Національний університет “Одеська політехніка”

Кафедра комп’ютеризовані системи та програмні технології

**КУРСОВА РОБОТА**

з дисципліни «Сучасні інтернет-технології»

Варіант 13

Студента 1 курсу, групи АТ–211

спеціальності «Автоматизація та комп'ютерно-інтегровані технології»

Квапіша О. О.

Керівник: доц. Сперанський В.О.

Національна шкала:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

     Члени комісії  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   \_\_\_\_\_Сперанський В.О.\_\_\_\_\_

(підпис)          (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_    \_\_\_\_\_Ілуца А.С.\_\_\_\_\_\_\_\_\_\_\_

(підпис)          (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)          (прізвище та ініціали)

м. Одеса – 2022 рік

1. Спроектувати структуру інформаційної бази свого варіанта завдання (Додаток А).
2. Створити основну таблицю інформаційної бази як динамічного списку об'єктів.
3. Передбачити функції контролю коректності всіх даних, що вводяться (наприклад, невід'ємний вік, дата продажу не раніше, ніж 100 років тому тощо).
4. Заповнити основну таблицю щонайменше 20 записами.
5. Забезпечити збереження та завантаження даних у зовнішній файл/з файлу.
6. Реалізувати інтерфейс користувача, який забезпечуватиме наступні функції:
7. створення інформаційної бази тієї предметної галузі, яка визначена постановкою завдання;
8. додавання нових записів до бази даних (з контролем коректності значень, що вводяться) в окремому вікні (модальному);
9. пошук записів по ключовому полю;
10. редагування записів бази даних у окремому вікні (модальному);
11. видалення непотрібних записів із підтвердженням дії;
12. збереження даних у файлі;
13. видачу звітів за запитом із можливістю сортування;
14. реалізувати систему нотифікацій в інтерфейсі, яка повідомлятиме про успішність/провальність дій в інтерфейсі (додавання запису, видалення тощо
15. Виконати оригінальну обробку, вказану у постановці конкретного завдання для вашого варіанту (Додаток А).
16. Організувати введення пароля під час операцій змін даних.
17. Додати довідкову систему користувача за створеною програмою.
18. Підготувати пояснювальну записку за розробленим проектом.

**ДОДАТОК (А)**

Разработать информационную базу для автоматизации работы багажного отделения аэропорта.

Каждая запись содержит:

- фамилия, имя и отчество пассажира;

- *номер рейса;*

- номер багажной квитанции;

- количество мест багажа;

- суммарный вес багажа пассажира.

Предусмотреть*:*

а) выдачу сведений о пассажирах, вес багажа которых превышает 30 кг;

б) выдачу сведений о суммарном весе багажа всех пассажиров рейса;

в) выдачу сведений о местонахождении багажа (номер рейса) по заданному номеру багажной квитанции.

г) удаление записей о пассажирах с заданной фамилией.

**ЕТАПИ ПРОЕКТУВАННЯ**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Найменування | Зміст | Дата закінчення |
| 1. | Початок | Вивчення постановочного завдання, предметної галузі | 01.06.2022 |
| 2. | Теоретична підготовка | Вивчення літератури з дисципліни | 05.06.2022 |
| 3. | Подготовка базовых классов | Проработка структуры проекта, создание основного класса Main, написание функций-хелперов. | 10.06.2022 |
| 4. | Создание классов для работы с базой данных | Создание класса для сериализации и десериализации данных в файл JSON | 22.06.2022 |
| 5. | Написание  сервера | Прописывание и тестировка  классов для работы с запросами. Написание методов под них | 27.06.2022 |
| 6. | Создание фронта | Написание пользовательского интерфейса и возможности ввода/вывода данных, модальные окна под добавление и редактирование. | 28.06.2022 |
| 7. | Связь пользовательского интерфейса с бекендом | Корректировка кода под связь между бекендом и фронтенндом | 30.06.2022 |

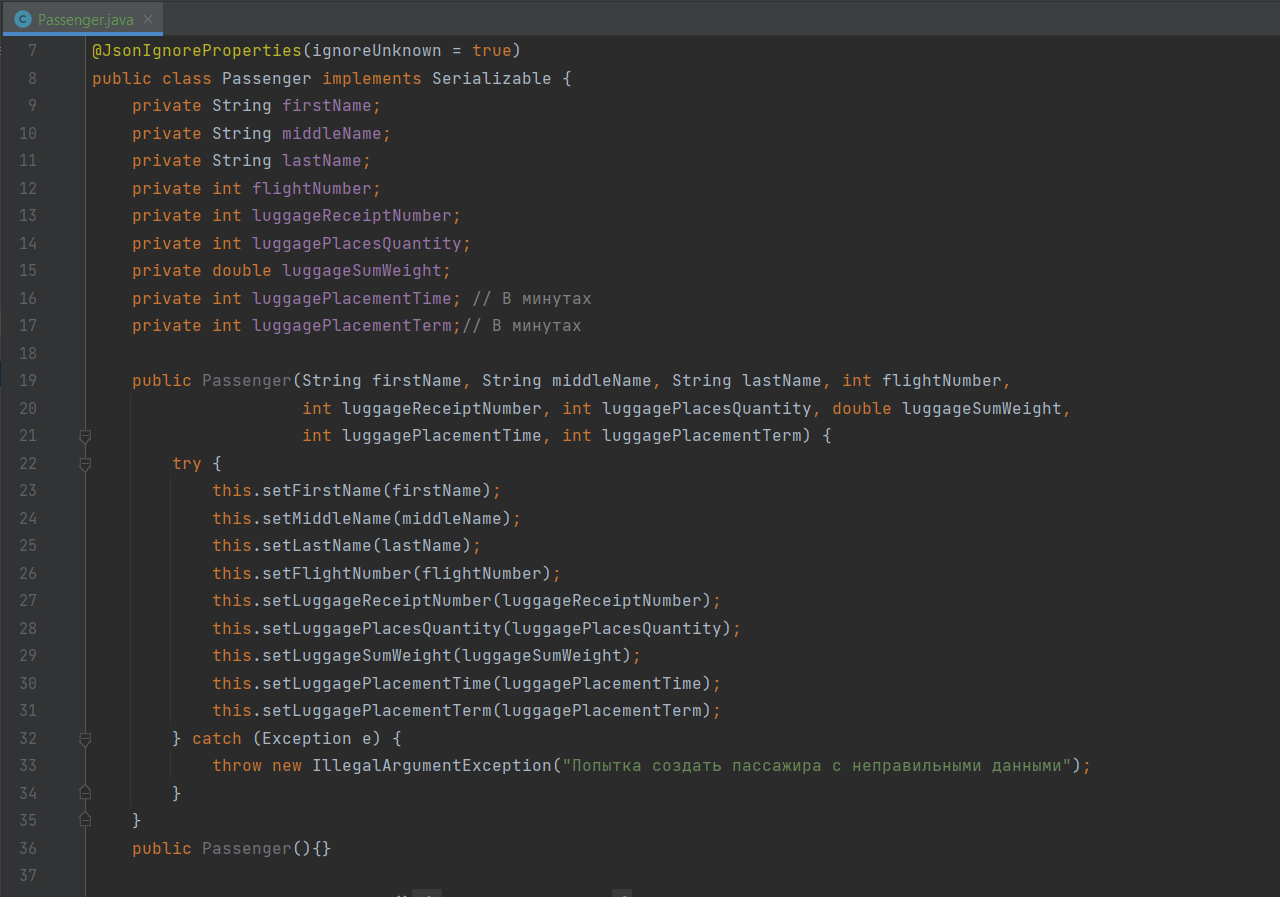
Використані технології:

