##### Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Дисциплина: Базы и банки данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**СИСТЕМА ПОКУПКИ БИЛЕТОВ НА АВИАРЕЙСЫ**

|  |  |
| --- | --- |
| Студент гр. 020605: | Сурма Е.В. |
| Руководитель: | Трофимович А.Ф. |

Минск 2023

РЕФЕРАТ

СИСТЕМА ПОКУПКИ БИЛЕТОВ НА АВИАРЕЙСЫ: курсовой проект / Е. В. Сурма – Минск: БГУИР, 2023, – п.з. – 50 с., чертежей (плакатов) – 1 л. формата А3.

Курсовой проект на тему "Система бронирования авиабилетов" разработан с целью создания веб-приложения, обеспечивающего удобную процедуру покупки билетов на авиарейсы. В пояснительной записке к проекту содержится введение, четыре раздела, заключение, список использованных источников и приложение, включающее примеры кода отдельных классов.

Для разработки данного веб-приложения был выбран язык программирования *Python*. Проект представляет собой автоматизированную систему, предоставляющую функциональность по покупке и бронированию авиабилетов на различные рейсы. Внутри проекта представлены различные диаграммы, которые иллюстрируют различные аспекты системы.

В результате выполнения курсовой работы была спроектирована и реализована база данных для веб-приложения, а также разработано само веб-приложение. Также в пояснительной записке представлены схемы базы данных и описаны различные алгоритмы, используемые в проекте.

Полученное приложение, разработанное в рамках курсового проекта, может быть использовано любым пользователем с доступом в Интернет. Оно включает множество схем и диаграмм, которые могут быть использованы для анализа работы приложения при необходимости. Данный проект служит примером для создания хорошей архитектуры веб-приложений, которая может быть применена в коммерческой разработке в настоящее время.

Учреждение образования

«Белорусский государственный университет информатики   
и радиоэлектроники»

Факультет информационных технологий и управления

УТВЕРЖДАЮ

Заведующий кафедрой

*––––––––––––––––––––––––*

(подпись)

––––––– –––––––––––2023г.

ЗАДАНИЕ

по курсовому проектированию

Студенту    *Сурме Екатерине Владимировне–––––––––––––––––––*

1. Тема проекта   *Система покупки билетов на авиарейсы.*

2. Срок сдачи студентом законченного проекта– –*4 мая 2023 г.–*

3. Исходные данные к проекту *Создание базы данных для хранения информации о пассажирах, авиарейсах, авиалиниях, бронировании билетов, разработка интерфейса для ввода информации о пассажирах и поиска доступных авиарейсов, реализация механизма бронирования билетов на выбранные авиарейсы, разработка функционала для проведения оплаты бронирования билетов, разработка интерфейса для просмотра и управления заказами пассажиров, реализация функционала для выдачи электронных билетов, разработка механизма управления данными о пассажирах, авиарейсах, авиалиниях, бронировании билетов и оплатах, реализация механизма автоматического определения стоимости билетов в зависимости от выбранных параметров.*

*Устройство должно быть подключено к сети. Скорость передачи информации должна быть выше 0.5 мегабит в секунду. На устройстве требуется иметь браузер для просмотра сайта. Требование к браузеру: браузер Internet Explorer 10 и выше, все остальные браузеры последних либо предпоследних версий.*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке) *Введение. 1. Анализ предметной области. 2. Проектирование системы. 3. Описание процесса реализации. 4. Руководство пользователя. Приложение А (справочное).–*

5. Консультант по проекту (с обозначением разделов проекта) *А.Ф. Трофимович*

6. Дата выдачи задания –––– *26 января 2023 г.–––*

7. Календарный график работы над проектом на весь период проектирования

(с обозначением сроков выполнения и трудоемкости отдельных этапов):

*оформление пояснительной записки и выполнение задания к 23.02 – 30 %*

*оформление пояснительной записки и выполнение задания к 30.03 – 60 %*

*оформление пояснительной записки и выполнение задания к 27.04 – 90 %*

*Защита курсового проекта c 04.05.2023 по 18.05.2023г.–*

Руководитель*– –––––––– А.Ф. Трофимович*

(подпись)

