**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
 РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФГБОУ ВО «БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Кафедра** «Информатика и программное обеспечение»

**КУРСОВАЯ РАБОТА ПО ДИСЦИПЛИНЕ**

**«Разработка web-узлов и приложений»**

ВЕБ-САЙТ ПО ПРОДАЖЕ АВИАБИЛЕТОВ

Выполнил студ. гр. О-18-ИВТ-1-по-Б, зач. кн. №18.0412

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Максимов Д.И.

«22» декабря 2020 г.

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доц. Белов Е.А.

«22» декабря 2020 г.

Брянск 2020

Содержание

[Введение 3](#_Toc59512100)

[Анализ предметной области 4](#_Toc59512101)

[1.1. Основные черты сайта по продаже авиабилетов 4](#_Toc59512102)

[1.1.1. Характерные признаки 4](#_Toc59512103)

[1.2. Сайт по поиску билетов AviaSales 4](#_Toc59512104)

[1.2.1. История сервиса 5](#_Toc59512105)

[1.2.2. Особенности сервиса 6](#_Toc59512106)

[Конструкторская часть 7](#_Toc59512107)

[2.1 Общая структура проекта 7](#_Toc59512108)

[2.2 Структура базы данных 9](#_Toc59512109)

[2.3 Технические решения 10](#_Toc59512110)

[2.3.1 Класс MainController 10](#_Toc59512111)

[2.3.2 Шаблон Hello.ftlh 11](#_Toc59512112)

[2.3.3 Функция seatchTickets 13](#_Toc59512113)

[2.3.4 Функция getTickets 15](#_Toc59512114)

[2.3.5 Класс RolesApiController 17](#_Toc59512115)

[2.3.6 Класс Airport 20](#_Toc59512116)

[2.3.7 Класс WebSecurityConfig 21](#_Toc59512117)

[Тестирование 23](#_Toc59512118)

[Заключение 32](#_Toc59512119)

[Список литературы 33](#_Toc59512120)

Введение

Авиа-туризм является одной из самых крупных, прибыльных и быстроразвивающихся отраслей в мире. Ведущие компании и студии ежегодно наблюдают стабильный прирост клиентов и, соответственно, увеличение доходов. На сегодняшний день туризм стал менее востребован, но это не отменяет спрос на него.

В этой курсовой работе будет рассмотрен сайт по продаже авиабилетов, а также будет разработан собственный сайт данного типа. Сайт будет содержать личный кабинет пользователя с отслеживанием купленных билетов, фильтр и покупку авиабилетов, включая функционал пересадок. Так же сайт будет обладать системой редактирования всех данных.

# Анализ предметной области

## Основные черты сайта по продаже авиабилетов

В последнее время все меньше и меньше людей ходят в турагентства для покупки авиабилетов, или же покупают их прямо в аэропорту. В связи с этим стали активно развиваться сайты по продаже авиабилетов онлайн.

Одним из характерных элементов такого сайта является панель фильтра на главной странице для поиска билетов, а также возможность покупать несколько билетов с возможностью пересадки, если прямого билета нет, или он стоит дороже.

### Характерные признаки

Каждый подобный сайт имеет свои характерные признаки:

* Сайт имеет возможность подбора авиабилета по дате вылета.
* Присутствует возможность побора авиабилетов с пересадками, что позволяет пользователю сэкономить, а иногда – это единственный способ попасть в пункт назначения.
* Присутствует возможность выбора других параметров, таких как: тип багажа, дата вылета.

## Сайт по поиску билетов AviaSales

Самым популярным и удобным сайтом по поиску авиабилетов, является, несомненно, AviaSales.

Aviasales.ru — российский метапоисковик авиабилетов, существующий с 2007 года. Основатель — Константин Калинов. Ежемесячная аудитория сервиса оценивается примерно в 15 миллионов человек. Штаб-квартира компании находится в Таиланде на острове Пхукет. В России у Aviasales ещё два офиса: в Москве и в Санкт-Петербурге. Метапоисковик принадлежит компании Go Travel Un Limited.[1]

### История сервиса

Ресурс появился в 2007 году как личный блог путешественника Константина Калинова, где аккумулировались спецпредложения авиакомпаний. Изначально с Калиновым работал только один редактор, получавший тогда все деньги заработанные с рекламы (от 30 до 50 долларов в месяц). Позже в компанию были привлечены программисты, которые добавили поисковый алгоритм по базе предложений авиакомпаний внутри ресурса.

Первый движок написал выпускник «Военмеха» Антон Кириллов, который впоследствии ушел из компании. В 2008 году с помощью Aviasales был забронирован первый авиабилет. Рейс «Москва — Париж».

Когда Калинов в 2009 году обратился к Максу Крайнову, чтобы оценить стоимость ресурса, тот построил финансовую модель компании и посоветовал подождать. Дела пошли в гору, и Калинов предложил Крайнову стать партнёром. В 2011 году Макс присоединился к Aviasales.

В этом же году появилась партнёрская программа Travelpayouts. За рубежом Aviasales работал под торговой маркой JetRadar с 2012 года. Сервис был локализован для 16 стран. В мае этого же года было запущено мобильное приложение, которое через два дня после появления в AppStore заняло первое место в категории «Путешествия», обогнав приложения «Яндекс.Такси» и Booking.com.

В 2013 году был запущен Hotellook — отельный поисковик. В феврале 2014 года сервис закрыл первый раунд инвестиций в размере $10 млн от фонда iTech Capital. Через некоторое время эту сделку назвали Deal of the year на премии Venture Award. Фонд получил миноритарную долю в компании, а управляющий партнер фонда Глеб Давидюк и инвестиционный менеджер Илья Баландин вошли в совет директоров Aviasales. Инвестиции привлекались для развития сервиса, маркетинга продуктов и расширения штата.

В январе 2016 года Aviasales подписала контракт с киберспортивной командой Natus Vincere, став одним из первых непрофильных спонсоров киберспорта в России.

