

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Факультет  
вычислительной техники

Кафедра  
«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация  
автоматизированных систем специального назначения»  
Специализация: «Эксплуатация вычислительных машин, комплексов,  
систем и сетей специального назначения»

**ДИПЛОМНЫЙ ПРОЕКТ**

на тему

**СИСТЕМА РЕГИСТРАЦИИ ПАССАЖИРОВ**  
**АЭРОПОРТА**

Студент С.Гурин 02.06.2025 Буренков Сергей Владимирович  
(подпись, дата) (ФИО полностью)

Руководитель В.Егоров 09.06.2025 Егоров В.Ю.  
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр А.С.Бычков 09.06.2025 Бычков А.С.  
(подпись, дата) (фамилия, инициалы)

Рецензент глав. специалист АО "НПП "Рубин" Василий Васильевич А.В.  
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 28.05.2025 г. № 16)

Заведующий кафедрой М.А.Митрохин Митрохин М.А.  
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо  
(протокол заседания ГЭК от 19.06.2025 № 1)

Секретарь ГЭК Е.И.Гурин Гурин Е.И.  
(подпись) (фамилия, инициалы)

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**  
**ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ**

Факультет

Вычислительной техники

Кафедра

Вычислительная техника

«Утверждаю»  
Заведующий кафедрой  
Митрохин М.А.  
«06» 05 2025 г.

**ЗАДАНИЕ  
НА ДИПЛОМНЫЙ ПРОЕКТ**

1. Студент Буренков Сергей Владимирович гр. 20ВОЭ1 факультета вычислительной техники.

2. Тема работы: «Система регистрации пассажиров аэропорта».

Тема утверждена приказом ПГУ № 601/о от 06.05.2025 г.

3. Руководитель работы к.т.н., доцент каф. ВТ «ПГУ» Егоров В.Ю.

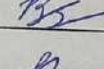
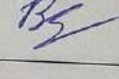
4. Задание на работу (назначение разработки, исходные данные и т.п.).

Разработать систему регистрации пассажиров аэропорта. Система регистрации пассажиров аэропорта должна выполнять следующие функции: добавление, изменение и удаление рейсов на самолет; добавление, изменение и удаление пассажиров; автоматическое добавление свободных мест на самолет при создании рейса, мониторинг занятости мест, добавление рейсов в корзину и удаление рейсов из корзины. Система регистрации пассажиров аэропорта должна разрабатываться с помощью среды разработки «Visual Studio Code». База данных должна разрабатываться для СУБД «PostgreSQL».

### 5. Перечень подлежащих разработке вопросов.

Требуется провести анализ предметной области, определить функциональный состав программы, выбрать стек технологий для реализации программы, разработать диаграммы: вариантов использования и развертывания, спроектировать модели базы данных: концептуальная, логическая и физическая, разработать структуры программы: клиентская и серверная части, с последующим написанием программного кода.

### 6. Календарный график выполнения работы

| № п/п | Наименование этапов работы           | Объем работы | Срок выполнения         | Подпись руководителя                                                                  |
|-------|--------------------------------------|--------------|-------------------------|---------------------------------------------------------------------------------------|
| 1.    | Анализ предметной области            | 10           | 06.05.2025 – 09.05.2025 |    |
| 2.    | Проектирование системы и базы данных | 10           | 10.05.2025 – 12.05.2025 |    |
| 3.    | Разработка системы и базы данных     | 30           | 13.05.2025 – 20.05.2025 |   |
| 4.    | Отладка и тестирование               | 10           | 21.05.2025 – 23.05.2025 |  |
| 5.    | Разработка экономической части       | 5            | 24.05.2025              |  |
| 6.    | Разработка БЖД раздела               | 5            | 25.05.2025              |  |
| 7.    | Оформление графической части         | 10           | 26.05.2025 – 28.05.2025 |  |
| 8.    | Оформление выпускной работы          | 20           | 29.05.2025 – 02.06.2025 |  |

Дата выдачи задания 06.05.2025 г.

Руководитель дипломного проекта

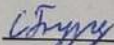


06.05.2025

(подпись, дата)

Егоров В.Ю.

Задание к исполнению принял студент



06.05.2025

(подпись, дата)

Буренков С.В.

Работу к защите допустить

Декан факультета



16.06.2025

(подпись, дата)

Фионова Л.Р.

## Реферат

Пояснительная записка 125 листов, 7 рисунков, 24 таблиц и 4 приложения.  
СИСТЕМА РЕГИСТРАЦИИ ПАССАЖИРОВ АЭРОПОРТА, БАЗА ДАННЫХ,  
СЕРВЕР, КЛИЕНТ

Объектом исследования является процесс контроля работы аэропорта, учет рейсов, мест на самолет и пассажиров.

Цель работы – разработка системы регистрации пассажиров аэропорта.

В процессе работы необходимо выполнить анализ предметной области, сформулировать техническое задание для разрабатываемой системы, разработать диаграммы вариантов использования и развертывания, спроектировать концептуальную, логическую и физическую модели базы данных, разработать клиентскую и серверную структуры программы, с последующим написанием программного кода.

В результате проделанной работы разработана система регистрации пассажиров аэропорта, состоящая из базы данных и клиент-серверного приложения для работы с ней.

База данных спроектирована с использованием онлайн инструмента для создания диаграмм и схем Diagrams.net. База данных разработана для СУБД PostgreSQL, приложение реализовано в среде разработки Visual Studio Code.

|           |      |               |                    |          |                                                                          |  |  |  |  |
|-----------|------|---------------|--------------------|----------|--------------------------------------------------------------------------|--|--|--|--|
|           |      |               |                    |          | ПГУ 1.090501.03.001 ПЗ                                                   |  |  |  |  |
| Изм.      | Лист | № докум.      | Подпись            | Дата     |                                                                          |  |  |  |  |
| Разраб.   |      | Буренков С.В. | <i>С.Буренков</i>  | 29.08.20 | Система регистрации пассажиров<br>аэропорта<br><br>Пояснительная записка |  |  |  |  |
| Провер.   |      | Егоров В.Ю.   | <i>В.Егоров</i>    | 29.08.20 |                                                                          |  |  |  |  |
| Н. Контр. |      | Бычков А.С.   | <i>А.Бычков</i>    | 29.08.20 |                                                                          |  |  |  |  |
|           |      |               | <i>В.А.Сидоров</i> | 29.08.20 | ФВТ, каф. ВТ<br>гр 20В031                                                |  |  |  |  |
| Утверд.   |      | Митрохин М.А. | <i>М.Митрохин</i>  | 29.08.20 |                                                                          |  |  |  |  |

## Содержание

|                                                                             |    |
|-----------------------------------------------------------------------------|----|
| Перечень сокращений и обозначений .....                                     | 5  |
| Введение .....                                                              | 6  |
| 1 Анализ предметной области.....                                            | 8  |
| 1.1 Описание предметной области.....                                        | 8  |
| 1.2 Анализ аналогов.....                                                    | 9  |
| 1.3 Цели создаваемой системы.....                                           | 10 |
| 1.4 Анализ целевой аудитории .....                                          | 10 |
| 2 Техническое задание .....                                                 | 11 |
| 2.1 Назначение разработки .....                                             | 11 |
| 2.2 Требования к программе .....                                            | 11 |
| 2.2.1 Потребности к функциям системы.....                                   | 11 |
| 2.2.2 Запросы к надежности системы.....                                     | 12 |
| 2.2.3 Требования к характеристикам средств для работы системы.....          | 13 |
| 2.3 Требования к содержанию программной документации .....                  | 13 |
| 2.5 Порядок контроля и приемки между исполнителем и заказчиком .....        | 14 |
| 3 Объяснение причины выбора конкретных средств для разработки системы ..... | 15 |
| 3.1 Visual Studio Code .....                                                | 15 |
| 3.2 JavaScript.....                                                         | 15 |
| 3.3 Серверная часть .....                                                   | 16 |
| 3.4 Клиентская часть.....                                                   | 18 |
| 3.5 Diagrams.net .....                                                      | 19 |
| 4 Проектирование системы .....                                              | 20 |
| 4.1 Концептуальное проектирование базы данных.....                          | 20 |
| 4.2 Логическое проектирование базы данных .....                             | 23 |
| 4.3 Физическое проектирование базы данных.....                              | 27 |
| 5 Разработка системы.....                                                   | 31 |
| 5.1 Описание структуры программы .....                                      | 31 |
| 5.2 Реализация регистрации и авторизации пользователей в системе .....      | 38 |

|                                                                         |     |
|-------------------------------------------------------------------------|-----|
| 5.3 Реализация выбора места на рейс в системе.....                      | 40  |
| 6 Функциональное тестирование.....                                      | 43  |
| 7 Оценка затрат на разработку системы .....                             | 48  |
| 7.1 Организация производства системы регистрации .....                  | 48  |
| 7.2 Расчет себестоимости программного обеспечения.....                  | 52  |
| 8 Общие принципы обеспечения безопасности.....                          | 56  |
| 8.1 Актуальность вопросов безопасности человека при работе с ПЭВМ ..... | 56  |
| 8.2 Требования к помещениям для работы с ПЭВМ.....                      | 56  |
| 8.3 Техника безопасности при работе с ПЭВМ.....                         | 57  |
| Заключение .....                                                        | 59  |
| Список использованных источников .....                                  | 60  |
| Приложение А – Диаграммы UML, модели, структуры.....                    | 61  |
| Приложение Б – Результаты функционального тестирования.....             | 69  |
| Приложение В – Листинг программы.....                                   | 96  |
| Приложение Г – Презентация .....                                        | 123 |

## Перечень сокращений и обозначений

В настоящей работе применяются следующие сокращения и обозначения

БД – база данных

ЕСПД – единая система программной документации

МВД – Министерство внутренних дел

НФ – нормальная форма

ПЭВМ – персональная электронно-вычислительная машина

ФСБ – Федеральная служба безопасности

FK (Foreign Key) – внешний ключ

ID (Identifier) – идентификатор сущности

PK (Primary Key) – первичный ключ

RESTful API – архитектурный стиль для разработки интернет сервисов

SQL (Structured Query Language) – язык структурированных запросов

UML (Unified Modeling Language) – унифицированный язык моделирования

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 5    |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

## Введение

Системы регистрации играют важную роль в жизни человека и общества в целом. Они используются в тех отраслях, где необходимо автоматизировать повторяющиеся процессы. Также они позволяют организовать правильное взаимодействие между пользователем и системой, которое приводит к минимизации ошибок, при работе с данной системой. К системам регистрации можно отнести онлайн сервисы, образовательные системы, государственные системы специального назначения, такие как – МВД, ФСБ, аэропорты, медицинские учреждения и многие другие.

К основным целям систем регистрации можно отнести: автоматизацию повторяющихся процессов, обеспечение безопасности хранения данных и разграничение прав доступа к ним, в зависимости от роли пользователя, отслеживание действий пользователей в системе для учета и анализа использованных ресурсов, получение необходимой информации в короткий срок.

Системы регистрации работают с использованием баз данных. База данных (БД) — это организованная совокупность данных, хранящаяся в электронном виде на компьютере или специальном сервере, структурированная таким образом, чтобы облегчить их хранение, управление и доступ к ним. Основная цель базы данных — эффективно хранить и управлять большими объемами информации, а также предоставлять инструменты для работы с этими данными. Базы данных позволяют быстро находить нужную информацию по определенным критериям [1]. Каждая база данных привязана к определённой предметной области. Разработка базы данных происходит поэтапно. Первоначально создаётся концептуальная модель, которая характеризует таблицы вместе с их атрибутами и показывает связи между таблицами. После разрабатывается логическая модель, которая уже будет описывать больше деталей, но всё также, не будет иметь привязки к определённой СУБД. В данную модель добавляется описание внешних ключей, указываются ограничения атрибутов сущности. Затем проектируется физическая модель, которая описывает то, как данные фактически будут храниться в базе данных.

Основная задача систем регистрации — это идентификация пользователей, с помощью этого система может предоставлять различные виды доступа к ресурсам. Для проверки, что пользователь имеет доступ к конкретным ресурсам системы, необходимо доказательство личности, например, посредством логина и пароля. Также это предотвращает несанкционированный доступ в программу. Информация о логине, пароле, а также роли пользователя находится на сервере. Для организации защиты информации, и минимизации взлома аккаунтов пассажиров, в случае утечки данных с сервера, пароль пользователя хранится в зашифрованном виде.

Системы регистрации – это важное средство для автоматизации повторяющихся процессов в определённой предметной области, упорядочивание доступа, обеспечение безопасности хранения данных. Кроме того, с помощью данных систем происходит анализ предметной области, для выяснения приоритетных используемых ресурсов пользователем. Они используются во многих сферах человеческой деятельности и развиваются с каждым днём.

|      |      |          |         |      |                                      |      |
|------|------|----------|---------|------|--------------------------------------|------|
|      |      |          |         |      | <p><i>ПГУ 1.090501.03.001 ПЗ</i></p> | Лист |
|      |      |          |         |      |                                      | 7    |
| Изм. | Лист | № докум. | Подпись | Дата |                                      |      |

# 1 Анализ предметной области

## 1.1 Описание предметной области

Система регистрации пассажиров аэропорта предназначена для контроля работы аэропорта. Она позволяет автоматизировать повторяющиеся процессы, которые возникают при работе аэропорта. Также она позволяет хранить данные в безопасном – электронном виде, без использования физических носителей информации. Данная система представляет возможность взаимодействия между пассажиром и аэропортом, а также между администратором и аэропортом.

Предметной областью данной системы является аэропорт. В настоящее время люди достаточно часто пользуются услугами аэропорта, так как им необходимо преодолевать длинные расстояния за минимально затраченное время. Кроме того, вырос спрос Россиян на путешествия с использованием авиаперелетов. По результатам летнего сезона турпоток из России за рубеж вырастет не менее чем на 15%, а по ряду направлений он будет более существенный [2]. Также со временем все больше развиваются города и строительство аэропортов растёт. Именно поэтому тема разработки данной системы является актуальной.

Система регистрации пассажиров аэропорта должна включать в себя информацию о пользователях данной системы, о зарегистрированных пассажирах, о доступных рейсах и местах на самолет.

Система должна разграничивать доступ для администратора системы и для обычного пользователя. Администратор должен иметь функции добавления, редактирования, удаления рейсов. Пользователь системы регистрации пассажиров аэропорта должен иметь весь функционал, необходимый для покупки авиабилета. Этот функционал включает регистрацию пассажиров, выбор доступного рейса, возможность сортировки доступных рейсов, выбор места в самолете на конкретный рейс, а также добавление рейса, который нужно купить, в «корзину».

Данная система должна иметь клиент-серверную архитектуру. Так как сервер позволяет централизованно хранить информацию с использованием базы данных,

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 8    |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

также это упрощает управление, резервное копирование и безопасность данных. Клиентская часть отвечает за удобный пользовательский интерфейс и взаимодействие с пользователем.

## 1.2 Анализ аналогов

Для того, чтобы определиться с функционалом разрабатываемой системы регистрации пассажиров аэропорта, необходимо познакомиться с аналогами, которые используются в аэропортах России. Такими системами являются: Astra DCS, Asteros Contact Air, IS – DCS (Integrated System – Departure Control System).

Astra DCS — это автоматизированная система управления пассажирскими авиаперелетами. Данная система имеет функционал для регистрации пассажиров, обработки данных о рейсах и управления билетами. Она поддерживает интеграцию с различными системами аэропортов и авиакомпаний. Также, данная система имеет мультязычный интерфейс. Эта система используется в Московском аэропорте Домодедово.

Asteros Contact Air — это система для автоматизации процессов в аэропортах. В её функционал входит регистрация пассажиров, управление рейсами и билетами, а также обработка багажа. Кроме того, она имеет весь необходимый функционал для мониторинга, учёта и анализа данных. Эта система используется в Московском аэропорте Шереметьево, где внедряется экспресс регистрация пассажиров.

IS-DCS (Integrated System – Departure Control System) — это автоматизированная система, которая необходима для управления процессами регистрации, вылета и посадки пассажиров. Она обеспечивает автоматизацию на всех этапах, начиная от регистрации пассажира и заканчивая посадкой на рейс. Также данная система отвечает за печать посадочных талонов. Эта система используется в Московском аэропорте Шереметьево.

Исходя из анализа аналогов, делается заключение о том, что данная система будет использоваться малыми аэропортами, те, которые располагаются в городах с малым населением. Это позволит интегрировать информационные технологии обработки информации даже в малые аэропорты. Система будет иметь интуитивно

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 9    |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

понятный и удобный интерфейс, что повысит удовлетворенность пассажиров. Функционал системы будет иметь только те операции, которые нужны каждому пользователю.

### 1.3 Цели создаваемой системы

К целям создания системы можно отнести:

- повышение эффективности работы аэропорта в целом;
- рост скорости покупки авиабилетов;
- рост скорости регистрации пассажиров;
- обеспечение контроля рейсов и мест на самолет;
- обеспечение контроля зарегистрированных пассажиров;
- обеспечение контроля рейсов пассажиров для покупки;
- повышение комфорта работы администратора аэропорта.

### 1.4 Анализ целевой аудитории

Разработка системы регистрации пассажиров аэропорта является актуальной для малых аэропортов, находящихся в не больших городах или в труднодоступных населенных пунктах. Например, в Чукотском автономном округе есть несколько аэропортов, в том числе в городах с небольшим населением: Анадырь (Угольный), Певек, Кепервеем, Бухта Провидения [3].

Делая вывод, основной целевой группой являются сотрудники и пассажиры аэропортов, расположенных в малых городах.

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 10   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

## 2 Техническое задание

### 2.1 Назначение программы

Данная программа будет использована сотрудником аэропорта и пассажирами, целями которых является покупка авиабилета. Администратор аэропорта сможет не контактировать на прямую с пассажирами, а удалённо сообщать информацию о рейсах. Пользователь сможет осуществлять регистрацию и покупку авиабилета везде, где есть интернет.

### 2.2 Требования к программе

#### 2.2.1 Потребности к функциям системы

Программа будет иметь следующий функционал:

- учет пользователей системы;
- учет рейсов;
- учет мест на конкретный рейс;
- учет пассажиров;
- учет купленных билетов;
- хранение данных:
- данные о пользователях системы;
- данные о рейсах;
- данные о свободных или занятых местах на конкретный рейс;
- данные о пассажирах;
- данные о купленных билетах.

Входными данными системы регистрации должны быть:

- данные о рейсах;
- данные о местах на рейс;
- данные о пассажирах.

Выходными данными системы регистрации являются купленные билеты.

Диаграмма вариантов использования представлена на рисунке 1.

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 11   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

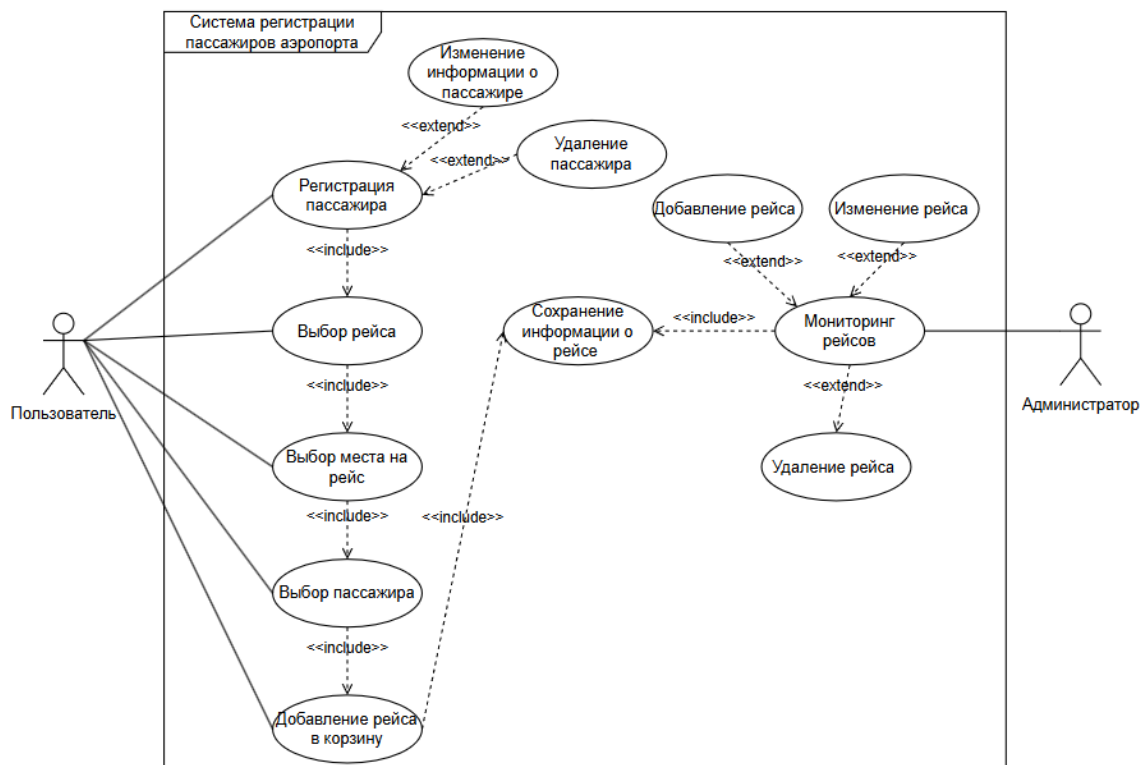


Рисунок 1 – Диаграмма вариантов использования системы регистрации пассажиров аэропорта

Управляющими элементами в данной системе выступают пользователь и администратор. Пользователь может зарегистрировать пассажира, также, при необходимости, изменить информацию о пассажире или удалить пассажира. Кроме этого, для покупки авиабилета, пользователь должен выбрать конкретный рейс, затем выбрать зарегистрированного пассажира и добавить рейс в корзину. Администратор может просматривать все рейсы, и при необходимости, добавлять рейс, изменять информацию о рейсе или удалять рейс. Вся информация о рейсах сохраняется в базу данных.

## 2.2.2 Запросы к надёжности системы

Система должна функционировать в непрерывном режиме, код программы должен храниться на сервере и быть разделён на интуитивно понятные файлы, для того чтобы, в случае возникновения непредвиденной ошибки, любой специализированный программист смог в кратчайшие сроки вернуть работоспособность программы. Также система регистрации должна обеспечивать

защиту данных и разграничение прав доступа от несанкционированного входа на основе процедуры аутентификации пользователей. Кроме того, система обеспечивает защиту данных пользователей, хранящихся на сервере, методом хеширования пароля. Также, программа должна помогать пользователю выявлять ошибку, в его совершенных действиях, путём вывода на экран соответствующего сообщения.

### 2.2.3 Требования к характеристикам средств для работы системы

Система регистрации пассажиров аэропорта может использоваться всеми людьми, имеющими вычислительную машину и доступ в интернет, так как программа может работать на техники с не высокими техническими характеристиками. Учитывая развитие современного общества, почти каждая вычислительная техника имеет в своем составе эти минимальные технические характеристики.

Серверная часть данной системы, которая будет постоянно обрабатывать данные и хранить информацию о работе аэропорта, должна обязательно иметь в своём техническом составе средне-производительный процессор с тактовой частотой от 2 ГГц, оперативную память с частотой от 2400 МГц и объёмом памяти от 8 Гб, для обеспечения хранения большого количества данных необходим жёсткий диск имеющий объём памяти от 1 Тб.

## 2.3 Требование к содержанию программной документации

Данный программный продукт должен в обязательном порядке содержать в своём составе техническую документацию, которая необходима для предоставления заказчику всей информации о системе, также наличие документации будет подтверждать, что заказчик имеет все права на использование системы. Программная документация должна содержать описание технического задания, которое описывает функции разработанной системы, требования к характеристикам средств для успешной работы программы. Также документация имеет в своём составе описание всех диаграмм. Кроме того, она должна содержать полное объяснение проектирования базы данных на всех этапах. В документации

|      |      |          |         |      |                        |      |
|------|------|----------|---------|------|------------------------|------|
|      |      |          |         |      | ПГУ 1.090501.03.001 ПЗ | Лист |
|      |      |          |         |      |                        |      |
| Изм. | Лист | № докум. | Подпись | Дата |                        | 13   |

обязательно должно находиться описание структуры системы вместе с кодом программы, расположенным в отдельном приложении. Для того, чтобы заказчик понимал, что перед ним находится полностью работоспособный продукт, документация должна включать в себя функциональное тестирование и описание применения.

Техническая документация должна оформляться в соответствии с требованиями Единой системы программной документации (ЕСПД).

## **2.5 Порядок контроля и приемки между исполнителем и заказчиком**

Приёмка системы в обязательном порядке должна осуществляться только в присутствии заказчика, который должен подготовить тестовые наборы данных, для того чтобы полностью удостовериться в работоспособности приложения. Исполнитель обязан предъявить полностью работающий продукт, вместе с технической документацией. Приложение должно соответствовать всем требованиям технического задания. В условиях работы приложения заказчик должен проверить действия сотрудника аэропорта, а именно: вход в систему под ролью администратора, добавление, изменение и удаление рейсов. Также заказчик должен проверить все возможные действия пассажиров аэропорта, то есть обычных пользователей, к этим действиям относится: регистрация и вход в систему, добавление, изменение, удаление информации о пассажире, выбор рейса, места на самолет и зарегистрированного пассажира, а после добавление рейса в корзину. После приемки исполнитель должен внедрить программный продукт в среду пользования заказчика. Также обсуждается дальнейшее сотрудничество между исполнителем и заказчиком, которое подразумевает дальнейшее сопровождение системы.

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               |      |
| Изм. | Лист | № докум. | Подпись | Дата |                               | 14   |

### 3 Объяснение причины выбора конкретных средств для разработки системы

Исходя из потребностей к функциям системы регистрации пассажиров аэропорта были выбраны наиболее современные, удобные, подходящие под эти цели средства разработки. В качестве компилятора применяется Visual Studio Code. В данной программе написана клиентская и серверная часть приложения на языке JavaScript. Для разработки серверной части (backend) используется следующий стек технологий – Express, Node JS, PostgreSQL, Sequelize. Для разработки клиентской части (frontend) – React JS, Axios, React-router-dom, Mob-X, React Bootstrap. Для проектирования базы данных используется редактор – Diagrams.net.

#### 3.1 Visual Studio Code

Visual Studio Code (VS Code) — это современный, удобный компилятор кода для разработчиков, который работает на Windows, macOS и Linux. Данный редактор кода поддерживает язык программирования JavaScript, также имеет широкий функционал для написания, отладки и тестирования кода. Редактор используют разработчики для создания WEB приложений, RESTful API и других программных продуктов. Visual Studio Code имеет функцию подсветки синтаксиса, что делает код более читаемым. Также, среди явных плюсов можно выделить его бесплатность, что делает его доступным для всех разработчиков. Кроме того, данный компилятор имеет встроенный терминал, что позволяет выполнять команды, не выходя из программы. Данная программа быстро запускается и хорошо работает на маломощных компьютерах. Также можно настраивать внешний вид программы под свои предпочтения.

#### 3.2 JavaScript

JavaScript — это высокоуровневый, интерпретируемый, язык программирования, используемый для создания WEB приложений. Данный язык поддерживается во всех современных браузерах. Кроме того, благодаря платформе

Node.js на этом языке программирования можно писать серверную часть приложения. JavaScript поддерживает большое количество фреймворков, таких как React, Angular, Vue.js и jQuery. Также данный язык поддерживает асинхронное программирование с помощью колбеков, промисов и async/await, что позволяет выполнять множество задач, без блокировки основного потока работы программы. Он позволяет выполнять валидацию форм и работать с данными в реальном времени. Кроме того, он поддерживает объектно – ориентированное программирование, что позволяет работать со сложными структурами данных. JavaScript отлично совместим с такими языками программирования, как HTML и CSS.

### 3.3 Серверная часть

В качестве фреймворка для написания серверной части на Node JS используется Express. Это современный и гибкий WEB – фреймворк, который упрощает разработку WEB – приложений. Он включает в себя набор инструментов и функций, необходимых для написания сервера приложения. Express представляет возможность для работы с запросами и маршрутами. Кроме того, фреймворк отлично работает с middleware-функциями, которые используются для реализации аутентификации, обработки ошибок, проверки запросов. Также express предоставляет возможности маршрутизации, с помощью которых разработчики определяют, как обрабатывать различные HTTP-запросы на основе URL и метода запроса.

В качестве платформы для написания серверной части на языке JavaScript используется Node JS. Это открытая среда выполнения, которая позволяет запускать JavaScript-код на сервере. Node.js часто используется для создания веб-серверов, RESTful API, реального времени приложений (например, чатов) и многого другого. Одной из основных привлекательных особенностей Node.js является скорость. JavaScript-код, выполняемый в среде Node.js, может быть в два раза быстрее, чем код, написанный на компилируемых языках, вроде C или Java, и на порядки быстрее интерпретируемых языков наподобие Python или Ruby.

|      |      |          |         |      |                        |      |
|------|------|----------|---------|------|------------------------|------|
|      |      |          |         |      | ПГУ 1.090501.03.001 ПЗ | Лист |
|      |      |          |         |      |                        |      |
| Изм. | Лист | № докум. | Подпись | Дата |                        | 16   |

Причиной подобного является неблокирующая архитектура платформы, а конкретные результаты зависят от используемых тестов производительности, но, в целом, Node.js — это очень быстрая платформа [4]. Кроме того, Node.js работает на различных операционных системах, включая Windows, macOS и Linux. Он имеет встроенный менеджер пакетов npm (Node Package Manager), который позволяет легко устанавливать и управлять сторонними библиотеками и модулями.

PostgreSQL выступает в роли системы управления базами данных (СУБД). СУБД руководят структурой базы данных, состоящей из множества таблиц. Данная система даёт возможность взаимодействовать пользователю с базой данных. Сама структура базы данных хранится в виде набора файлов, чтобы увидеть содержимое понадобится СУБД. Данная система управления является объектно-реляционной. Это бесплатная система с открытым исходным кодом. Она необходима для обработки структурированной информации. Также данная СУБД поддерживает транзакции с соблюдением принципов атомарности, согласованности, изолированности, долговечности, именно это обеспечивает надежность и целостность данных. PostgreSQL позволяет создавать собственные типы данных, функции и операторы. СУБД позволяет работать с реляционными и объектно-ориентированными базами данных, создавать, изменять, удалять записи, отправлять транзакции — совокупность последовательных запросов на языке SQL. Запросы в этой СУБД выполняются параллельно, что ускоряет обработку больших объёмов информации.