Java, Jackson, HttpServer, HTML, CSS, Bootstrap, JS

**ДЕТАЛЬНИЙ ОПИС ПРОЕКТУ**

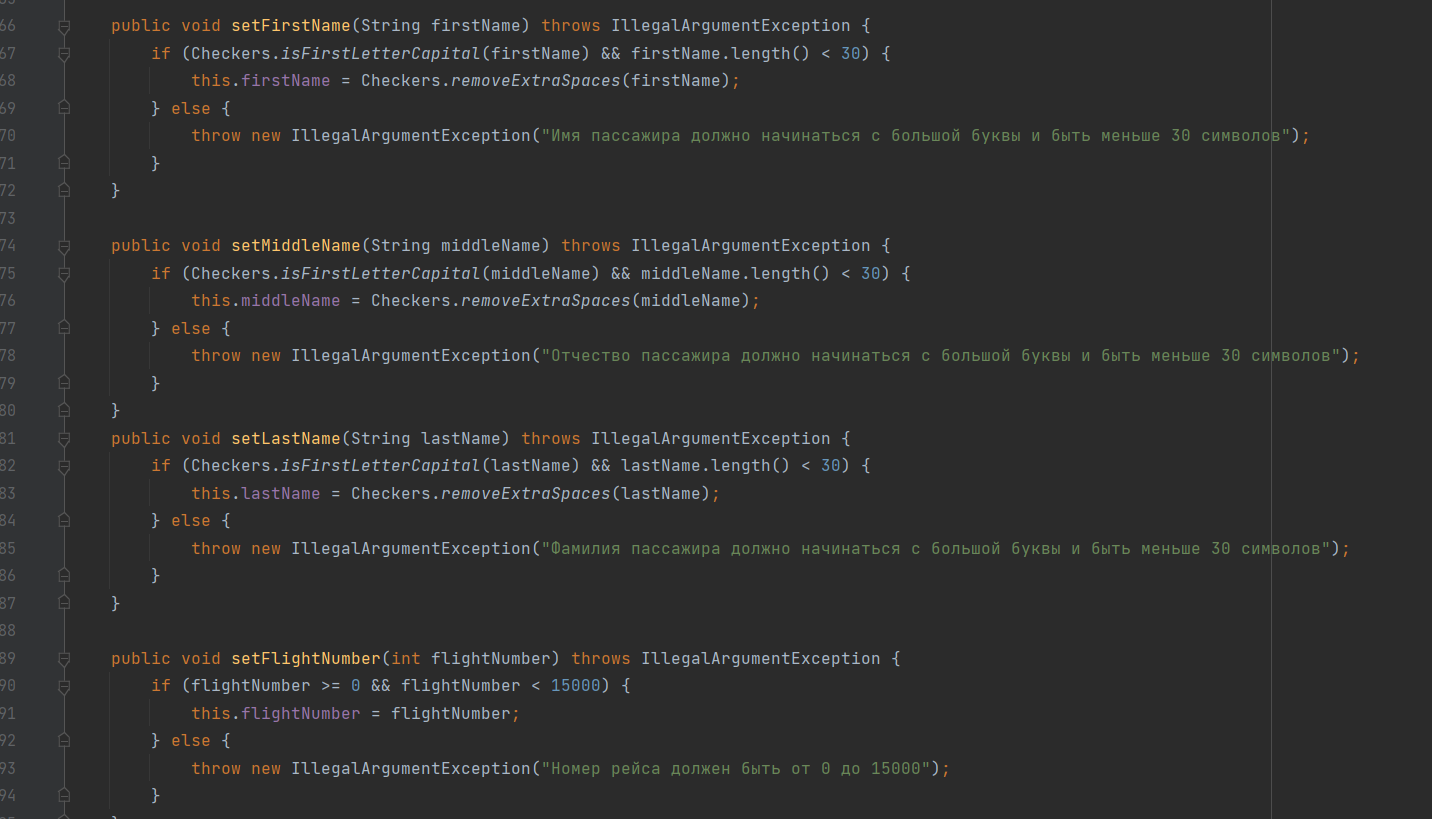
**BACK-END**

1. Создание класса «Passenger», для хранения данных о багаже каждого отдельного пассажира, прописаны поля и конструктор (рис. 1.1).

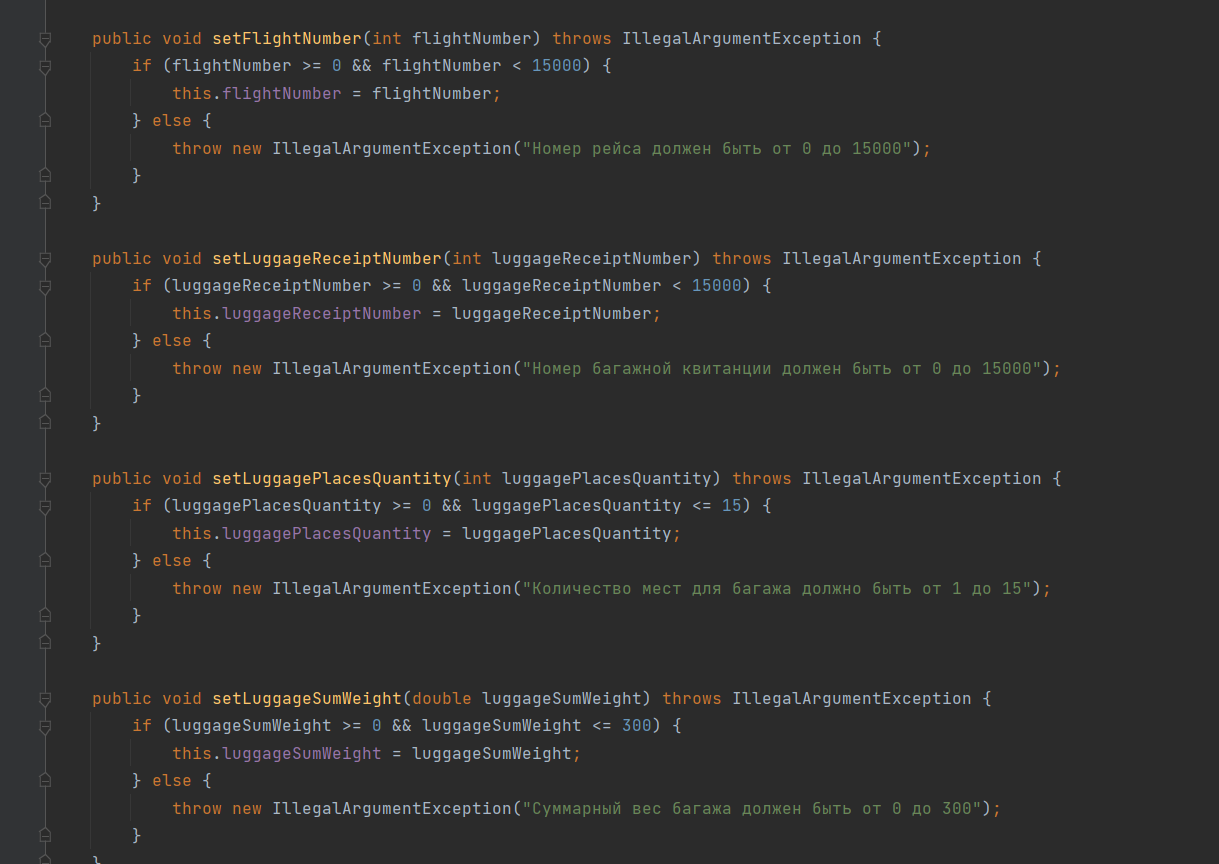


(рис. 1.1)

Также созданы геттеры и сеттеры, в конструкторе используется заполнение только через сеттеры чтобы все данные проверялись при заполнении (рис. 1.2, 1.3, 1.4).



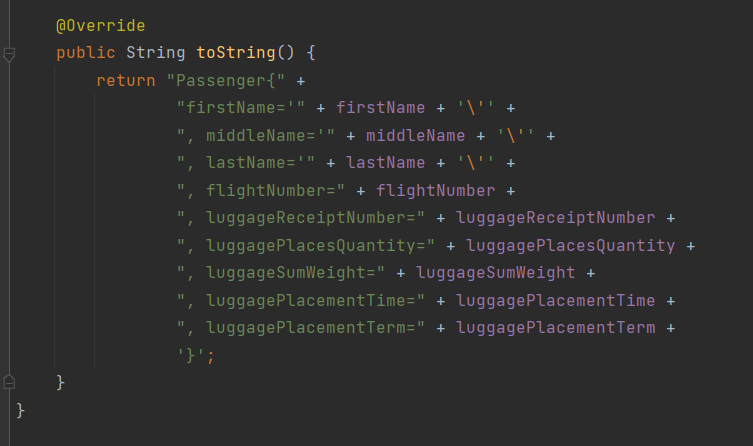
(рис. 1.2)

 (рис. 1.3)

(рис. 1.4)