В марте 2017 года стало известно, что компания Go Travel Un Limited решила сократить персонал на 20 % (в общей сложности 32 человека) и переформировать другие свои проекты JetRadar и Hotellook. По словам управляющего директора Макса Крайнова, это решение было связано с ужесточением конкуренции на российском рынке после того, как Priceline Group (владелец Booking.com и Kayak) приобрела Momondo Group (туристические поисковики Momondo и Cheapflights). Aviasales планировал высвободить ресурсы, чтобы укрепить свои позиции на российском рынке. По итогам реструктуризации Hotellook должен был стать частью Aviasales и работать под брендом «Aviasales Отели», а JetRadar превращался в часть партнёрской программы Travelpayouts.

Высвободив ресурсы, метапоисковик сфокусировался на рынках России и стран СНГ. Это решение привело к увеличению выручки на отечественном рынке на 63%. В итоге Hotelook превратился в часть Travelpayouts, и сейчас отели на Aviasales бронируются с помощью Booking.com, с которым компания вступила в стратегическое партнёрство в июне 2019 года.

В 2019 году Skift поместил Aviasales в топ 25 перспективных тревел-стартапов.

### Особенности сервиса

В отличие от заблуждения многих пользователей, Aviasales является метапоисковым сервисом и не продаёт билеты. Найдя нужный билет, пользователь переходит на страницу авиакассы или авиакомпании, чтобы совершить оплату. Оплата происходит на странице авиакомпании или авиакассы либо прямо в интерфейсе поисковика.

Кроме поисковых функций, ресурс оставляет за собой право модерировать выдачу — в случае, если поставщик манипулирует стоимостью перевозки, его исключают.

# Конструкторская часть

## Общая структура проекта

Проект написан на Java 15 с использованием фреймворка Spring. В качестве шаблонизатора используется FreeMarker. Данные хранятся в базе данных MySQL, доступ к которой осуществляется с помощью Spring Data репозиториев. Сайт располагается на Apache. Авторизация производится с помощью Spring Security.

Рассмотрим структуру проекта (см. рис. 1). Фреймворк Spring MVC обеспечивает архитектуру паттерна Model — View — Controller (Модель — Отображение (далее — Вид) — Контроллер) при помощи слабо связанных готовых компонентов.

Паттерн MVC разделяет аспекты приложения (логику ввода, бизнес-логику и логику UI), обеспечивая при этом свободную связь между ними.

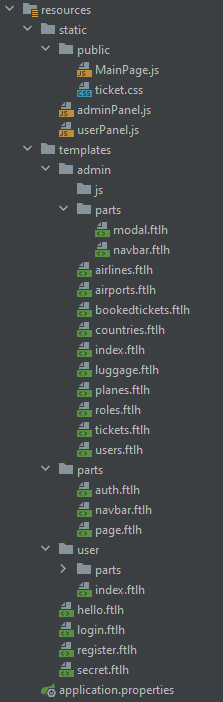
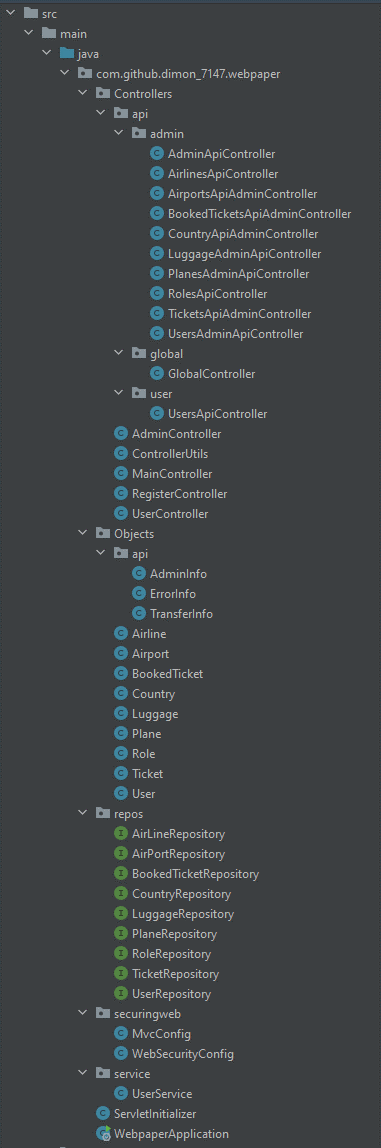
Model (Модель) инкапсулирует (объединяет) данные приложения, в целом они будут состоять из POJO (Java-объектов, или «бинов»).

View (Отображение, Вид) отвечает за отображение данных Модели, — как правило, генерируя HTML, которые мы видим в своём браузере.

Controller (Контроллер) обрабатывает запрос пользователя, создаёт соответствующую Модель и передаёт её для отображения в Вид.

Рассмотрим основные директории:

* Controllers – в данной директории располагаются контроллеры, обрабатывающие запросы.
* Object – Java-объекты, хранящиеся в БД, за исключением поддиректории api, объекты в ней служат для возращения json объектов по запросу на api.
* Repos, Service – репозитории для взаимодействия с базой данных.
* SecurityWeb – настройки безопасности приложения – ограничение доступа к определенному пути ресурса без авторизации.

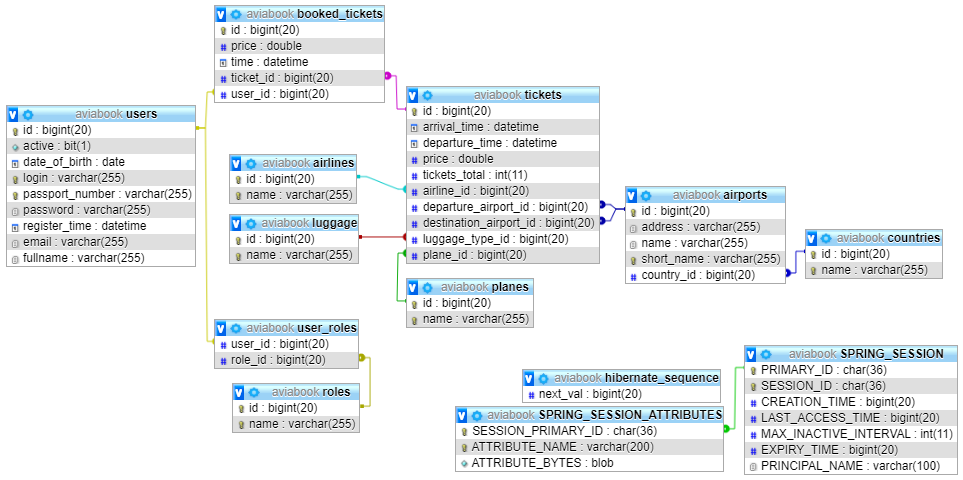


**Рис. 1 Структура приложения.**

## Структура базы данных

Для хранения всей информации, безусловно, необходимо было разработать базу данных. Мной был выбран СУБД MySql, поскольку он считается для меня более удобным.

