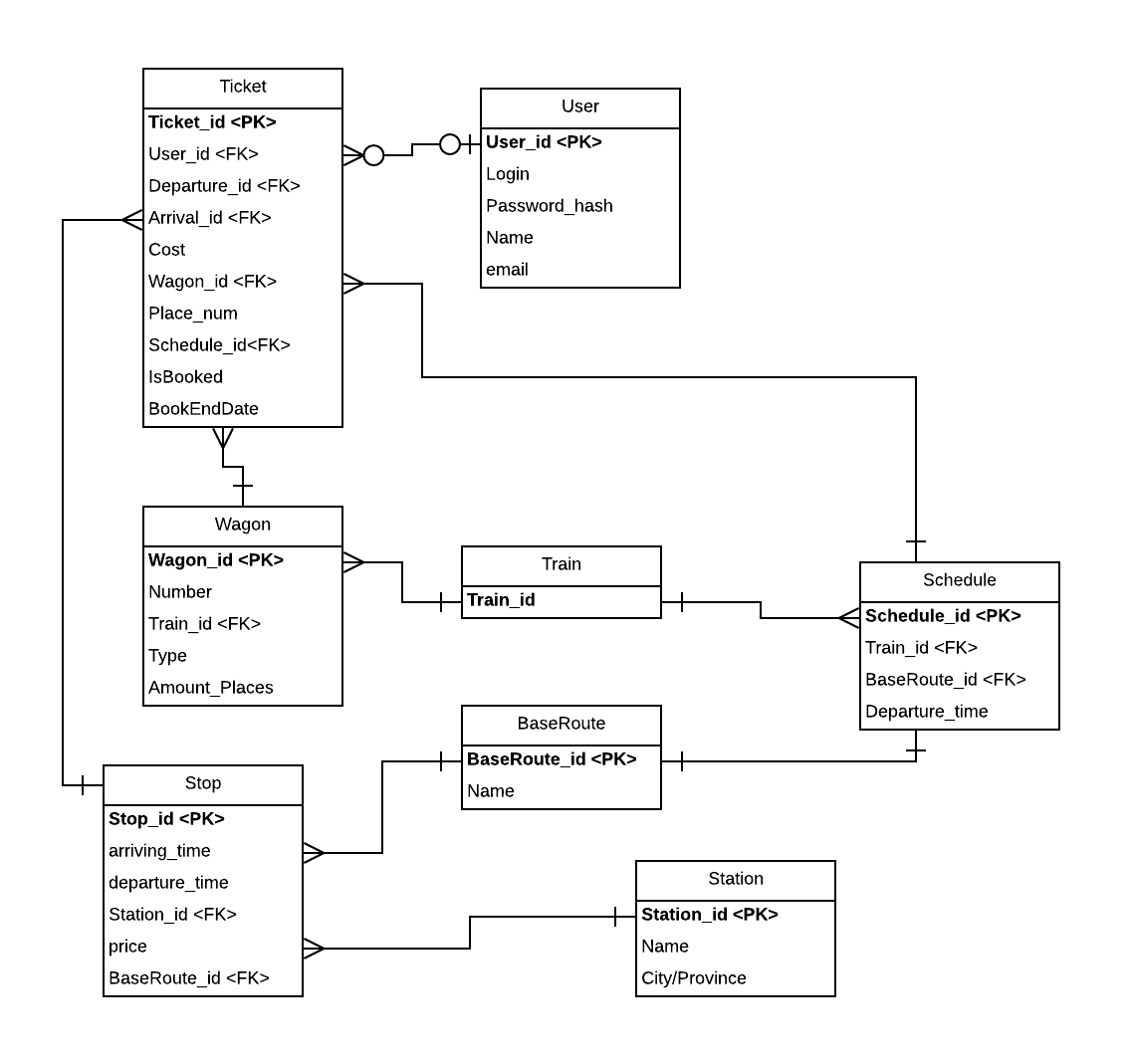
## Use Case



## Deployment Diagram

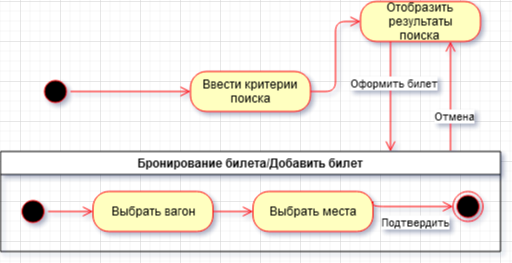


## ER-Diagram



## State Diagrams

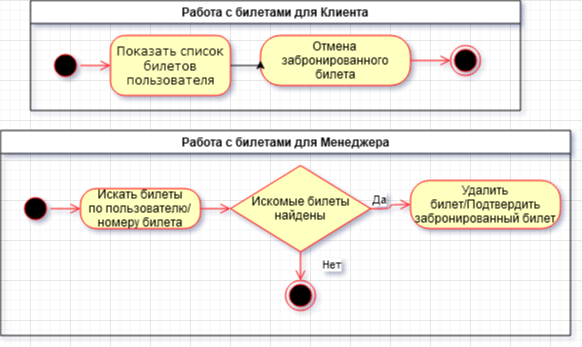
### Поиск рейса и оформление билета



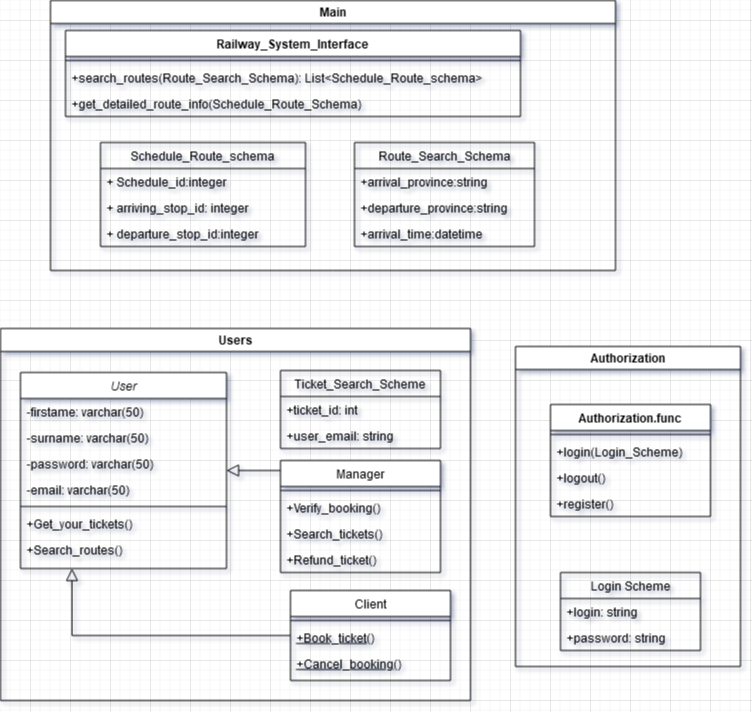