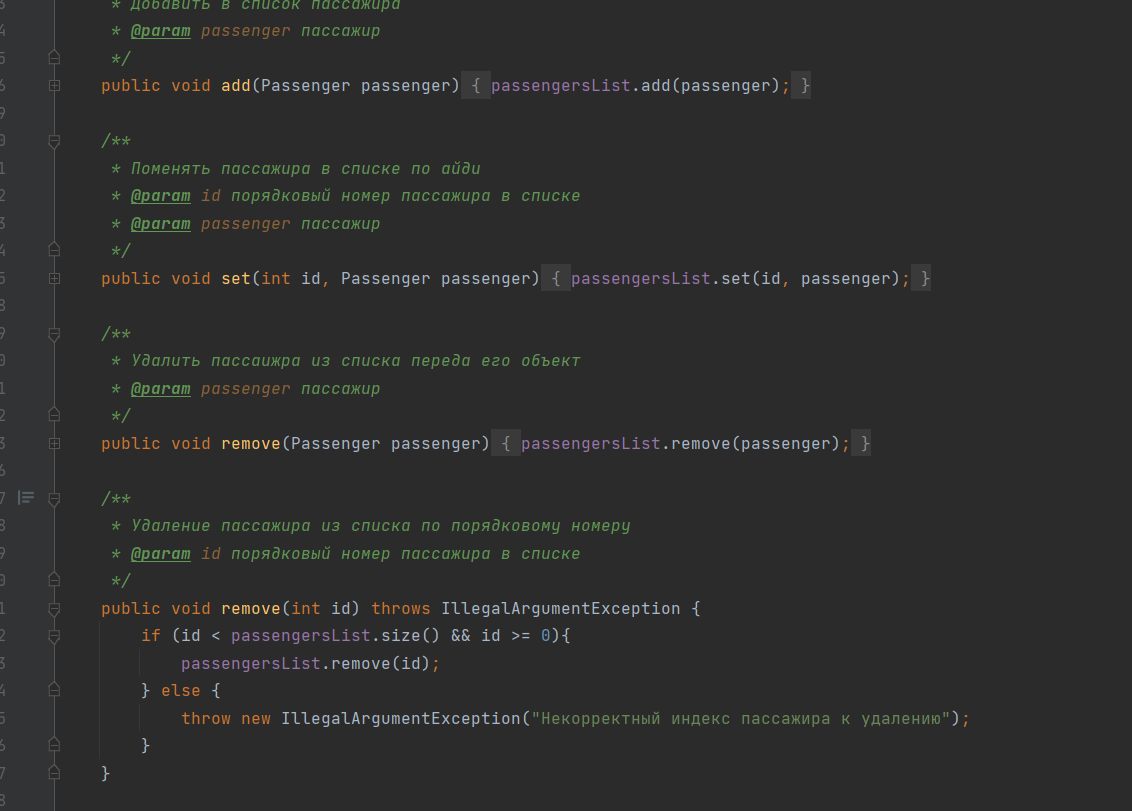
Сортировка и прочие операции, которые не требуют редактирования данных по возможности перенесены на фронтенд, чтобы не нагружать бекенд дополнительными запросами.

Также переопределяем для класса метод toString:



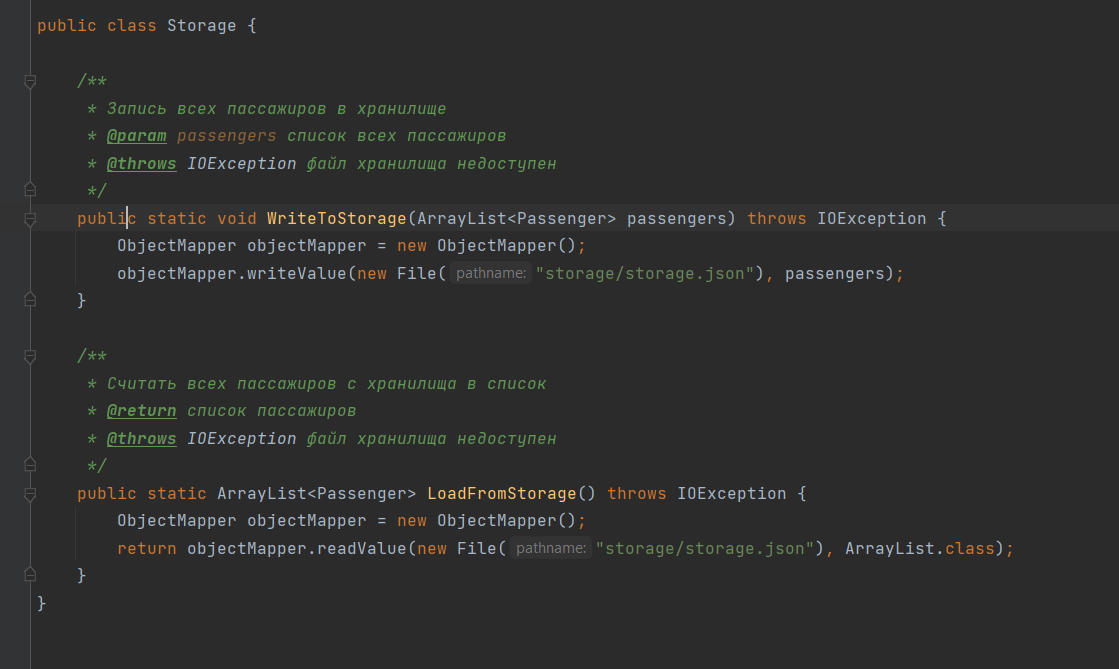
(рис. 1.5)

2. Создаем класс «Passengers» где храним список всех багажей пассажиров (ArrayList of Passenger) для применения множественных действий (удаление, сортировка и т.д.). Создаем методы добавления объектов в список, удаления из него объектов по айди и по копии объекта, и перезаписи по айди. (см. рис. 1.6)



(рис. 1.6)

3. Был создан класс «Storage», где прописаны методы сериализации и десериализации списка. (как одно из требований Jackson, классы «Passenger» и «Passengers» реализуют интерфейс «java.io.Serializable» для добавления возможности сериализации и десериализации) (см. рис. 1.7)



(рис. 1.7)

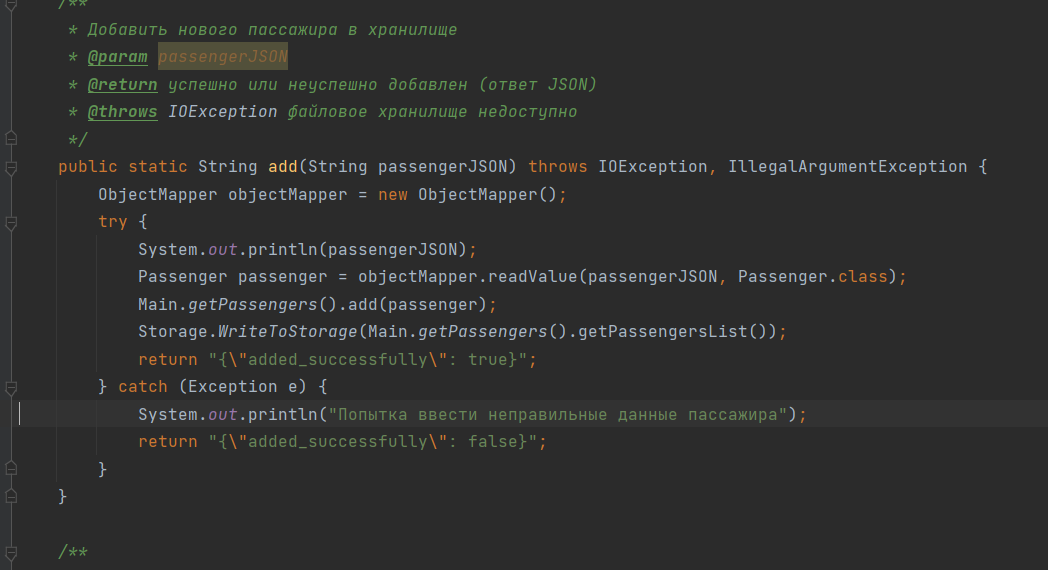
4. Был создан класс «Servlet», занимающийся управлением сервером, получением, отправкой и редактированием данных. Прописаны методы работы под этого. (рис. 1.8, получение запросов и передача их на Handling)



(рис. 1.8)

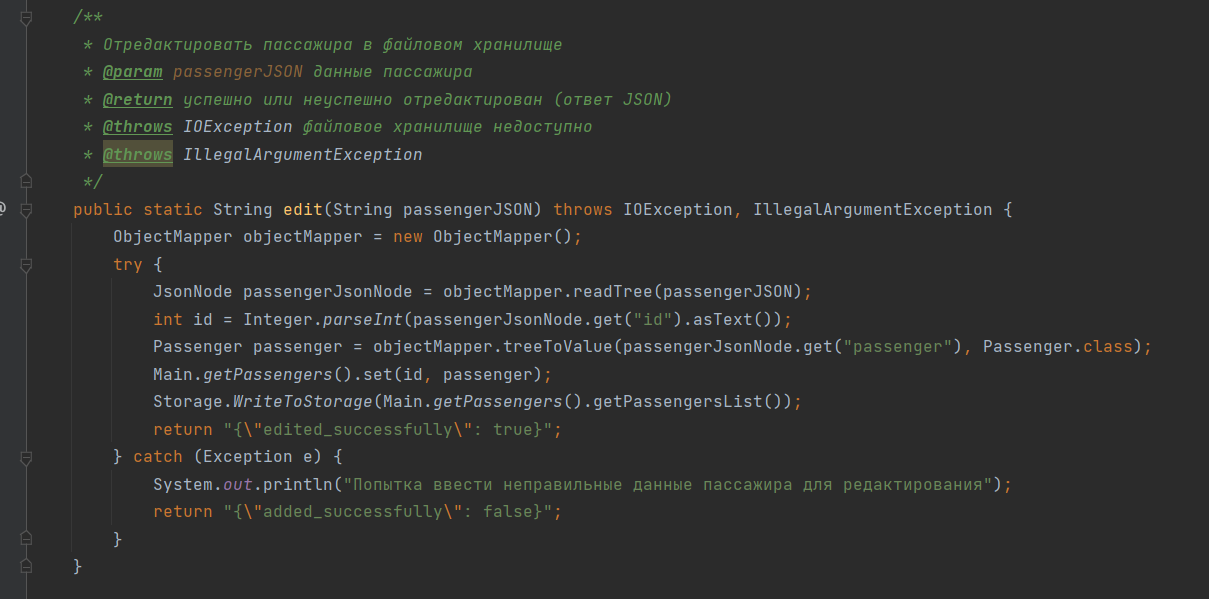
Для разграничения ссылок для получения запросов (эндпоинты) и их обработки был создан класс «Handling».