Рассмотрим диаграмму на рис. 2:



**Рис. 2 Блок-Схема БД.**

База данных содержит 13 таблиц, 3 из которых системные – они нужны для корректной работы Spring и хранения сессии пользователей.

Рассмотрим предназначение основных таблиц:

* Tickets – основная таблица. Необходима для хранения данных о билетах на продажу.
* Booked\_Tickets – хранит информацию о приобретённых билетах.
* Users – информация о пользователе: логин, пароль, паспортные данные, дата регистрации, ФИО, дата рождения, email, активность профиля.
* Roles – хранит информацию о ролях. Таблицы связываются друг с другом отношением «Много-ко-Многим» с помощью дополнительной таблицы user\_roles.
* Airlines – Авиакомпании.
* Countries – Список стран.
* Luggage – Информация о багаже.
* Planes – Список самолетов.
* Airports – Список аэропортов.

## Технические решения

Рассмотрим наиболее важные фрагменты кода:

### Класс MainController

Данный класс является контроллером, обсуживающим главную страницу сайта.

Рассмотрим его на рис. 3



**Рис. 3 Контроллер главной страницы**

При получении запроса на отображение главной страницы, контроллер проверяет авторизацию пользователя с помощью аннотации @ AuthenticationPrincipal. В случае, если пользователь авторизирован, он добавляется на модель для отображения дополнительных ссылок в панели навигации. Если же нет, то в модель помещается пользователь с id -1, что позволяет макету идентифицировать, что пользователь не авторизован.

### Шаблон Hello.ftlh

Рассмотрим шаблон hello.ftlh на рис. 4.

В начале шаблона происходит импорт заголовка страницы со стилями bootstrap с помощью макроса, функционал которых предоставляет шаблонизатор FreeMarker. Затем происходит определение авторизации пользователя и отображение навигационной панели в зависимости от привилегий пользователя. Затем происходит обход по списку аэропортов, типов багажей и их добавление в селекторы для последующего выбора пользователем.

Так же в шаблоне содержится скрытый класс с id ticketsMain, который служит для вывода найденных билетов на страницу с помощью технологии Ajax и ЯП JavaScript с библиотекой JQuery.



**Рис. 4 Шаблон главной страницы**

### Функция seatchTickets

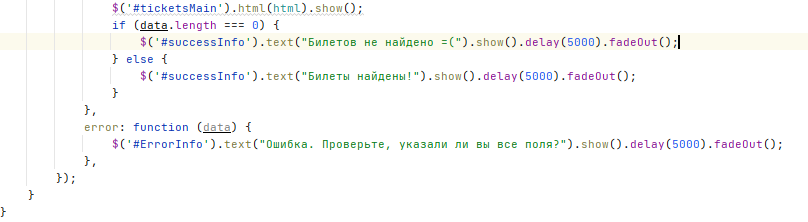
Рассмотрим функцию searchTickets() (см. рис. 5-7):



**Рис. 5 Функция seatchTickets() 1/3**



**Рис. 6 Функция seatchTickets() 2/3**



**Рис. 7 Функция seatchTickets() 3/3**

Данная функция вызывается при нажатии на кнопку поиска билетов. В зависимости от выбранного варианта поиска (с пересадками, или без), выполняются разные ajax запросы. Затем происходит извлечение данных из полученного json и их занесение в виде html кода на страницу. В момент обработки информации пользователь видит иконку загрузки. В случае успешной загрузки, будет выведено окошко с информацией, как и в случае неудачи.

### Функция getTickets

Рассмотрим получение списка билетов (см. рис. 8-9). При обращении GET запросом по адресу /api/public/tickets и передачей обязательных параметров: id аэропортов, даты полета, происходит выдача списка билетов. Но перед этим происходит фильтрация – пользователю не будут отданы билеты с истекшим сроком. В случае, если пользователь ищет билеты без пересадок, то ему выдается список билетов с помощью репозитория.

В случае, если пользователь ищет билеты с пересадками, то сначала выбираются все билеты, у которых отправочный пункт – выбранный пользователем аэропорт. Затем каждый билет обходится в поисках нужного конечного аэропорта. Если такой билет был найден, то он добавляется в список, который в дальнейшем вместе с списком билетов добавляется в объект класса TransferInfo и выводится в виде json объекта.



**Рис. 8 Функция getTickets 1/2**



**Рис. 9 Функция getTickets 2/2**

### Класс RolesApiController

Рассмотрим класс RolesApiController (см. рис. 10) для понимания того, как реализованы API запросы на добавление, обновление, удаление, получение данных таблиц в панели администратора.

Рассмотрим функцию addRole, которая принимает POST запросы по адресу /api/admin/roles. На вход подается объект класса Role. Spring сам расшифровывает json и преобразует его в объект класса. Затем создается новая роль, в которую заносятся данные, переданные в запросе. Это нужно для того, чтобы пользователь не смог внести некорректные данные (к примеру, указав id – это делается автоматически). После чего роль добавляется в репозиторий (БД), а клиенту выводится информация о том, что роль успешно добавлена.

Теперь рассмотрим функцию removeRole, которая принимает DELETE запросы по адресу /api/admin/roles/{id}, где вместо {id} – id удаляемой роли. В функции проверяется существование роли по заданному идентификатору, если она существует, то она удаляется из БД, и клиенту выводится информация о том, что роль удалена, или же в противном случае, что роль не была удалена.

Рассмотрим функцию getRoles. Она выводит список всех пользователей. В данном случае все довольно просто – выводятся все данные из репозитория таблицы роли.