В качестве ORM для работы с базой данных на Node JS используется Sequelize. Это инструмент для организации взаимодействия между фреймворком и базой данных. Этот ORM предоставляет возможность быстрого написания базы данных, без необходимости использования языка SQL. С помощью программного кода создаются таблицы и устанавливаются ассоциации между ними (например, один-к-одному, один-ко-многим и многие-ко-многим). Кроме того, sequelize позволяет добавлять валидацию данных на уровне модели и использовать хуки

(hooks) для выполнения действий до или после определенных событий (например, перед созданием или обновлением записи).

### 3.4 Клиентская часть

В качестве библиотеки для создания пользовательских интерфейсов используется React JS. Это библиотека для языка программирования JavaScript с открытым исходным кодом. Она помогает быстро и легко реализовать реактивность — явление, когда в ответ на изменение одного элемента меняется все остальное. Библиотека предназначена для - создания функциональных интерактивных веб-интерфейсов, работая с которыми, не нужно постоянно обновлять страницу, быстрой и удобной реализации отдельных компонентов и страниц целиком — элементы в React легко использовать повторно, работы с серверной частью сайта или разработки мобильных приложений, разработки одностраничных и многостраничных приложений [5]. React использует виртуальный DOM (Document Object Model), который позволяет эффективно обновлять интерфейс. Вместо изменения реального DOM, React сначала обновляет виртуальный DOM, а затем сравнивает его с реальным, уменьшая количество операций и улучшая производительность. Также React использует JSX (JavaScript XML) — синтаксис, который позволяет писать HTML-подобный код внутри JavaScript. Это делает код более читаемым и удобным для разработки интерфейсов.

Для выполнения HTTP-запросов в JavaScript используется библиотека Axios. Axios использует промисы для обработки асинхронных операций, что позволяет легко управлять результатами запросов и обрабатывать ошибки. Также данная библиотека поддерживает HTTP-методы, такие как GET, POST, PUT, DELETE, PATCH, что позволяет выполнять различные операции с данными. Axios умеет преобразовывать ответы в формате JSON, что упрощает работу с данными, полученными от сервера. С помощью Axios можно создавать интерсепторы для перехвата запросов и ответов, что позволяет добавлять логику, такую как аутентификация или обработка ошибок, перед отправкой запроса или после получения ответа.

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 18   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

В качестве библиотеки для маршрутизации в приложении, созданных с помощью React, используется React-router-dom. Она позволяет управлять навигацией между страницами в одностраничных приложениях, обеспечивая плавный переход между ними без перезагрузки страницы. Библиотека позволяет определять маршруты, которые связывают URL-адреса с компонентами. Это позволяет отображать разные компоненты в зависимости от текущего URL. Кроме того, библиотека управляет историей навигации, позволяя использовать кнопки "Назад" и "Вперед" в браузере для перемещения между страницами.

В качестве библиотеки для управления состоянием в приложении, созданных с помощью React, используется MobX. Он использует реактивные принципы. Это означает, что изменения в состоянии приводят к обновлению компонентов, которые зависят от этого состояния. В MobX состояние представляется в виде наблюдаемых объектов (observable), а действия (actions) используются для изменения этого состояния. Это необходимо для организации четкой структуры и разделения логики в приложении. Также он обновляет только те компоненты, которые зависят от изменившегося состояния. Это позволяет убрать ненужные перерисовки и улучшает производительность приложения.

В качестве библиотеки компонентов используется React Bootstrap. Эта библиотека используется для создания пользовательских интерфейсов и имеет в своем составе различные кнопки, таблицы, модальные окна, навигационные панели, поля ввода данных и многое другое. Он позволяет настраивать стили с помощью CSS и встроенных классов.

### 3.5 Diagrams.net

Для проектирования базы данных или создания диаграмм используется Diagrams.net. Это бесплатный онлайн-сервис, который позволяет создавать и редактировать диаграммы, различные схемы и прочие визуальные представления данных. Diagrams.net предлагает удобный интерфейс с перетаскиванием элементов, а также множество шаблонов, фигур, стрелок, таблиц и многое другое, что делает его удобным.

|      |      |          |         |      |                        |      |
|------|------|----------|---------|------|------------------------|------|
|      |      |          |         |      | ПГУ 1.090501.03.001 ПЗ | Лист |
|      |      |          |         |      |                        |      |
| Изм. | Лист | № докум. | Подпись | Дата |                        | 19   |

## 4 Проектирование системы

Диаграмма развертывания системы регистрации пассажиров аэропорта представлена на рисунке 2.

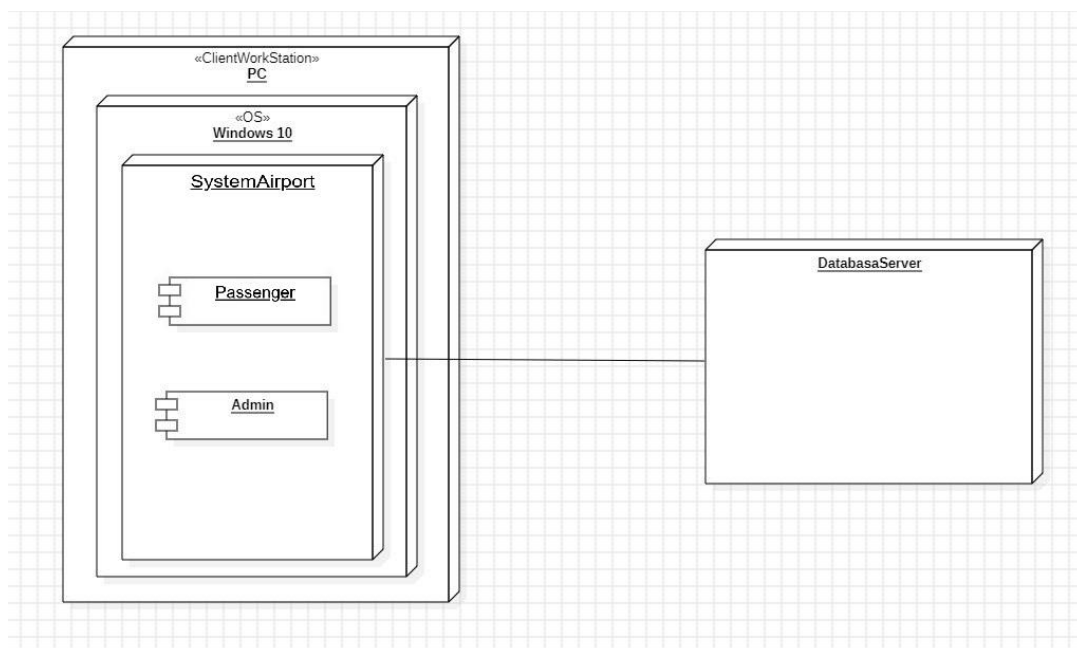


Рисунок 2 – Диаграмма развертывания

### 4.1 Концептуальное проектирование базы данных

Концептуальная модель демонстрируется на рисунке 3 и содержит пять таблиц. Сущность user – пользователи системы, в данную таблицу заносятся данные администраторов и пассажиров. По умолчанию при регистрации всегда в базу данных заносится пользователь с ролью USER, для того чтобы добавить дополнительного пользователя с ролью ADMIN потребуется обратиться к разработчику для изменения значения по умолчанию. Количество администраторов системы изначально проговаривается с заказчиком. Сущность passenger – информация о пассажирах. В данную таблицу заносятся данные о зарегистрированных пассажирах конкретного пользователя. Сущность flight – информация о рейсах. Эта таблица содержит информацию о доступных рейсах. Сущность seat – информация о состоянии мест конкретного рейса. Сущность basket

|      |      |          |         |      |
|------|------|----------|---------|------|
|      |      |          |         |      |
| Изм. | Лист | № докум. | Подпись | Дата |

ПГУ 1.090501.03.001 ПЗ

Лист

20

– корзина пользователя. Эта окончательная таблица, в которую пользователь заносит рейсы для покупки.

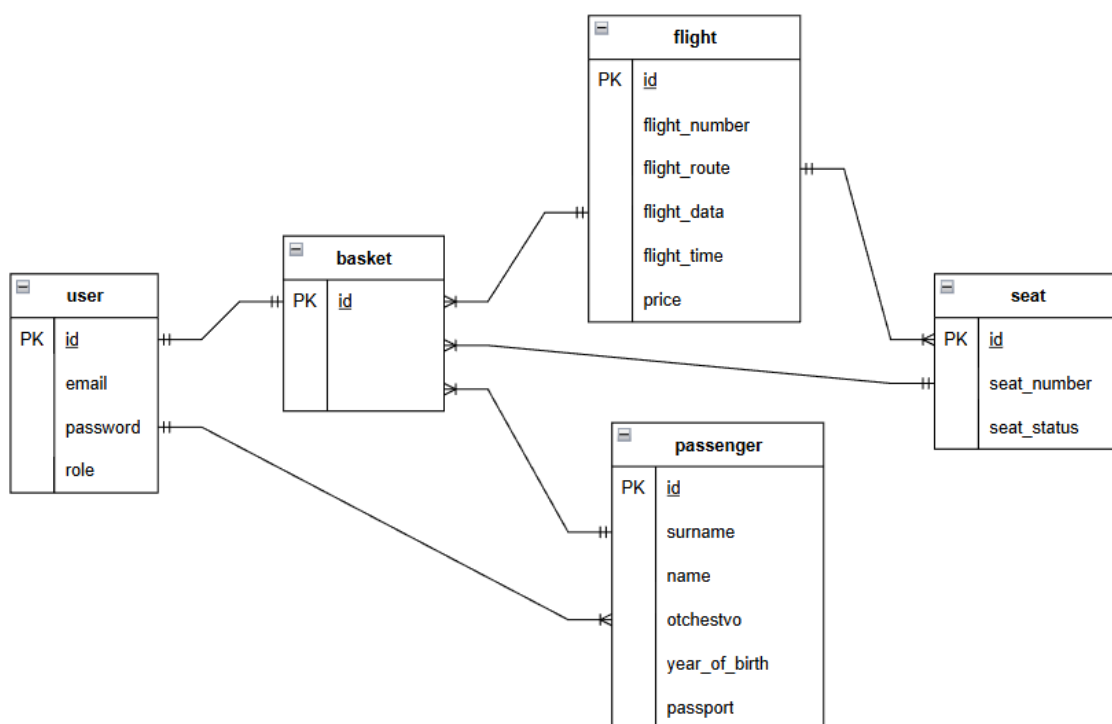


Рисунок 3 – Концептуальная модель базы данных

Описание атрибутов сущностей концептуальной модели приведено в таблицах 1 – 5.

Таблица 1 – Описание атрибутов сущности «user»

| Имя атрибута | Назначение          | Первичный ключ |
|--------------|---------------------|----------------|
| <u>id</u>    | id пользователя     | PK             |
| email        | Почта пользователя  |                |
| password     | Пароль пользователя |                |
| role         | Роль пользователя   |                |

Таблица 2 – Описание атрибутов сущности «basket»

| Имя атрибута | Назначение              | Первичный ключ |
|--------------|-------------------------|----------------|
| <u>id</u>    | id корзины пользователя | PK             |

Таблица 3 - Описание атрибутов сущности «passenger»

| Имя атрибута  | Назначение             | Первичный ключ |
|---------------|------------------------|----------------|
| <u>id</u>     | id пассажира           | РК             |
| surname       | Фамилия пассажира      |                |
| name          | Имя пассажира          |                |
| otchestvo     | Отчество пассажира     |                |
| year_of_birth | Год рождения           |                |
| passport      | Серия и номер паспорта |                |

Таблица 4 - Описание атрибутов сущности «flight»

| Имя атрибута  | Назначение            | Первичный ключ |
|---------------|-----------------------|----------------|
| <u>id</u>     | id рейса              | РК             |
| flight_number | Номер рейса           |                |
| flight_route  | Маршрут полета        |                |
| flight_data   | Дата вылета           |                |
| flight_time   | Время вылета/прибытия |                |
| price         | Цена рейса            |                |

Таблица 5 – Описание атрибутов сущности «seat»

| Имя атрибута | Назначение     | Первичный ключ |
|--------------|----------------|----------------|
| <u>id</u>    | id места       | РК             |
| seat_number  | Название места |                |
| seat_status  | Статус места   |                |

Сущность «user» связана с сущностью «basket» связью с показателем кардинальности «один к одному», так как один пользователь может иметь только одну корзину с какими-либо рейсами. В сущности, user в качестве первичного ключа выбран атрибут «id». В сущности, basket в качестве первичного ключа выбран атрибут «id».

Сущность «user» связана с сущностью «passenger» связью с показателем кардинальности «один ко многим», так как один пользователь может зарегистрировать множество пассажиров. В сущности, user в качестве первичного ключа выбран атрибут «id». В сущности, passenger в качестве первичного ключа выбран атрибут «id».

Сущность «flight» связана с сущностью «seat» связью с показателем кардинальности «один ко многим», так как один рейс может иметь много мест. В сущности, flight в качестве первичного ключа выбран атрибут «id». В сущности, seat в качестве первичного ключа выбран атрибут «id».

Сущность «flight» связана с сущностью «basket» связью с показателем кардинальности «один ко многим», так как в одной корзине может находиться много рейсов. В сущности, flight в качестве первичного ключа выбран атрибут «id». В сущности, basket в качестве первичного ключа выбран атрибут «id».

Сущность «seat» связана с сущностью «basket» связью с показателем кардинальности «один ко многим», так как в одной корзине может находиться много занятых мест разных рейсов. В сущности, seat в качестве первичного ключа выбран атрибут «id». В сущности, basket в качестве первичного ключа выбран атрибут «id».

Сущность «passenger» связана с сущностью «basket» связью с показателем кардинальности «один ко многим», так как в одной корзине может находиться много зарегистрированных пассажиров. В сущности, passenger в качестве первичного ключа выбран атрибут «id». В сущности, basket в качестве первичного ключа выбран атрибут «id».

## 4.2 Логическое проектирование базы данных

После, опираясь на концептуальную модель, проектируется логическая, представленная на рисунке 4.

|      |      |          |         |      |                        |      |
|------|------|----------|---------|------|------------------------|------|
|      |      |          |         |      | ПГУ 1.090501.03.001 ПЗ | Лист |
|      |      |          |         |      |                        |      |
| Изм. | Лист | № докум. | Подпись | Дата |                        | 23   |

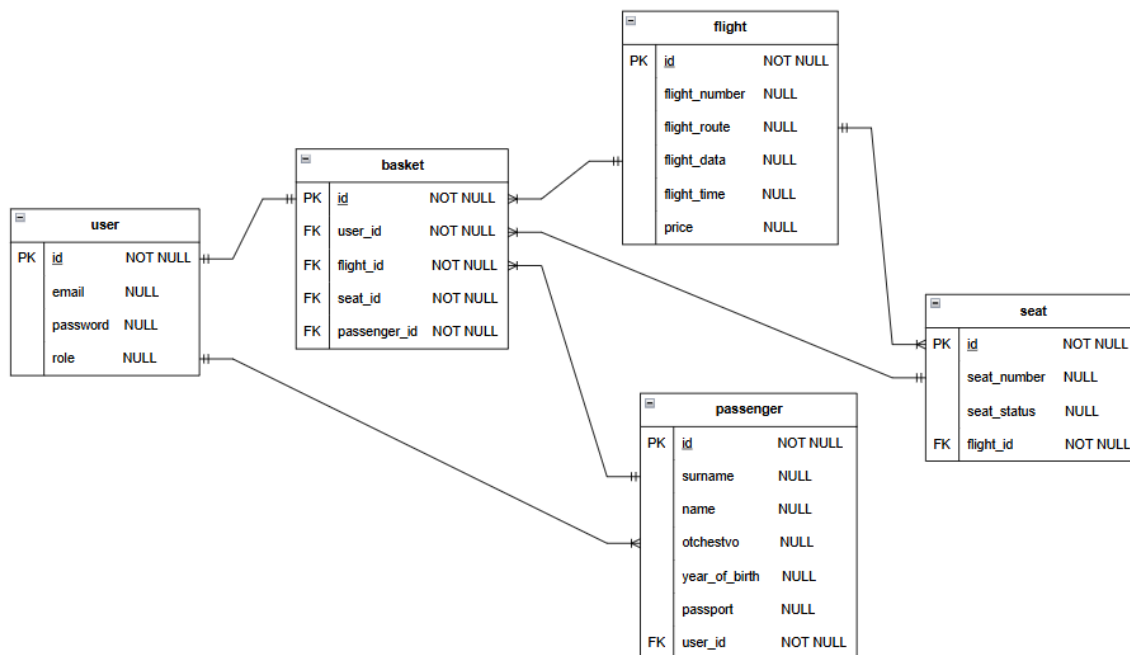


Рисунок 4 – Логическая модель базы данных

Описание атрибутов сущностей логической модели приведено в таблицах 6 – 10.

Таблица 6 – Описание атрибутов сущности «user»

| Имя атрибута | Назначение          | Первичный ключ | Внешний ключ |
|--------------|---------------------|----------------|--------------|
| <u>id</u>    | id пользователя     | PK             |              |
| email        | Почта пользователя  |                |              |
| password     | Пароль пользователя |                |              |
| role         | Роль пользователя   |                |              |

Таблица 7 – Описание атрибутов сущности «basket»

| Имя атрибута | Назначение      | Первичный ключ | Внешний ключ |
|--------------|-----------------|----------------|--------------|
| <u>id</u>    | id корзины      | PK             |              |
| user_id      | id пользователя |                | FK           |
| flight_id    | id рейса        |                | FK           |
| seat_id      | id места        |                | FK           |
| passenger_id | id пассажира    |                | FK           |

Таблица 8 - Описание атрибутов сущности «passenger»

| Имя атрибута  | Назначение                | Первичный ключ | Внешний ключ |
|---------------|---------------------------|----------------|--------------|
| <u>id</u>     | id пассажира              | PK             |              |
| surname       | Фамилия пассажира         |                |              |
| name          | Имя пассажира             |                |              |
| otchestvo     | Отчество пассажира        |                |              |
| year_of_birth | Год рождения              |                |              |
| passport      | Серия и номер<br>паспорта |                |              |
| user_id       | id пользователя           |                | FK           |

Таблица 9 - Описание атрибутов сущности «flight»

| Имя атрибута  | Назначение               | Первичный ключ | Внешний ключ |
|---------------|--------------------------|----------------|--------------|
| <u>id</u>     | id рейса                 | PK             |              |
| flight_number | Номер рейса              |                |              |
| flight_route  | Маршрут полета           |                |              |
| flight_data   | Дата вылета              |                |              |
| flight_time   | Время<br>вылета/прибытия |                |              |
| price         | Цена рейса               |                |              |

Таблица 10 – Описание атрибутов сущности «seat»

| Имя атрибута | Назначение     | Первичный ключ | Внешний ключ |
|--------------|----------------|----------------|--------------|
| <u>id</u>    | id места       | PK             |              |
| seat_number  | Название места |                |              |
| seat_status  | Статус места   |                |              |
| flight_id    | id рейса       |                | FK           |

В логическую модель уже добавляются внешние ключи, которые необходимы для взаимодействия между таблицами.

В сущность passenger добавлен внешний ключ user\_id из сущности user. Данный внешний ключ позволяет закрепить за определённым пользователем зарегистрированных пассажиров. Также это необходимо для того, чтобы у разных пользователей был доступ только к своим зарегистрированным пассажирам.

В сущность seat добавлен внешний ключ flight\_id из сущности flight. Данный внешний ключ позволяет закрепить определённые места за конкретным рейсом. При создании администратором рейса автоматически генерируются свободные места, у которых столбец flight\_id заполняется первичным ключом id из таблицы flight. Это позволяет делать запрос к базе данных и узнавать информацию только о местах конкретного рейса.

В сущность basket добавлены внешние ключи: user\_id из сущности user, flight\_id из сущности flight, seat\_id из сущности seat, passenger\_id из сущности passenger. Данные внешние ключи, в рамках реализации программы, необходимы для объединения всех данных о купленном рейсе в корзине пользователя. Так внешний ключ user\_id закрепляет корзину рейсов за конкретным пользователем. Также данный внешний ключ необходим для того, чтобы сохранять определённые рейсы для покупки за конкретным пользователем. Если пользователь выйдет из системы и потом обратно вернётся в неё, произойдёт запрос в базу данных по данному внешнему ключу user\_id и пользователь увидит все раньше добавленные им рейсы для покупки. Внешний ключ flight\_id закрепляет определённый рейс за корзиной пользователя. Внешний ключ seat\_id закрепляет определённое место за корзиной. Внешний ключ passenger\_id закрепляет определённого зарегистрированного пассажира.

Также в логическую модель добавляются определённые ограничения. На рисунке 4 можно увидеть два типа ограничений NULL и NOT NULL. Второй тип ограничения преимущественно используется в составе первичного и внешнего ключа. Он означает что данный атрибут должен обязательно быть заполнен

данными. Ограничение NULL напротив сообщает, что данная ячейка в таблице базы данных может оставаться при определённых условиях пустой.

Учитывая все эти особенности, была спроектирована логическая модель базы данных.

### 4.3 Физическое проектирование базы данных

Под конец, опираясь на логическую модель базы данных, разрабатывается физическая, продемонстрированная на рисунке 5. При создании этой модели, уже учитываются особенности конкретной СУБД, а именно PostgreSQL.

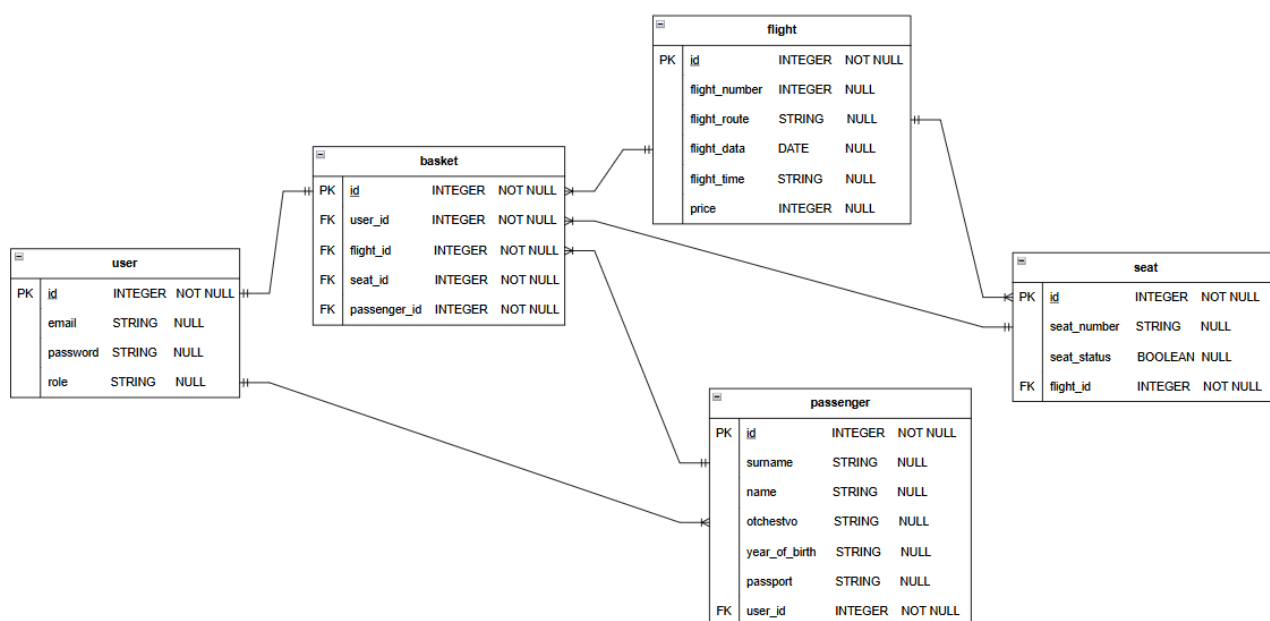


Рисунок 5 – Физическая модель базы данных

Физическая модель данных содержит пять сущностей. Описание таблиц физической модели приведено в таблицах 11 - 15.

Таблица 11 – Описание полей таблицы «user»

| Имя атрибута | Физическое имя | Нулевое значение | Ключевое поле | Тип данных |
|--------------|----------------|------------------|---------------|------------|
| <u>id</u>    | id             | Не допустимо     | PK            | INTEGER    |

| Имя атрибута | Физическое имя | Нулевое значение | Ключевое поле | Тип данных |
|--------------|----------------|------------------|---------------|------------|
| email        | email          | Допустимо        |               | STRING     |
| password     | password       | Допустимо        |               | STRING     |
| role         | role           | Допустимо        |               | STRING     |

Таблица 12 – Описание полей таблицы «basket»

| Имя атрибута | Физическое имя | Нулевое значение | Ключевое поле | Тип данных |
|--------------|----------------|------------------|---------------|------------|
| <u>id</u>    | id             | Не допустимо     | PK            | INTEGER    |
| user_id      | user_id        | Не допустимо     | FK            | INTEGER    |
| flight_id    | flight_id      | Не допустимо     | FK            | INTEGER    |
| seat_id      | seat_id        | Не допустимо     | FK            | INTEGER    |
| passenger_id | passenger_id   | Не допустимо     | FK            | INTEGER    |

Таблица 13 – Описание полей таблицы «flight»

| Имя атрибута  | Физическое имя | Нулевое значение | Ключевое поле | Тип данных |
|---------------|----------------|------------------|---------------|------------|
| <u>id</u>     | id             | Не допустимо     | PK            | INTEGER    |
| flight_number | flight_number  | Допустимо        |               | INTEGER    |
| flight_route  | flight_route   | Допустимо        |               | STRING     |
| flight_data   | flight_data    | Допустимо        |               | DATE       |
| flight_time   | flight_time    | Допустимо        |               | STRING     |

| Имя атрибута | Физическое имя | Нулевое значение | Ключевое поле | Тип данных |
|--------------|----------------|------------------|---------------|------------|
| price        | price          | Допустимо        |               | INTEGER    |

Таблица 14 – Описание полей таблицы «seat»

| Имя атрибута | Физическое имя | Нулевое значение | Ключевое поле | Тип данных |
|--------------|----------------|------------------|---------------|------------|
| <u>id</u>    | id             | Не допустимо     | PK            | INTEGER    |
| seat_number  | seat_number    | Допустимо        |               | STRING     |
| seat_status  | seat_status    | Допустимо        |               | BOOLEAN    |
| Имя атрибута | Физическое имя | Нулевое значение | Ключевое поле | Тип данных |
| flight_id    | flight_id      | Не допустимо     | FK            | INTEGER    |

Таблица 15 – Описание полей таблицы «passenger»

| Имя атрибута  | Физическое имя | Нулевое значение | Ключевое поле | Тип данных |
|---------------|----------------|------------------|---------------|------------|
| <u>id</u>     | id             | Не допустимо     | PK            | INTEGER    |
| surname       | surname        | Допустимо        |               | STRING     |
| name          | name           | Допустимо        |               | STRING     |
| otchestvo     | otchestvo      | Допустимо        |               | STRING     |
| year_of_birth | year_of_birth  | Допустимо        |               | STRING     |
| passport      | passport       | Допустимо        |               | STRING     |
| user_id       | user_id        | Не допустимо     | FK            | INTEGER    |

В физическую модель добавляются типы данных для каждого атрибута. Так на рисунке 5 можно увидеть следующие разновидности типов данных: INTEGER – целочисленный тип данных, который необходим для работы с целыми числами, кроме того с данным типом данных, в условиях разработки, удобно работать с математическими операциями, например сложение с другим числом перед добавлением в базу данных; STRING – строковый тип данных, который удобен в использовании при сочетании буквенно-цифровых символов; BOOLEAN – логический тип данных, который позволяет использовать только два значения true или false. В условиях разработки системы регистрации используется только в столбце seat\_status, для учета состояния мест. По умолчанию столбец заполняется значениями false, а при покупке билета значение меняется на true. Это будет позволять в дальнейшем отрисовывать место, как занятое; DATE – тип данных, для работы с датами и временем. Для работы с этим типом данных стоит учитывать, то в каком формате данные хранятся и отображаются в конкретном СУБД. Это необходимо учитывать, чтобы при запросе данных из базы, отобразить пользователю дату в приемлемом для него виде.

Учитывая все эти особенности, была спроектирована физическая модель базы данных.

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               |      |
| Изм. | Лист | № докум. | Подпись | Дата |                               | 30   |

# 5 Разработка системы

## 5.1 Описание структуры программы

Клиент – серверное приложение реализовано с помощью языка программирования JavaScript в среде разработки Visual Studio Code и имеет структуру в виде отдельных частей клиента и сервера. Серверная часть приложения представлена на рисунке 6.