Так работает обработка запроса на добавление нового пассажира в список (add). JSON строка содержит поля багажа пассажира и их значения для дальнейшего создания объекта Passenger, затем происходит десериализация и сохранение значений во временную переменную passenger, десериализация основного файла базы данных во временный список объектов, и добавление нового объекта в список с последующей сериализацией и записью обновленного списка в файл. (см. рис. 1.9)



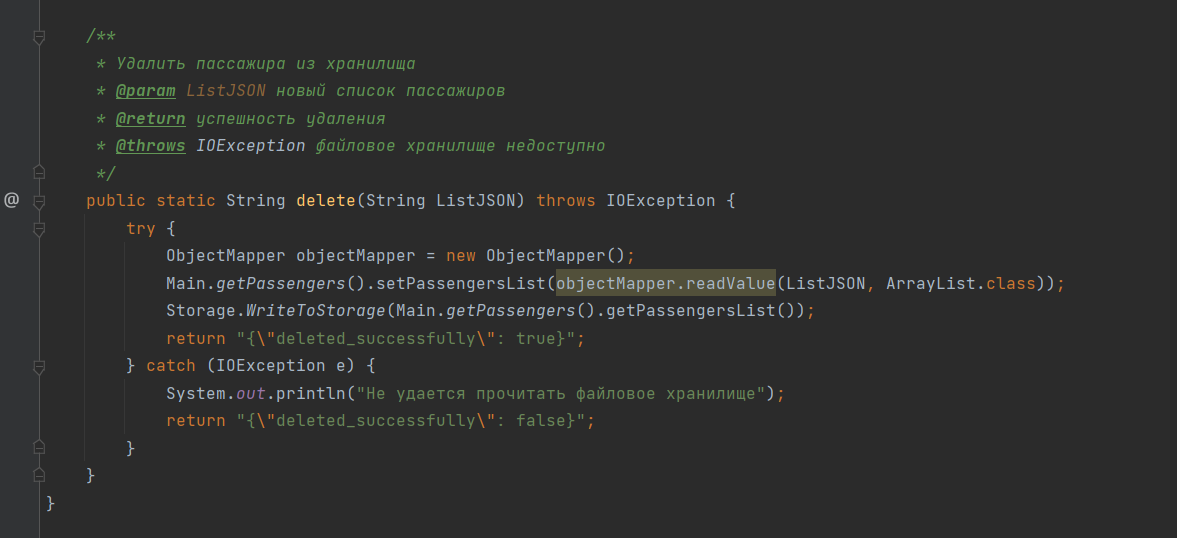
(рис. 1.9)

Таким же образом работает и запрос на редактирование, парсим полученный с фронтенда JSON-объект и получаем айди и все поля объекта используя методы Jackson. Записываем полученные данные вместе со всеми необходимыми проверками в хранилище и отправляем на фронтенд ответ что данные успешно были записаны в хранилище (см. рис. 1.10)



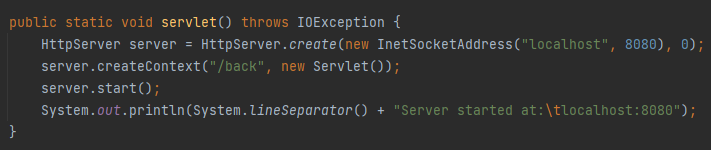
(рис. 1.10)

Запрос на удаление пассажира. От фронтенда сервер получается список всех пассажиров за исключением текущего, записывает его во временный объект, а затем в хранилище, и после отправляет ответ о корректном выполнении операции. (см. рис. 1.11)



(рис. 1.11)

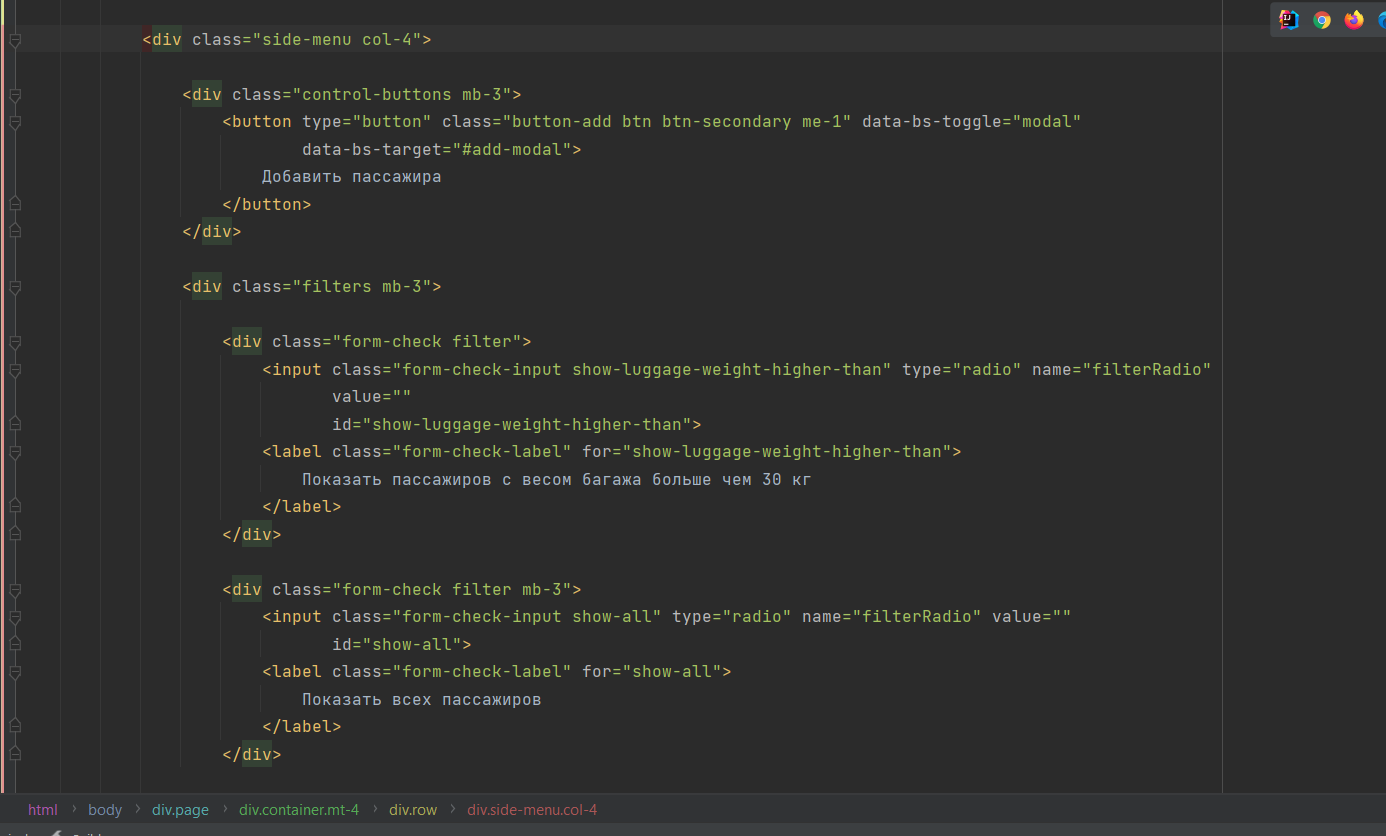
Чтобы сервер запустился, прописываем в классе Main метод для запуска сервера на http://localhost:8080/back.