Перейдем к функции getRole. Она обрабатывает GET запросы по адресу /api/admin/roles/ id}, где вместо {id} – id роли. Она нужна для получения информации о роли с конкретным идентификатором. Аналогично предудущим функциям, делается запрос в репозиторий, в случае успеха возвращается объект Role.

В заключение, рассмотрим последнюю функцию в данном классе – saveRole. Она служит для обновления информации о существующей роли по протоколу PUT и адресу /api/admin/roles/ id}, где вместо {id} – id роли. В ней происходит валидация полей роли с помощью аннотации @Valid. Критерии валидации описываются в классе самого объекта таблицы. Затем, в случае успеха, роль обновляется в БД (Репозиторий идентифицирует ее по ID).



**Рис. 10 Класс RolesApiController**

.

.

### Класс Airport

На примере данного класса мы рассмотрим как реализованы сущности

Таблиц (см. рис. 11):



**Рис. 11 Класс Airport**

Перед классом стоят аннотации @Entity и @Table. Они необходимы для понимания фреймворком того, что данный класс является представлением таблицы в БД.

С помощью аннотаций над методами задаются характеристики полей в БД. К примеру, у id есть аннотации @Id, @GeneratedValue и @Collumn.

@Id нужен для идентификации сущности, @GeneratedValue отвечает за автоматическую генерацию значений (автоинкримент), а @Column нужен для понимания того, что данное поле должно храниться в базе данных. К тому же, у @Collumn есть дополнительные параметры, к примеру, можно определить возможность null значений в колонке.

Рассмотрим поле country. В нем мы видим две аннотации: @ManyToOne и @JoinColumn. Первая аннотация определяет тип связи с другой таблицей, а параметр name @JoinColumn позволяет конкретно указать, с каким полем осуществляется связь.

### Класс WebSecurityConfig

Данный класс служит для настройки параметров безопасности сайта. Рассмотрим его на рис. 12:

В данном классе нас интересует метод configure(), отвечающий за безопасность. В нем разрешены запросы для всех по адресам «/», «/register», «/api/public/\*\*», «/public/\*\*». Ограничены только для пользователей запросы по адресу «/api/user/\*\*», и только для администрации по адресу «/api/admin/\*\*». Так тут описано поведение при успешной авторизации и выходе с сайта.

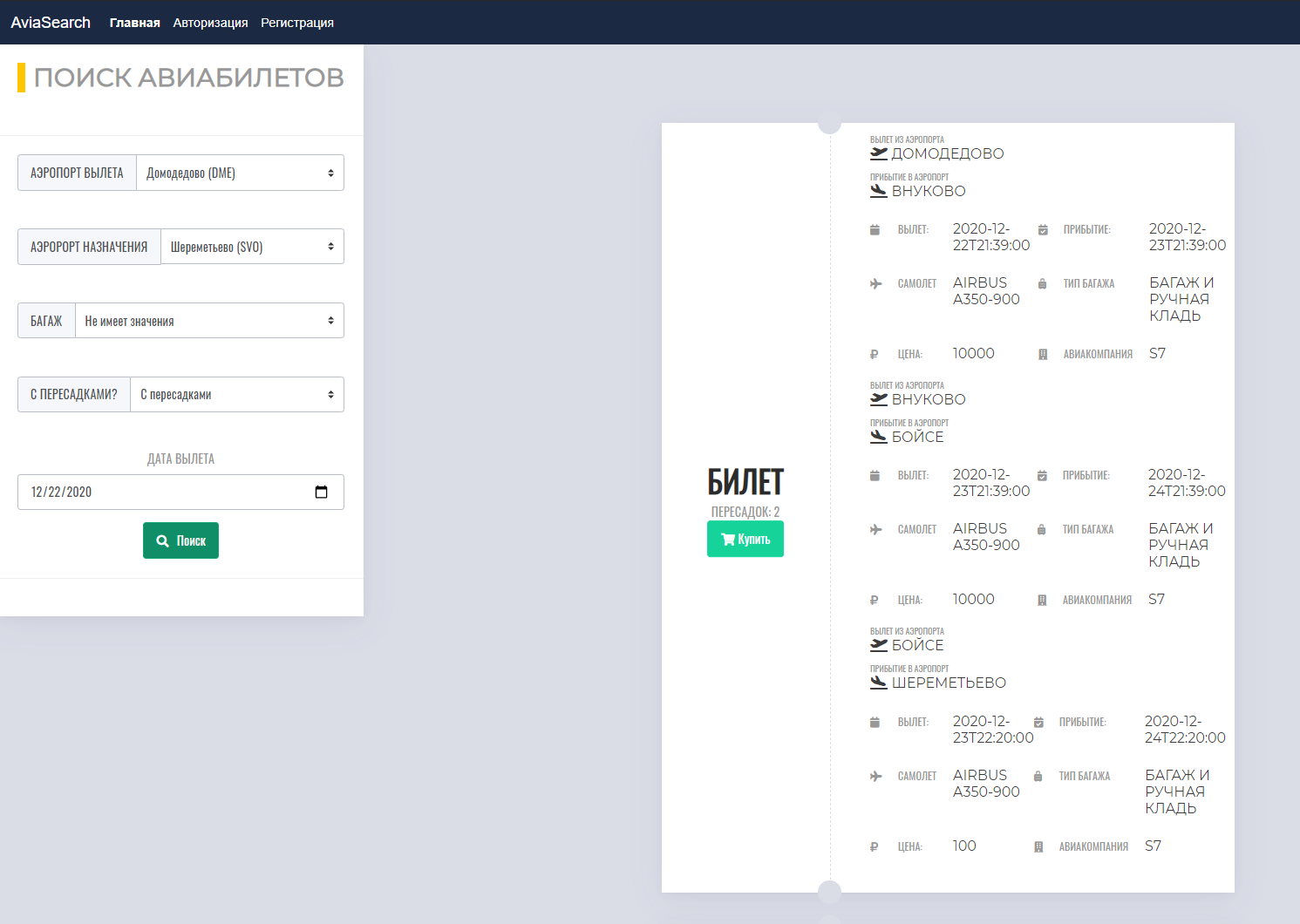


**Рис. 12 Класс WebSecurityConfig**

Тестирование

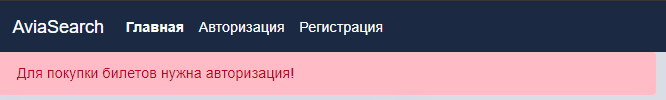
Проверим работу сайта в соответствии с заданными требованиями.