Задание принял к исполнению *–––––––\_ Е.В. Сурма*

(подпись)

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc134658707)

[1 ОБЗОР ИССЛЕДУЕМОЙ ОБЛАСТИ 6](#_Toc134658708)

[1.1 Описание предметной области 6](#_Toc134658709)

[1.2 Обзор аналогов 9](#_Toc134658710)

[1.3 Постановка задачи 13](#_Toc134658711)

[2 Проектирование системы ПОКУПКИ БИЛЕТОВ НА АВИАРЕЙСЫ 14](#_Toc134658712)

[2.1 Проектирование базы данных 14](#_Toc134658713)

[2.2 Проектирование приложения 20](#_Toc134658714)

[3 разработка системы покупки билетов на авиарейсы 29](#_Toc134658715)

[3.1 Выбор средств реализации (СУБД) 29](#_Toc134658716)

[3.2 Выбор средств реализации (ЯП) 33](#_Toc134658717)

[3.3 Реализация базы данных 37](#_Toc134658718)

[3.4 Программная реализация приложения 39](#_Toc134658719)

[3.5 Инструкция пользователя 41](#_Toc134658720)

[Заключение 48](#_Toc134658721)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 49](#_Toc134658722)

[ПРИЛОЖЕНИЕ А 50](#_Toc134658723)

[ВЕДОМОСТЬ КУРСОВОГО ПРОЕКТА 51](#_Toc134658724)

# Введение

Авиаперевозки являются одной из самых быстроразвивающихся отраслей в мире. С каждым годом число людей, путешествующих на самолетах, увеличивается, что требует от авиакомпаний постоянного совершенствования своих услуг и улучшения качества обслуживания. Одним из ключевых компонентов успешной работы авиакомпаний является продажа авиабилетов. В связи с этим в настоящее время все большее число авиакомпаний ориентируется на использование автоматизированных систем продажи билетов для упрощения процесса продажи, повышения качества обслуживания и улучшения удобства для пассажиров.

Целью данного курсового проекта является разработка системы продажи авиабилетов, которая будет обеспечивать удобство и простоту в использовании, быстроту обработки заказов, а также точность и надежность при обработке платежей. В рамках данной работы будет рассмотрена основная функциональность, требования к системе, ее архитектура, используемые технологии и инструменты.

Основной задачей проекта является разработка полноценной системы продажи авиабилетов, которая будет позволять пассажирам приобретать авиабилеты через веб-интерфейс. Система должна обеспечивать простоту в использовании, быстроту обработки заказов и точность при обработке платежей. Кроме того, система должна обладать возможностью быстрой и удобной настройки ценовой политики для авиакомпании, что позволит ей эффективно управлять своими доходами и прибыльностью.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучение требований к системе и функциональных возможностей.
2. Проектирование системы и ее архитектуры.
3. Разработка веб-интерфейса системы.

В качестве исходных данных для разработки системы будут использованы данные о рейсах, ценах и наличии свободных мест на рейсах, предоставляемые авиакомпаниями

В результате выполнения проекта ожидается создание современной и эффективной системы продажи авиабилетов, которая будет полезна как для авиакомпаний, так и для их клиентов, и поможет сделать процесс покупки авиабилетов более комфортным и быстрым.

# ОБЗОР ИССЛЕДУЕМОЙ ОБЛАСТИ

## Описание предметной области

Сегодня одним из самых популярных способов покупки авиабилетов является онлайн-покупка через интернет-сервисы. Такие сервисы позволяют клиентам самостоятельно выбирать маршрут, дату, время вылета и прилета, а также сравнивать цены на авиабилеты различных авиакомпаний. В СНГ наиболее популярными онлайн-сервисами для покупки авиабилетов являются Яндекс.Авиабилеты, *Aviasales* (ранее известный как *JetRadar*), *KAYAK* и *Skyscanner*. Эти сервисы предлагают пользователям широкий выбор авиакомпаний, различные фильтры и опции для поиска наиболее подходящего рейса по цене и времени, а также возможность оформления бронирования и оплаты билета онлайн.