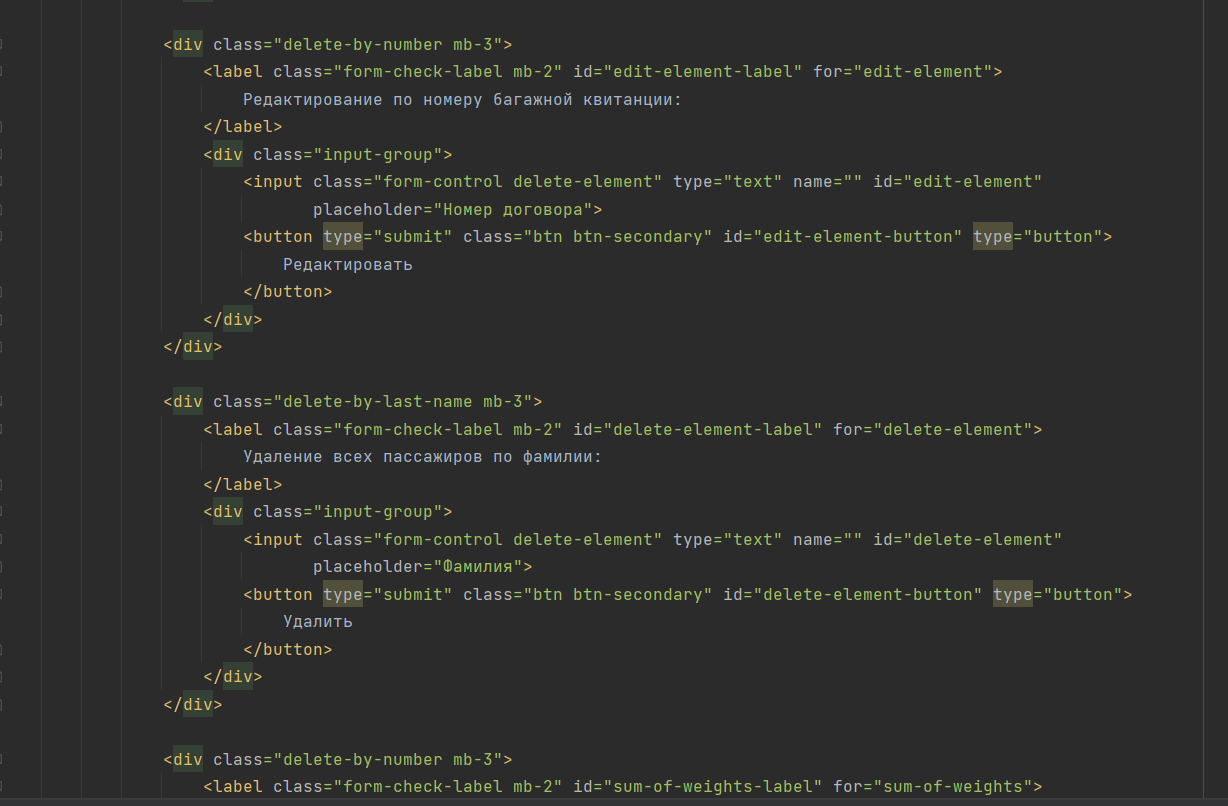


(рис. 1.12)

**FRONT-END**

1. Был написан клиентский интерфейс для отображения, изменения, сортировки, и поиска информации, создания модальных окон и их связи между собой. Для стилизации использовался Bootstrap вместе с его джаваскрипт-библиотекой bootstrap-bundle.min.js.

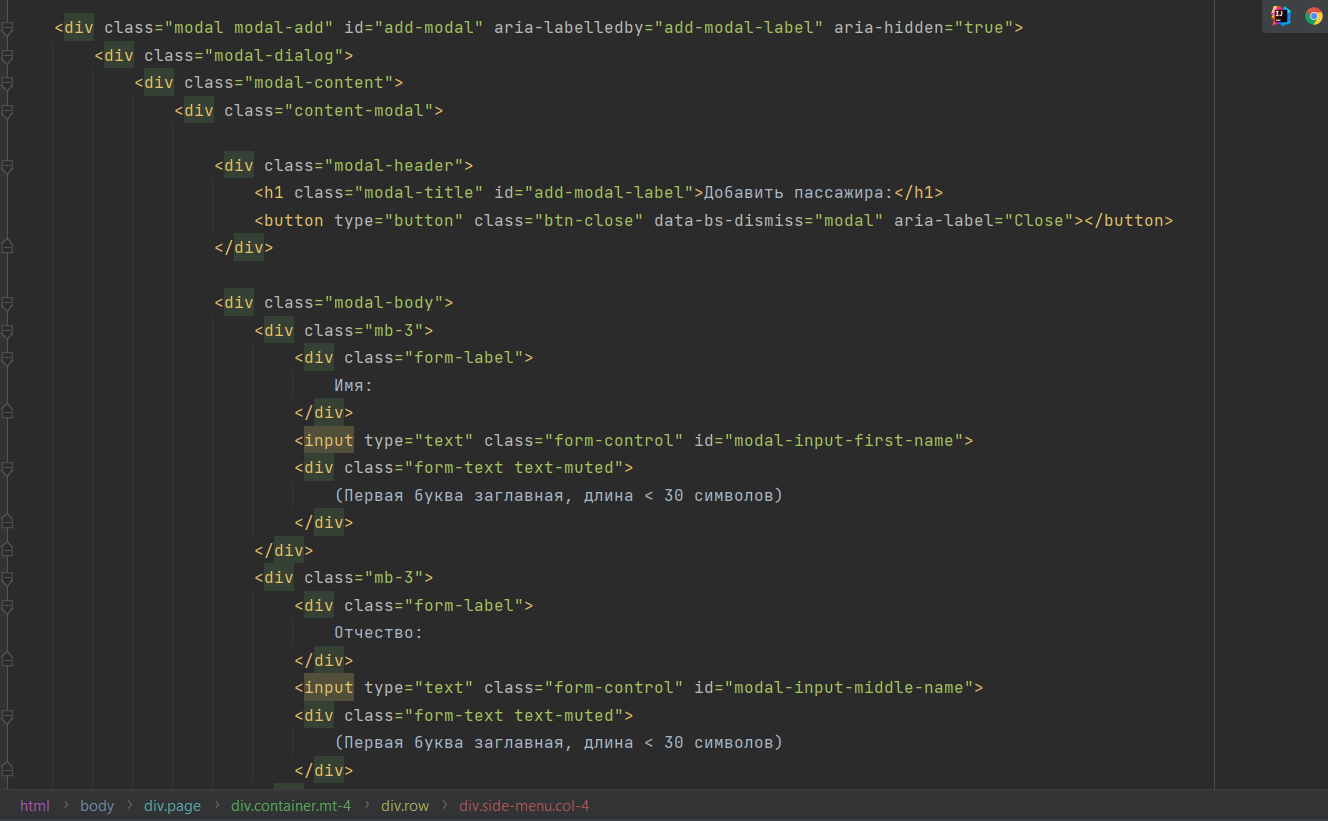
 (рис. 2.1, добавление и фильтры пассажиров)



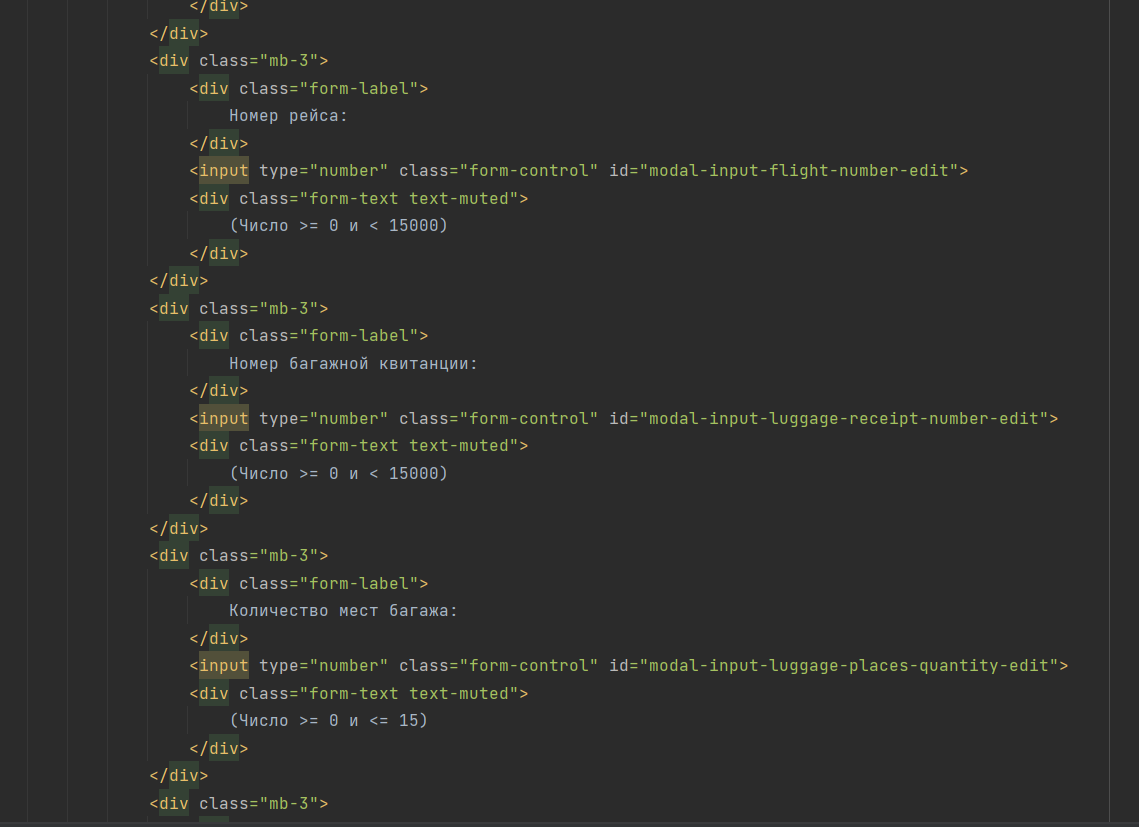
(рис. 2.2, редактирование пассажиров, удаление всех пассажиров по заданной фамилии)