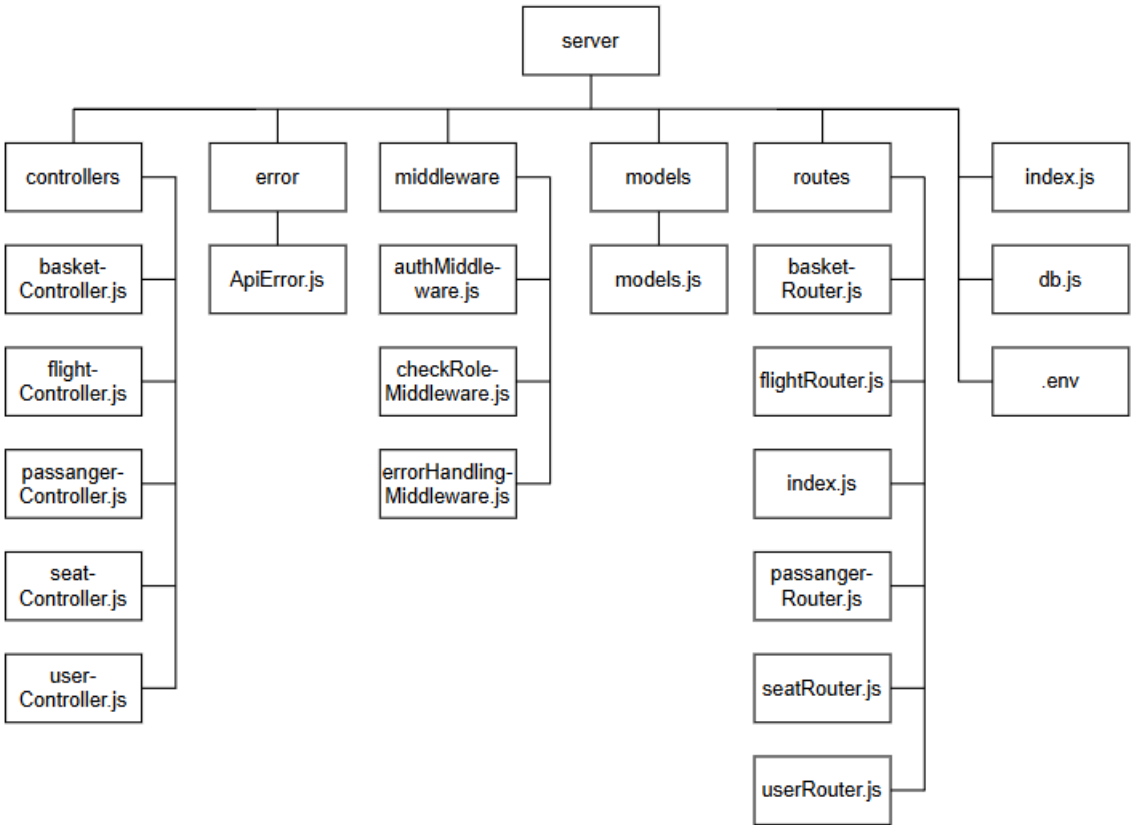


Рисунок 6 — Серверная часть приложения

Описание структуры серверной части представлено в таблице 16.

Таблица 16 – Описание структуры серверной части

| Название            | Описание                                                                                        |
|---------------------|-------------------------------------------------------------------------------------------------|
| basketController.js | Обработчик логики запросов на добавление, удаление, получение данных корзины из таблицы basket. |

| Название                    | Описание                                                                                                                                                                                |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| flightController.js         | Обработчик логики запросов на добавление, изменение, удаление, получение данных рейсов из таблицы flight.                                                                               |
| passangerController.js      | Обработчик логики запросов на добавление, изменение, удаление, получение данных пассажиров из таблицы passenger.                                                                        |
| seatController.js           | Обработчик логики запросов на добавление, изменение, удаление, получение данных мест конкретного рейса из таблицы seat.                                                                 |
| userController.js           | Обработчик логики запросов для регистрации пользователя, входа в программу, а также он необходим для проверки состояния входа пользователя.                                             |
| ApiError.js                 | Данный файл необходим для обработки ошибок и присвоения кода ошибки.                                                                                                                    |
| authMiddleware.js           | Здесь происходит проверка авторизации с использованием JSON Web Token (JWT).                                                                                                            |
| checkRoleMiddleware.js      | В данном коде реализована middleware-функция для проверки роли пользователя. Данная middleware-функция позволяет ограничивать доступ к определённым ресурсам, хранящимся на сервере.    |
| ErrorHandling-Middleware.js | В данном коде реализована middleware-функция для обработки ошибок. Эта функция предназначена для перехвата и обработки ошибок, которые могут возникнуть в процессе выполнения запросов. |
| models.js                   | В данном коде описывается создание таблиц и устанавливаются взаимосвязи между этими таблицами с использованием ORM sequelize.                                                           |

| Название           | Описание                                                                                                                                                                                |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| basketRouter.js    | В данном коде устанавливаются маршруты к функциям для выполнения запросов к таблице basket.                                                                                             |
| flightRouter.js    | В данном коде устанавливаются маршруты к функциям, middleware-функции checkRoleMiddleware для выполнения запросов к таблице flight.                                                     |
| index.js           | В данном коде создается основной маршрутизатор (router) для приложения. Этот маршрутизатор объединяет различные подмаршрутизаторы, которые обрабатывают запросы к различным частям API. |
| passangerRouter.js | Здесь происходит работа с маршрутами к таблице passenger.                                                                                                                               |
| seatRouter.js      | Здесь происходит работа с маршрутами к таблице seat.                                                                                                                                    |
| userRouter.js      | В данном коде устанавливаются маршруты к функциям, middleware-функции authMiddleware для выполнения запросов к таблице user.                                                            |
| .env               | В данном файле задаётся переменное окружение, также в нем содержится секретный ключ для расшифровки JWT токена.                                                                         |
| db.js              | В данном файле происходит создание базы данных с использованием ORM sequelize.                                                                                                          |
| index.js           | В данном файле происходит запуск сервера.                                                                                                                                               |

Клиентская часть приложения представлена на рисунке 7.

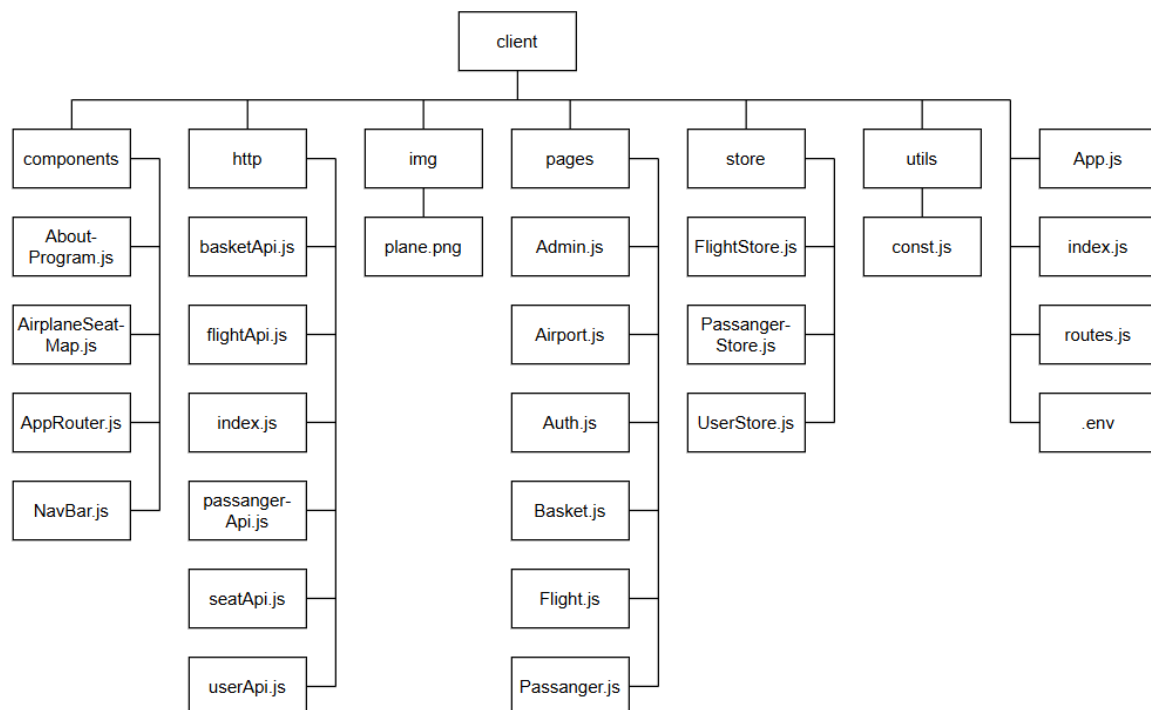


Рисунок 7 — Клиентская часть приложения

Описание структуры клиентской части представлено в таблице 17.

Таблица 17 – Описание структуры клиентской части

| Название           | Описание                                                                                                                                                                                                                           |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AboutProgram.js    | Данный код открывает модальное окно после нажатия кнопки «О программе».                                                                                                                                                            |
| AirplaneSeatMap.js | Данный код отображает схему мест в самолете, разделенную на бизнес-класс и эконом-класс. Компонент позволяет пользователю выбирать свободные места и визуально цветом помечает занятые и свободные места, а также выбранное место. |
| AppRouter.js       | Данный код отвечает за маршрутизацию в приложении на React с использованием библиотеки react-router-dom. Этот компонент определяет, какие маршруты доступны пользователю в зависимости от его статуса авторизации и роли.          |

| Название        | Описание                                                                                                                                                                                                                           |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NavBar.js       | В данном коде создается панель навигации вместе с кнопками, которым присваиваются маршруты для навигации по приложению.                                                                                                            |
| basketApi.js    | В данном коде реализованы функции для работы с корзиной в приложении. Эти функции позволяют добавлять элементы в корзину, получать содержимое корзины и удалять элементы из нее.                                                   |
| flightApi.js    | В данном коде реализованы функции для работы с рейсами в приложении. Эти функции позволяют создавать рейсы, получать список доступных рейсов, удалять необходимые рейсы, изменять информацию о рейсе.                              |
| index.js        | В данном коде создаются экземпляры Axios для выполнения HTTP-запросов к API, а также настраивается интерсептор для авторизованных запросов.                                                                                        |
| passangerApi.js | В данном коде реализованы функции для работы с пассажирами в приложении. Эти функции позволяют добавлять информацию о пассажире, изменять информацию о пассажире, удалять информацию о пассажире, получать информацию о пассажире. |
| seatApi.js      | В данном коде реализована функция для работы с местами конкретного рейса в приложении. Эта функция позволяет получать состояние мест.                                                                                              |

| Название       | Описание                                                                                                                                                                                                                                                 |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| userApi.js     | В данном коде реализованы функции для работы с пользователями приложения. Эти функции позволяют регистрировать пользователя в приложении, осуществлять вход пользователя в приложение, осуществлять проверку пользователя на авторизацию.                |
| plane.png      | Картинка самолета                                                                                                                                                                                                                                        |
| Admin.js       | В данном коде создаётся страница администратора, позволяющая добавлять, изменять, удалять рейсы.                                                                                                                                                         |
| Airport.js     | В данном коде создаётся главная страница приложения, которая вида всем пользователям.                                                                                                                                                                    |
| Auth.js        | В данном коде создаётся страница авторизации пользователя.                                                                                                                                                                                               |
| Basket.js      | В данном коде создаётся страница корзины пользователя с рейсами для покупки.                                                                                                                                                                             |
| Flight.js      | В данном коде создаётся страница для поиска и покупки рейсов.                                                                                                                                                                                            |
| Passanger.js   | В данном коде создаётся страница для регистрации пассажиров.                                                                                                                                                                                             |
| FlightStore.js | В данном коде реализован класс FlightStore, который используется для управления состоянием данных о рейсах с использованием библиотеки MobX, что позволяет автоматически отслеживать изменения и обновлять таблицу рейсов на сайте при изменении данных. |

| Название          | Описание                                                                                                                                                                                                                                        |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PassangerStore.js | Здесь реализован класс для управления состоянием пассажиров.                                                                                                                                                                                    |
| UserStore.js      | В данном коде реализован класс UserStore, который используется для управления состоянием данных о пользователях с использованием библиотеки MobX, что позволяет автоматически отслеживать изменения и обновлять интерфейс при изменении данных. |
| const.js          | В данном коде описаны маршруты для навигации по страницам.                                                                                                                                                                                      |
| App.js            | В данном коде реализован основной компонент приложения, который управляет состоянием пользователя, выполняет проверку авторизации и отображает интерфейс приложения.                                                                            |
| Index.js          | В данном коде происходит инициализация и настройка контекста для управления состоянием в приложении. Этот контекст будет использоваться для предоставления доступа к различным хранилищам состояния в компонентах приложения.                   |
| routes.js         | В данном коде описаны маршруты для авторизованных и не авторизованных пользователей, а также для администратора приложения.                                                                                                                     |
| .env              | В данном коде описан URL сервера.                                                                                                                                                                                                               |

## 5.2 Реализация регистрации и авторизации пользователей в системе

Регистрация и авторизация пользователей в системе регистрации пассажиров аэропорта, реализована на клиентской и серверной частях.

На клиентской части, функции, для работы с аутентификацией пользователей, располагаются в файле «userApi.js».

Сначала импортируются объекты \$authHost и \$host, которые представляют собой экземпляры, создаваемые с помощью библиотеки Axios, для зарегистрированных и не зарегистрированных пользователей. Также импортируется функция jwtDecode, которая необходима для декодирования токена.

Функция registration принимает переменные email и password, затем отправляет запрос для создания пользователя в базе данных, добавляя информацию о роли пользователя. После токен сохраняется в локальное хранилище localStorage. Функция возвращает декодированный токен, который содержит информацию о почте, пароле и роли пользователя.

Функция login принимает переменные email и password, затем отправляет запрос для поиска пользователя в базе данных. Если пользователь с такими данными существует, тогда токен сохраняется в локальное хранилище localStorage.

Функция check предназначена для проверки того, что пользователь с конкретными данными email и password вошёл в систему. Она отправляет запрос на адрес с помощью \$authHost, требующий наличия токена. Если запрос успешен, токен (JWT) сохраняется в локальное хранилище localStorage. Возвращается декодированный токен.

Страница авторизации/регистрации пользователей располагается в файле «Auth.js». В данный файл импортируются маршруты AIRPORT\_ROUTE, LOGIN\_ROUTE, REGISTRATION\_ROUTE из файла «const.js» для перемещения между страницей регистрации и авторизации пользователей, а также маршрут первоначальной страницы, после успешного входа в систему. Также в данный файл

импортируются функции registration, login из файла «userApi.js» для работы с регистрацией и авторизацией пользователя.

В функции click, если пользователь находится на странице авторизации вызывается функция login, если он находится на странице регистрации вызывается функция registration. После успешного выполнения функции, обновляется состояние пользователя в MobX (хранилище информации) и происходит навигация на страницу AIRPORT\_ROUTE (начальная страница приложения). Если возникает ошибка, выводится соответствующее сообщение об этом.

В файл «App.js» импортируется функция check из файла «userApi.js», для проверки аутентификации пользователя.

В данном фрагменте кода используется useEffect, для выполнения проверки аутентификации. Если в localStorage есть токен, вызывается функция check, которая проверяет наличие токена. Если токен верный, состояние пользователя обновляется, и устанавливается флаг setIsAuth в true, который означает, что пользователь находится в системе. В случае ошибки выводится сообщение в консоль, и состояние пользователя очищается в user.setUser.

На серверной части, методы, для обработки запросов пользователей на регистрацию, вход, проверку аутентификации, располагаются в файле «userController.js».

В данном файле импортируется класс ApiError, который используется для обработки ошибок. Также импортируется библиотека bcrypt для хеширования пароля и импортируется таблица пользователя User, содержащая столбцы email, password, role.

Функция generateJwt генерирует токен, который содержит идентификатор пользователя id, его электронную почту email и роль пользователя role. Токен закрепляется с использованием секретного ключа SECRET\_KEY, и имеет срок действия 24 часа.

Класс User Controller содержит три метода: registration, login и check.

Метод registration принимает данные email, password и role из тела запроса. Если данные не содержат информации возвращается ошибка. Далее проверяется, существует ли пользователь с таким email в базе данных, если существует, возвращается соответствующее сообщение об этом. Затем, для защиты информации, находящейся в базе данных, пароль хешируется с использованием bcrypt и происходит создание нового пользователя. Под конец генерируется токен для нового пользователя и возвращается в ответе.

Метод login принимает email и password из тела запроса. Если пользователя с данной электронной почтой не существует, возвращается соответствующая ошибка. Затем сравнивается введенный пароль с зашифрованным, в случае несовпадения паролей, пользователь получает сообщение об этом. В конце создаётся токен и возвращается в ответе функции.

Метод check нужен для проверки состояния пользователя.

### 5.3 Реализация выбора места на рейс в системе

Выбор места на рейс в системе регистрации пассажиров аэропорта реализован на клиентской и серверной частях.

На клиентской части в файле «Flight.js» таблица доступных рейсов для покупки заполняется информацией и из базы данных. Рядом с каждым рейсом располагается кнопка «Выбрать место». Когда пользователь нажимает эту кнопку для конкретного рейса, вызывается функция handleSelectSeat.

Данная функция загружает места для выбранного рейса с помощью функции fetchSeatsByFlight, которая отправляет запрос о состоянии мест в базу данных, fetchSeatsByFlight располагается в файле seatApi.js. Затем идёт открытие модального окна. В случае ошибки выводится сообщение об этом.

Внутри модального окна отображается схема расположения мест с помощью компонента AirplaneSeatMap. Также, внутри модального окна располагается компонент Dropdown для выбора зарегистрированного пассажира. А в футере модального окна находятся кнопки «Добавить в корзину» и «Закрыть». Когда

пользователь нажимает кнопку «Добавить в корзину», вызывается функция `handleAddToBasket`.

В данной функции, если место не выбрано выводится сообщение об этом, если пассажир не выбран, также выводится соответствующее сообщение. Если все условия выполнены, происходит добавление рейса в корзину с помощью функции `addToBasket`. Когда пользователь нажимает кнопку «Закрыть», модальное окно закрывается.

Компонент `AirplaneSeatMap` отвечает за отображение схемы мест в самолете, также он выделяет цветом свободные места (зелёный и синий цвета, в зависимости от класса эконом или бизнес), занятые места (красный цвет), выбранное место (жёлтый цвет).

Компонент, расположенный в файле `AirplaneSeatMap.js` принимает: `seats` – массив мест, содержащий информацию о каждом месте, `selectedSeat` – текущее выбранное место, `onSelectSeat` – функция, вызываемая при выборе места. Затем места разделяются на бизнес-класс и эконом-класс по номеру ряда. Бизнес-класс включает ряды 1 и 2, а эконом-класс — ряды 3, 4, 5, 6. Функция `groupSeatsByRow` группирует места по номерам рядов и сортирует их по номерам мест. Функция `renderRow` отрисовывает ряд мест, разделяя их на левую и правую стороны в зависимости от класса (бизнес или эконом). Места отображаются с помощью кнопок, которые вызывают функцию `renderSeat`. Функция `renderSeat` создает кнопку для каждого места. Цвет кнопки зависит от статуса места: занято – красный, свободно – основной цвет (бизнес — синий, эконом — зеленый), выбрано – желтый. При наведении курсора на кнопки выводятся вспомогательные сообщения о состоянии места. При нажатии на кнопку (если место свободно) вызывается функция `onSelectSeat`, в которую помещается выбранное место. Под конец компонент `AirplaneSeatMap` возвращает отрисованные места с заголовками.

В файле «`seatApi.js`» располагается функция `fetchSeatsByFlight`, которая отвечает за получение информации о местах для конкретного рейса.

Функция отправляет запрос на адрес с помощью \$authHost, требующий наличия токена. Она принимает параметр flightId, который является идентификатором рейса, для которого нужно получить информацию о местах. Функция возвращает данные о рейсе.

На серверной части, в файле «seatController.js» реализован контроллер SeatController, который отвечает за обработку запросов, связанных с местами.

В данном файле импортируется класс ApiError, который, используется для обработки ошибок. Импортируется таблица мест базы данных Seat. В методе getSeatsByFlight извлекается идентификатор рейса flightId из параметров запроса. Затем происходит запрос к базе данных для поиска всех мест по идентификатору рейса. Если запрос к базе данных успешен, возвращается ответ с массивом мест. Если произошла ошибка, вызывается middleware для работы с ошибкой, используя next(ApiError.internal(e.message)).

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 42   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

## 6 Функциональное тестирование

Функциональное тестирование предполагает проверку всех реализованных функций на предмет соответствия их техническому заданию.

После входа в систему открывается главная страница приложения системы регистрации пассажиров аэропорта (рисунок Б.1). На данной странице доступны кнопки «О программе» и «Вход в систему». После нажатия на кнопку «О программе», откроется модальное окно (рисунок Б.2). После нажатия на кнопку «Вход в систему», откроется страница авторизации (рисунок Б.3). Данная система имеет две роли пользователей, чтобы войти в систему под ролью администратора, необходимо ввести следующие данные: в поле «Введите вашу электронную почту» необходимо ввести «admin», а в поле «Введите ваш пароль» необходимо ввести «admin» (рисунок Б.4). Чтобы удостовериться в правильности введенного пароля, необходимо нажать на надпись «Показать пароль», в качестве результата мы увидим введенный пароль (рисунок Б.5). После нажатия на кнопку «Войти» откроется система под ролью администратора (рисунок Б.6). На данной странице доступны следующие кнопки «Админ панель» (данная кнопка доступна только для роли администратора), «О программе», «Регистрация пассажира», «Купить билет на рейс», «Корзина» (данные кнопки необходимы для проверки работы системы) и «Выйти из системы». После нажатия на кнопку «Админ панель», откроется страница администратора (рисунок Б.7). Данная страница необходима для добавления, редактирования и удаления рейсов на самолёт. Для проверки функции добавления рейса на данной странице, необходимо ввести тестовые данные: в поле «Введите номер рейса...» введём «1», выберем маршрут полёта «Пенза - Ульяновск», выберем дату вылета «30/05/2025», в поле «Введите время вылета/прибытия...» введём «4:00/6:00», в поле «Введите цену рейса...» введём «7000» (рисунок Б.8). После необходимо нажать на кнопку «Добавить рейс», в качестве результата мы увидим добавленный рейс (рисунок Б.9). Для проверки функции редактирования рейса, необходимо нажать на кнопку «Выбрать», расположенную рядом с тем рейсом, который необходимо отредактировать

|      |      |          |         |      |                        |      |
|------|------|----------|---------|------|------------------------|------|
|      |      |          |         |      | ПГУ 1.090501.03.001 ПЗ | Лист |
|      |      |          |         |      |                        |      |
| Изм. | Лист | № докум. | Подпись | Дата |                        | 43   |

(рисунок Б.10). После нажатия на кнопку «Выбрать», на этом месте появится новая кнопка «Отменить» и все данные выбранного рейса переместятся в поля административной панели для работы с рейсами (рисунок Б.11). Если необходимо отменить данную операцию нужно нажать на кнопку «Отменить», в качестве результата увидим страницу, как на рисунке Б.10. Затем, для проверки функции редактирования рейса, необходимо изменить данные: в поле «Введите номер рейса...» введём «6», выберем маршрут полёта «Пенза – Минеральные воды», оставим дату вылета без изменений «27/05/2025», в поле «Введите время вылета/прибытия...» введём «14:00/18:00», в поле «Введите цену рейса...» введём «10000» (рисунок Б.12). Далее необходимо нажать на кнопку «Редактировать рейс», в качестве результата мы увидим отредактированный рейс (рисунок Б.13). Для проверки функции удаления рейса, необходимо нажать на кнопку «Удалить», расположенную рядом с тем рейсом, который необходимо удалить (рисунок Б.14). Затем необходимо подтвердить данную операцию, нажав на кнопку «Удалить» (рисунок Б.15). В качестве результата, рейс, который необходимо было удалить, пропадёт из таблицы (рисунок Б.16). Чтобы выйти из системы и сменить роль пользователя необходимо нажать на кнопку «Выйти из системы», в качестве результата появится главная страница, как на рисунке Б.1. Чтобы зарегистрировать нового пользователя, необходимо нажать на кнопку «Вход в систему», в качестве результата появится страница, как на рисунке Б.3, затем необходимо нажать на надпись «Зарегистрироваться...», в качестве результата появится страница регистрации (рисунок Б.17). Далее, чтобы зарегистрировать нового пользователя введём следующие тестовые данные: в поле «Введите вашу электронную почту» введём «user2@mail.ru», а в поле «Введите ваш пароль» введём «user2» (рисунок Б.18). Затем необходимо нажать на кнопку «Зарегистрироваться», в качестве результата появится начальная страница пользователя с кнопками «О программе», «Регистрация пассажира», «Купить билет на рейс», «Корзина», «Выйти из системы» (рисунок Б.19). Чтобы зарегистрировать пассажира необходимо нажать на кнопку «Регистрация пассажира», в качестве результата увидим страницу

регистрации пассажиров (рисунок Б.20). Данная страница необходима для добавления, редактирования и удаления пассажиров. Для проверки функции добавления пассажира на данной странице, необходимо ввести тестовые данные: в поле «Введите фамилию пассажира...» введём «Буренков», в поле «Введите имя пассажира...» введём «Сергей», в поле «Введите отчество пассажира...» введём «Владимирович», в поле «Введите дату рождения пассажира...» введём «08.10.2002», в поле «Введите серию и номер паспорта...» введём «11 11 111111» (рисунок Б.21). После необходимо нажать на кнопку «Добавить пассажира», в качестве результата мы увидим добавленного пассажира в таблицу (рисунок Б.22). Зарегистрируем еще одного пассажира (рисунок Б.23). Для проверки функции редактирования пассажира, необходимо нажать на кнопку «Выбрать», расположенную рядом с тем пассажиром, которого необходимо отредактировать (рисунок Б.24). После нажатия на кнопку «Выбрать», на этом месте появится новая кнопка «Отменить» и все данные выбранного пассажира переместятся в поля регистрации пассажира (рисунок Б.25). Если необходимо отменить данную операцию нужно нажать на кнопку «Отменить», в качестве результата увидим страницу, как на рисунке Б.24. Затем, для проверки функции редактирования информации о пассажире, необходимо изменить данные: в поле «Введите фамилию пассажира...» оставим «Буренков», в поле «Введите имя пассажира...» оставим «Сергей», в поле «Введите отчество пассажира...» оставим «Владимирович», в поле «Введите дату рождения пассажира...» введём «08.10.2000», в поле «Введите серию и номер паспорта...» оставим «11 11 111111» (рисунок Б.26). Далее необходимо нажать на кнопку «Редактировать пассажира», в качестве результата мы увидим отредактированного пассажира (рисунок Б.27). Для проверки функции удаления пассажира, необходимо нажать на кнопку «Удалить», расположенную рядом с тем пассажиром, которого необходимо удалить (рисунок Б.28). Затем необходимо подтвердить данную операцию, нажав на кнопку «Удалить» (рисунок Б.29). В качестве результата, пассажир, которого необходимо было удалить, пропадёт из таблицы (рисунок Б.30). Чтобы купить билет на самолет необходимо

нажать на кнопку «Купить билет на рейс», в качестве результата увидим страницу покупки авиабилетов (рисунок Б.31). На данной странице существует возможность отсортировать билеты по маршруту полета и дате вылета, а также выбрать свободное место в самолете на конкретном рейсе. Чтобы отсортировать рейсы по маршруту полета необходимо выбрать из выпадающего списка (рисунок Б.32), конкретный рейс, например: «Пенза – Минеральный воды» (рисунок Б.33), затем необходимо нажать на кнопку «Отсортировать», в качестве результата на странице появятся отсортированные рейсы по маршруту полета «Пенза – Минеральный воды» (рисунок Б.34). Чтобы отменить сортировку необходимо нажать на кнопку «Отменить сортировку», в качестве результата увидим страницу как на рисунке Б.31. Чтобы отсортировать рейсы по дате вылета необходимо выбрать дату (рисунок Б.35), затем необходимо нажать на кнопку «Отсортировать», в качестве результата на странице появятся отсортированные рейсы по дате вылета (рисунок Б.36). Также существует возможность отсортировать рейсы по маршруту полета и дате вылета одновременно (рисунок Б.37). Для покупки авиабилета необходимо нажать на кнопку «Выбрать место», рядом с тем рейсом, который предполагается купить (рисунок Б.38). После нажатия на кнопку, откроется модальное окно для выбора места и зарегистрированного пассажира (рисунок Б.39). Синим и зелёным цветом помечаются свободные места, различия только в классе обслуживания, красным цветом помечено занятое место. Для покупки авиабилета необходимо выбрать место, нажав по нему, место сразу же поменяет цвет на желтый (рисунок Б.40). Затем необходимо выбрать зарегистрированного пассажира из списка (рисунок Б.41). Если место не выбрано выведется соответствующее сообщение об этом (рисунок Б.42). Если пассажир не выбран выведется соответствующее сообщение об этом (рисунок Б.43). Далее для покупки авиабилета после выбора места и пассажира, необходимо нажать на кнопку «Добавить в корзину», в качестве результата увидим соответствующее сообщение (рисунок Б.44). Теперь, нажав на кнопку «Корзина», откроется страница с авиабилетами готовыми к покупке (рисунок Б.45), на данной странице располагается только что купленный рейс.

Также добавленное место в корзину теперь помечено красным цветом (рисунок Б.46). Кроме того, на данной странице существует возможность удалить рейс из корзины, нажав на кнопку «Удалить», расположенную рядом с рейсом, который необходимо удалить (рисунок Б.47). Затем необходимо подтвердить данную операцию, нажав на кнопку «Удалить» (рисунок Б.48). В качестве результата увидим страницу корзины с отсутствующим рейсом (рисунок Б.49). После выхода из системы и при следующем заходе с этими же данными электронной почты и пароля вход будет осуществляться через страницу авторизации, при неверном вводе электронной почты, будет выведено соответствующее сообщение об этом (рисунок Б.50). Также при вводе неправильного пароля будет выведено соответствующее сообщение об этом (рисунок Б.51).

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               |      |
| Изм. | Лист | № докум. | Подпись | Дата |                               | 47   |

## 7 Оценка затрат на разработку системы

### 7.1 Организация производства системы регистрации

Разработка системы регистрации происходит в течении 4 месяцев на собственной территории общей площадью 28 кв.м. Содержание помещения в месяц, учитывая оплату интернета, коммунальных услуг, в которые входит плата за холодное водоснабжение, горячее водоснабжение, электроснабжение, тепловую энергию (в холодное время года) и вывоз мусора, составит приблизительно 20000 рублей в месяц.

Для разработки системы приобретено оборудование (системные блоки), периферийные устройства (компьютерные мышки, клавиатуры, колонки, мониторы, источники бесперебойного питания, Wi-Fi роутер, принтер), лицензионное программное обеспечение (программы для разработки системы, антивирусы) и дополнительные ресурсы (компьютерные столы, компьютерные кресла, коврики для компьютерной мышки, настольные светильники, различные канцтовары, огнетушитель).

Данные по потребности в оборудовании, периферийных устройствах, лицензионном программном обеспечении и дополнительных ресурсах представлены соответственно в таблицах 18, 19, 20 и 21.