При заходе на сайт должна отображаться панель поиска билетов, а также ссылки на страницу авторизации и регистрации.



**Рис. 13 Главная страница сайта**

Видим (см. рис. 13), что действительно отображается поиск авиабилетов с возможностью фильтрации, а также то, что он работает корректно. Попробуем купить билет, хотя мы не авторизованы.



**Рис. 14 Уведомление о необходимости авторизации**

Видим (см. рис. 14), что данное действие пользователя предусмотрено и сайт выдает ошибку о том, что пользователь не авторизован.

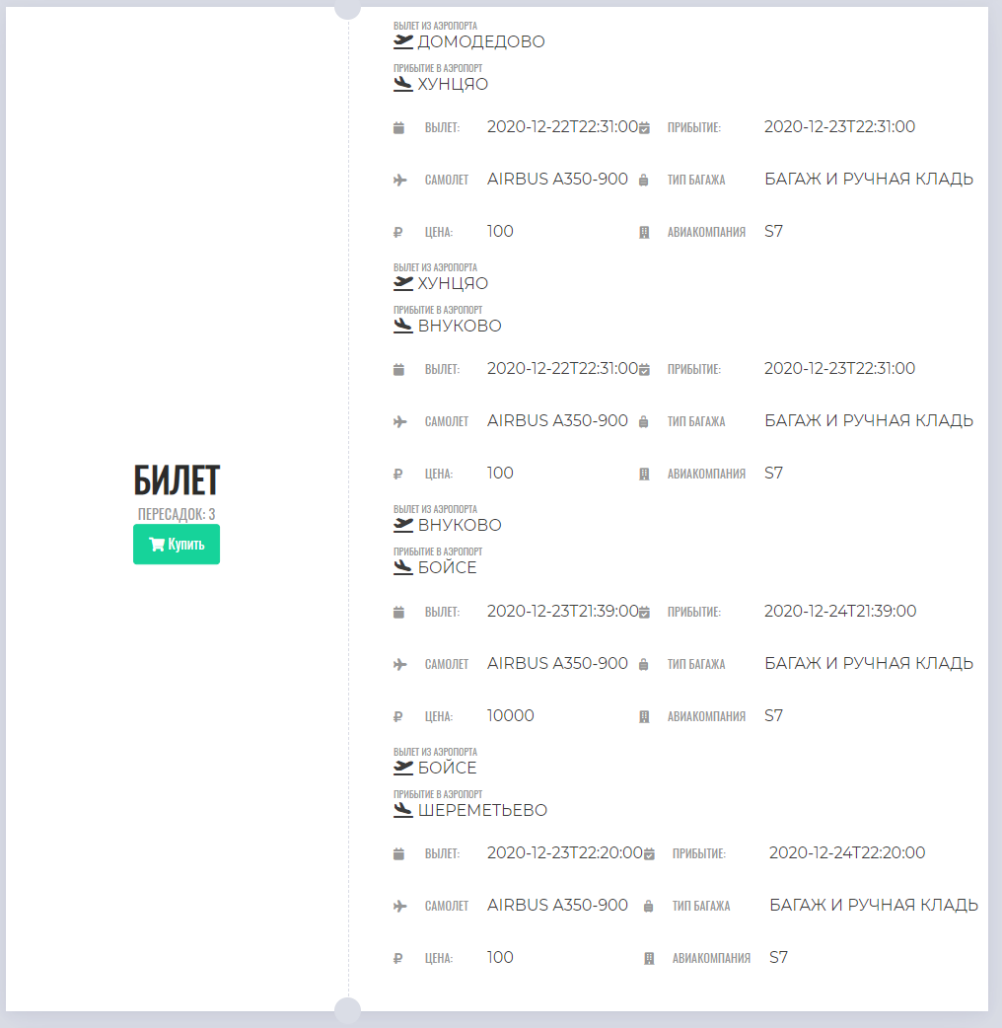
Проверим работу системы регистрации пользователя. Попробуем зарегистрироваться, а затем войти на сайт. После авторизации у нас должно быть 3 ссылки в панели навигации: главная, личный кабинет, выход.

Видим (см. рис. 15), что авторизация и регистрация работает корректно.



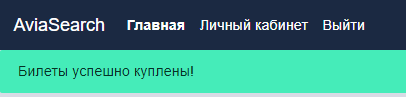
**Рис. 15 Изменение ссылок на панели навигации**

Теперь попробуем купить билет, будучи авторизованными на сайте. Для усложнения задачи, купим билет с тремя пересадками, в таком случае у нас должно появиться 4 билета.