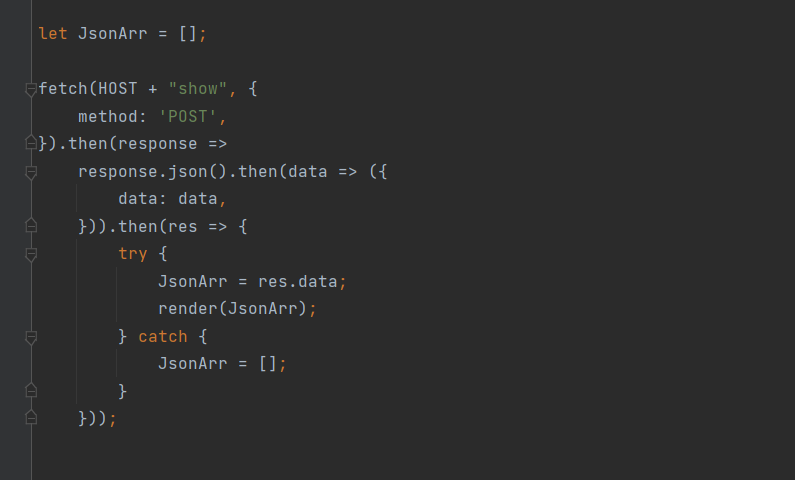


(рис. 2.2, таблица пассажиров, наполняется джаваскриптом с данных, полученных с бекенда)



(рис. 2.3, модальное окно добавления пассажира)

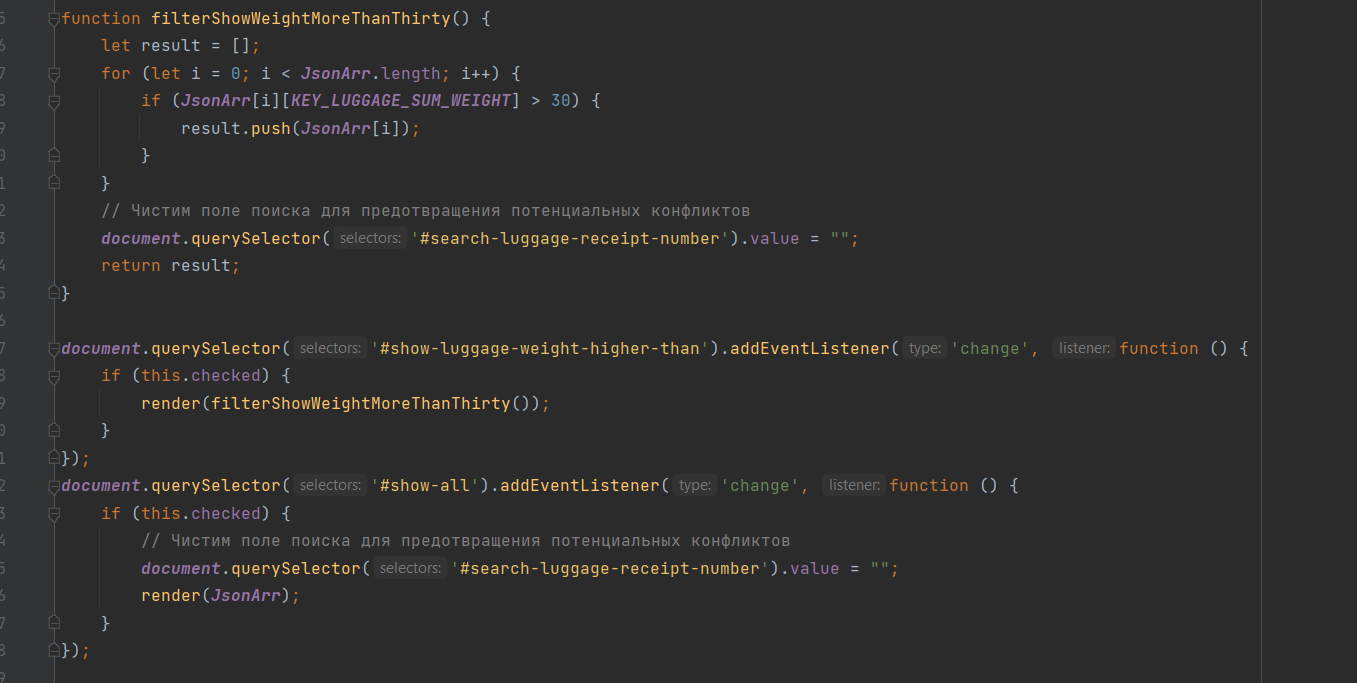
 (рис. 2.4, модальное окно для редактирования пассажира)



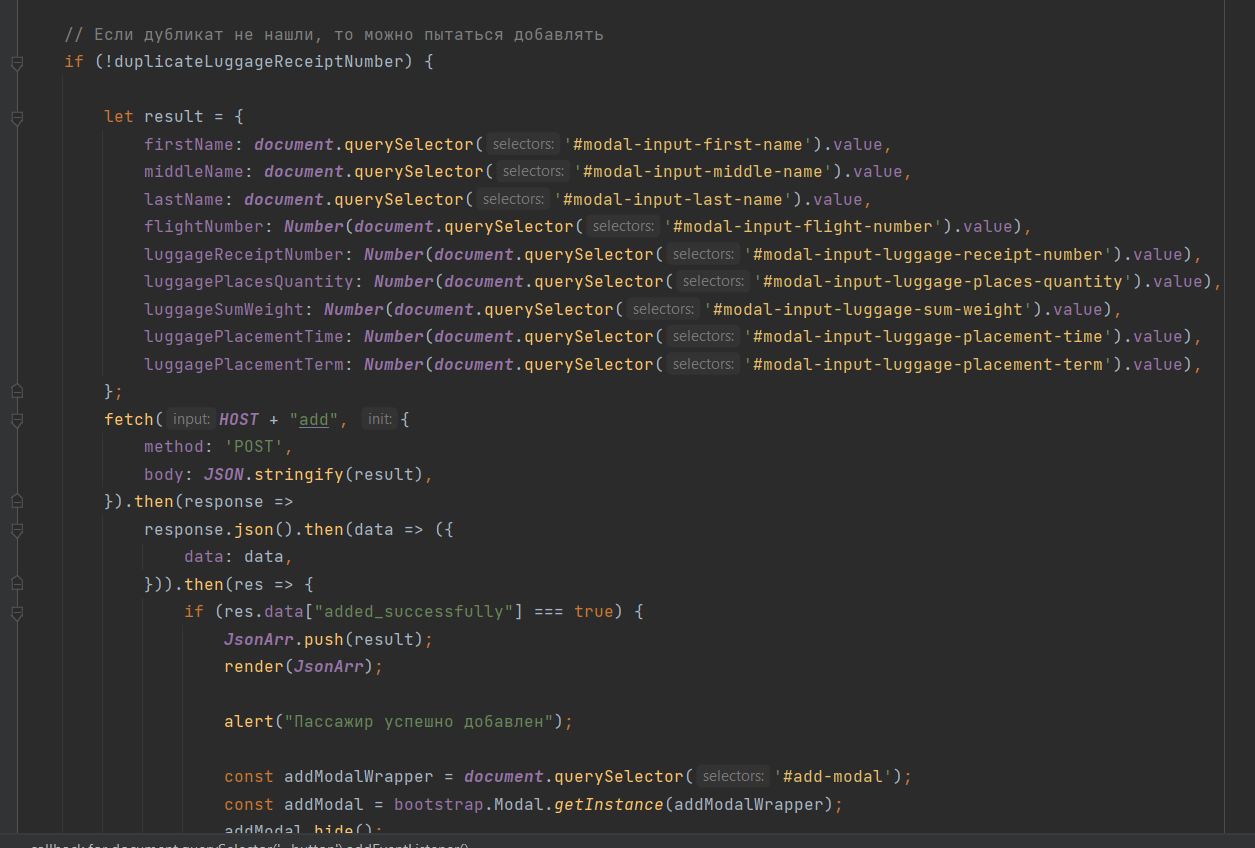
(рис. 2.5, изначальное отображение данных)