### Авторизация/Регистрация



### Работа с билетами

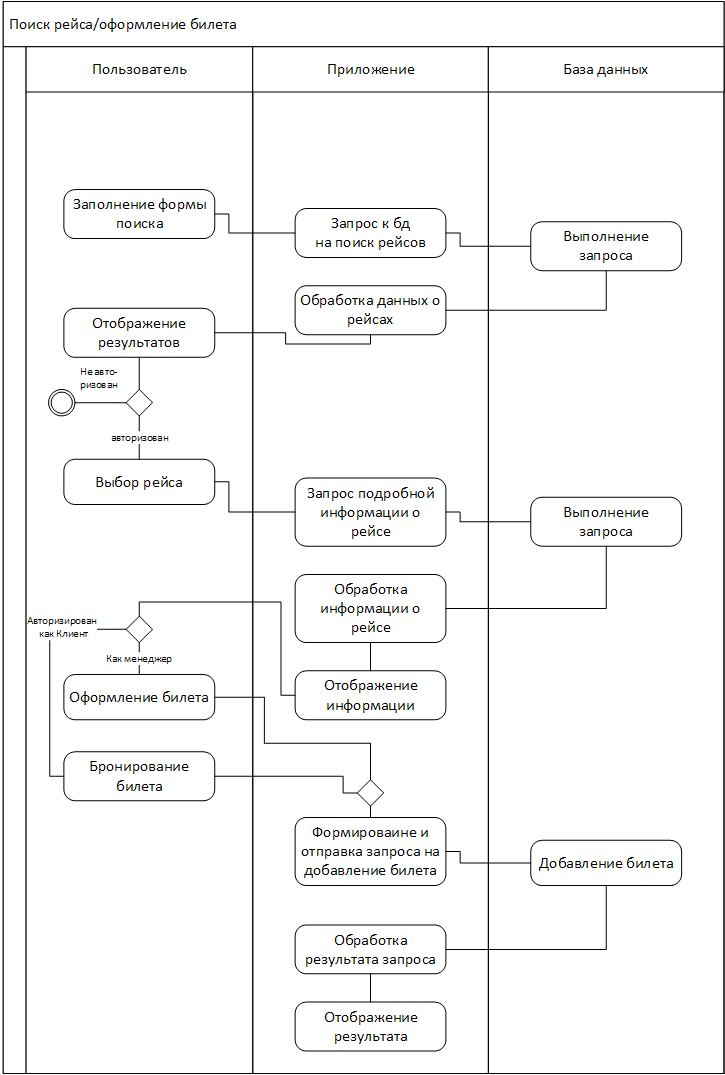


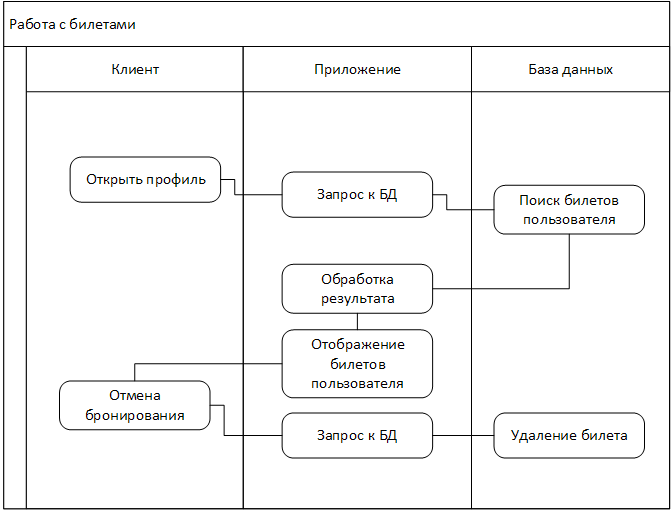
## Class Diagram

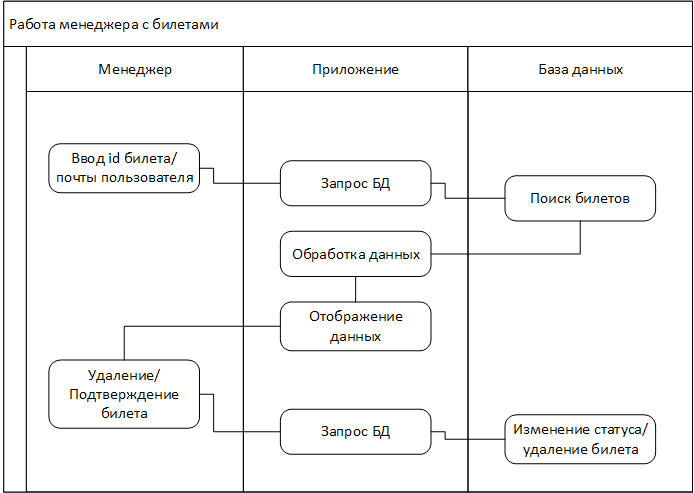


## State Diagrams

### 

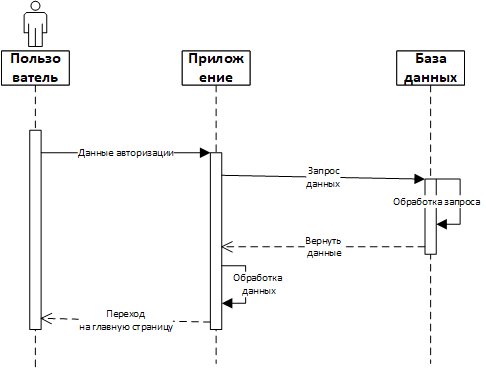




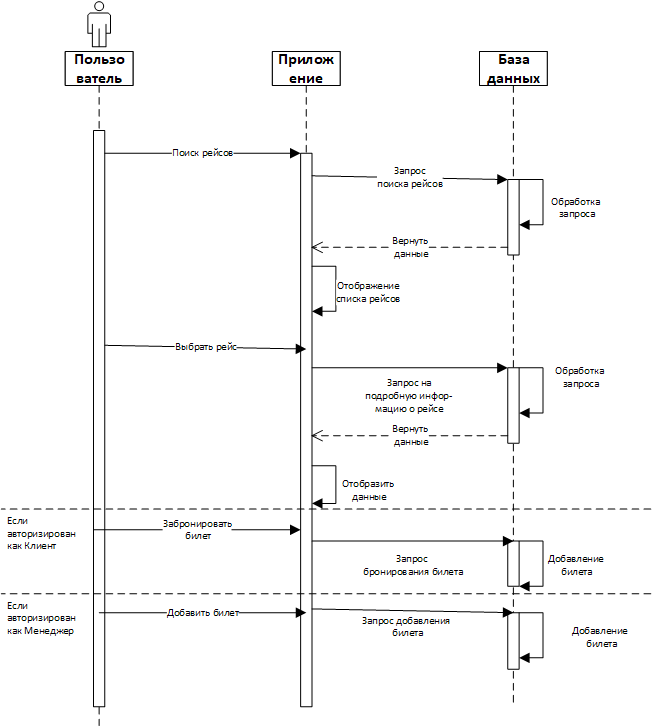