Таблица 18 – Потребность в оборудовании

| Оборудование                   | Источник покупки          | Количество | Ориентировочная цена за единицу, рублей | Потребляемая мощность, Ватт |
|--------------------------------|---------------------------|------------|-----------------------------------------|-----------------------------|
| Системный блок DEXP Atlas H482 | Магазин электроники «DNS» | 4          | 31799                                   | 350                         |

Таблица 19 – Потребность в периферийных устройствах

| Устройство                                       | Источник покупки             | Количество | Ориентировочная цена за единицу, рублей | Потребляемая мощность, Ватт |
|--------------------------------------------------|------------------------------|------------|-----------------------------------------|-----------------------------|
| Компьютерная мышь проводная<br>ARDOR GAMING Fury | Магазин электроники<br>«DNS» | 4          | 1599                                    | 4                           |
| Клавиатура проводная<br>Logitech K120            | Магазин электроники<br>«DNS» | 4          | 1099                                    | 5                           |
| Колонки DEXP R350                                | Магазин электроники<br>«DNS» | 4          | 1599                                    | 40                          |
| Монитор MSI PRO MP273A 27"                       | Магазин электроники<br>«DNS» | 4          | 11999                                   | 200                         |
| ИБП DEXP CEE-E 650VA                             | Магазин электроники<br>«DNS» | 4          | 4000                                    | 390                         |
| Wi-Fi роутер HUAWEI AX3                          | Магазин электроники<br>«DNS» | 1          | 3599                                    | 50                          |
| Принтер лазерный<br>Canon                        | Магазин электроники<br>«DNS» | 1          | 17899                                   | 259                         |

Таблица 20 – Потребность в лицензионном программном обеспечении

| Лицензионное программное обеспечение | Источник                                                                                                                    | Цена, рублей | Срок действия лицензии, дней |
|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|--------------|------------------------------|
| Microsoft Windows 10                 | Входит в комплект системного блока «DEXP Atlas H482»                                                                        | -            | бессрочная                   |
| Visual Studio Code                   | Интернет сайт URL:<br><a href="https://code.visualstudio.com/download">https://code.visualstudio.com/download</a>           | -            | Бессрочная                   |
| Антивирус Dr.Web Security Space      | Магазин электроники «DNS»                                                                                                   | 1599         | 365                          |
| СУБД PostgreSQL                      | Интернет-магазин Postgresql URL:<br><a href="https://www.postgresql.org/download/">https://www.postgresql.org/download/</a> | -            | бессрочная                   |

Таблица 21 – Потребность в дополнительных ресурсах

| Ресурс                                    | Источник                  | Количество, штук | Ориентировочная цена за единицу, рублей |
|-------------------------------------------|---------------------------|------------------|-----------------------------------------|
| Стол компьютерный ARDOR GAMING Iron shell | Магазин электроники «DNS» | 4                | 14999                                   |
| Компьютерное кресло Aceline               | Магазин электроники «DNS» | 4                | 3899                                    |
| Коврик для мышки ARDOR GAMING             | Магазин электроники «DNS» | 4                | 399                                     |

| Ресурс                           | Источник                        | Количество,<br>штук | Ориентировочная<br>цена за единицу,<br>рублей |
|----------------------------------|---------------------------------|---------------------|-----------------------------------------------|
| Настольный светильник<br>СТАРТ   | Магазин<br>электроники<br>«DNS» | 4                   | 1250                                          |
| Канцтовары                       | Магазин «Лист»                  | 4                   | 1000                                          |
| Огнетушитель<br>Ярпожинвест ОП-3 | Магазин «Все<br>инструменты»    | 1                   | 2000                                          |

Разработка системы регистрации пассажиров аэропорта осуществлялась командой разработчиков, представленных в таблице 22.

Таблица 22 – Состав команды разработчиков

| Специалист                                 | Количество, человек | Описание занятости                                                                                                                                                   |
|--------------------------------------------|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Менеджер<br>проекта                        | 1                   | Управление проектом, координация разработчиков, взаимодействие с заказчиком, планирование задач.                                                                     |
| Программист<br>(Full-stack<br>разработчик) | 1                   | Проектирование базы данных, клиентской и серверной частей приложения. Написание программного кода всего приложения. Разработка дизайна пользовательского интерфейса. |
| Тестировщик                                | 1                   | Составление плана тестирования. Тестирование всех частей приложения на наличие ошибок. Взаимодействие с программистом для исправления продукта.                      |

| Специалист           | Количество, человек | Описание занятости                                       |
|----------------------|---------------------|----------------------------------------------------------|
| Технический писатель | 1                   | Создание документации для пользователей и разработчиков. |

Данные по основной заработной плате разработчиков системы регистрации пассажиров аэропорта и их количественному составу приведены в таблице 23.

Таблица 23 – Сведения о заработной плате

| Специалист                           | Количество, человек | Сумма заработной платы, рублей |
|--------------------------------------|---------------------|--------------------------------|
| Менеджер проекта                     | 1                   | 100000                         |
| Программист (Full-stack разработчик) | 1                   | 90000                          |
| Тестировщик                          | 1                   | 60000                          |
| Технический писатель                 | 1                   | 55000                          |

## 7.2 Расчет себестоимости программного обеспечения

Для планирования бюджета необходимо рассчитать себестоимость программного обеспечения. Это позволит определить все затраты, связанные с разработкой, внедрением и поддержкой, а также оценить рентабельность проекта.

Себестоимость разработки системы регистрации пассажиров аэропорта будет складываться из прямых затрат, косвенных затрат и непредвиденных затрат. К прямым затратам относится заработная плата разработчиков системы, затраты на приобретение оборудования, периферийных устройств, лицензионного программного обеспечения, а также затраты на дополнительные ресурсы необходимые для разработки автоматизированной системы. К косвенным затратам относятся расходы, связанные с арендой офиса. Также необходимо учитывать, что в процессе разработки могут произойти непредвиденные обстоятельства. Поэтому

необходимо закладывать определённый процент от общей суммы всех затрат на непредвиденные расходы.

Для расчета общих сумм затрат необходимо учитывать количество сотрудников и то, что для каждого разработчика необходимо своё рабочее место.

Общая сумма затрат на оборудование складывается из таблицы 18 и составляет 127196 рублей.

Общая сумма затрат на периферийные устройства складывается из таблицы 19 и составляет 102682 рублей.

Общая сумма затрат на лицензионное программное обеспечение складывается из таблицы 20 и составляет 1599 рублей.

Общая сумма затрат на дополнительные ресурсы складывается из таблицы 21 и составляет 88188 рублей.

Общая сумма затрат на аренду офисного помещения в течении 4 месяцев составит 80000 рублей.

Затраты на заработную плату включают затраты на основную заработную плату разработчиков и отчисления на страховые взносы.

Общая сумма затрат на заработанную плату разработчиков складывается из таблицы 23 и составляет 305000 рублей в месяц. Так как проект разрабатывается в течении 4 месяцев затраты составят 1220000 рублей.

Для расчёта отчислений с зарплаты сотрудников в течении 4 месяцев необходимо учитывать НДФЛ (налог на доходы физических лиц) и отчисления на страховые взносы. К страховым взносам относятся отчисления в Пенсионный фонд РФ, в фонд обязательного медицинского страхования РФ, в фонд социального страхования РФ.

С 1 января 2025 года применяется новая прогрессивная шкала НДФЛ, где размер процентной ставки зависит от величины доходов в год. Так как программа разрабатывается в течении 4 месяцев, и общая сумма доходов каждого из сотрудников не превышает 2,4 млн рублей, процентная ставка НДФЛ будет 13 %.

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 53   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

Отчисления на страховые взносы производится согласно «Налоговому кодексу Российской Федерации», так как компания осуществляет деятельность в области IT, она может претендовать на льготную ставку (п. 5 ст. 427 НК РФ) равную 7,6% [6].

Данные по суммам затрат на каждого сотрудника с учётом НДФЛ и страховых взносов представлены в таблице 24.

Таблица 24 – Затраты на заработанную плату

| Сотрудник                                  | Заработная<br>плата,<br>рублей | НДФЛ,<br>13% | Страховые<br>взносы, 7,6<br>% | Затраты за 1<br>месяц | Сумма<br>затрат за 4<br>месяца |
|--------------------------------------------|--------------------------------|--------------|-------------------------------|-----------------------|--------------------------------|
| Менеджер<br>проекта                        | 100000                         | 13000        | 7600                          | 120600                | 482400                         |
| Программист<br>(Full-stack<br>разработчик) | 90000                          | 11700        | 6840                          | 108540                | 434160                         |
| Тестировщик                                | 60000                          | 7800         | 4560                          | 72360                 | 289440                         |
| Технический<br>писатель                    | 55000                          | 7150         | 4180                          | 66330                 | 265320                         |

Общая сумма затрат на заработанную плату разработчиков, с учётом НДФЛ и страховых взносов складывается из таблицы 24 и составляет 367830 рублей в месяц. Так как проект разрабатывается в течении 4 месяцев затраты составят 1471320 рублей. Сумма отчислений за 4 месяца на НДФЛ и страховые взносы составит:

$$1471320 - 1220000 = 251320 \text{ руб}$$

Обслуживание оборудования и периферийных устройств не будет учитываться, так как они приобретались с годовой гарантией, а на разработку системы необходимо 4 месяца.

Себестоимость разработки вычисляется как сумма всех затрат, необходимых для реализации проекта. Общая сумма всех затрат представлена в таблице 25.

Таблица 25 – Общая сумма всех затрат

| Статья затрат                                         | Сумма отчислений, рублей |
|-------------------------------------------------------|--------------------------|
| Приобретение оборудования                             | 127196                   |
| Приобретение периферийных устройств                   | 102682                   |
| Приобретение лицензионного программного обеспечения   | 1599                     |
| Затраты на дополнительные ресурсы                     | 88188                    |
| Затраты на заработную плату, с учётом налогообложения | 1471320                  |
| Затраты на аренду офиса                               | 80000                    |
| Итого:                                                | 1870985                  |

Общая сумма затрат на разработку системы составит 1870985 рублей. Кроме того, необходимо заложить 5% от общей суммы всех затрат на непредвиденные расходы. Таким образом, сумма необходимая для реализации проекта системы регистрации пассажиров аэропорта составит:

$$1870985 * 1,05 = 1964534,25 \text{ руб}$$

Данная сумма будет соответствовать себестоимости системы регистрации пассажиров аэропорта.

## 8 Общие принципы обеспечения безопасности

### 8.1 Актуальность вопросов безопасности человека при работе с ПЭВМ

Каждый год растёт количество сотрудников в компаниях использующих ПЭВМ (персональных электронных вычислительных машин), поэтому необходимо помнить про безопасность работы человека за ПЭВМ, для сохранения его здоровья и трудоспособности. Правильное расположение ПЭВМ, а также нахождение человека за рабочем местом с использованием ПЭВМ регламентируется санитарно-эпидемиологическими требованиями, действующими на всей территории Российской Федерации. Данные требования необходимо обязательно учитывать, так как работа за ПЭВМ имеет вредные факторы:

- физические факторы, к которым относятся повышенные уровни переменного электромагнитного и электростатического полей;
- психофизиологические факторы, воздействующие на пользователя, приводят к его физическим и нервно-психическим перегрузкам. Физическая перегрузка, как длительное статическое напряжение мышц. К факторам, приводящим к нервно-психическим перегрузкам, можно отнести повышенную неравномерность распределения яркости в поле зрения пользователя [7].

### 8.2 Требования к помещениям для работы с ПЭВМ

Чтобы правильно организовать помещение для работы с ПЭВМ необходимо учитывать следующие факторы:

- естественное освещение, которое должно осуществляться через окна, расположенные преимущественно на север и северо-восток и иметь жалюзи или внешние козырьки для регулирования освещения;
- искусственное освещение, которое необходимо при недостатке естественного освещения, оно должно быть равномерным и не давать лишних бликов при работе за ПЭВМ;

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 56   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

- расположение рабочих мест для работы с ПЭВМ не допускается в подвальных помещениях;
- согласно СанПиН 2.2.2/2.4.1340-03 «Санитарно-эпидемиологические правила и нормативы» [8], для внутренней отделки интерьера помещений, где расположены ПЭВМ, должны использоваться диффузно отражающие материалы с коэффициентом отражения для потолка - 0,7 - 0,8; для стен - 0,5 - 0,6; для пола - 0,3 - 0,5;
- в помещении должны быть установлены огнетушители;
- в помещении должен находиться план эвакуации на случай чрезвычайной ситуации;
- уровень шума должен находиться в пределах допустимых норм;
- согласно СанПиН 2.2.2/2.4.1340-03 «Санитарно-эпидемиологические правила и нормативы» [8] расстояние между рабочими местами должно составлять менее 1,2 метра;
- электропроводка должна соответствовать требованиям безопасности;
- помещение должно быть оборудовано защитным заземлением и электрическим щитком, с автоматическим отключением электричества при коротком замыкании.

### **8.3 Техника безопасности при работе с ПЭВМ**

Технику безопасности необходимо соблюдать для сохранения здоровья человека и предотвращение несчастных случаев. Нужно быть ознакомленным со следующим рядом правил:

- необходимо осуществлять регулярную проверку оборудования, электропроводки, сетевых фильтров и не допускать перегрузки электрических цепей;
- пользоваться только оборудованием, имеющим соответствующие сертификаты безопасности;
- сотрудники компаний должны проходить регулярные инструктажи безопасности при работе с ПЭВМ;

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 57   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

– необходимо избегать накопление легковоспламеняющихся материалов на рабочем месте;

– нельзя самостоятельно вскрывать оборудование для ремонта;

– в современное время также не стоит забывать про установку на ПЭВМ антивирусной защиты, которая предотвратит выход электронной машины из строя;

Кроме того, человек не должен забывать и про собственное здоровье, для этого необходимо делать регулярные перерывы для физической разминки.

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 58   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

## Заключение

В течении дипломного проектирования была разработана система регистрации пассажиров аэропорта. Были реализованы все необходимые функции для корректной работы приложения.

Была написана вся необходимая документация, доказывающая то, что программа удовлетворяет требованиям технического задания. Рассмотрены современные средства разработки, позволяющие проектировать, отлаживать, тестировать программу удобными методами. Были описаны модели базы данных от начала проектирования и до конца. Также рассмотрены структуры программы со стороны клиента и сервера. Описаны принципы работы важных частей программного кода приложения, а именно регистрация и авторизация пользователей, а также выбор пассажиром конкретного места на рейс. Кроме того, с помощью тестовых наборов данных было проведено функциональное тестирование, показывающее то, как пользоваться программой, что делать в случае всплывающих сообщений и доказывающее то, что система работает без перебоев и внезапно возникающих ошибок.

В разделе экономики удалось взглянуть на программу с экономической точки зрения. Было выяснено сколько необходимо человеческих ресурсов и денежных средств для разработки системы регистрации пассажиров аэропорта.

В разделе безопасности жизнедеятельности было выяснено на сколько важно знать технику безопасности при работе с ПЭВМ, а также было обозначено, то какую важную роль играют помещения, в которых должны находиться вычислительные машины.

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 59   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

### Список использованных источников

1. Определение базы данных [Электронный ресурс] – URL - <https://beseller.by/blog/bazy-dannykh-i-subd/> (дата 15.02.2025).
2. Статистика турпотока [Электронный ресурс] – URL - <https://tass.ru/ekonomika/23962053> (дата 15.02.2025).
3. Аэропорты в не больших городах [Электронный ресурс] – URL - <https://www.vokrugsveta.ru/articles/gde-redko-stupaet-noga-passazhira-posmotrite-na-10-malenkikh-aeroportov-rossii-id890581/> (дата 10.03.2025).
4. Руководство по Node.js [Электронный ресурс] – URL - <https://habr.com/ru/companies/ruvds/articles/422893/> (дата 18.03.2025).
5. Руководство по React.js [Электронный ресурс] – URL - <https://blog.skillfactory.ru/glossary/react/> (дата 20.03.2025)
6. Льготная ставка для it компаний [Электронный ресурс] – URL - [https://www.consultant.ru/document/cons\\_doc\\_LAW\\_93256/56dcda2dd757d46bd18a3dee5607ff5b18abd75d/](https://www.consultant.ru/document/cons_doc_LAW_93256/56dcda2dd757d46bd18a3dee5607ff5b18abd75d/) (дата 14.04.2025)
7. Вредные факторы ПЭВМ [Электронный ресурс] – URL - <https://adm-verhotury.ru/social/helth/media/2017/5/29/opasnyie-i-vrednyie-factoryi-pri-rabote-s-pevm/> (дата 26.04.2025)
8. СанПиН 2.2.2/2.4.1340-03 «Санитарно-эпидемиологические правила и нормативы». Изд-во стандартов, 2003. – 42, 47с.

## Приложение А

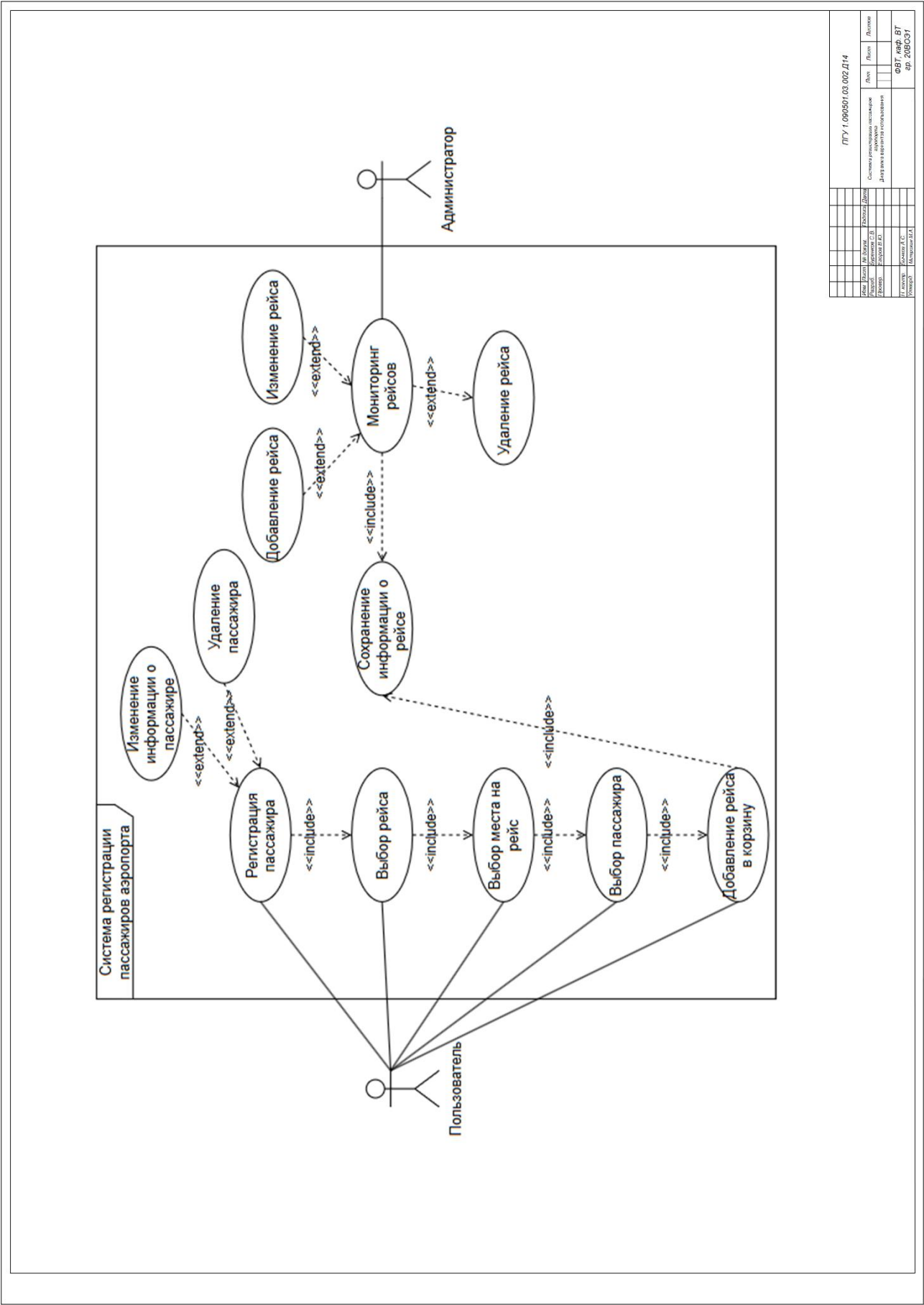
(обязательное)

Система регистрации пассажиров аэропорта

Диаграммы UML, модели, структуры

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 61   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

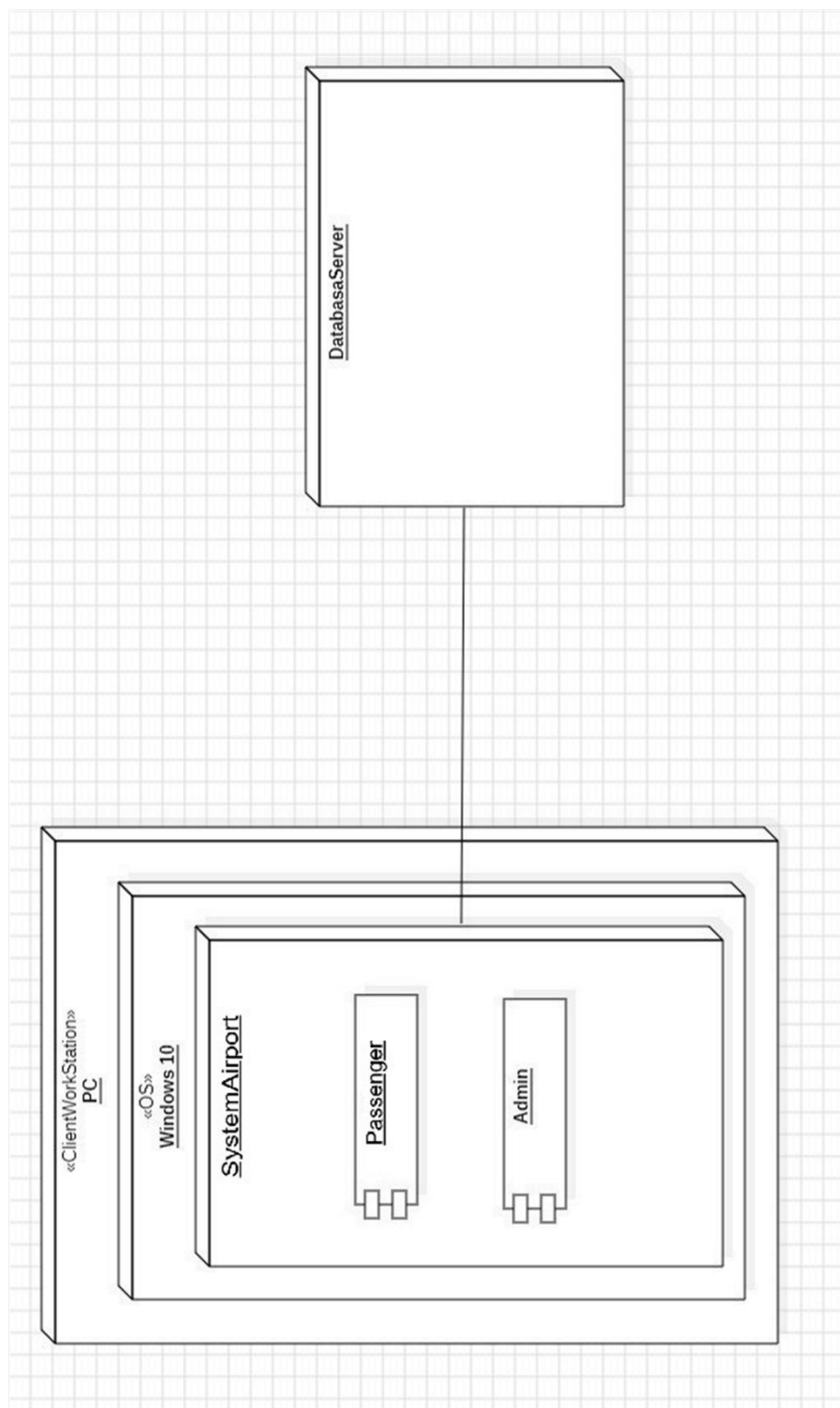
Приложение А.1 – Диаграмма вариантов использования



|                        |     |          |      |       |                                          |  |  |  |  |
|------------------------|-----|----------|------|-------|------------------------------------------|--|--|--|--|
| ПУ 1.090501.03.002 П14 |     |          |      |       |                                          |  |  |  |  |
| Имя                    | Пол | М. Долг. | Роль | Долг. | Система регистрации пассажиров аэропорта |  |  |  |  |
| Роль                   | Пол | М. Долг. | Роль | Долг. | Данные по регистрации пассажиров         |  |  |  |  |
| Роль                   | Пол | М. Долг. | Роль | Долг. | ФВТ, мф, ВТ                              |  |  |  |  |
| Роль                   | Пол | М. Долг. | Роль | Долг. | ФВТ, мф, ВТ                              |  |  |  |  |
| Роль                   | Пол | М. Долг. | Роль | Долг. | ФВТ, мф, ВТ                              |  |  |  |  |

|      |      |          |         |      |
|------|------|----------|---------|------|
| Изм. | Лист | № докум. | Подпись | Дата |
|      |      |          |         |      |

## Приложение А.2 – Диаграмма развертывания

[illegible]

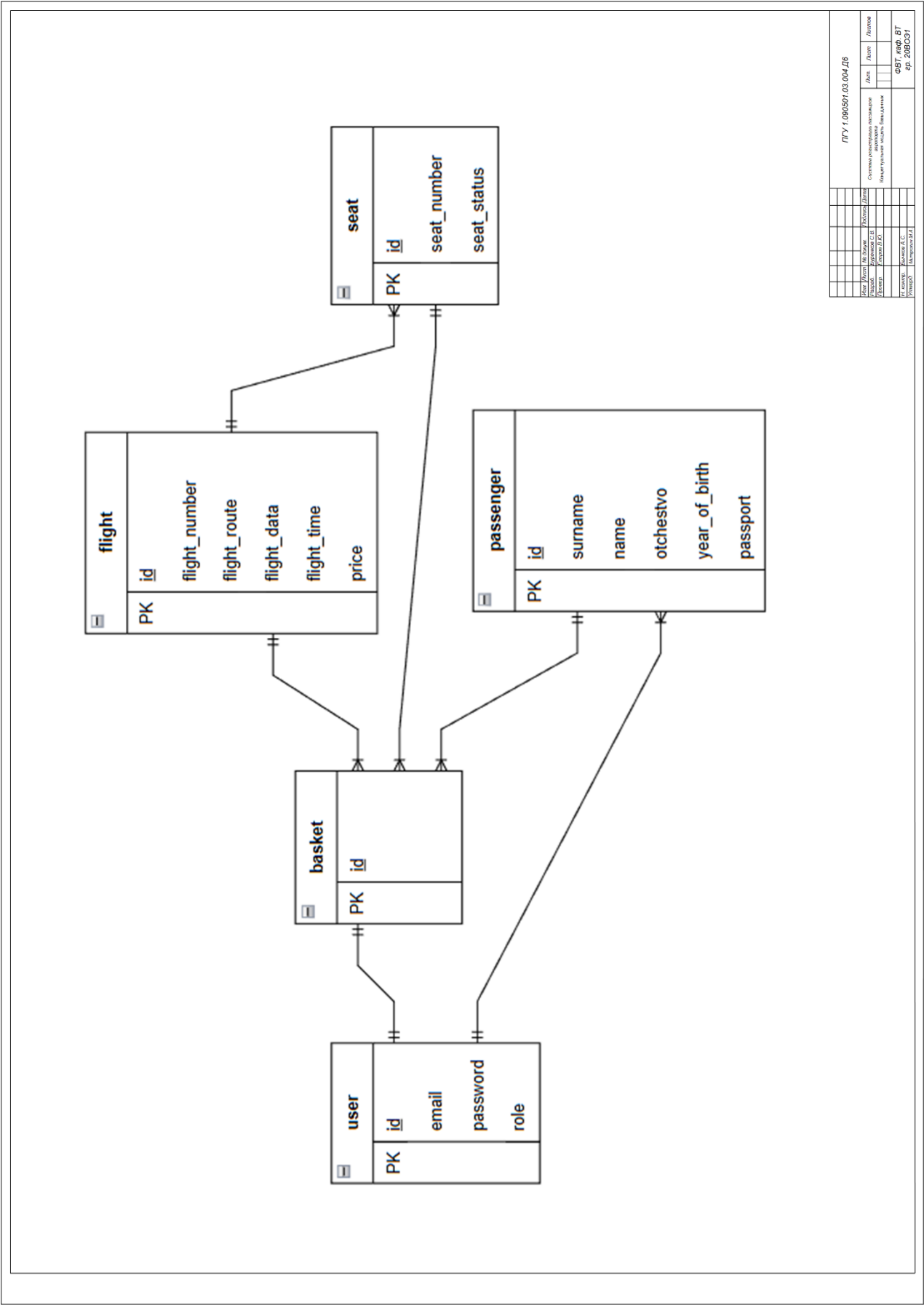
|      |      |          |         |      |
|------|------|----------|---------|------|
|      |      |          |         |      |
|      |      |          |         |      |
| Изм. | Лист | № докум. | Подпись | Дата |