(рис. 2.6, функция рендеринга таблицы, вызывается как минимум один раз при первой загрузке)



(рис. 2.7, фильтры)

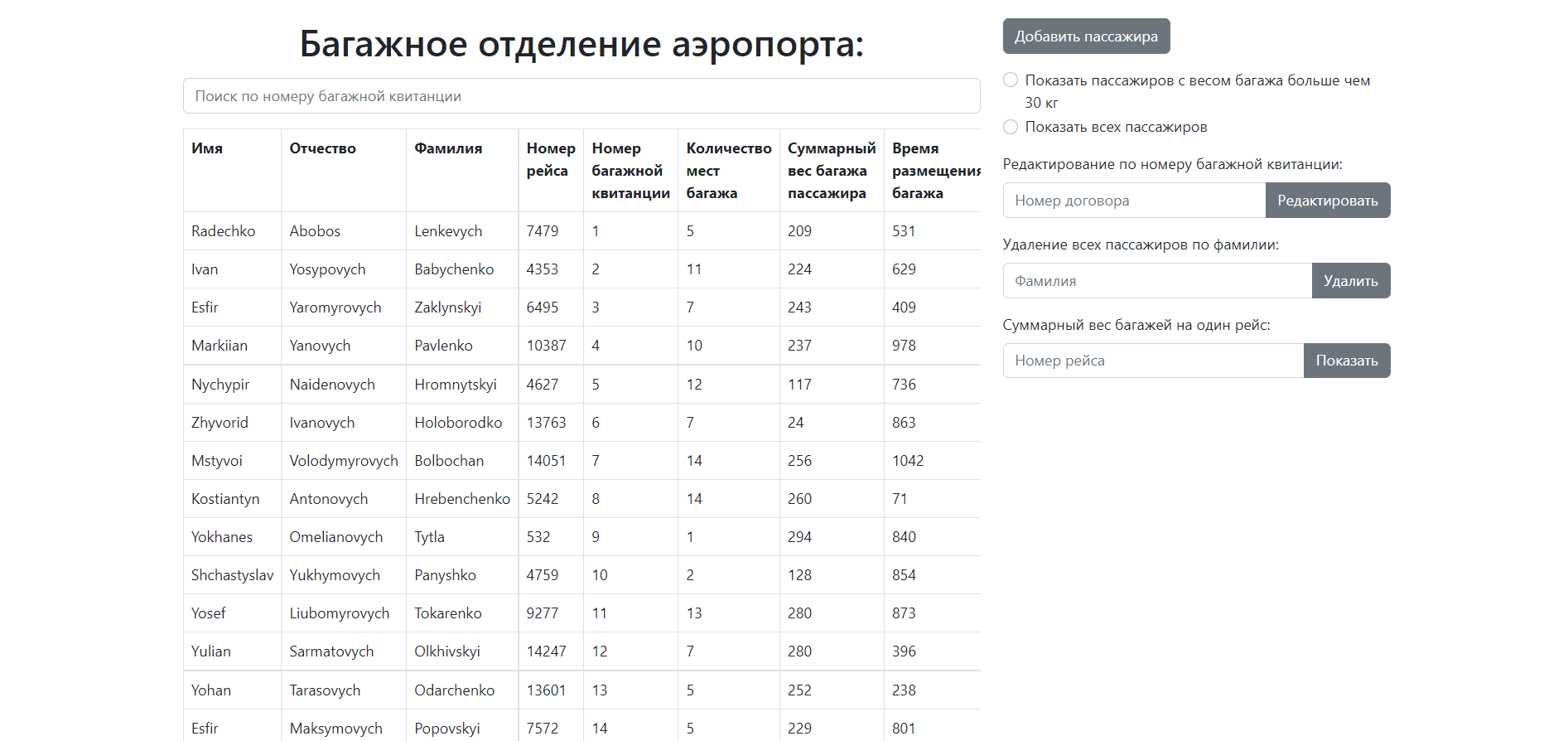


(рис. 2.10, добавление нового элемента, значения берутся с полей для ввода)

**ПОЛЬЗОВАТЕЛЬСКАЯ ИНСТРУКЦИЯ**

ГЛАВНАЯ СТРАНИЦА

Общий вид главной страницы (см. рис. 3.1)

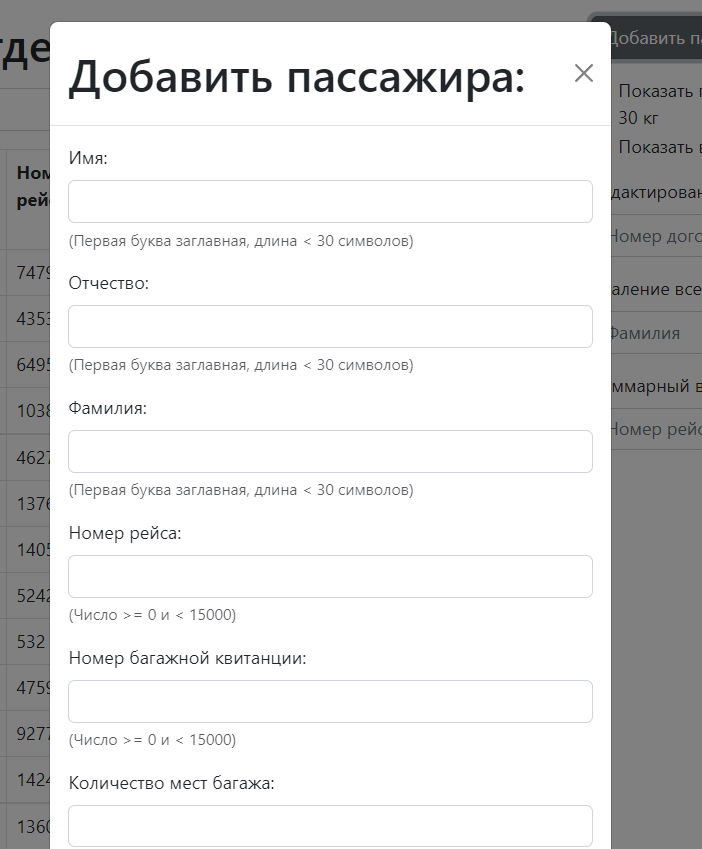


(рис. 3.1, общий вид главной страницы)

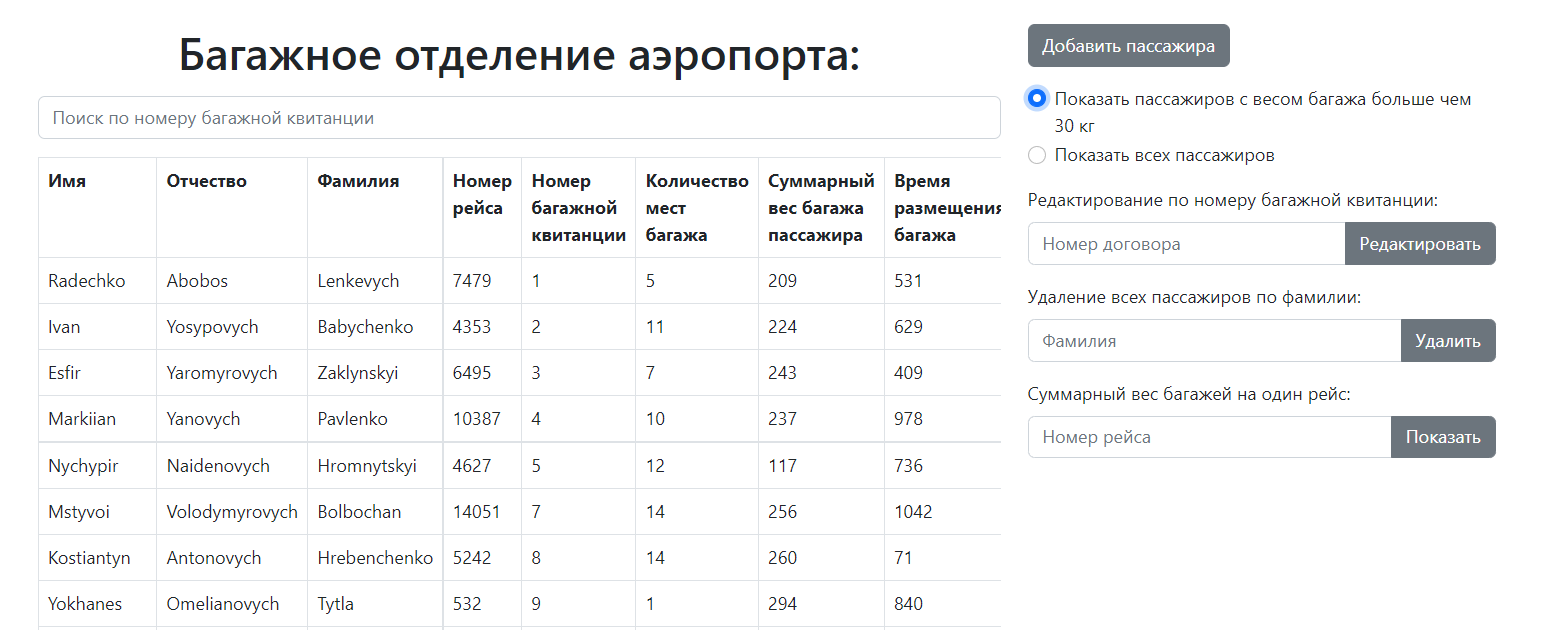
Основные объекты главной страницы (см. рис. 3.1):