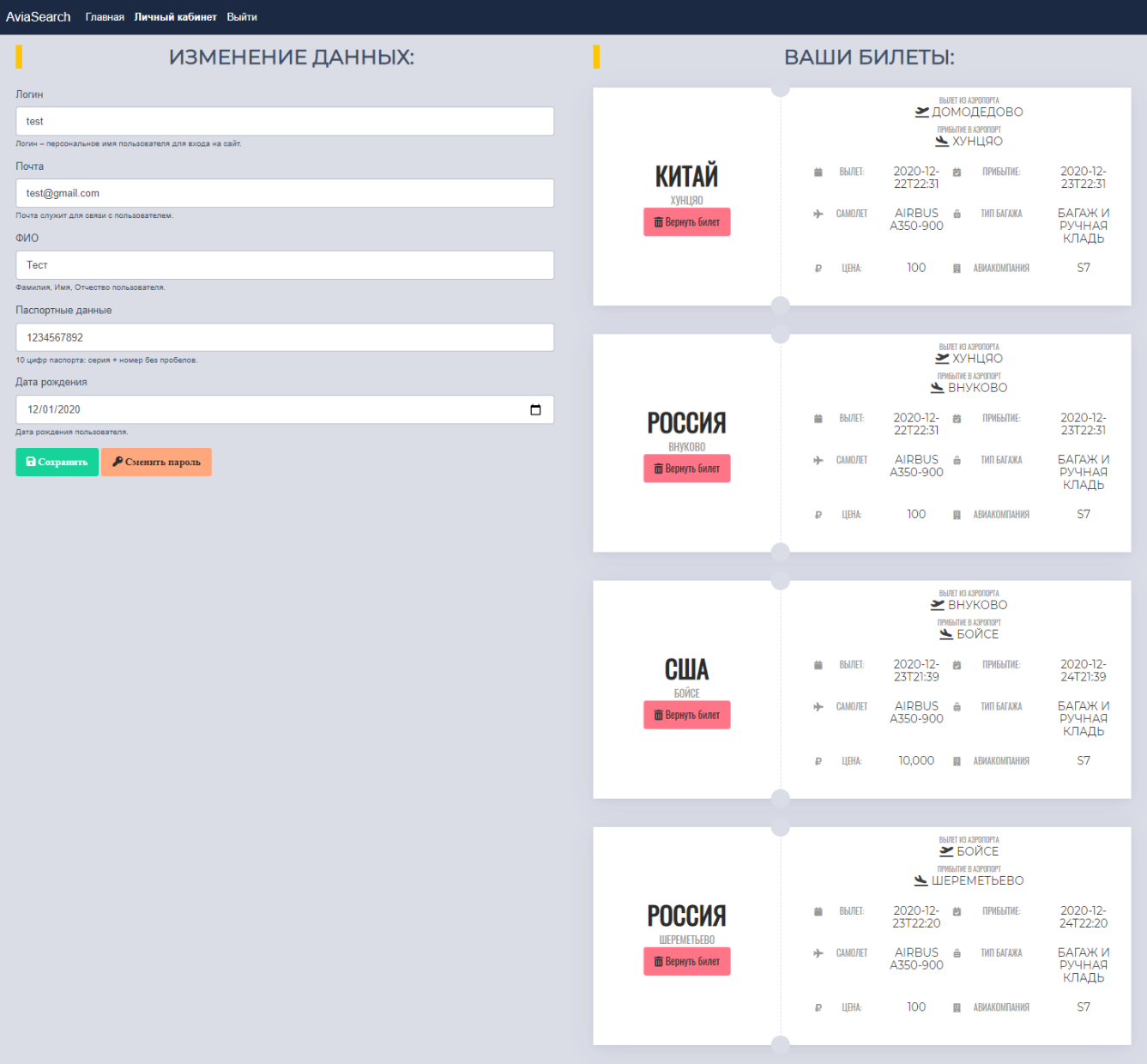


**Рис. 16 Выбранный билет для покупки**

По нажатию на кнопку «Купить» отобразилось всплывающее сообщение о том, что билеты куплены (см. рис. 17). Проверим это перейдя в личный кабинет (см. рис. 18).

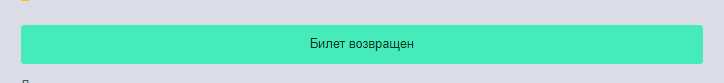


**Рис. 17 Уведомление о том, что билет куплен**



**Рис. 18 Купленные билеты в личном кабинете**

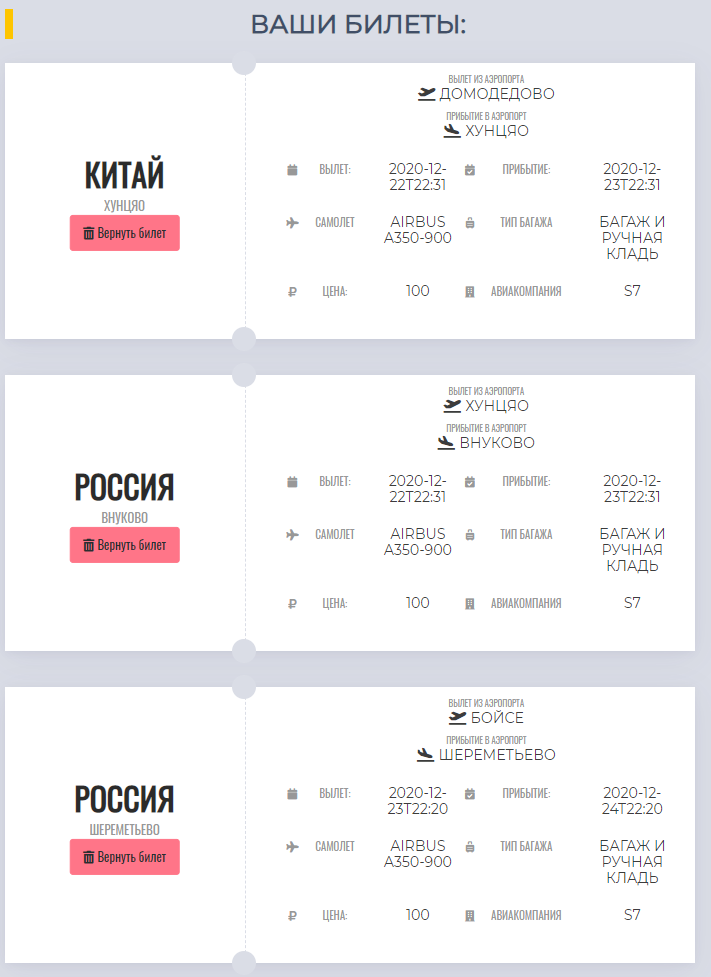
Как мы видим, билеты действительно были куплены. Попробуем вернуть билет Внуково-Бойсе. Вылезло сообщение о том, что билет возвращен (см. рис. 19).