Однако, при покупке авиабилетов через онлайн-сервисы, клиенты могут столкнуться с рядом проблем, таких как ограничения на выбор места в самолете, возможность отмены или изменения бронирования, дополнительные комиссии за оплату билета и другие. Кроме того, многие авиакомпании предлагают специальные тарифы и скидки только при покупке билета через их официальный сайт или офис продаж. Это означает, что клиенты могут упустить возможность сэкономить деньги, если будут покупать билеты только через онлайн-сервисы.

С другой стороны, традиционная технология продажи билетов на авиарейсы до сих пор остается востребованной. Особенно это касается пожилых людей и людей, не имеющих доступа к Интернету или не умеющих пользоваться компьютером. Кроме того, многие пассажиры предпочитают получить консультацию от специалистов в офисах продаж и узнать о всех возможных вариантах перелета, сроках и условиях бронирования.

В данном контексте разработка автоматизированной системы продажи авиабилетов может значительно улучшить качество обслуживания клиентов авиакомпаний и агентств бронирования билетов, а также сократить время ожидания и очередей в офисах продаж. Кроме того, автоматизация процесса продажи билетов позволит сократить расходы на обслуживание и увеличить эффективность работы компаний. Такие системы могут быть оснащены алгоритмами, которые учитывают предпочтения и потребности клиентов, а также обеспечивают быстрое и удобное бронирование билетов с использованием различных платежных систем. Кроме того, автоматизированные системы могут предоставлять клиентам более подробную информацию о рейсах, возможность выбора места в самолете, а также информацию об услугах и дополнительных возможностях, которые предоставляются авиакомпанией.

Однако, внедрение автоматизированных систем продажи авиабилетов может столкнуться с некоторыми трудностями, такими как необходимость обучения персонала и настройка системы в соответствии с требованиями компании. Кроме того, такие системы требуют постоянного технического обслуживания и обновления, чтобы обеспечить стабильную и безопасную работу.

Тем не менее, автоматизация процесса продажи авиабилетов имеет огромный потенциал для оптимизации работы авиакомпаний и агентств бронирования билетов, а также улучшения качества обслуживания клиентов. При правильном подходе и использовании современных технологий, автоматизированные системы продажи билетов могут стать надежным инструментом для развития бизнеса и улучшения конкурентоспособности компаний в авиационной отрасли.