ПГУ 1.090501.03.001 ПЗ

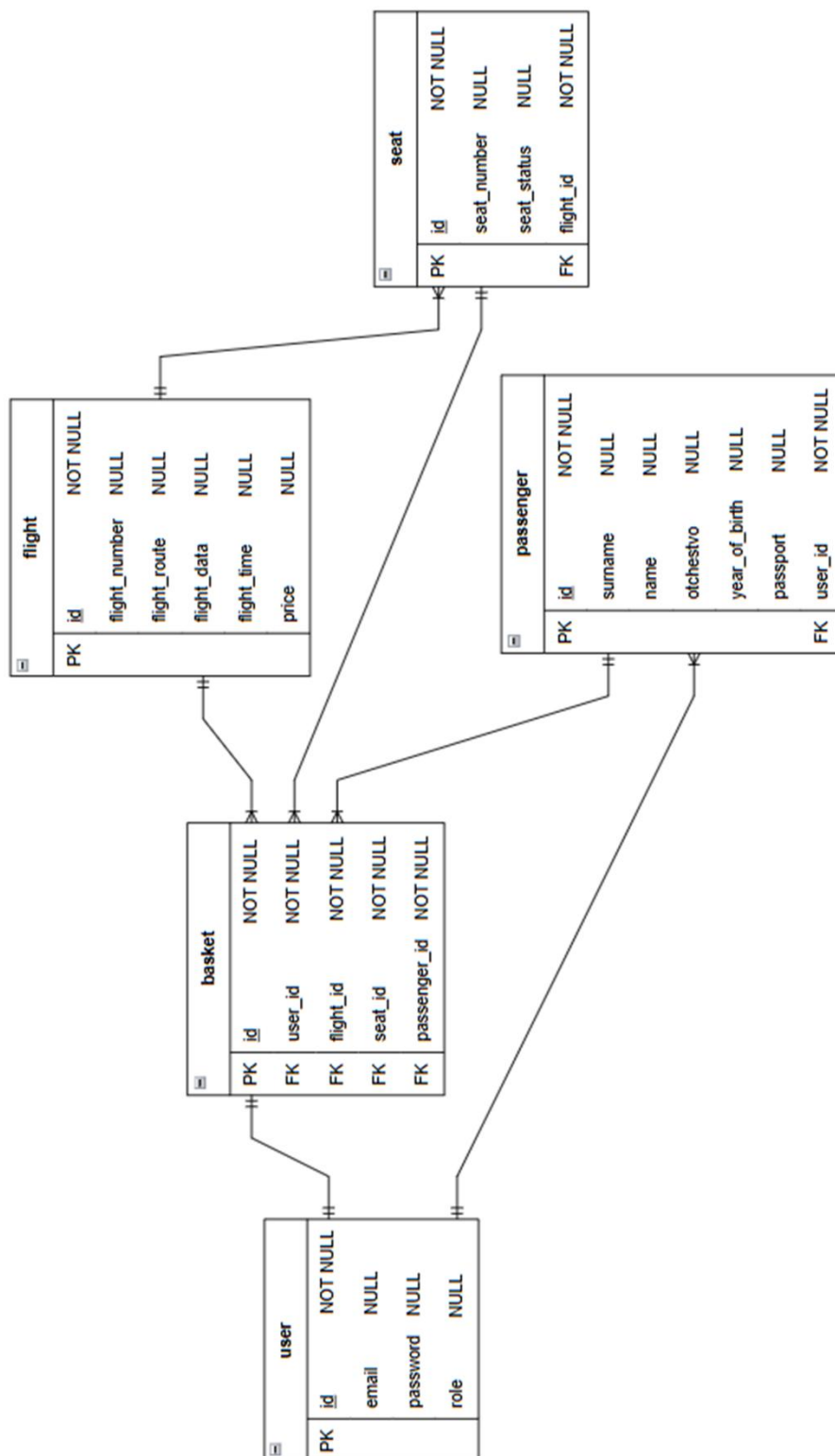
Лист

63

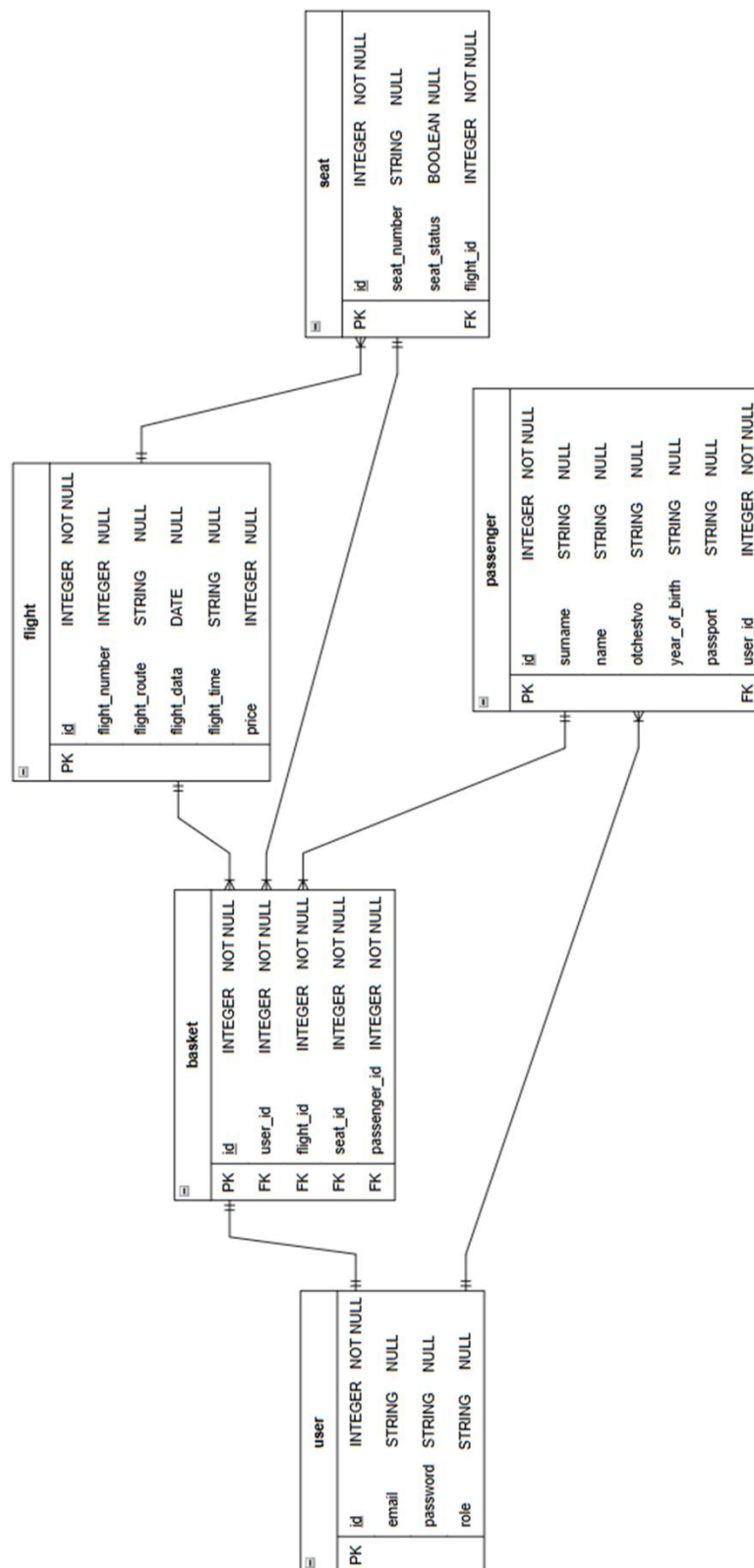
Приложение А.3 – Концептуальная модель базы данных



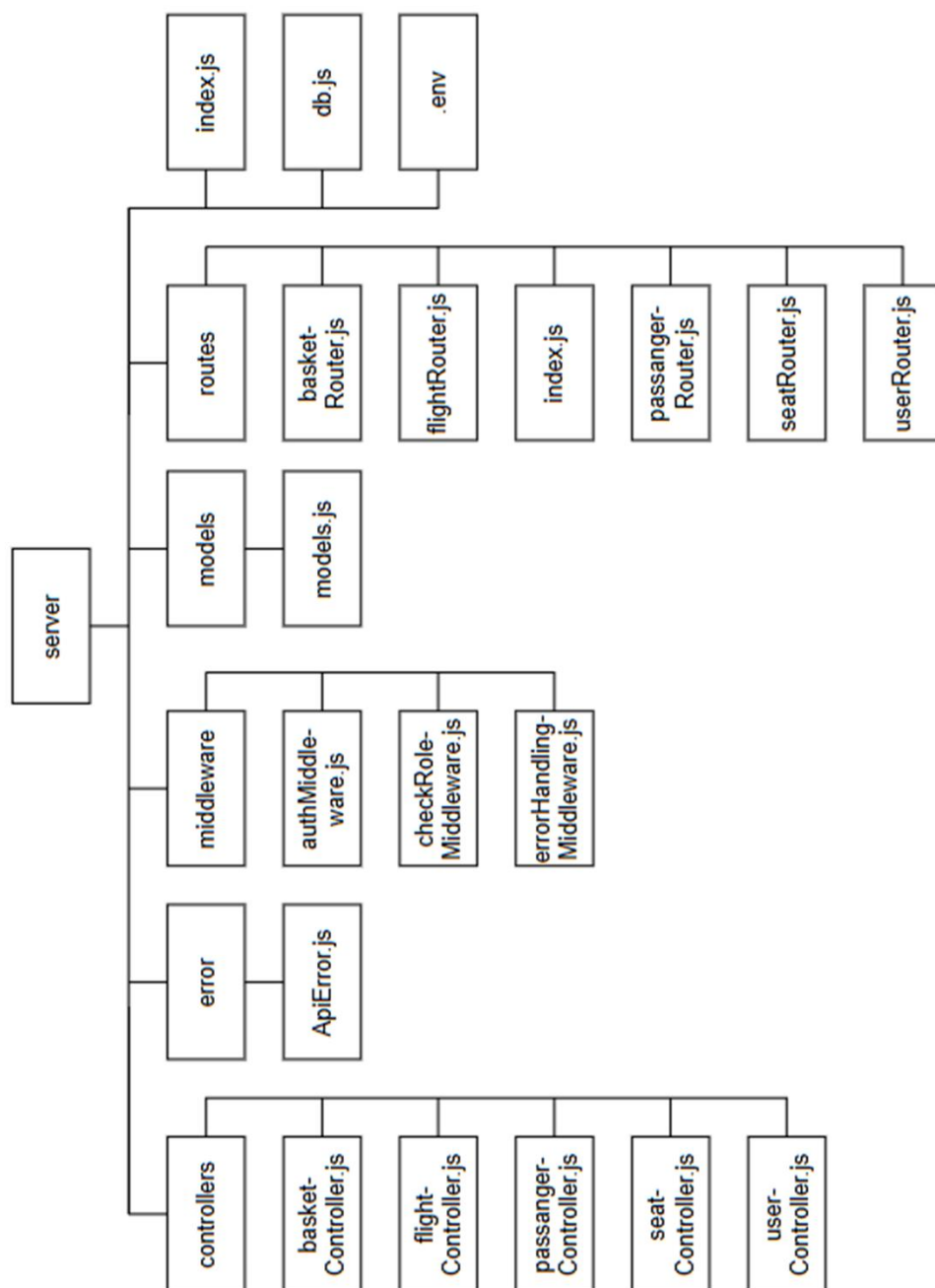
## Приложение А.4 – Логическая модель базы данных

[illegible]

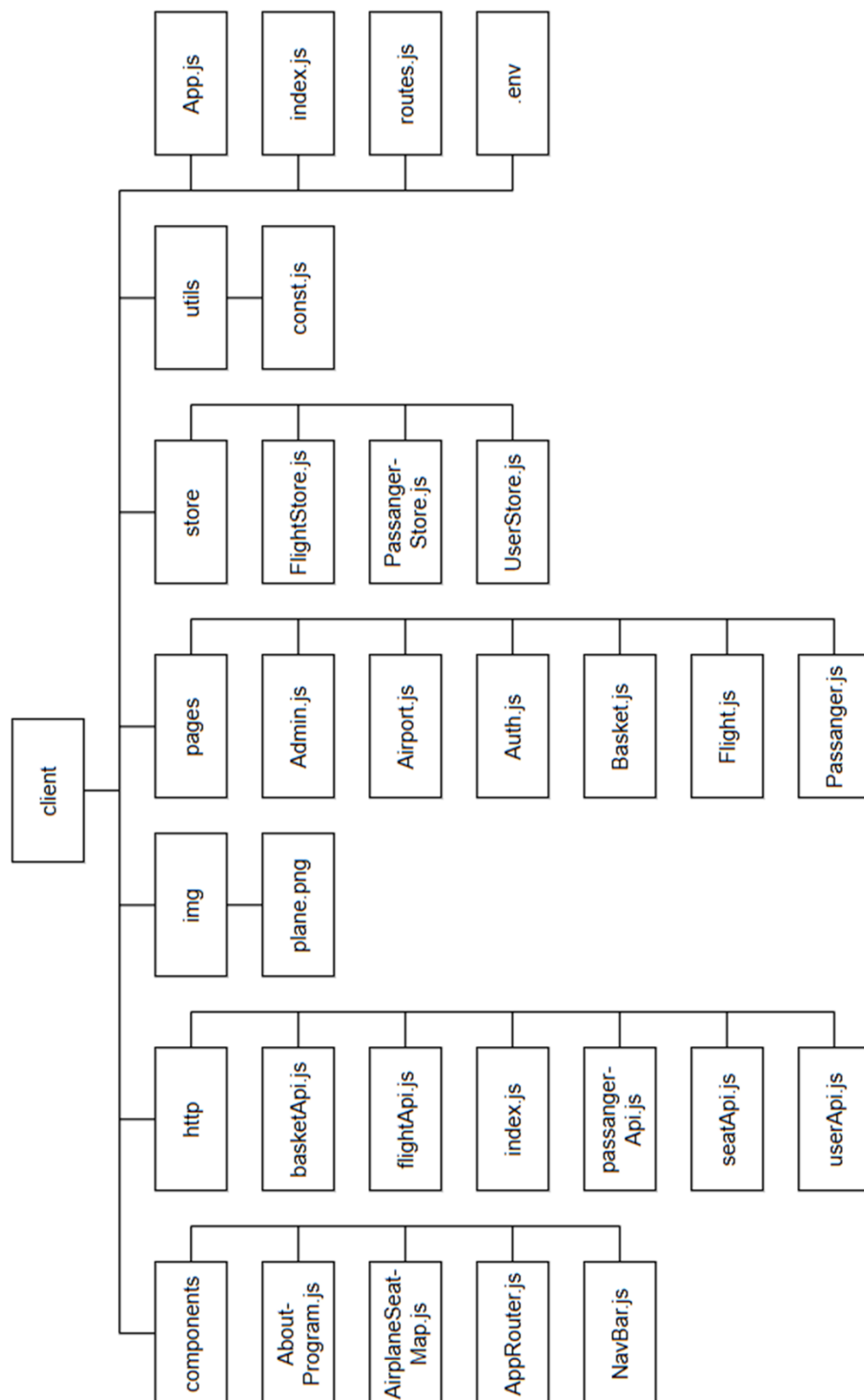
## Приложение А.5 – Физическая модель базы данных

[illegible]

## Приложение А.6 – Структура серверной части приложения

[illegible]

## Приложение А.7 – Структура клиентской части приложения

[illegible]

## Приложение Б

(обязательное)

Система регистрации пассажиров аэропорта

Результаты функционального тестирования

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 69   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

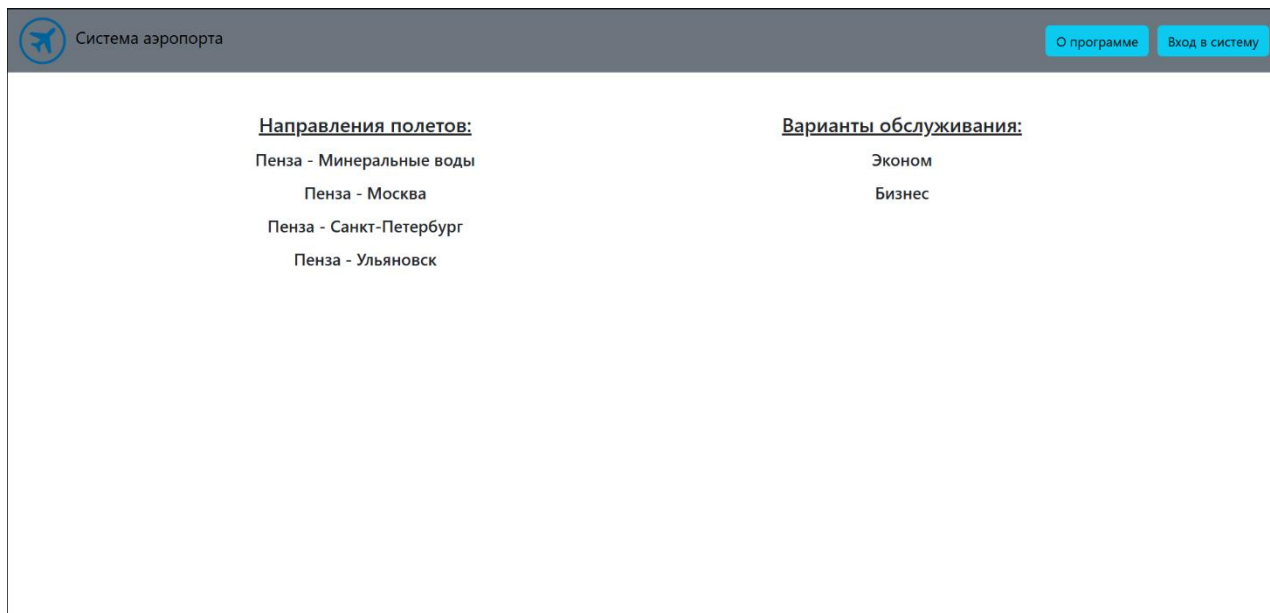


Рисунок Б.1 – Начальная страница системы

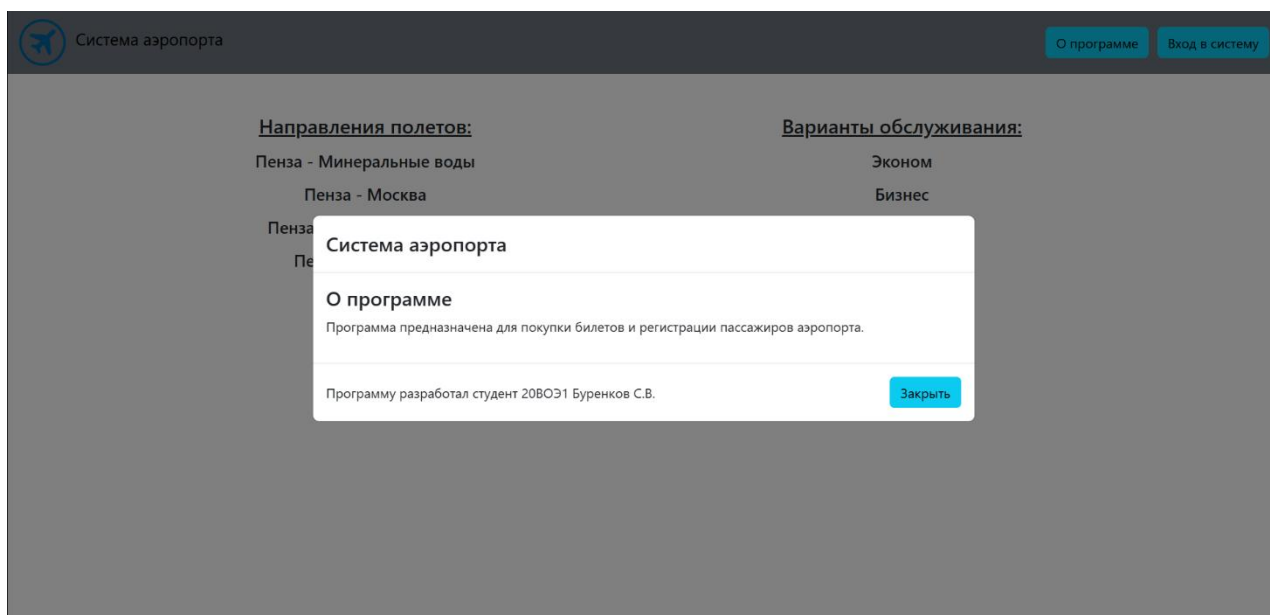


Рисунок Б.2 – Нажатие кнопки «О программе»







Система аэропорта

Добро пожаловать, admin

Админ панель

О программе

Регистрация пассажира

Купить билет на рейс

Корзина

Выйти из системы

Административная панель для работы с рейсами

Введите номер рейса...

Выберите маршрут полета...

Выберите дату вылета...

Добавить рейс

Введите время вылета/прибытия...

Введите цену рейса...

Редактировать рейс

Список доступных рейсов:

| Номер рейса | Маршрут полета           | Дата вылета | Время вылета/прибытия | Цена рейса в рублях |         |         |
|-------------|--------------------------|-------------|-----------------------|---------------------|---------|---------|
| 2           | Пенза - Москва           | 27.05.2025  | 12:00/15:00           | 8000                | Удалить | Выбрать |
| 3           | Пенза - Москва           | 28.05.2025  | 16:00/18:00           | 8000                | Удалить | Выбрать |
| 4           | Пенза - Санкт Петербург  | 24.05.2025  | 20:00/23:30           | 11000               | Удалить | Выбрать |
| 5           | Пенза - Минеральные воды | 27.05.2025  | 23:00/28.05.2025 3:00 | 10000               | Удалить | Выбрать |

Рисунок Б.7 – Страница административной панели



Система аэропорта

Добро пожаловать, admin

Админ панель

О программе

Регистрация пассажира

Купить билет на рейс

Корзина

Выйти из системы

Административная панель для работы с рейсами

1

Пенза - Ульяновск

30/05/2025

Добавить рейс

4:00/6:00

7000

Редактировать рейс

Список доступных рейсов:

| Номер рейса | Маршрут полета           | Дата вылета | Время вылета/прибытия | Цена рейса в рублях |         |         |
|-------------|--------------------------|-------------|-----------------------|---------------------|---------|---------|
| 2           | Пенза - Москва           | 27.05.2025  | 12:00/15:00           | 8000                | Удалить | Выбрать |
| 3           | Пенза - Москва           | 28.05.2025  | 16:00/18:00           | 8000                | Удалить | Выбрать |
| 4           | Пенза - Санкт Петербург  | 24.05.2025  | 20:00/23:30           | 11000               | Удалить | Выбрать |
| 5           | Пенза - Минеральные воды | 27.05.2025  | 23:00/28.05.2025 3:00 | 10000               | Удалить | Выбрать |

Рисунок Б.8 – Ввод тестовых данных







Система аэропорта

Добро пожаловать, admin

Админ панель

О программе

Регистрация пассажира

Купить билет на рейс

Корзина

Выйти из системы

Административная панель для работы с рейсами

Введите номер рейса...

Выберите маршрут полета...

Выберите дату вылета...

Добавить рейс

Введите время вылета/прибытия...

Введите цену рейса...

Редактировать рейс

Список доступных рейсов:

| Номер рейса | Маршрут полета           | Дата вылета | Время вылета/прибытия | Цена рейса в рублях |         |         |
|-------------|--------------------------|-------------|-----------------------|---------------------|---------|---------|
| 3           | Пенза - Москва           | 28.05.2025  | 16:00/18:00           | 8000                | Удалить | Выбрать |
| 4           | Пенза - Санкт Петербург  | 24.05.2025  | 20:00/23:30           | 11000               | Удалить | Выбрать |
| 5           | Пенза - Минеральные воды | 27.05.2025  | 23:00/28.05.2025 3:00 | 10000               | Удалить | Выбрать |
| 1           | Пенза - Ульяновск        | 30.05.2025  | 4:00/6:00             | 7000                | Удалить | Выбрать |
| 6           | Пенза - Минеральные воды | 27.05.2025  | 14:00/18:00           | 10000               | Удалить | Выбрать |

Рисунок Б.13 – Результат редактирования



Система аэропорта

Добро пожаловать, admin

Админ панель

О программе

Регистрация пассажира

Купить билет на рейс

Корзина

Выйти из системы

Административная панель для работы с рейсами

Введите номер рейса...

Выберите маршрут полета...

Выберите дату вылета...

Добавить рейс

Введите время вылета/прибытия...

Введите цену рейса...

Редактировать рейс

Список доступных рейсов:

| Номер рейса | Маршрут полета           | Дата вылета | Время вылета/прибытия | Цена рейса в рублях |         |         |
|-------------|--------------------------|-------------|-----------------------|---------------------|---------|---------|
| 3           | Пенза - Москва           | 28.05.2025  | 16:00/18:00           | 8000                | Удалить | Выбрать |
| 4           | Пенза - Санкт Петербург  | 24.05.2025  | 20:00/23:30           | 11000               | Удалить | Выбрать |
| 5           | Пенза - Минеральные воды | 27.05.2025  | 23:00/28.05.2025 3:00 | 10000               | Удалить | Выбрать |
| 1           | Пенза - Ульяновск        | 30.05.2025  | 4:00/6:00             | 7000                | Удалить | Выбрать |
| 6           | Пенза - Минеральные воды | 27.05.2025  | 14:00/18:00           | 10000               | Удалить | Выбрать |

Рисунок Б.14 – Кнопка «Удалить» рядом с рейсом

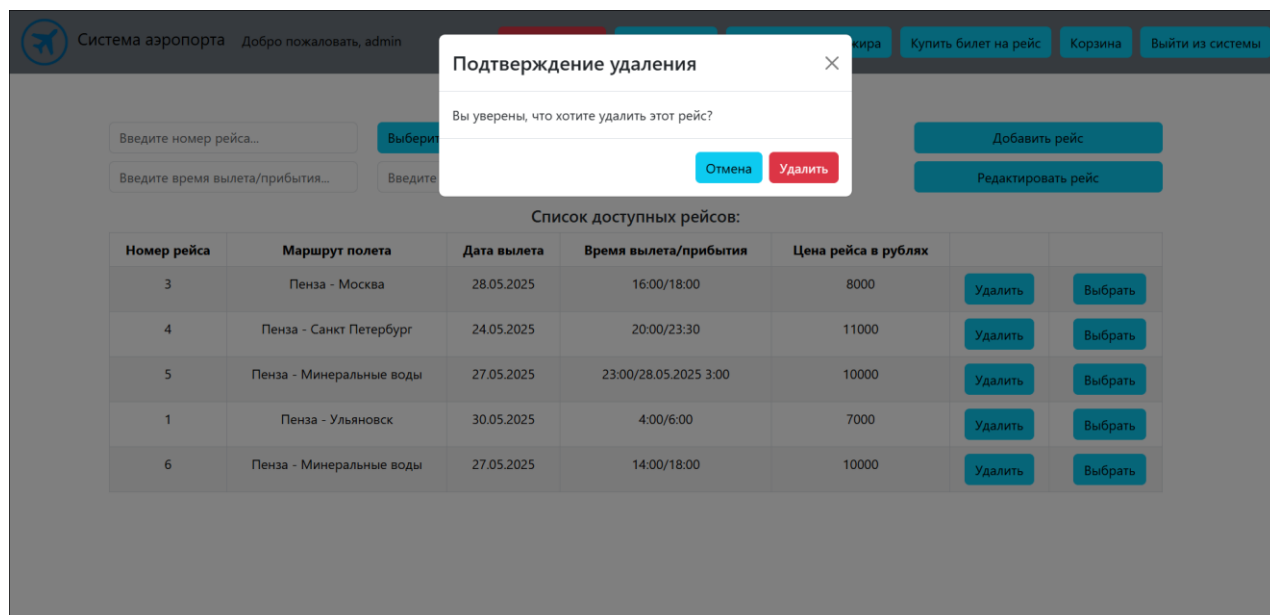


Рисунок Б.15 – Подтверждение удаления рейса

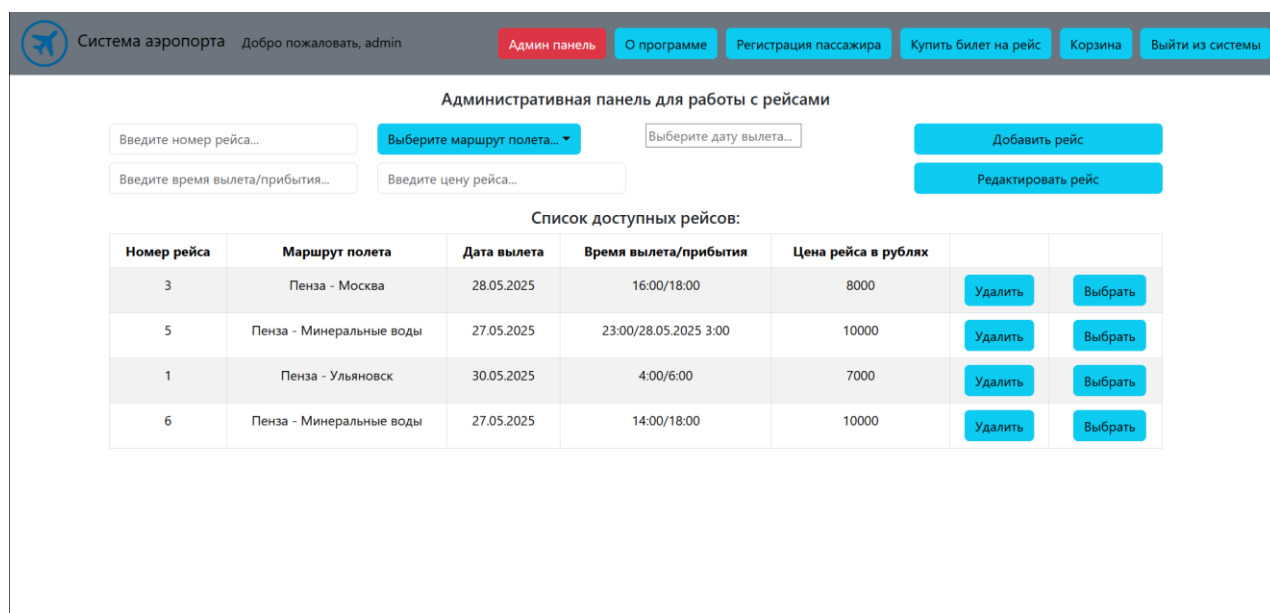


Рисунок Б.16 – Результат удаления рейса


Система аэропорта

О программе
Вход в систему

Регистрация

Введите вашу электронную почту...

Введите ваш пароль...