## Sequence Diagrams

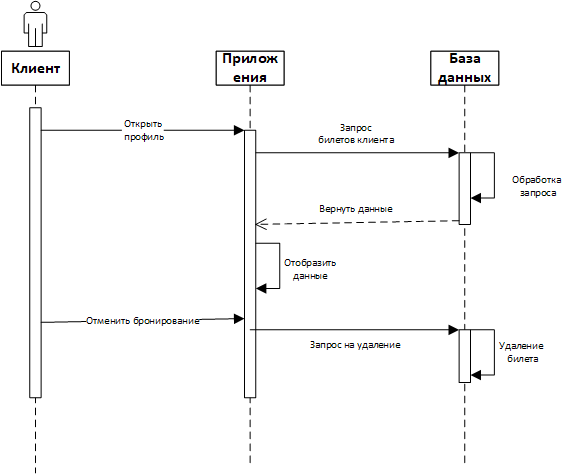
### Авторизация



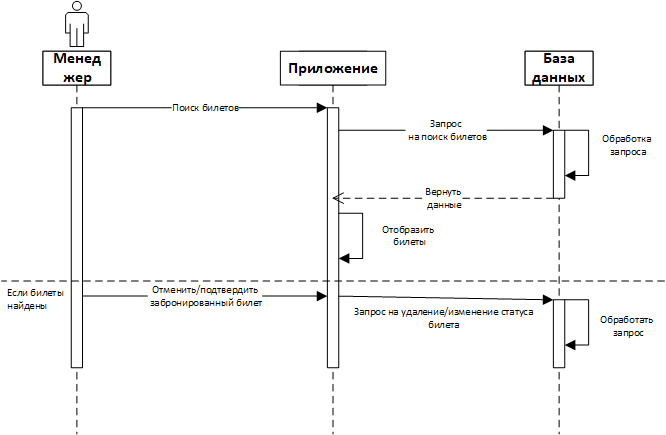
### Поиск рейсов и оформление билетов



### Отмена бронирования клиентом



### Работа менеджера с билетами



# 4. Сценарии воронок конверсии

Сценарий для регистрации пользователя: перейти на страницу регистрации -> Ввести данные -> нажать кнопку «Регистрация»

Сценарий для авторизации пользователя: перейти на страницу авторизации -> Ввести данные -> нажать кнопку «Войти»

Сценарий для поиска рейса: перейти на страницу поиска -> Ввести данные -> нажать кнопку «Поиск»