1. Таблица, отображающая багаж всех пассажиров
2. Поиск по номеру багажной квитанции
3. Кнопка вызова модального окна для добавления пассажира
4. Фильтры на радиокнопках
5. Поле для ввода номера багажной квитанции для вызова модального окна для редактирования этого багажа
6. Удаление всех пассажиров по фамилии
7. Показать суммарный вес багажа на введенный номер рейса

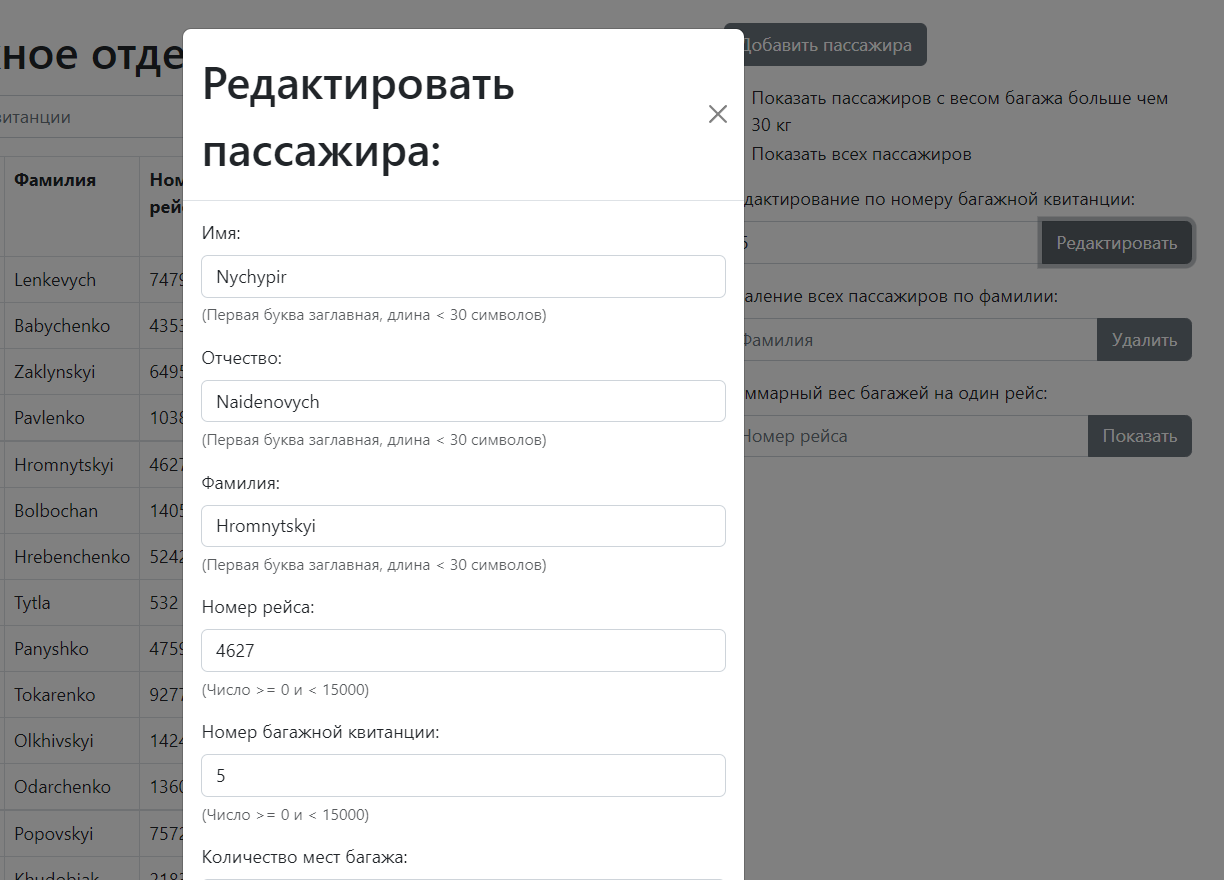
ДОБАВЛЕНИЕ ПАССАЖИРА



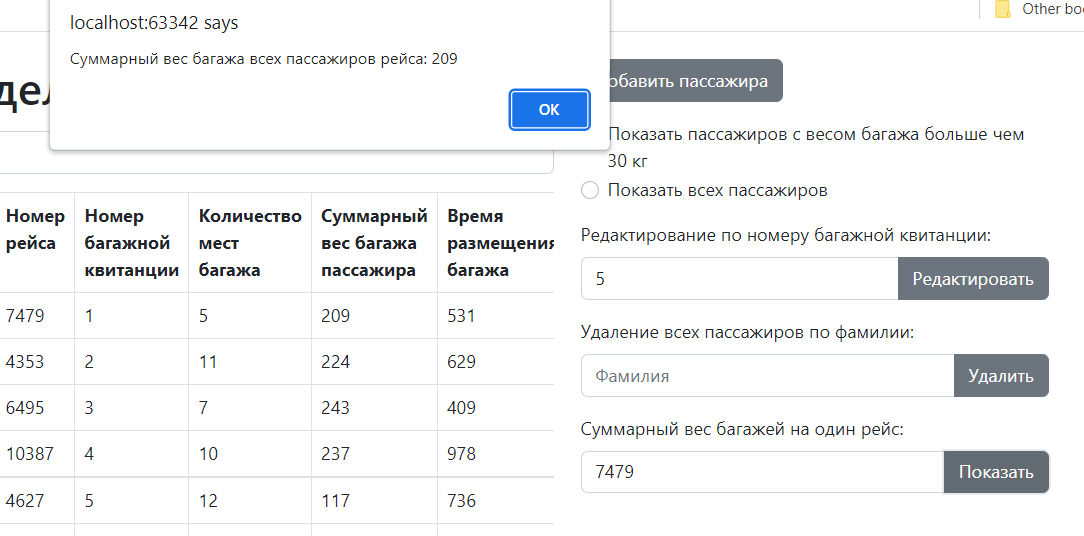
(рис. 3.2, модальное окно для добавления)



(Рис 3.3, пример сортировки радиокнопками)



(Рис 3.4, пример редактирования багажа по номеру квитанции)



(рис. 3.3, пример суммарного веса багажа на один рейс)

**ВЫВОД**

При выполнении курсовой работы была создана база данных и веб-приложение для просмотра/редактирования/удаления данных. Для выполнения поставленной задачи были использованы современные технологии программирования (Java, Jackson, Bootstrap, CSS, JS)

Были исследованы процесс и требуемые навыки создания локального сервера и его связи с клиентским интерфейсом.

Была исследована и более подробно понята сущность веб-приложений, интернет-технологий, серверов и построения пользовательского интерфейса.

**ИСТОЧНИКИ**

1. Документация HTTP-сервера:

<https://docs.oracle.com/javase/8/docs/jre/api/net/httpserver/spec/com/sun/net/httpserver/HttpServer.html>;

5. Пример использования HttpServer:

<https://github.com/it-school/SimpleServer.git>;

2. Jackson документация:

<https://github.com/FasterXML/jackson-docs>

3. Использование Fetch Api:

<https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch>;

5. Использование Json в Javascript:

<https://learn.javascript.ru/json>

4. Компиляция проекта в .jar файл:

<https://www.jetbrains.com/help/idea/working-with-artifacts.html#build_artifacts>;6.

7. Java - The Complete Reference. Helbert Shield