[Показать пароль?](#)
[Забыли пароль?](#)

[Есть аккаунт? Войдите...](#)
[Зарегистрироваться](#)

Рисунок Б.17 – Окно регистрации пользователя


Система аэропорта

О программе
Вход в систему

Регистрация

user2@mail.ru

user2

[Скрыть пароль?](#)
[Забыли пароль?](#)

[Есть аккаунт? Войдите...](#)
[Зарегистрироваться](#)

Рисунок В.18 – Ввод тестовых входных данных




Система аэропорта
Добро пожаловать, user2@mail.ru

О программе
Регистрация пассажира
Купить билет на рейс
Корзина
Выйти из системы

Зарегистрировать пассажира

Буренков

Сергей

Владимирович

Добавить пассажира

08.10.2002

11 11 111111

Редактировать пассажира

Список зарегистрированных пассажиров:

| Фамилия | Имя | Отчество | Дата рождения | Серия и номер паспорта |  |  |
|---------|-----|----------|---------------|------------------------|--|--|
|         |     |          |               |                        |  |  |

Рисунок Б.21 – Ввод тестовых данных


Система аэропорта
Добро пожаловать, user2@mail.ru

О программе
Регистрация пассажира
Купить билет на рейс
Корзина
Выйти из системы

Зарегистрировать пассажира

Введите фамилию пассажира...

Введите имя пассажира...

Введите отчество пассажира...

Добавить пассажира

Введите дату рождения пассажира...

Введите серию и номер паспорта...

Редактировать пассажира

Список зарегистрированных пассажиров:

| Фамилия  | Имя    | Отчество     | Дата рождения | Серия и номер паспорта |         |         |
|----------|--------|--------------|---------------|------------------------|---------|---------|
| Буренков | Сергей | Владимирович | 08.10.2002    | 11 11 111111           | Удалить | Выбрать |

Рисунок Б.22 – Результат добавления пассажира


Система аэропорта
Добро пожаловать, user2@mail.ru
О программе
Регистрация пассажира
Купить билет на рейс
Корзина
Выйти из системы

Зарегистрировать пассажира

Введите фамилию пассажира...

Введите имя пассажира...

Введите отчество пассажира...

Добавить пассажира

Введите дату рождения пассажира...

Введите серию и номер паспорта...

Редактировать пассажира

Список зарегистрированных пассажиров:

| Фамилия  | Имя    | Отчество     | Дата рождения | Серия и номер паспорта |         |         |
|----------|--------|--------------|---------------|------------------------|---------|---------|
| Буренков | Сергей | Владимирович | 08.10.2002    | 11 11 111111           | Удалить | Выбрать |
| Лазарева | Лилия  | Владимировна | 05.01.2006    | 22 22 222222           | Удалить | Выбрать |

Рисунок Б.23 – Добавление второго пассажира


Система аэропорта
Добро пожаловать, user2@mail.ru
О программе
Регистрация пассажира
Купить билет на рейс
Корзина
Выйти из системы

Зарегистрировать пассажира

Введите фамилию пассажира...

Введите имя пассажира...

Введите отчество пассажира...

Добавить пассажира

Введите дату рождения пассажира...

Введите серию и номер паспорта...

Редактировать пассажира

Список зарегистрированных пассажиров:

| Фамилия  | Имя    | Отчество     | Дата рождения | Серия и номер паспорта |         |         |
|----------|--------|--------------|---------------|------------------------|---------|---------|
| Буренков | Сергей | Владимирович | 08.10.2002    | 11 11 111111           | Удалить | Выбрать |
| Лазарева | Лилия  | Владимировна | 05.01.2006    | 22 22 222222           | Удалить | Выбрать |

Рисунок Б.24 – Кнопка «Выбрать» рядом с пассажиром


Система аэропорта
Добро пожаловать, user2@mail.ru

О программе
Регистрация пассажира
Купить билет на рейс
Корзина
Выйти из системы

Зарегистрировать пассажира

Буренков
08.10.2002

Сergeй
11 11 111111

Владимирович

Добавить пассажира

Редактировать пассажира

Список зарегистрированных пассажиров:

| Фамилия  | Имя    | Отчество     | Дата рождения | Серия и номер паспорта |         |          |
|----------|--------|--------------|---------------|------------------------|---------|----------|
| Буренков | Сергей | Владимирович | 08.10.2002    | 11 11 111111           | Удалить | Отменить |
| Лазарева | Лилия  | Владимировна | 05.01.2006    | 22 22 222222           | Удалить | Выбрать  |

Последний: 2000

Рисунок Б.25 – Результат нажатия кнопки «Выбрать»


Система аэропорта
Добро пожаловать, user2@mail.ru

О программе
Регистрация пассажира
Купить билет на рейс
Корзина
Выйти из системы

Зарегистрировать пассажира

Буренков
08.10.2000

Сergeй
11 11 111111

Владимирович

Добавить пассажира

Редактировать пассажира

Список зарегистрированных пассажиров:

| Фамилия  | Имя    | Отчество     | Дата рождения | Серия и номер паспорта |         |          |
|----------|--------|--------------|---------------|------------------------|---------|----------|
| Буренков | Сергей | Владимирович | 08.10.2002    | 11 11 111111           | Удалить | Отменить |
| Лазарева | Лилия  | Владимировна | 05.01.2006    | 22 22 222222           | Удалить | Выбрать  |

Рисунок Б.26 – Редактирование данных


Система аэропорта Добро пожаловать, user2@mail.ru

О программе
Регистрация пассажира
Купить билет на рейс
Корзина
Выйти из системы

Зарегистрировать пассажира

Введите фамилию пассажира...

Введите имя пассажира...

Введите отчество пассажира...

Добавить пассажира

Введите дату рождения пассажира...

Введите серию и номер паспорта...

Редактировать пассажира

Список зарегистрированных пассажиров:

| Фамилия  | Имя    | Отчество     | Дата рождения | Серия и номер паспорта |         |         |
|----------|--------|--------------|---------------|------------------------|---------|---------|
| Лазарева | Лилия  | Владимировна | 05.01.2006    | 22 22 222222           | Удалить | Выбрать |
| Буренков | Сергей | Владимирович | 08.10.2000    | 11 11 111111           | Удалить | Выбрать |

Рисунок Б.27 – Результат редактирования данных


Система аэропорта Добро пожаловать, user2@mail.ru

О программе
Регистрация пассажира
Купить билет на рейс
Корзина
Выйти из системы

Зарегистрировать пассажира

Введите фамилию пассажира...

Введите имя пассажира...

Введите отчество пассажира...

Добавить пассажира

Введите дату рождения пассажира...

Введите серию и номер паспорта...

Редактировать пассажира

Список зарегистрированных пассажиров:

| Фамилия  | Имя    | Отчество     | Дата рождения | Серия и номер паспорта |         |         |
|----------|--------|--------------|---------------|------------------------|---------|---------|
| Лазарева | Лилия  | Владимировна | 05.01.2006    | 22 22 222222           | Удалить | Выбрать |
| Буренков | Сергей | Владимирович | 08.10.2000    | 11 11 111111           | Удалить | Выбрать |

Рисунок Б.28 – Кнопка «Удалить»

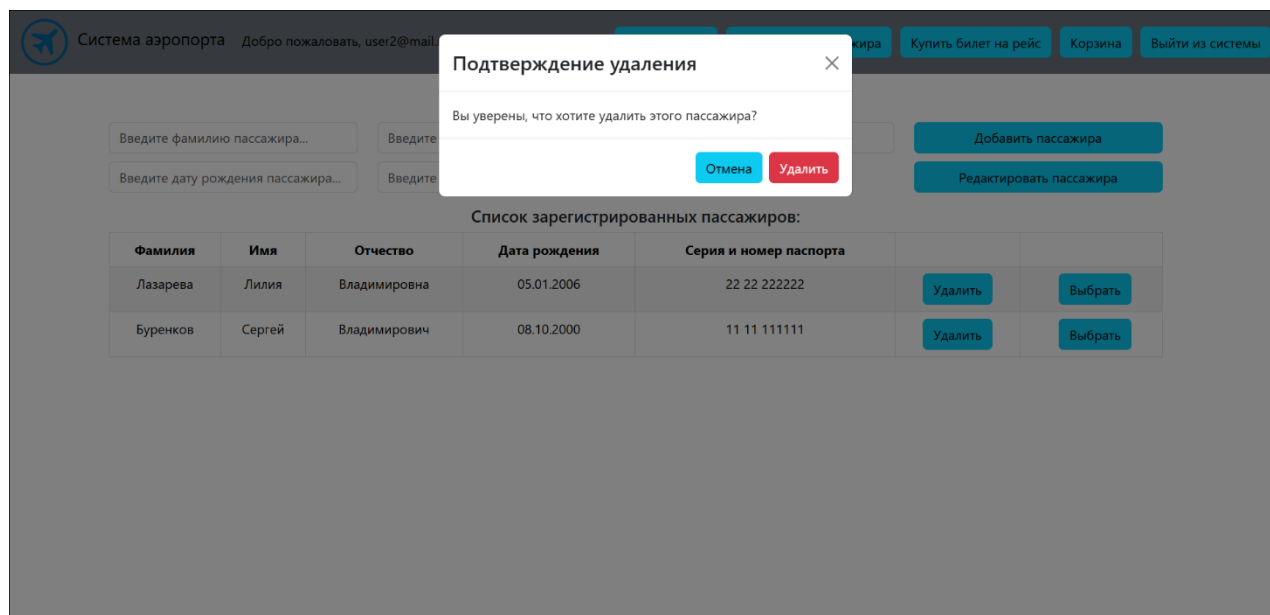


Рисунок Б.29 – Подтверждение удаления

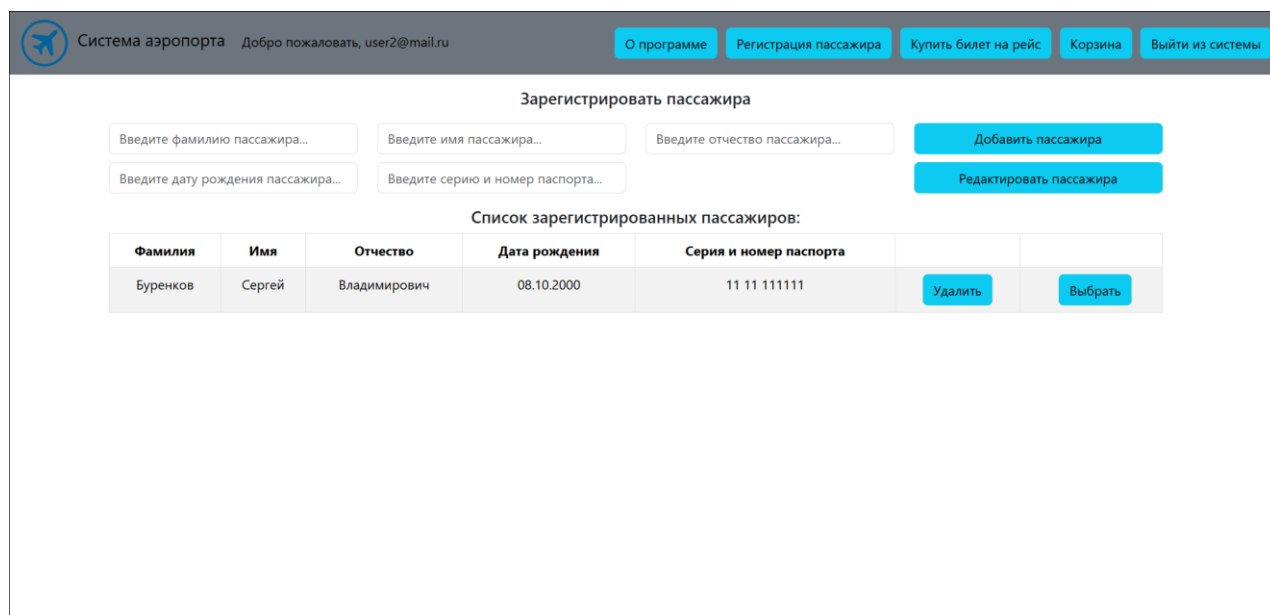


Рисунок Б.30 – Результат удаления

Система аэропорта
Добро пожаловать, user2@mail.ru

О программе
Регистрация пассажира
Купить билет на рейс
Корзина
Выйти из системы

Отсортировать рейсы по...

Маршрут полета
Дата вылета

Отсортировать
Отменить сортировку

Список доступных рейсов:

| Номер рейса | Маршрут полета           | Дата вылета | Время вылета/прибытия | Цена рейса в рублях |               |
|-------------|--------------------------|-------------|-----------------------|---------------------|---------------|
| 3           | Пенза - Москва           | 28.05.2025  | 16:00/18:00           | 8000                | Выбрать место |
| 5           | Пенза - Минеральные воды | 27.05.2025  | 23:00/28.05.2025 3:00 | 10000               | Выбрать место |
| 1           | Пенза - Ульяновск        | 30.05.2025  | 4:00/6:00             | 7000                | Выбрать место |
| 6           | Пенза - Минеральные воды | 27.05.2025  | 14:00/18:00           | 10000               | Выбрать место |

Рисунок Б.31 – Страница покупки рейсов

Система аэропорта
Добро пожаловать, user2@mail.ru

О программе
Регистрация пассажира
Купить билет на рейс
Корзина
Выйти из системы

Отсортировать рейсы по...

Маршрут полета
Дата вылета

Пенза - Минеральные воды
Пенза - Москва
Пенза - Санкт Петербург
Пенза - Ульяновск

Отменить сортировку
Список доступных рейсов:

| Номер рейса | Маршрут полета           | Дата вылета | Время вылета/прибытия | Цена рейса в рублях |               |
|-------------|--------------------------|-------------|-----------------------|---------------------|---------------|
| 3           | Пенза - Москва           |             | 16:00/18:00           | 8000                | Выбрать место |
| 5           | Пенза - Минеральные воды | 27.05.2025  | 23:00/28.05.2025 3:00 | 10000               | Выбрать место |
| 1           | Пенза - Ульяновск        | 30.05.2025  | 4:00/6:00             | 7000                | Выбрать место |
| 6           | Пенза - Минеральные воды | 27.05.2025  | 14:00/18:00           | 10000               | Выбрать место |

Рисунок Б.32 – Нажатие кнопки «Маршрут полета»



Система аэропорта

Добро пожаловать, user2@mail.ru

О программе

Регистрация пассажира

Купить билет на рейс

Корзина

Выйти из системы

Отсортировать рейсы по...

Пенза - Минеральные воды

Дата вылета

Отсортировать

Отменить сортировку

Список доступных рейсов:

| Номер рейса | Маршрут полета           | Дата вылета | Время вылета/прибытия | Цена рейса в рублях |               |
|-------------|--------------------------|-------------|-----------------------|---------------------|---------------|
| 3           | Пенза - Москва           | 28.05.2025  | 16:00/18:00           | 8000                | Выбрать место |
| 5           | Пенза - Минеральные воды | 27.05.2025  | 23:00/28.05.2025 3:00 | 10000               | Выбрать место |
| 1           | Пенза - Ульяновск        | 30.05.2025  | 4:00/6:00             | 7000                | Выбрать место |
| 6           | Пенза - Минеральные воды | 27.05.2025  | 14:00/18:00           | 10000               | Выбрать место |

Рисунок Б.33 – Выбор сортировки по маршруту полета «Пенза – Минеральные ВОДЫ»



Система аэропорта

Добро пожаловать, user2@mail.ru

О программе

Регистрация пассажира

Купить билет на рейс

Корзина

Выйти из системы

Отсортировать рейсы по...

Пенза - Минеральные воды

Дата вылета

Отсортировать

Отменить сортировку

Список доступных рейсов:

| Номер рейса | Маршрут полета           | Дата вылета | Время вылета/прибытия | Цена рейса в рублях |               |
|-------------|--------------------------|-------------|-----------------------|---------------------|---------------|
| 5           | Пенза - Минеральные воды | 27.05.2025  | 23:00/28.05.2025 3:00 | 10000               | Выбрать место |
| 6           | Пенза - Минеральные воды | 27.05.2025  | 14:00/18:00           | 10000               | Выбрать место |

Рисунок Б.34 – Результат сортировки по маршруту полета



Система аэропорта

Добро пожаловать, user2@mail.ru

О программе

Регистрация пассажира

Купить билет на рейс

Корзина

Выйти из системы

Отсортировать рейсы по...

Маршрут полета ▾

28/05/2025

Отсортировать

Отменить сортировку

Список доступных рейсов:

| Номер рейса | Маршрут полета           | Дата вылета | Время вылета/прибытия | Цена рейса в рублях |               |
|-------------|--------------------------|-------------|-----------------------|---------------------|---------------|
| 3           | Пенза - Москва           | 28.05.2025  | 16:00/18:00           | 8000                | Выбрать место |
| 5           | Пенза - Минеральные воды | 27.05.2025  | 23:00/28.05.2025 3:00 | 10000               | Выбрать место |
| 1           | Пенза - Ульяновск        | 30.05.2025  | 4:00/6:00             | 7000                | Выбрать место |
| 6           | Пенза - Минеральные воды | 27.05.2025  | 14:00/18:00           | 10000               | Выбрать место |

Рисунок Б.35 – Выбор сортировки по дате «28/05/2025»



Система аэропорта

Добро пожаловать, user2@mail.ru

О программе

Регистрация пассажира

Купить билет на рейс

Корзина

Выйти из системы

Отсортировать рейсы по...

Маршрут полета ▾

28/05/2025

Отсортировать

Отменить сортировку

Список доступных рейсов:

| Номер рейса | Маршрут полета | Дата вылета | Время вылета/прибытия | Цена рейса в рублях |               |
|-------------|----------------|-------------|-----------------------|---------------------|---------------|
| 3           | Пенза - Москва | 28.05.2025  | 16:00/18:00           | 8000                | Выбрать место |

Рисунок Б.36 – Результат сортировки по дате



Система аэропорта

Добро пожаловать, user2@mail.ru

О программе

Регистрация пассажира

Купить билет на рейс

Корзина

Выйти из системы

Отсортировать рейсы по...

Пенза - Минеральные воды

29/05/2025

Отсортировать

Отменить сортировку

Список доступных рейсов:

| Номер рейса | Маршрут полета           | Дата вылета | Время вылета/прибытия | Цена рейса в рублях |               |
|-------------|--------------------------|-------------|-----------------------|---------------------|---------------|
| 7           | Пенза - Минеральные воды | 29.05.2025  | 10:00/14:00           | 10000               | Выбрать место |

Рисунок Б.37 – Результат сортировки по маршруту полета «Пенза – Минеральные воды» и по дате «29/05/2025»



Система аэропорта

Добро пожаловать, user2@mail.ru

О программе

Регистрация пассажира

Купить билет на рейс

Корзина

Выйти из системы

Отсортировать рейсы по...

Маршрут полета

Дата вылета

Отсортировать

Отменить сортировку

Список доступных рейсов:

| Номер рейса | Маршрут полета           | Дата вылета | Время вылета/прибытия | Цена рейса в рублях |               |
|-------------|--------------------------|-------------|-----------------------|---------------------|---------------|
| 3           | Пенза - Москва           | 28.05.2025  | 16:00/18:00           | 8000                | Выбрать место |
| 5           | Пенза - Минеральные воды | 27.05.2025  | 23:00/28.05.2025 3:00 | 10000               | Выбрать место |
| 1           | Пенза - Ульяновск        | 30.05.2025  | 4:00/6:00             | 7000                | Выбрать место |
| 6           | Пенза - Минеральные воды | 27.05.2025  | 14:00/18:00           | 10000               | Выбрать место |
| 7           | Пенза - Минеральные воды | 29.05.2025  | 10:00/14:00           | 10000               | Выбрать место |

Рисунок Б.38 – Выбор кнопки «Выбрать место» рядом с рейсом

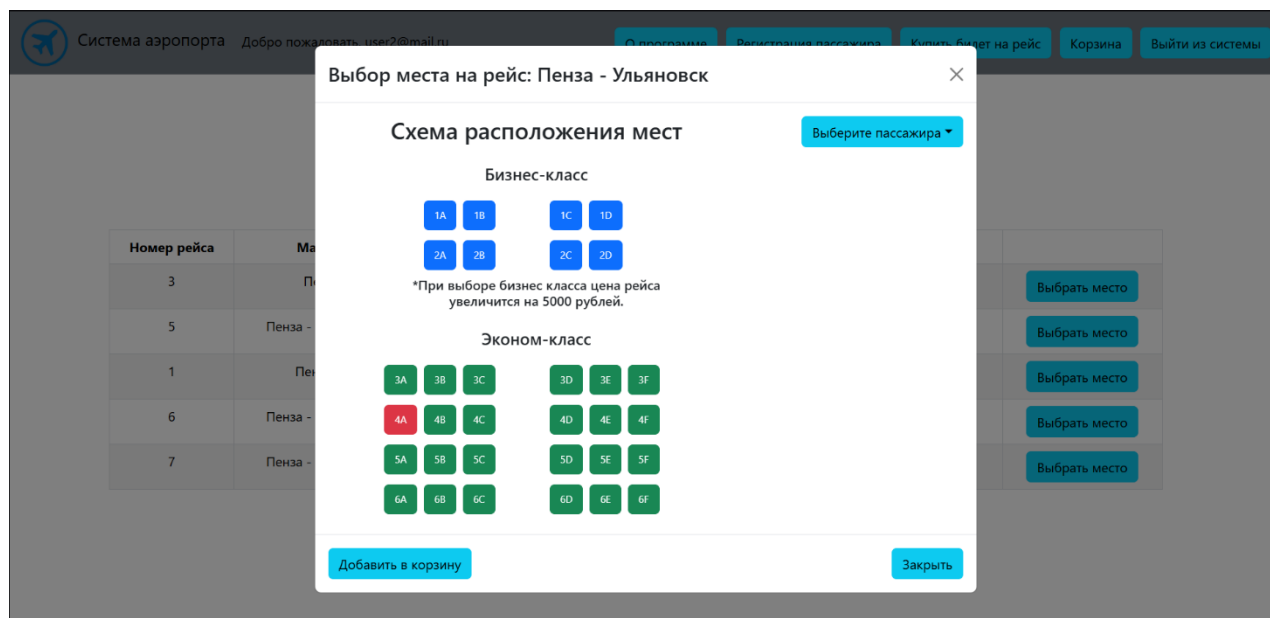


Рисунок Б.39 – Модальное окно схемы расположения мест

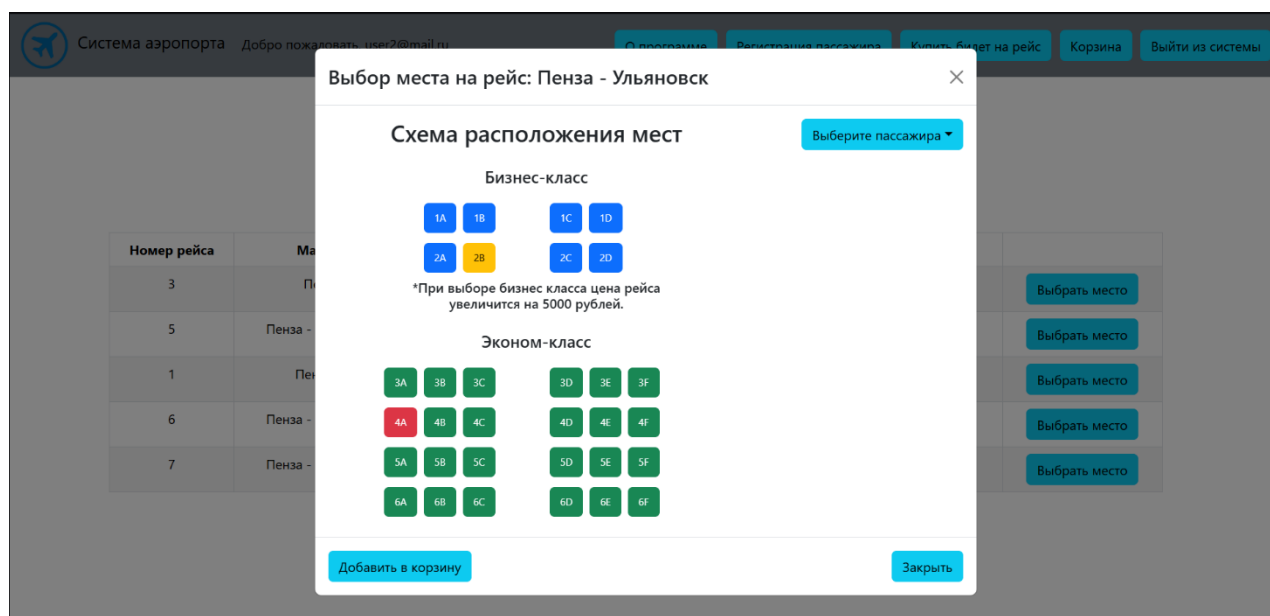


Рисунок Б.40 – Выбор места «2В»