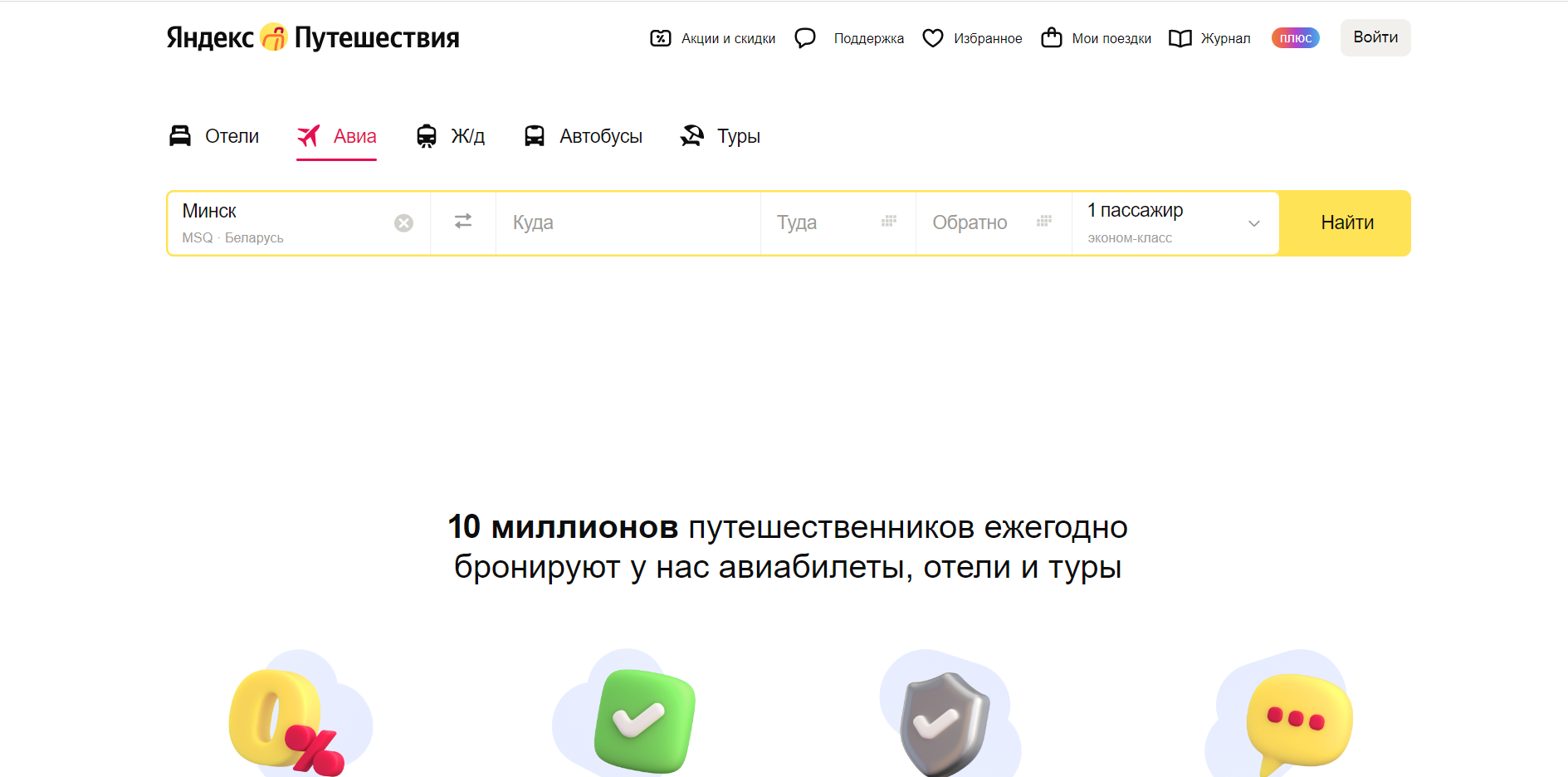


Рисунок 1.1 – Главная страница *travel.yandex.ru*

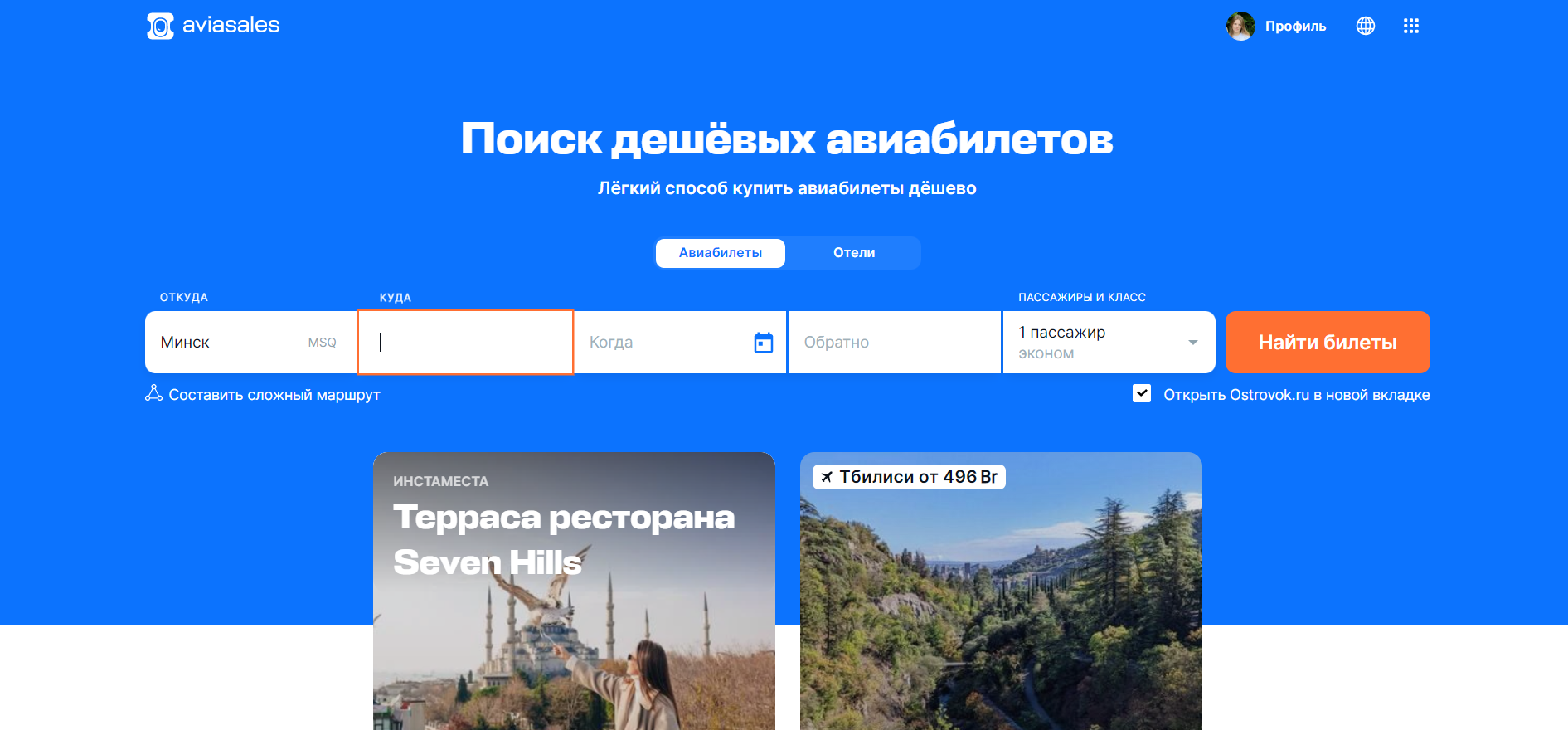


Рисунок 1.2 – Главная страница *aviasales.by*

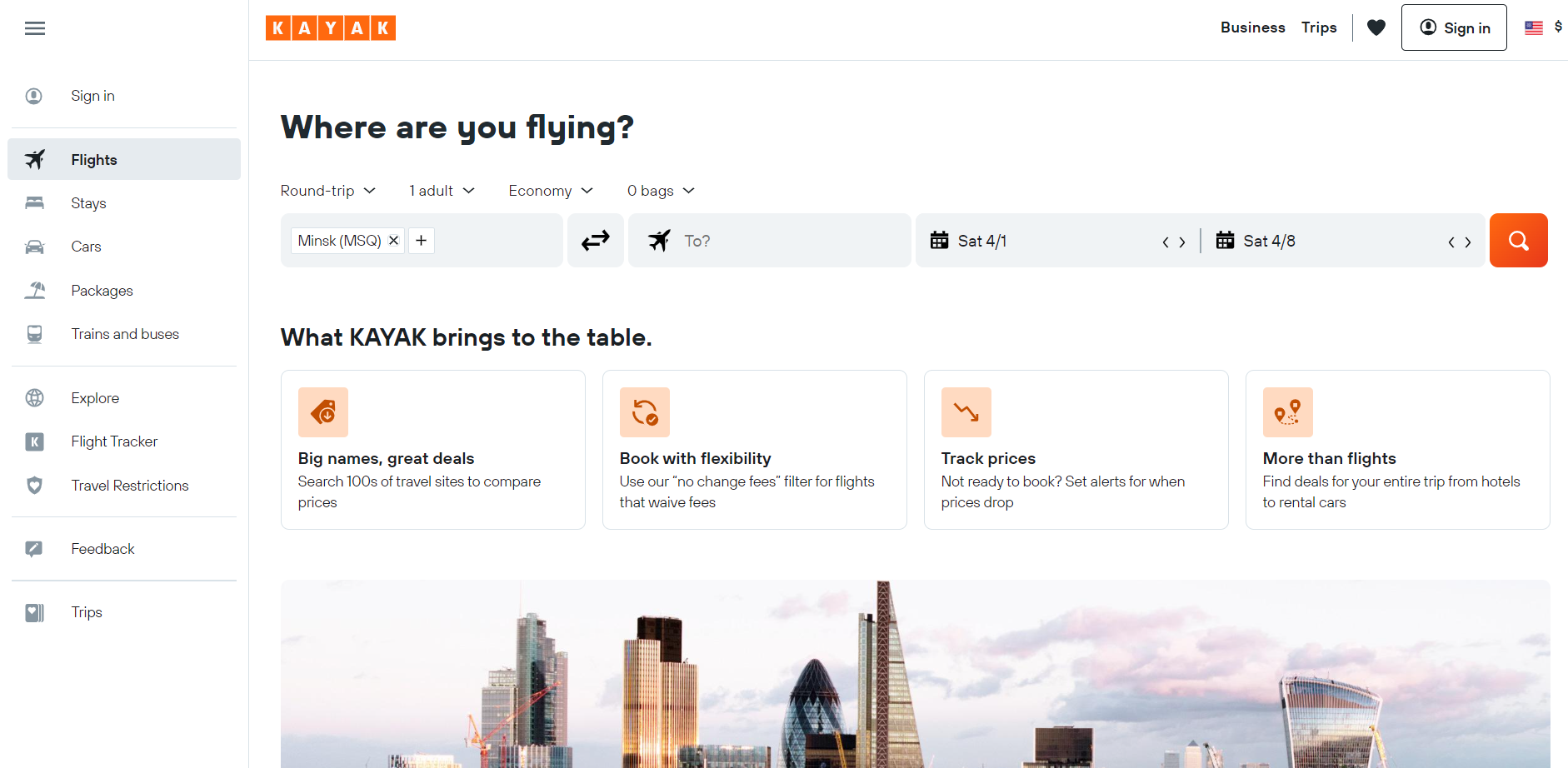


Рисунок 1.3 – Главная страница *kayak.com*

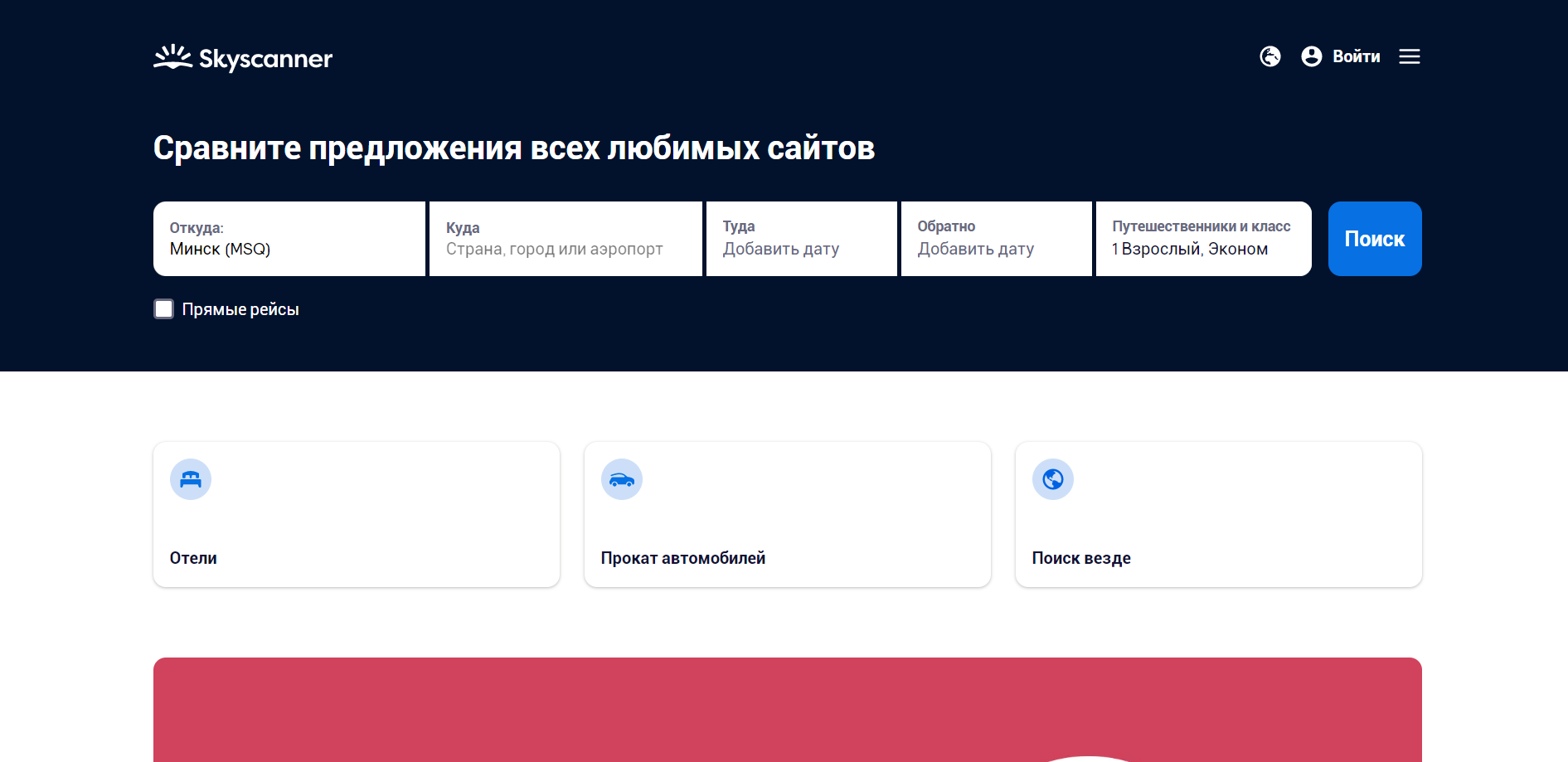


Рисунок 1.4 – Главная страница *ru.skyscanner.com*