# 

# 5. Тестирование

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Название | Предпосылка | Шаг | Ожидаемый результат | Отметка о прохождении | Комментарий |
| **1.1** | Авторизация пользователя | Пользователь зарегистрирован, но не авторизован | 1. Ввод логина и пароля  2. Нажатие на кнопку подтверждения | Вход в систему | Пройдено | Пользователь прошел авторизацию |
| **1.2** | Выход пользователя из системы | Выполнение сценария 1.1 | 1. Пользователь нажимает кнопку выхода из системы | Пользователь теряет доступ к части приложения, требующей авторизации | Пройдено | Выход из системы осуществлен |
| **1.3** | Регистрация | Пользователь не зарегистрирован | 1. Заполнение формы регистрации  2. Нажатие на кнопку подтверждения | В БД появляется данные о пользователе, пользователь получает доступ к бронированию билетов | Пройдено | Пользователь прошел регистрацию |
| **2.1** | Поиск рейсов | Пользователь хочет посмотреть расписание рейсов | 1. Пользователь выбирает соответствующие города отправления и прибытия, а также предельную дату прибытия  2. Пользователь нажимает на кнопку поиска | Пользователю отображается список рейсов соответствующих критериям | Пройдено | Пользователь получил список рейсов |
| **3.1** | Бронирование билета | Выполнение сценария 1.1 и 2.1 пользователем с ролью "Клиент" | 1. Пользователь выбирает рейс из списка предложенных  2. В всплывающем окне пользователь выбирает место в одном из вагонов поезда.  3. Пользователь нажимает кнопку бронирования. | В БД добавляется новый билет со статусом "Забронирован" | Пройдено | Пользователь Забронировал билет |
| **3.2** | Отмена бронирования | Выполнение сценария 3.1 пользователем с ролью "Клиент" | 1. Пользователь открывает свой профиль, где ему отображаются его билеты  2. Пользователь нажимает кнопку "Отменить" напротив выбранного билета со статусом "Забронирован" | Из БД удаляется запись о билете | Пройдено | Пользователь успешно отменил бронирование |
| **4.1** | Подтверждение бронирования | Выполнение сценария 1.1 пользователем с ролью "Менеджер", произошла плата за забронированный билет в физической точке продажи | 1. Пользователь ищет билет по его номеру/данным его владельца  2. Пользователь нажимает кнопку подтвердить | Статус билета меняется на "Подтвержденный" | Пройдено | Забронированный был подтверждён |
| **4.2** | Добавление билета в базу | Выполнение сценария 1.1, 2.1 пользователем с ролью "Менеджер", билет покупается в физической точке без бронирования | 1. Пользователь выбирает рейс из списка предложенных  2. В всплывающем окне пользователь выбирает место в одном из вагонов поезда  3. Пользователь нажимает кнопку оформления билета | Добавления билета в базу | Пройдено | Купленный билет отображается в базе данных системы |
| **4.3** | Возврат билета | Выполнение сценария 1.1 пользователем с ролью "Менеджер", билет возвращается в физической точке продажи | 1. Пользователь ищет билет по его номеру/данным его владельца  2. Пользователь нажимает кнопку отмены билета | Удаление билета из базы | Пройдено | Возврат билета отображен в базе данных системы |
| **5.1** | Негативный сценарий регистрации пользователя | Пользователь не зарегистрирован | 1. Пользователь выбирает пункт "Авторизоваться" на главной странице  2. Ввод уже существующей в системе электронной почты пользователя в форму.  3. Ввод остальных полей регистрации пользователя в форму.  4. Пользователь нажимает кнопку подтверждения | Предупреждение пользователя об ошибке посредством сообщения | Пройдено | Сообщение об ошибке высветилось |
| **5.2** | Негативный сценарий авторизации пользователя | Пользователь зарегистрирован, но не авторизован | 1. Пользователь вводит некорректную пару электронной почты и пароля.  2. Пользователь нажимает кнопку подтверждения | Предупреждение пользователя об ошибке посредством сообщения | Пройдено | Сообщение об ошибке высветилось |

# Заключение

Заявленный в техническом задании проект был полностью реализован командой из двух участников. Он соответствует следующим требованиям:

* Система помогает гостям находить рейсы поездов.
* Система предоставляет возможность пользователям забронировать билеты, а также просматривать список своих билетов.
* Система позволяет менеджерам оформлять билеты, подтверждать покупку или отменять бронирование билетов.
* Данное приложение полностью удовлетворяет требованиям, излагаемым в техническом задании. Заданный функционал отлажен и работает.

# Отчет о проделанной работе

Техническое задание составили: Чернышов К.С., Василега В.В.

Василега В.В:

* Диаграммы: Классов, Развертывания, Активностей, Состояний, Последовательностей;
* Back-end (Flask);
* Подключение базы данных;
* Тестирование.

Чернышов К.С:

* Диаграммы: Use Case, ER;
* Front-end (Vue + vuex, axios - для запросов к серверу);
* Подключение Яндекс метрики;
* Размещение на хостинге.