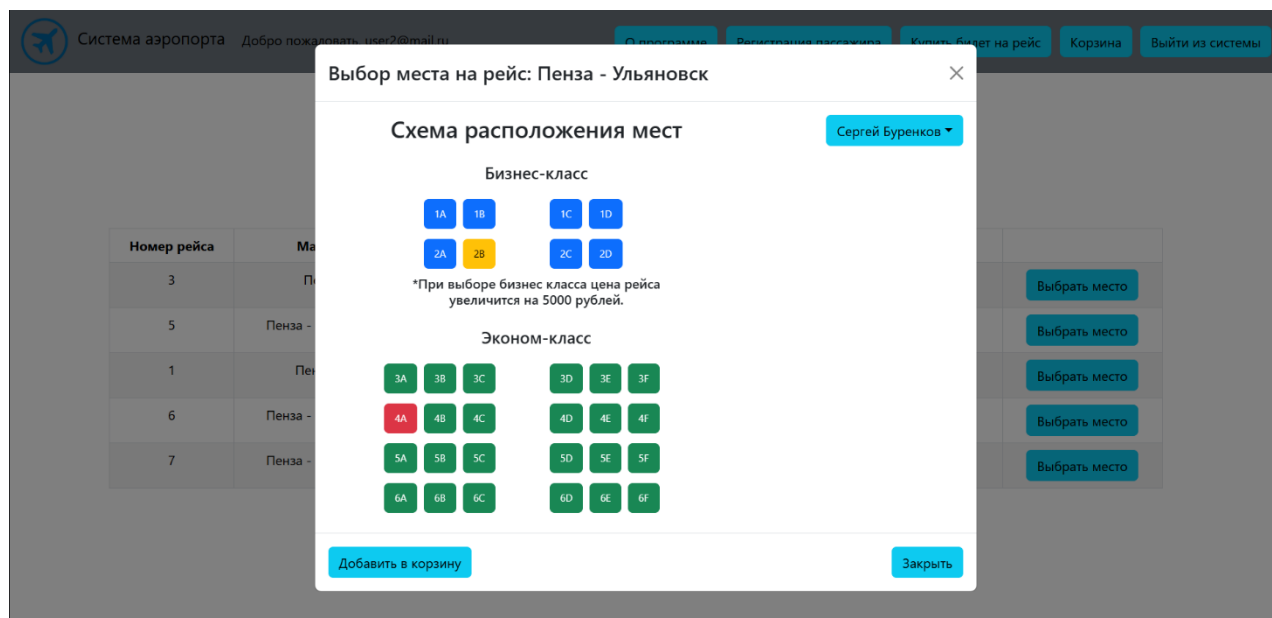


Рисунок Б.41 – Выбор зарегистрированного пассажира

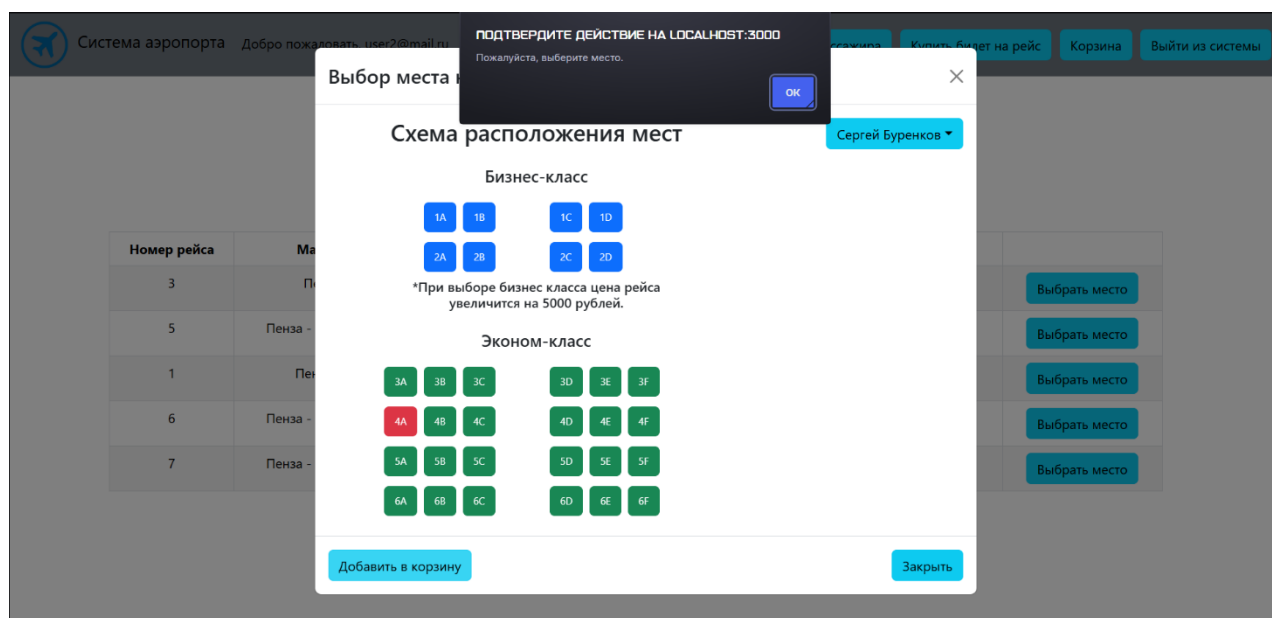


Рисунок Б.42 – Предупреждение о том, что место не выбрано

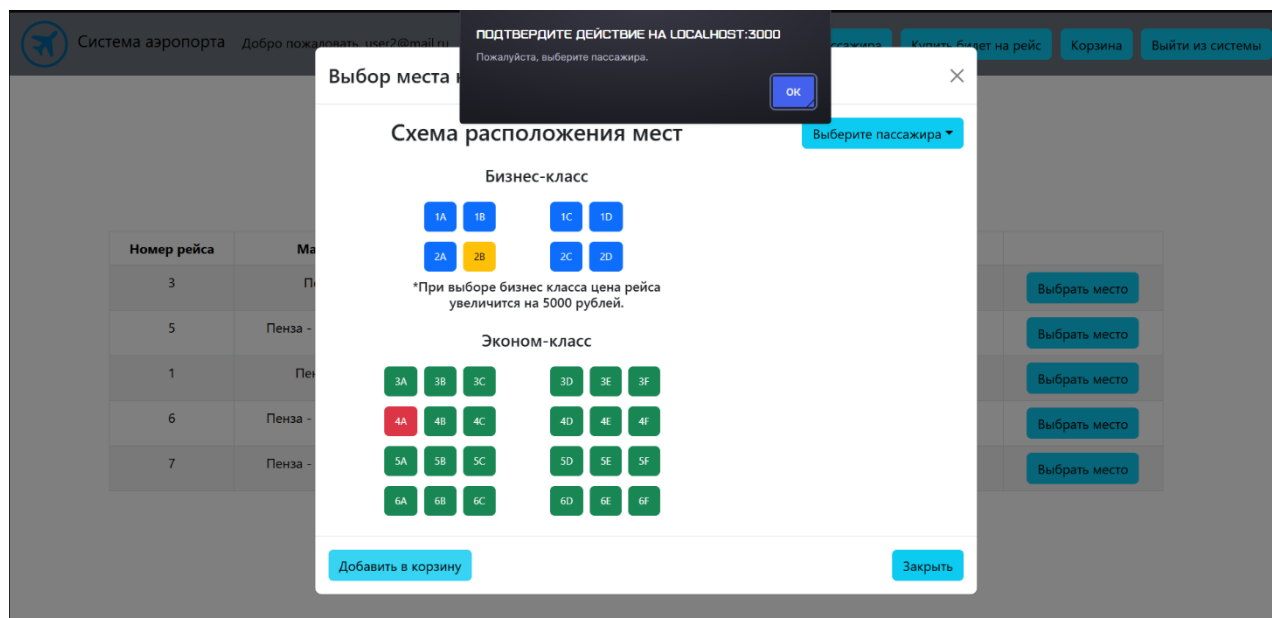


Рисунок Б.43 – Предупреждение о том, что пассажир не выбран

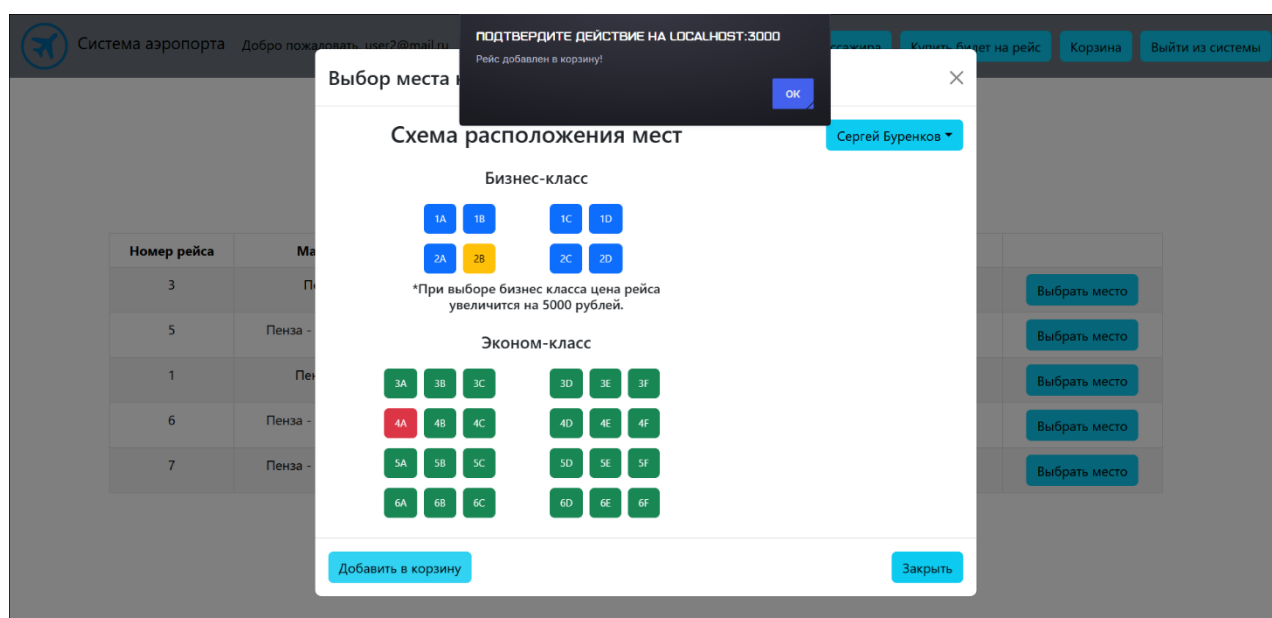


Рисунок Б.44 – Сообщение о том, что рейс добавлен в корзину

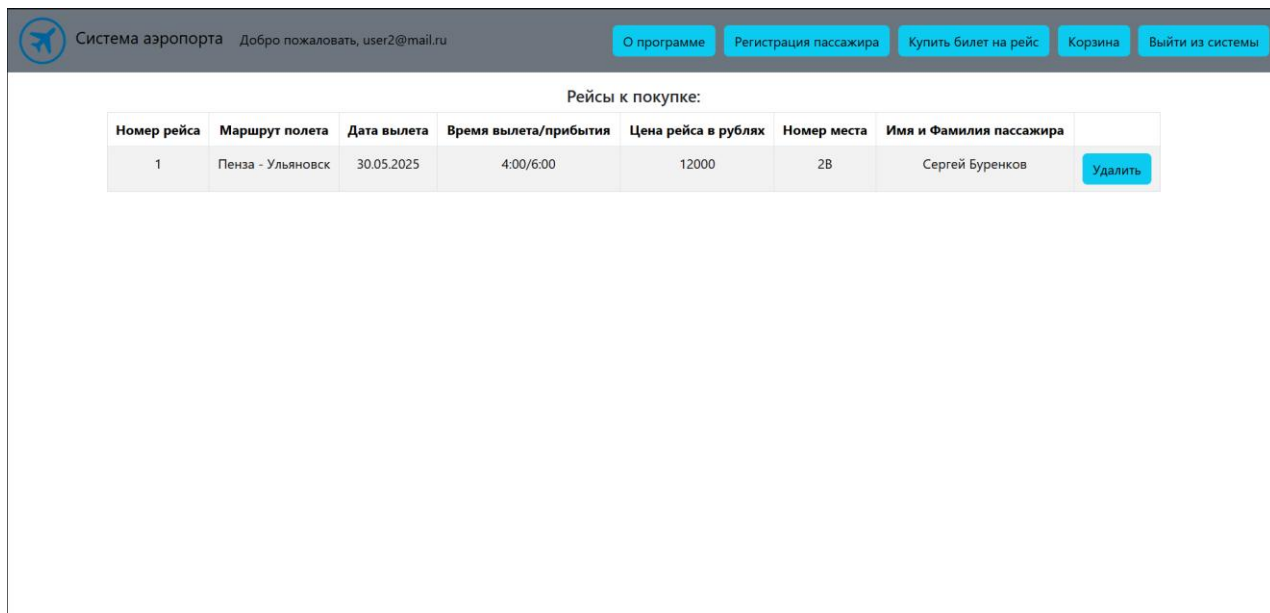


Рисунок Б.45 – Страница корзины

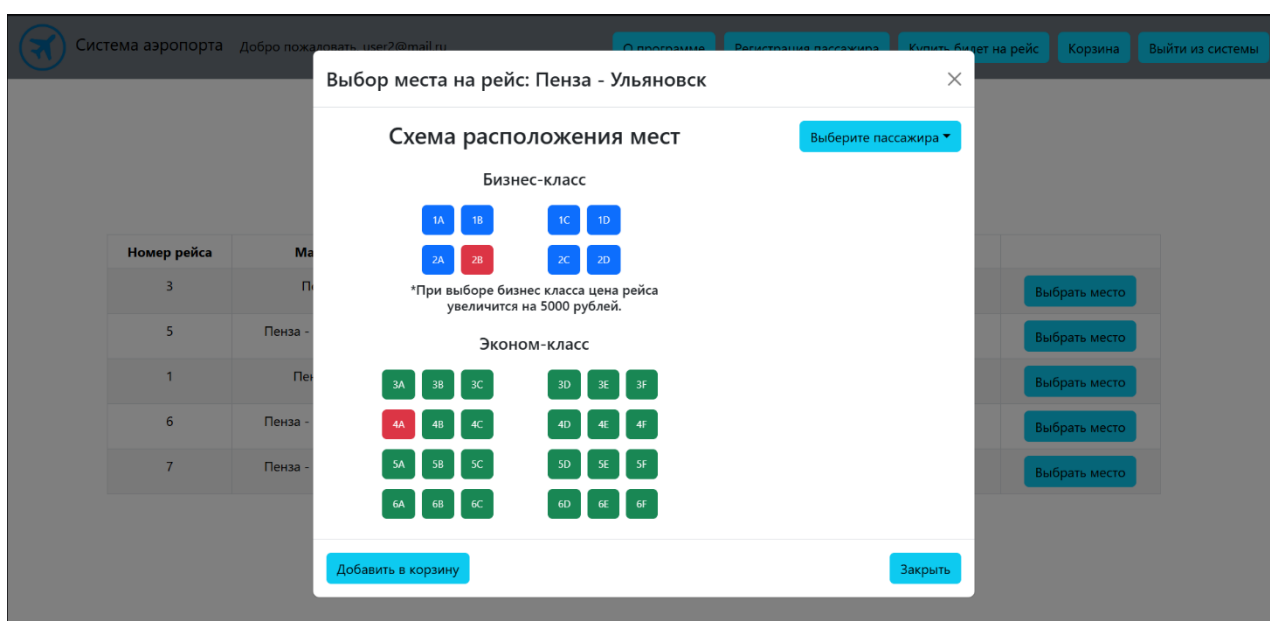


Рисунок Б.46 – Отображение, что место «2В» теперь занято

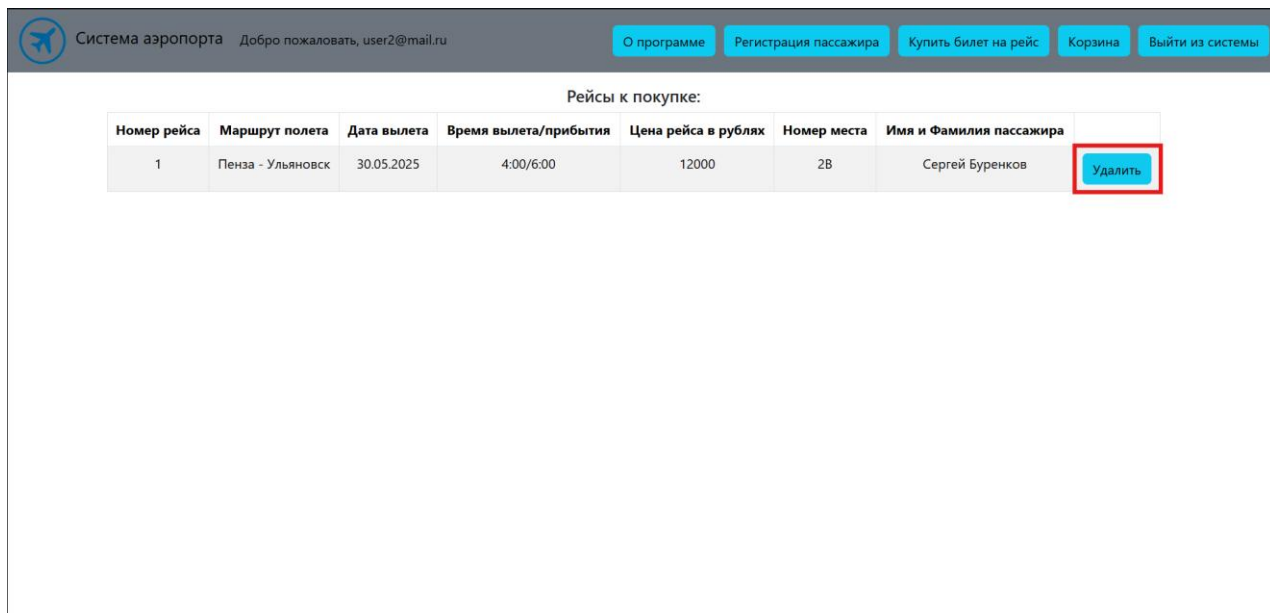


Рисунок Б.47 – Нажатие кнопки «Удалить»

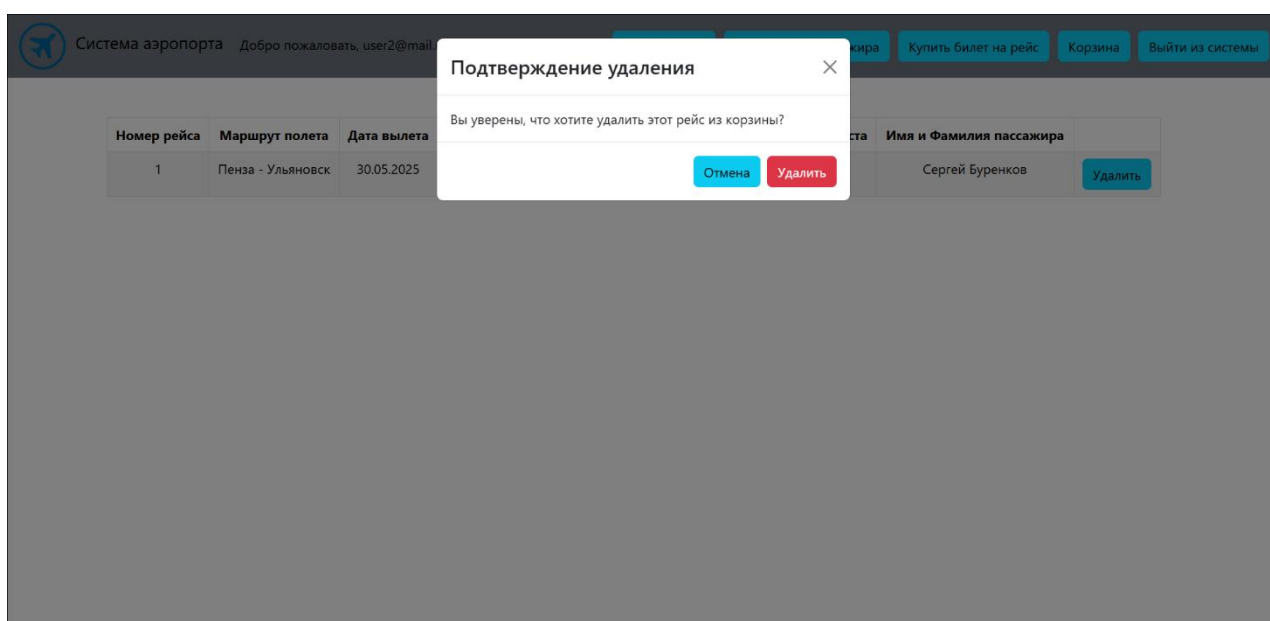


Рисунок Б.48 – Подтверждение удаления


Система аэропорта
Добро пожаловать, user2@mail.ru

О программе
Регистрация пассажира
Купить билет на рейс
Корзина
Выйти из системы

Рейсы к покупке:

| Номер рейса | Маршрут полета | Дата вылета | Время вылета/прибытия | Цена рейса в рублях | Номер места | Имя и Фамилия пассажира |
|-------------|----------------|-------------|-----------------------|---------------------|-------------|-------------------------|
|             |                |             |                       |                     |             |                         |

Рисунок Б.49 – Результат удаления


Система аэропорта

ПОДТВЕРДИТЕ ДЕЙСТВИЕ НА LOCALHOST:3000  
Пользователь не найден

ОК

О программе
Вход в систему

Авторизация

ser@mail.ru

.....

Показать пароль?
Забыли пароль?

Нет аккаунта?
Зарегистрироваться...

Войти

Рисунок Б.50 – Предупреждение «Пользователь не найден»

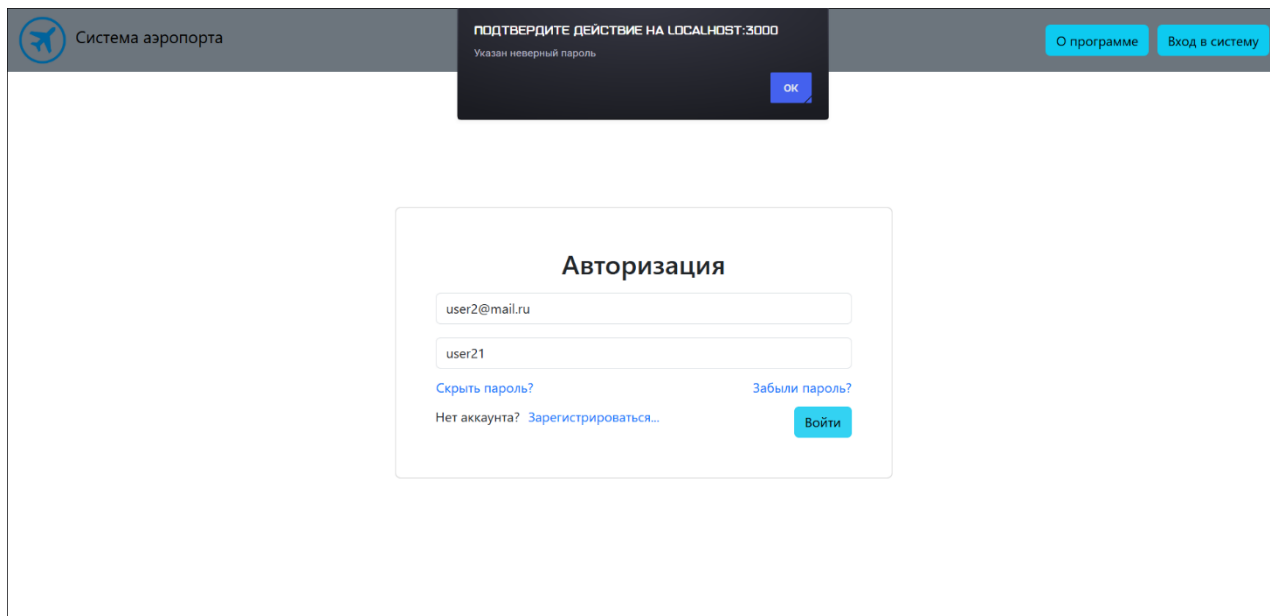


Рисунок Б.51 – Предупреждение «Указан неверный пароль»

## Приложение В

(обязательное)

Система регистрации пассажиров аэропорта

Листинг программы

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 96   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

### Файл «*index.js*»

```
require('dotenv').config()
const express = require('express')
const sequelize = require('./db')
const cors = require('cors')
const router = require('./routes/index')
const errorHandler = require('./middleware/ErrorHandlerMiddleware')
const PORT = process.env.PORT || 5000
const app = express()
app.use(cors())
app.use(express.json())
app.use('/api', router)
app.use(errorHandler)
const start = async () => {
  try {
    await sequelize.authenticate()
    await sequelize.sync()
    app.listen(PORT, () => console.log(`Server started on port
    ${PORT}`))
  } catch (e) {
    console.log(e)
  }
}
start()
```

### Файл «*db.js*»

```
const {Sequelize} = require('sequelize')
module.exports = new Sequelize(
  process.env.DB_NAME,
  process.env.DB_USER,
  process.env.DB_PASSWORD,
  {
    dialect: 'postgres',
    host: process.env.DB_HOST,
    port: process.env.DB_PORT
```

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 97   |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

```

    }
)

```

### Файл «models.js»

```

const sequelize = require('../db');
const { DataTypes } = require('sequelize');
const User = sequelize.define('user', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement:
true },
  email: { type: DataTypes.STRING, unique: true, allowNull: false
},
  password: { type: DataTypes.STRING, allowNull: false },
  role: { type: DataTypes.STRING },
});
const Basket = sequelize.define('basket', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement:
true },
});
const Flight = sequelize.define('flight', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement:
true },
  flight_number: { type: DataTypes.INTEGER },
  flight_route: { type: DataTypes.STRING },
  flight_data: { type: DataTypes.DATE },
  flight_time: { type: DataTypes.STRING },
  price: { type: DataTypes.INTEGER },
});
const Seat = sequelize.define('seat', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement:
true },
  seat_number: { type: DataTypes.STRING },
  seat_status: { type: DataTypes.BOOLEAN, defaultValue: false },
});
const Passenger = sequelize.define('passenger', {

```

|      |      |          |         |      |                        |      |
|------|------|----------|---------|------|------------------------|------|
|      |      |          |         |      | ИГУ 1.090501.03.001 ПЗ | Лист |
|      |      |          |         |      |                        | 98   |
| Изм. | Лист | № докум. | Подпись | Дата |                        |      |

```

    id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement:
true },
    name: { type: DataTypes.STRING },
    otchestvo: { type: DataTypes.STRING },
    surname: { type: DataTypes.STRING },
    year_of_birth: { type: DataTypes.STRING },
    passport: { type: DataTypes.STRING },
  });
User.hasOne(Basket);
Basket.belongsTo(User);
User.hasMany(Passenger);
Passenger.belongsTo(User);
Flight.hasMany(Basket);
Basket.belongsTo(Flight);
Seat.hasMany(Basket);
Basket.belongsTo(Seat);
Passenger.hasMany(Basket);
Basket.belongsTo(Passenger);
Flight.hasMany(Seat);
Seat.belongsTo(Flight);
module.exports = {
  User,
  Basket,
  Flight,
  Seat,
  Passenger,
};

```

#### Файл «*routes/index.js*»

```

const Router = require('express')
const router = new Router()
const userRouter = require('./userRouter')
const flightRouter = require('./flightRouter')
const passangerRouter = require('./passangerRouter')
const basketRouter = require('./basketRouter')

```

|      |      |          |         |      |                                                                  |      |
|------|------|----------|---------|------|------------------------------------------------------------------|------|
|      |      |          |         |      | <p style="text-align: center;"><i>ПГУ 1.090501.03.001 ПЗ</i></p> | Лист |
|      |      |          |         |      |                                                                  |      |
| Изм. | Лист | № докум. | Подпись | Дата |                                                                  | 99   |

```

const seatRouter = require("../seatRouter")
router.use('/user', userRouter)
router.use('/flight', flightRouter)
router.use('/passanger', passangerRouter)
router.use('/basket', basketRouter)
router.use('/seat', seatRouter)
module.exports = router

```

### Файл «*PassangerController.js*»

```

const ApiError = require("../error/ApiError")
const {Passenger} = require('../models/models')
class PassangerController {
  async create(req, next){
    try{
      const {name, otchestvo, surname, year_of_birth,
passport, userId} = req.body
      const passanger = await Passenger.create({name,
otchestvo, surname, year_of_birth, passport, userId})
    }catch (e) {
      next(ApiError.badRequest(e.message))
    }
  }
  async getAll( req, res, next) {
    try {
      const passangers = await Passenger.findAll();
      return res.json(passangers);
    } catch (e) {
      next(ApiError.internal(e.message));
    }
  }
  async getOne(req, res){
    const {id} = req.params
    const passanger = await Passenger.findOne(
      {
        where: {id}

```

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 100  |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

```

        }
    )
    return res.json(passanger)
}
async delete(req) {
    const { id } = req.params;
    const passanger = await Passenger.destroy(
        {
            where: { id }
        }
    )
}
async update(req, res, next) {
    try {
        const { id } = req.params;
        const { name, otchestvo, surname, year_of_birth,
passport } = req.body;
        const passanger = await Passenger.update(
            { name, otchestvo, surname, year_of_birth, passport
},
            { where: { id } }
        );
        return res.json({passanger});
    } catch (e) {
        next(ApiError.badRequest(e.message));
    }
}
}
module.exports = new PassangerController()

```

### Файл «*FlightController.js*»

```

const ApiError = require("../error/ApiError")
const {Flight, Seat} = require('../models/models')
const { Op } = require('sequelize');
class FlightController {

```

|      |      |          |         |      |                                                                  |      |
|------|------|----------|---------|------|------------------------------------------------------------------|------|
|      |      |          |         |      | <p style="text-align: center;"><i>ПГУ 1.090501.03.001 ПЗ</i></p> | Лист |
|      |      |          |         |      |                                                                  | 101  |
| Изм. | Лист | № докум. | Подпись | Дата |                                                                  |      |

```

    async create(req, next){
        try{
            const {flight_number, flight_route, flight_data,
flight_time, price} = req.body
            const flight = await Flight.create({flight_number,
flight_route, flight_data, flight_time, price})
            const businessSeats = [
                { seat_number: '1A', flightId: flight.id },
                { seat_number: '1B', flightId: flight.id },
                { seat_number: '1C', flightId: flight.id },
                { seat_number: '1D', flightId: flight.id },
                { seat_number: '2A', flightId: flight.id },
                { seat_number: '2B', flightId: flight.id },
                { seat_number: '2C', flightId: flight.id },
                { seat_number: '2D', flightId: flight.id },
            ];
            const economySeats = [
                { seat_number: '3A', flightId: flight.id },
                { seat_number: '3B', flightId: flight.id },
                { seat_number: '3C', flightId: flight.id },
                { seat_number: '3D', flightId: flight.id },
                { seat_number: '3E', flightId: flight.id },
                { seat_number: '3F', flightId: flight.id },
                { seat_number: '4A', flightId: flight.id },
                { seat_number: '4B', flightId: flight.id },
                { seat_number: '4C', flightId: flight.id },
                { seat_number: '4D', flightId: flight.id },
                { seat_number: '4E', flightId: flight.id },
                { seat_number: '4F', flightId: flight.id },
                { seat_number: '5A', flightId: flight.id },
                { seat_number: '5B', flightId: flight.id },
                { seat_number: '5C', flightId: flight.id },
                { seat_number: '5D', flightId: flight.id },
                { seat_number: '5E', flightId: flight.id },
                { seat_number: '5F', flightId: flight.id },
            ];
        }
    }
}

```

```

        { seat_number: '6A', flightId: flight.id },
        { seat_number: '6B', flightId: flight.id },
        { seat_number: '6C', flightId: flight.id },
        { seat_number: '6D', flightId: flight.id },
        { seat_number: '6E', flightId: flight.id },
        { seat_number: '6F', flightId: flight.id },
    ];

    await Seat.bulkCreate([...businessSeats,
...economySeats]);

    }catch (e) {
        next(ApiError.badRequest(e.message))
    }
}

async getAll(req, res) {
    const { route, service, date } = req.query;
    const whereConditions = {};
    let flights;
    if (route) {
        whereConditions.flight_route = route;
    }
    if (date){
        whereConditions.flight_data = {
            [Op.gte]: new Date(date + 'T00:00:00Z'), // Начало
дня
            [Op.lt]: new Date(date + 'T23:59:59Z') // Конец дня
        }
    }

    flights = await Flight.findAll({ where: whereConditions });
    return res.json(flights);
}

async getOne(req, res){
    const {id} = req.params
    const flight = await Flight.findOne(
        {
            where: {id}

```

```

        }

    )

    return res.json(flight)
}

async delete(req, res, next) {
    const { id } = req.params;
    try {
        await Seat.destroy({
            where: { flightId: id }
        });
        const flight = await Flight.destroy({
            where: { id }
        });
        return res.json({ message: "Рейс успешно удалён." });
    } catch (e) {
        next(ApiError.badRequest(e.message));
    }
}

async update(req, res, next) {
    try {
        const { id } = req.params;
        const { flight_number, flight_route, flight_data,
flight_time, price } = req.body;
        const flight = await Flight.update(
            { flight_number, flight_route, flight_data,
flight_time, price },
            { where: { id } }
        );
        return res.json({flight});
    } catch (e) {
        next(ApiError.badRequest(e.message));
    }
}

}

module.exports = new FlightController()

```

### Файл «Admin.js»

```
import { observer } from "mobx-react-lite";
import React, { useState, useEffect, useContext } from "react";
import { Table, Container, Row, Col, FormControl, Button,
DropdownButton, Dropdown, Modal } from "react-bootstrap";
import { createFlight, updateFlight, fetchFlights, fetchOneDelete }
from "../http/flightAPI";
import DatePicker from "react-datepicker";
import "react-datepicker/dist/react-datepicker.css";
import { Context } from "../index";
import moment from 'moment';

const Admin = observer( () => {
  const { flight } = useContext(Context);
  const [number, setNumber] = useState('')
  const [route, setRoute] = useState('')
  const [date, setDate] = useState(null);
  const [time, setTime] = useState('')
  const [price, setPrice] = useState('')
  const [selectedFlightId, setSelectedFlightId] = useState(null);
  const [isFlightSelected, setIsFlightSelected] = useState(false);
  const [showModal, setShowModal] = useState(false);
  const [flightToDelete, setFlightToDelete] = useState(null);
  useEffect(() => {
    const loadFlights = async () => {
      const flights = await fetchFlights();
      flight.setFlights(flights.slice());
    };
    loadFlights();
  }, []);
  const handleDelete = (id) => {
    setFlightToDelete(id);
    setShowModal(true);
  };
  const confirmDelete = async () => {
```

```

        await fetchOneDelete(flightToDelete);
        flight.setFlights(flight.flights.filter(flight => flight.id
!== flightToDelete));
        setShowModal(false);
    };
    const addFlight = async () => {
        try {
            await createFlight({
                flight_number: number,
                flight_route: route,
                flight_data: date,
                flight_time: time,
                price: price
            });
            resetFields();
            const flights = await fetchFlights();
            flight.setFlights(flights.slice());
        } catch (error) {
            console.error("Ошибка при добавлении рейса:", error);
        }
    }
    const updateFlightOne = async () => {
        try {
            await updateFlight(selectedFlightId, {
                flight_number: number,
                flight_route: route,
                flight_data: date,
                flight_time: time,
                price: price
            });
            resetFields();
            const flights = await fetchFlights();
            flight.setFlights(flights.slice());
        } catch (error) {
            console.error("Ошибка при обновлении рейса:", error);
        }
    }