**Рис. 19 Уведомление о том, что билет возвращен**

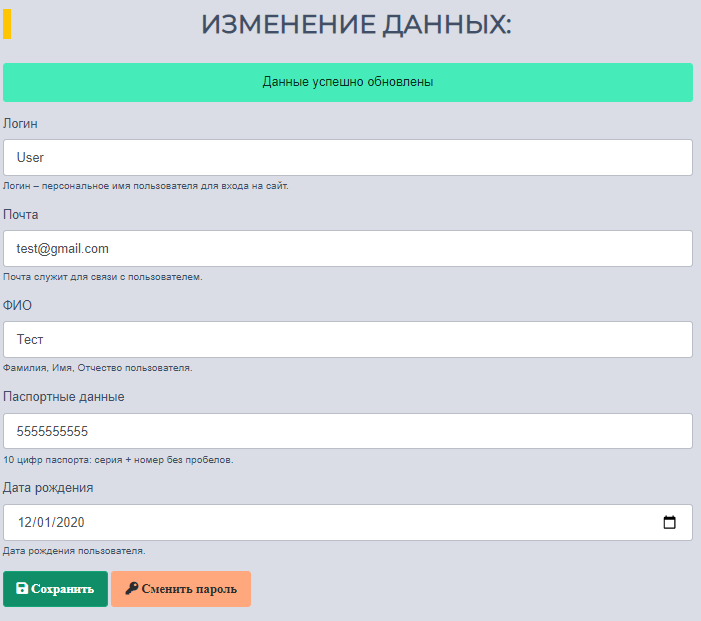
Обновим страницу и проверим, пропал ли билет. Как мы видим, он

действительно пропал (см. рис. 20):



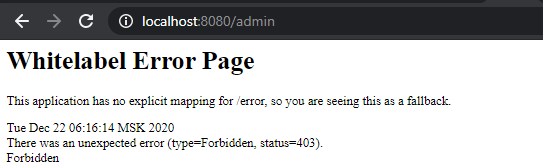
**Рис. 20 Обновленный список билетов**

Попробуем изменить данные пользователя, к примеру, паспортные данные. Как мы видим, они действительно изменились (см. рис. 21):



**Рис. 21 Успешное изменение данных пользователя**

Перейдем к тестированию панели администратора, но перед этим проверим, может ли пользователь, не имеющий привилегии администратора, попасть на нее. Как мы видим (см. рис. 22), возвращается ошибка 403.



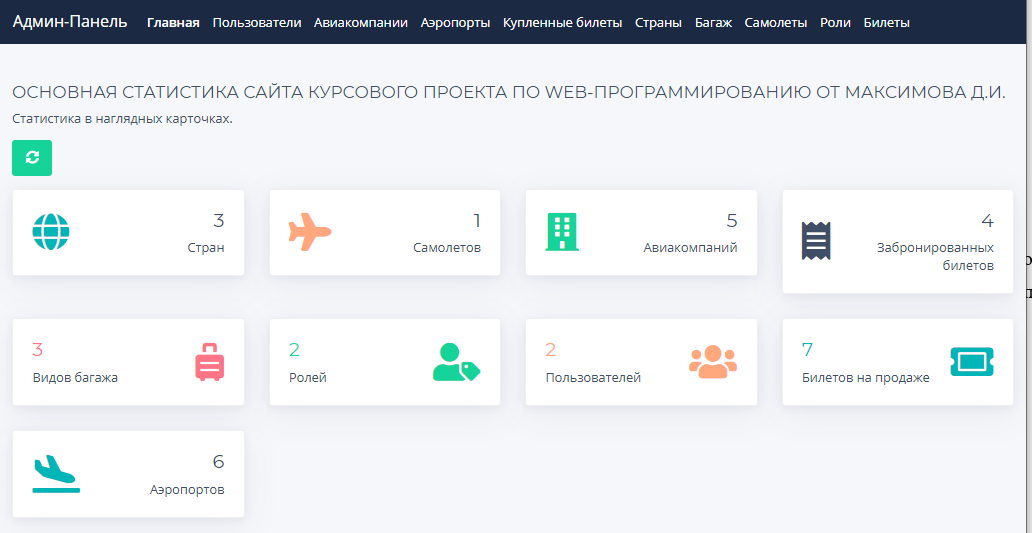
**Рис. 22 Ошибка при входе в панель**

Пробуем авторизироваться с учетной записи администратора и зайти в панель администратора. Как мы видим (см. рис. 24), при авторизации у нас появилась новая ссылка, ведущая на панель.



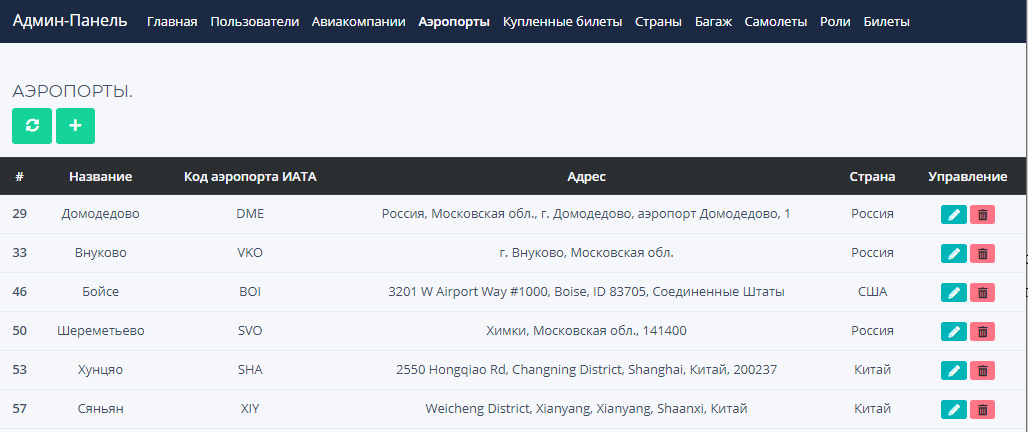
**Рис. 23 Обновленная панель навигации**

Как мы видим, у нас действительно получилось зайти в панель (см. рис. 24):

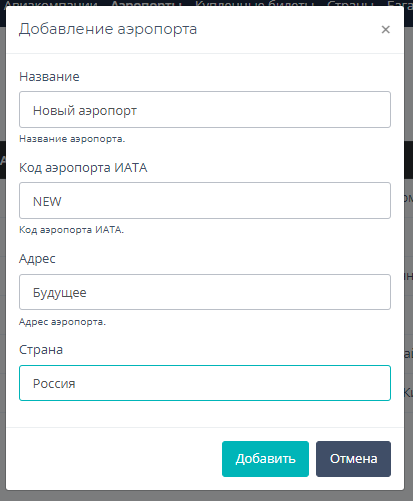


**Рис. 24 Главная страница панели администратора**

Перейдем на любую страницу редактирования данных, к примеру «Аэропорты» (см. рис. 25). Попробуем добавить новый (см. рис. 26-27), отредактировать его (см. рис. 28-30), а затем удалить (см. рис. 31-32).