```

```

    }
  }

  const resetFields = () => {
    setNumber('');
    setRoute('');
    setDate(null);
    setTime('');
    setPrice('');
    setSelectedFlightId(null);
    setIsFlightSelected(false);
  };

  const handleSelectFlight = (flightItem) => {
    setNumber(flightItem.flight_number);
    setRoute(flightItem.flight_route);
    setDate(new Date(flightItem.flight_data));
    setTime(flightItem.flight_time);
    setPrice(flightItem.price);
    setSelectedFlightId(flightItem.id);
    setIsFlightSelected(true);
  };

  const handleCancelSelection = () => {
    resetFields();
  };

  return (
    <Container>
      <Row className = "w-100 justify-content-center">
        <Col className="text-center">
          <h5 className="mt-3">Административная панель для
работы с рейсами</h5>
        </Col>
      </Row>
      <Row className = "w-100" style={{ marginTop: "10px" }}>
        <Col>
          <FormControl
            value = {number}

```

```

        onChange={e => setNumber(e.target.value)}
        placeholder = "Введите номер рейса..."
    />
</Col>
<Col>
    <DropDownButton variant="info" title={route ||
"Выберите маршрут полета..."} onSelect={(eventKey) =>
setRoute(eventKey)} >
        <DropDown.Item eventKey="Пенза - Минеральные
воды">Пенза - Минеральные воды</DropDown.Item>
        <DropDown.Divider />
        <DropDown.Item eventKey="Пенза -
Москва">Пенза - Москва</DropDown.Item>
        <DropDown.Divider />
        <DropDown.Item eventKey="Пенза - Санкт
Петербург">Пенза - Санкт Петербург</DropDown.Item>
        <DropDown.Divider />
        <DropDown.Item eventKey="Пенза -
Ульяновск">Пенза - Ульяновск</DropDown.Item>
    </DropDownButton>
</Col>
<Col>
    <DatePicker
        selected={date}
        onChange={date => setDate(date)}
        placeholderText="Выберите дату вылета..."
        dateFormat="dd/MM/yyyy"
        isClearable
    />
</Col>
<Col>
    <Button variant="info" onClick={() =>
addFlight() && window.location.reload()} className="w-100 justify-
content-center">Добавить рейс</Button>
</Col>

```

|      |      |          |         |      |                                                                  |      |
|------|------|----------|---------|------|------------------------------------------------------------------|------|
|      |      |          |         |      | <p style="text-align: center;"><i>ПГУ 1.090501.03.001 ПЗ</i></p> | Лист |
|      |      |          |         |      |                                                                  | 108  |
| Изм. | Лист | № докум. | Подпись | Дата |                                                                  |      |

```

        </Row>
        <Row className = "w-100 justify-content-center" style =
{{marginTop: "10px"}}>
            <Col>
                <FormControl
                    value = {time}
                    onChange={e => setTime(e.target.value)}
                    placeholder = "Введите время
вылета/прибытия..."
                />
            </Col>
            <Col>
                <FormControl
                    value = {price}
                    onChange={e => setPrice((e.target.value))}
                    placeholder = "Введите цену рейса..."
                />
            </Col>
            <Col>
            </Col>
            <Col>
                <Button variant="info" onClick={updateFlightOne}
className="w-100 justify-content-center">Редактировать рейс</Button>
            </Col>
        </Row>
        <Row className="w-100 justify-content-center">
            <Col className="text-center">
                <div>
                    <h5 className='mt-3'>Список доступных
рейсов:</h5>
                    <Table striped bordered hover>
                        <thead>
                            <tr>
                                <th>Номер рейса</th>
                                <th>Маршрут полета</th>

```

```

        <th>Дата вылета</th>
        <th>Время вылета/прибытия</th>
        <th>Цена рейса в рублях</th>
        <th></th>
        <th></th>
    </tr>
</thead>
<tbody>
    {flight.flights.map((flight) => (
        <tr key={flight.id}>
            <td>{flight.flight_number}</td>
            <td>{flight.flight_route}</td>
            <td>{moment(flight.flight_data).format('DD.MM.YYYY')}</td>
            <td>{flight.flight_time}</td>
            <td>{flight.price}</td>
            <td>
                <Button variant="info"
onClick={() => handleDelete(flight.id) } > Удалить </Button>
            </td>
            <td>
                {!isFlightSelected ||
selectedFlightId !== flight.id ? (
                    <Button variant="info"
style={{ marginLeft: "10px" }} onClick={() =>
handleSelectFlight(flight)}>Выбрать </Button>
                ) : (
                    <Button
variant="outline-info" style={{ marginLeft: "10px" }}
onClick={handleCancelSelection}>Отменить </Button>
                )}
            </td>
        </tr>
    ))}
</tbody>

```

```

        </Table>
      </div>
    </Col>
  </Row>
  <Modal show={showModal} onHide={() =>
setShowModal(false)}>
    <Modal.Header closeButton>
      <Modal.Title>Подтверждение
удаления</Modal.Title>
    </Modal.Header>
    <Modal.Body>Вы уверены, что хотите удалить этот
рейс?</Modal.Body>
    <Modal.Footer>
      <Button variant="info" onClick={() =>
setShowModal(false)}>
        Отмена
      </Button>
      <Button variant="danger" onClick={() =>
confirmDelete () && window.location.reload()}>
        Удалить
      </Button>
    </Modal.Footer>
  </Modal>
</Container>
);
});
export default Admin;

```

#### Файл «*Flight.js*»

```

import React, { useState, useContext, useEffect } from "react";
import { Container, Row, Col, Button, DropdownButton, Dropdown,
Table, Modal } from "react-bootstrap";
import { observer } from "mobx-react-lite";
import DatePicker from "react-datepicker";
import "react-datepicker/dist/react-datepicker.css";
import { fetchFlights } from "../http/flightAPI";

```

```

import { Context } from "../index";
import moment from 'moment';
import AirplaneSeatMap from '../components/AirplaneSeatMap';
import {addToBasket} from "../http/basketApi"
import {fetchSeatsByFlight} from "../http/seatApi"
import { fetchPassangers } from "../http/passangerAPI";
const Flight = observer( () => {
    const { flight } = useContext(Context);
    const { user } = useContext(Context);
    const { passanger } = useContext(Context);
    const [route, setRoute] = useState('')
    const [date, setDate] = useState(null);
    const [showModal, setShowModal] = useState(false);
    const [selectedFlight, setSelectedFlight] = useState(null);
    const [selectedSeat, setSelectedSeat] = useState(null);
    const [seats, setSeats] = useState([]);
    const [passengers, setPassengers] = useState([]);
    const [selectedPassengerId, setSelectedPassengerId] =
useState(null);
    useEffect(() => {
        const loadAllFlights = async () => {
            try {
                const allFlights = await fetchFlights({});
                flight.setFlights(allFlights);
                flight.setSortedFlights(allFlights);
            } catch (error) {
                console.error('Ошибка при загрузке рейсов:', error);
                flight.setFlights([]);
                flight.setSortedFlights([]);
            }
        };
        loadAllFlights();
    }, [flight]);
    useEffect(() => {
        const loadPassengers = async () => {

```

|      |      |          |         |      |                                                                  |      |
|------|------|----------|---------|------|------------------------------------------------------------------|------|
|      |      |          |         |      | <p style="text-align: center;"><i>ПГУ 1.090501.03.001 ПЗ</i></p> | Лист |
|      |      |          |         |      |                                                                  | 112  |
| Изм. | Лист | № докум. | Подпись | Дата |                                                                  |      |

```

        try {
            const fetchedPassengers = await fetchPassangers();
            setPassengers(fetchedPassengers);
        } catch (error) {
            console.error('Ошибка при загрузке пассажиров:',
error);
        }
    };
    loadPassengers();
}, []);
const loadFlights = async () => {
    try {
        const params = {};
        if (route) params.route = route;
        if (date) params.date =
date.toISOString().split('T')[0];
        const fetchedFlights = await fetchFlights(params);
        flight.setSortedFlights(fetchedFlights);
    } catch (error) {
        console.error('Ошибка при загрузке рейсов:', error);
        flight.setSortedFlights([]);
    }
};
const resetFilters = () => {
    setRoute('');
    setDate(null);
    flight.setSortedFlights(flight.flights);
};
const handleSelectSeat = async (flight) => {
    setSelectedFlight(flight);
    try {
        const fetchedSeats = await
fetchSeatsByFlight(flight.id);
        setSeats(fetchedSeats);
        setSelectedSeat(null);
    }

```

```

        setShowModal(true);
    } catch (error) {
        console.error('Ошибка при загрузке мест:', error);
        alert('Не удалось загрузить места для рейса.');
```

setSeats([]);

setSelectedSeat(null);

setShowModal(true);

}

};

const handleCloseModal = () => {

setShowModal(false);

setSelectedFlight(null);

setSelectedSeat(null);

};

const handleAddToBasket = async () => {

if (!selectedSeat) {

alert('Пожалуйста, выберите место.');

return;

}

if (!selectedPassengerId) {

alert('Пожалуйста, выберите пассажира.');

return;

}

try {

const userId = user.user.id;

await addToBasket(userId, selectedFlight.id,

selectedSeat.id, selectedPassengerId);

alert('Рейс добавлен в корзину!');

handleCloseModal();

} catch (error) {

alert(error.message);

}

};

```

    const filteredPassangers = passanger.passangers.filter(p =>
p.userId === user.user.id);
    return (
      <Container>
        <Row className="w-100 justify-content-center">
          <Col className="text-center">
            <h5 className="mt-3">Отсортировать рейсы
по...</h5>
          </Col>
        </Row>
        <Row className="w-100 justify-content-center">
          <Col>
            </Col>
            <Col className="d-flex justify-content-end">
              <DropDownButton variant="info" title={route ||
"Маршрут полета"} onSelect={(eventKey) => setRoute(eventKey)} >
                <DropDown.Item eventKey="Пенза - Минеральные
воды">Пенза - Минеральные воды</DropDown.Item>
                <DropDown.Divider />
                <DropDown.Item eventKey="Пенза -
Москва">Пенза - Москва</DropDown.Item>
                <DropDown.Divider />
                <DropDown.Item eventKey="Пенза - Санкт
Петербург">Пенза - Санкт Петербург</DropDown.Item>
                <DropDown.Divider />
                <DropDown.Item eventKey="Пенза -
Ульяновск">Пенза - Ульяновск</DropDown.Item>
              </DropDownButton>
            </Col>
            <Col className="text-center">
              <DatePicker
                selected={date}
                onChange={setDate}
                placeholderText="Дата вылета"
                dateFormat="dd/MM/yyyy"

```

```

        isClearable
      />
    </Col>
    <Col>
  </Col>
</Row>
<Row>
  <Col className="text-center">
    <Button variant="outline-info" className="mt-3"
onClick={ () => {
      loadFlights();
    }}>Отсортировать
  </Button>
  <Button variant="outline-info" className="mt-3"
style = {{ marginLeft: "10px"}} onClick={resetFilters}>Отменить
сортровку</Button>
  </Col>
</Row>
<Row className="w-100 justify-content-center">
  <Col className="text-center">
    <div>
      <h5 className='mt-3'>Список доступных
рейсов:</h5>
      <Table striped bordered hover>
        <thead>
          <tr>
            <th>Номер рейса</th>
            <th>Маршрут полета</th>
            <th>Дата вылета</th>
            <th>Время вылета/прибытия</th>
            <th>Цена рейса в рублях</th>
            <th></th>
          </tr>
        </thead>

```

```

<tbody>
{flight.sortedFlights.map((flight) => (
<tr key={flight.id}>
<td>{flight.flight_number}</td>
<td>{flight.flight_route}</td>
<td>{moment(flight.flight_data).format('DD.MM.YYYY')}</td>
<td>{flight.flight_time}</td>
<td>{flight.price}</td>
<td>
<Button
variant="info" onClick={() => handleSelectSeat(flight)}>Выбрать
место</Button>
</td>
</tr>
))}
</tbody>
</Table>
</div>
</Col>
</Row>
<Modal show={showModal} onHide={handleCloseModal} size =
'lg' centered>
<Modal.Header closeButton>
<Modal.Title>Выбор места на рейс:
{selectedFlight ? selectedFlight.flight_route : ''}</Modal.Title>
</Modal.Header>
<Modal.Body>
<Row>
<Col xs={8}>
<h3 className="mb-4 text-center">Схема
расположения мест</h3>
<AirplaneSeatMap
seats={seats}
selectedSeat={selectedSeat}
onSelectSeat={setSelectedSeat}

```

```

        />
    </Col>
    <Col className="d-flex justify-content-end">
        <DropDownButton
            variant="info"
            title={selectedPassengerId ?
filteredPassangers.find(p => p.id === selectedPassengerId)?.name + '
' + passengers.find(p => p.id === selectedPassengerId)?.surname :
"Выберите пассажира"}
            onSelect={(eventKey) => {
setSelectedPassengerId(parseInt(eventKey));
            }}
        >
            {filteredPassangers.map(passenger =>
(
                <DropDown.Item
key={passenger.id} eventKey={passenger.id.toString()}>
                    {passenger.name + ' ' +
passenger.surname}
                </DropDown.Item>
            ) ) }
        </DropDownButton>
    </Col>
</Row>
</Modal.Body>
<Modal.Footer>
    <div style={{ display: 'flex', justifyContent:
'space-between', width: '100%' }}>
        <Button variant="info"
onClick={handleAddToBasket}>
            Добавить в корзину
        </Button>
        <Button variant="info"
onClick={handleCloseModal}>
            Закрыть
    </div>

```

|      |      |          |         |      |                                                                  |      |
|------|------|----------|---------|------|------------------------------------------------------------------|------|
|      |      |          |         |      | <p style="text-align: center;"><i>ПГУ 1.090501.03.001 ПЗ</i></p> | Лист |
|      |      |          |         |      |                                                                  | 118  |
| Изм. | Лист | № докум. | Подпись | Дата |                                                                  |      |

```

        </Button>
      </div>
    </Modal.Footer>
  </Modal>
</Container>

);
});
export default Flight;

```

### Файл «Basket.js»

```

import React, { useState, useEffect, useContext } from "react";
import { Table, Container, Row, Col, Button, Modal } from "react-
bootstrap";
import { observer } from "mobx-react-lite";
import { fetchBasket, fetchOneDelete } from "../http/basketApi";
import moment from "moment";
import { Context } from "../index";
const Basket = observer(() => {
  const { user } = useContext(Context);
  const [basketItems, setBasketItems] = useState([]);
  const [showModal, setShowModal] = useState(false);
  const [itemToDelete, setItemToDelete] = useState(null);
  useEffect(() => {
    const loadBasket = async () => {
      try {
        const basket = await fetchBasket();
        setBasketItems(basket);
      } catch (error) {
        console.error("Ошибка при получении корзины:",
error);
      }
    };
    loadBasket();
  }, []);
  const handleDelete = (id, seatId) => {

```

|      |      |          |         |      |                                                                  |      |
|------|------|----------|---------|------|------------------------------------------------------------------|------|
|      |      |          |         |      | <p style="text-align: center;"><i>ПГУ 1.090501.03.001 ПЗ</i></p> | Лист |
|      |      |          |         |      |                                                                  | 119  |
| Изм. | Лист | № докум. | Подпись | Дата |                                                                  |      |

```

        setItemToDelete({ id, seatId });
        setShowModal(true);
    };
    const confirmDelete = async () => {
        if (itemToDelete) {
            try {
                await fetchOneDelete(itemToDelete.id,
itemToDelete.seatId);
                setBasketItems(basketItems.filter(item => item.id
!== itemToDelete.id));
                setShowModal(false);
            } catch (error) {
                console.error("Ошибка при удалении рейса:", error);
            }
        }
    };
    const filteredBasket = basketItems.filter(p => p.userId ===
user.user.id);
    const premiumSeats = ['1A', '1B', '1C', '1D', '2A', '2B', '2C',
'2D'];
    return (
        <Container>
            <Row className="w-100 justify-content-center">
                <Col className="text-center">
                    <div>
                        <h5 className='mt-3'>Рейсы к покупке:</h5>
                        <Table striped bordered hover>
                            <thead>
                                <tr>
                                    <th>Номер рейса</th>
                                    <th>Маршрут полета</th>
                                    <th>Дата вылета</th>
                                    <th>Время вылета/прибытия</th>
                                    <th>Цена рейса в рублях</th>
                                    <th>Номер места</th>
                                </tr>
                            </thead>
                        </Table>
                    </div>
                </Col>
            </Row>
        </Container>
    );

```

```

        <th>Имя и Фамилия пассажира</th>
        <th></th>
    </tr>
</thead>
<tbody>
    {filteredBasket.map((item) => {
        const isPremiumSeat =
premiumSeats.includes(item.seat.seat_number);
        const price = isPremiumSeat ?
item.flight.price + 5000 : item.flight.price;
        return (
            <tr key={item.id}>
<td>{item.flight.flight_number}</td>
<td>{item.flight.flight_route}</td>
<td>{moment(item.flight.flight_data).format('DD.MM.YYYY')}</td>
<td>{item.flight.flight_time}</td>

                <td>{price}</td>
<td>{item.seat.seat_number}</td>
<td>{item.passenger.name} {item.passenger.surname}</td>
                <td>
                    <Button
variant="info" onClick={() => handleDelete(item.id,
item.seat.id)}>Удалить</Button>
                </td>
            </tr>
        )
    })}
</tbody>
</Table>
</div>
</Col>
</Row>
<Modal show={showModal} onHide={() =>
setShowModal(false)}>
    <Modal.Header closeButton>

```

```

        <Modal.Title>Подтверждение
удаления</Modal.Title>
    </Modal.Header>
    <Modal.Body>Вы уверены, что хотите удалить этот рейс
из корзины?</Modal.Body>
    <Modal.Footer>
        <Button variant="info" onClick={ () =>
setShowModal(false) }>
            Отмена
        </Button>
        <Button variant="danger" onClick={ () =>
confirmDelete () && window.location.reload() }>
            Удалить
        </Button>
    </Modal.Footer>
</Modal>
</Container>
);
});
export default Basket;

```

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 122  |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

Приложение Г

(обязательное)

Система регистрации пассажиров аэропорта

Презентация

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 123  |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

ДИПЛОМНЫЙ ПРОЕКТ  
на тему

# СИСТЕМА РЕГИСТРАЦИИ ПАССАЖИРОВ АЭРОПОРТА

Выполнил:  
ст. гр. 20ВОЭ1 Буренков С.В.  
Руководитель:  
к.т.н., доцент каф. ВТ Егоров В.Ю.

## Слайд Г.1 – Тема дипломного проекта

**Цель разработки** – увеличение скорости покупки авиабилетов, обеспечение контроля рейсов и мест на самолет, обеспечение контроля зарегистрированных пассажиров, обеспечение контроля рейсов пассажиров для покупки.

В результате внедрения данной системы будет повышена эффективность работы аэропорта, снижена лишняя нагрузка на сотрудника аэропорта, повышена скорость регистрации пассажиров.

**Функционал системы** - добавление, изменение и удаление рейсов на самолет; добавление, изменение и удаление пассажиров; автоматическое добавление свободных мест на самолет при создании рейса, мониторинг занятости мест, добавление рейсов в корзину и удаление рейсов из корзины.

## Слайд Г.2 – Цель разработки, функционал системы

|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 124  |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

## Аналоги разрабатываемой системы

♦ **Astra DCS** — это автоматизированная система управления пассажирскими авиаперелетами. Данная система имеет функционал для регистрации пассажиров, обработки данных о рейсах и управления билетами. Она поддерживает интеграцию с различными системами аэропортов и авиакомпаний. Также, данная система имеет мультиязычный интерфейс. Эта система используется в Московском аэропорте Домодедово.

♦ **Asteros Contact Air** — это система для автоматизации процессов в аэропортах. В её функционал входит регистрация пассажиров, управление рейсами и билетами, а также обработка багажа. Кроме того, она имеет весь необходимый функционал для мониторинга, учёта и анализа данных. Эта система используется в Московском аэропорте Шереметьево, где внедряется экспресс регистрация пассажиров.

Слайд Г.3 – Аналоги разрабатываемой системы

## Аналоги разрабатываемой системы

♦ **IS-DCS (Integrated System – Departure Control System)** — это автоматизированная система, которая необходима для управления процессами регистрации, вылета и посадки пассажиров. Она обеспечивает автоматизацию на всех этапах, начиная от регистрации пассажира и заканчивая посадкой на рейс. Также данная система отвечает за печать посадочных талонов. Эта система используется в Московском аэропорте Шереметьево.

Слайд Г.4 – Аналоги разрабатываемой системы

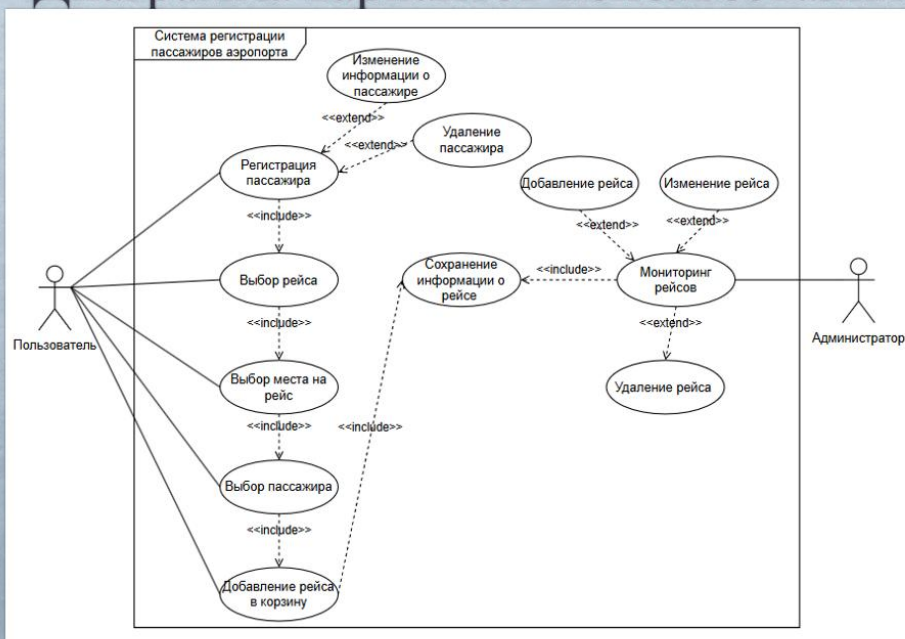
|      |      |          |         |      |                               |      |
|------|------|----------|---------|------|-------------------------------|------|
|      |      |          |         |      | <i>ПГУ 1.090501.03.001 ПЗ</i> | Лист |
|      |      |          |         |      |                               | 125  |
| Изм. | Лист | № докум. | Подпись | Дата |                               |      |

## Анализ целевой аудитории

- ♦ Разработка системы регистрации пассажиров аэропорта является актуальной для малых аэропортов, находящихся в не больших городах или в труднодоступных населенных пунктах. Например, в Чукотском автономном округе есть несколько аэропортов, в том числе в городах с небольшим населением: Анадырь (Угольный), Певек, Кеппервеем, Бухта Провидения.
- ♦ Делая вывод, основной целевой группой являются сотрудники и пассажиры аэропортов, расположенных в малых городах.

Слайд Г.5 – Анализ целевой аудитории

## Диаграмма вариантов использования



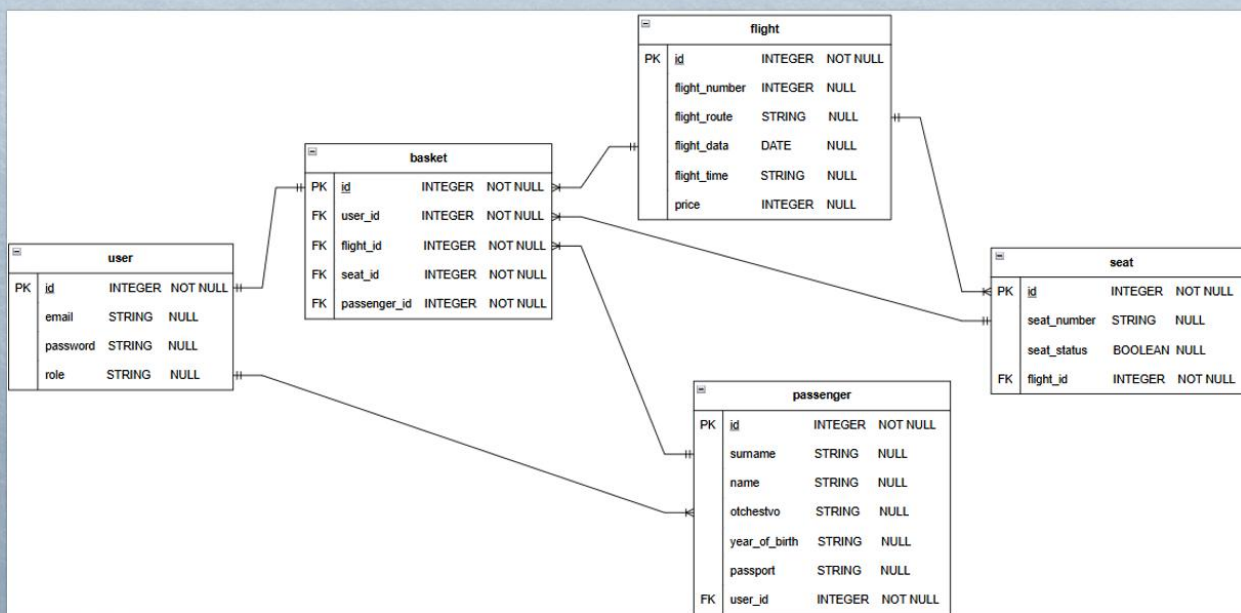
Слайд Г.6 – Диаграмма вариантов использования

## Выбранные средства разработки

На основании анализа требований к функциональным характеристикам клиент – серверного WEB приложения системы регистрации пассажиров аэропорта были выбраны наиболее актуальные средства разработки. Для написания кода используется программа Visual Studio Code. Используемый язык программирования – JavaScript. Для разработки серверной части (backend) используется следующий стек технологий – Express, Node JS, PostgreSQL, Sequelize. Для разработки клиентской части (frontend) – React JS, Axios, React-router-dom, Mob-X, React Bootstrap. Для проектирования базы данных используется редактор – Diagrams.net.

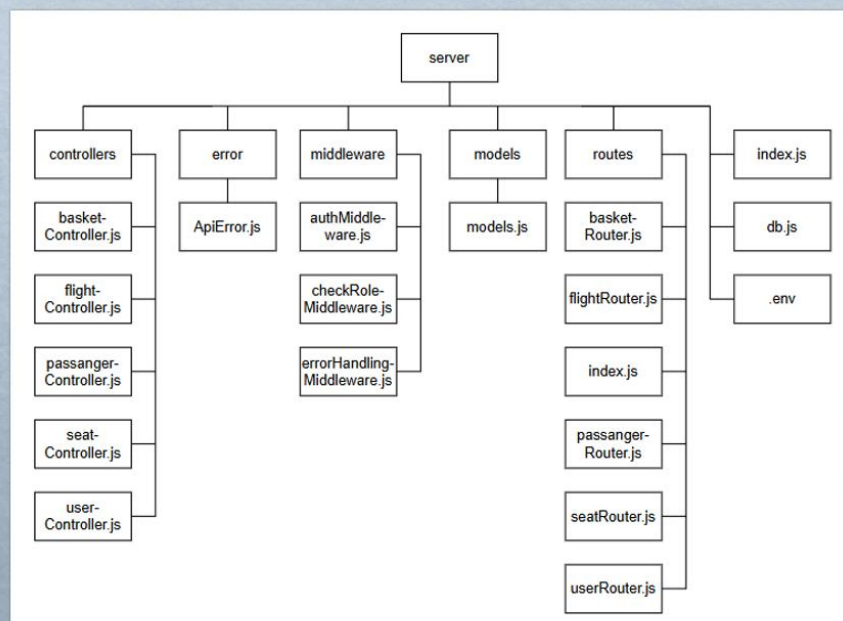
Слайд Г.7 – Выбранные средства разработки

## Физическая модель базы данных



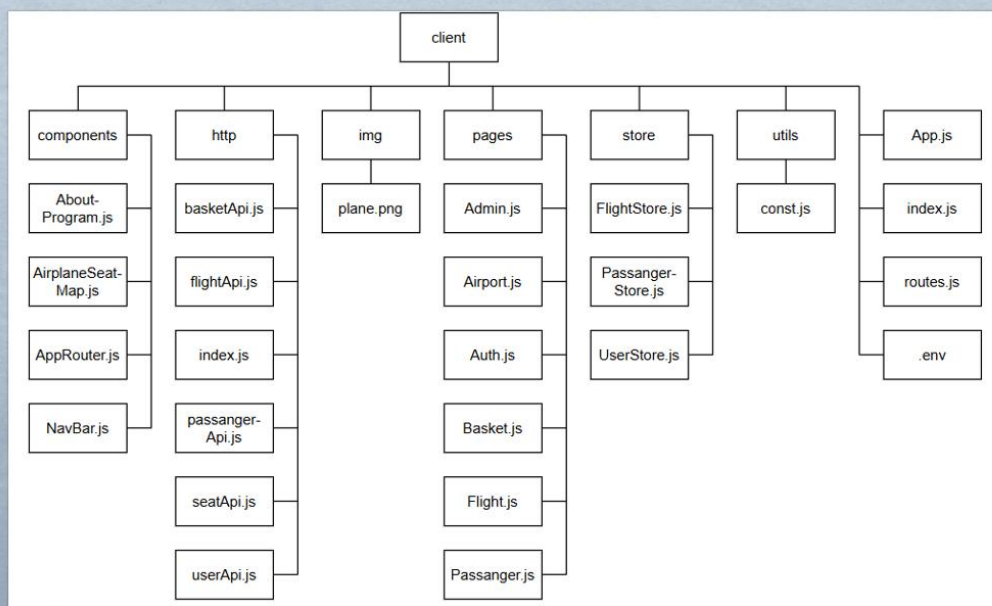
Слайд Г.8 – Физическая модель базы данных

## Структура серверной части приложения



Слайд Г.9 – Структура серверной части приложения

## Структура клиентской части приложения



Слайд Г.10 – Структура клиентской части приложения