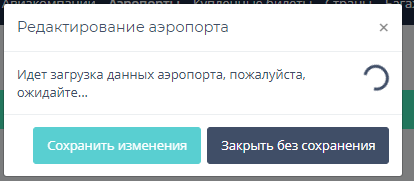
**Рис. 25 Список аэропортов**



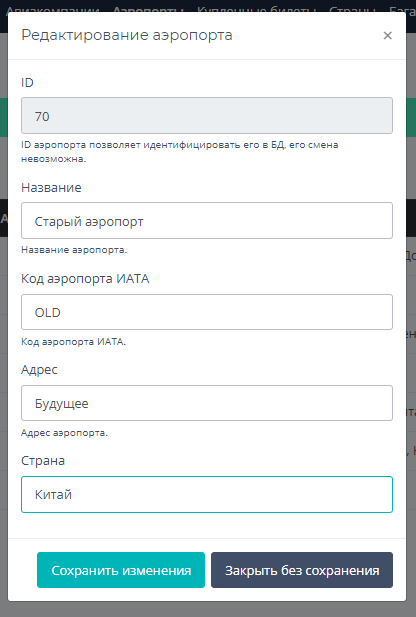
**Рис. 26 Добавление нового аэропорта**



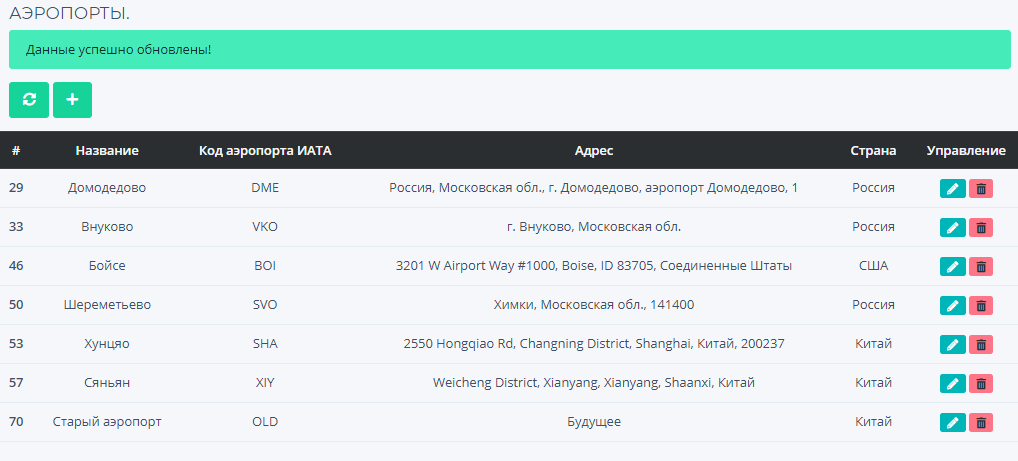
**Рис. 27 Появление нового аэропорта**



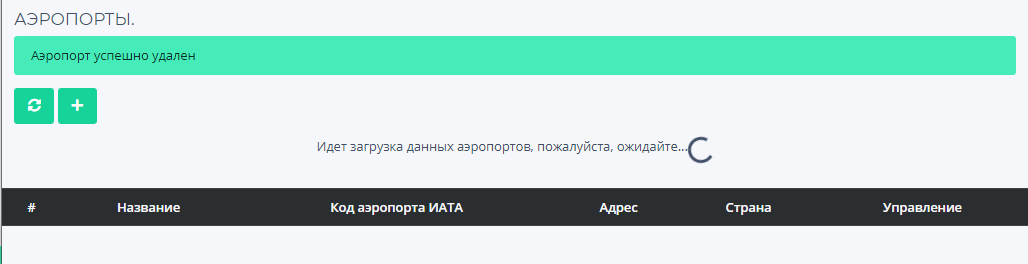
**Рис. 28 Загрузка редактора аэропорта**



**Рис. 29 Редактирование аэропорта**



**Рис. 30 Успешное обновление данных**



**Рис. 31 Удаление аэропорта**



**Рис. 32 Успешное удаление аэропорта**

Заключение

В ходе написания курсовой работы были реализованы все требуемые функции:

* Личный кабинет пользователя с отслеживанием заказов
* Возможность выбора подбора описка билетов с пересадками
* Система управления для редактирования расписаний, наличия билетов, управления пользователями

Данные функции так же успешно прошли проверку на корректную работу.

В случае дальнейшего развития программы, можно добавить расширенный фильтр, возможность покупки билетов на несколько пассажиров одновременно, улучшить дизайн.

# Список литературы

1. Википедия – свободная энциклопедия – AviaSales.ru [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/Aviasales.ru
2. Хабр – Spring MVC — основные принципы [Электронный ресурс] Режим доступа: https://habr.com/ru/post/336816/
3. Хабр – Spring Data JPA [Электронный ресурс] Режим доступа: <https://habr.com/ru/post/435114/>
4. YouTube – Spring Framework – alishev [Электронный ресурс] Режим доступа: https://www.youtube.com/playlist?list=PLAma\_mKffTOR5o0WNHnY0mTjKxnCgSXrZ
5. YouTube – Spring Boot 2 – letsCode [Электронный ресурс] Режим доступа: https://www.youtube.com/playlist?list=PLU2ftbIeotGpAYRP9Iv2KLIwK36-o\_qYk
6. Bootstrap – Documentation [Электронный ресурс] Режим доступа: <https://getbootstrap.com/docs/4.2/getting-started/introduction/>
7. FreeMarker – Manual [Электронный ресурс] Режим доступа: https://freemarker.apache.org/docs/index.html