## Обзор аналогов

С появлением интернета, возможность приобретения билетов онлайн избавила людей от необходимости посещать офисы продаж авиакомпаний и турагентств. Сегодня на рынке существует множество сервисов, позволяющих быстро и удобно купить авиабилеты.

Каждый из этих сервисов имеет свои преимущества и недостатки, которые важно учитывать при выборе оптимального варианта для покупки авиабилетов.

Яндекс.Авиабилеты – это сервис для покупки авиабилетов, разработанный компанией Яндекс. Он позволяет быстро и удобно найти подходящий рейс и купить билеты на него.

Один из главных плюсов Яндекс.Авиабилеты – это простой и удобный интерфейс. Сайт имеет интуитивно понятный дизайн и позволяет быстро находить необходимые рейсы. В верхней части страницы расположены основные фильтры поиска, такие как место отправления и прибытия, даты полетов, количество пассажиров и класс обслуживания(см.рис. 1.6). Также есть возможность выбрать прямой рейс или рейс с пересадкой, а также сравнить цены на различные даты.

Благодаря интеграции с крупными авиакомпаниями, такими как *Aeroflot*, *S7 Airlines, Rossiya Airlines* и другими, Яндекс.Авиабилеты предлагает более широкий выбор рейсов и направлений. Также можно выбрать класс обслуживания и оплатить билеты онлайн с помощью банковской карты или электронного кошелька.

Яндекс.Авиабилеты также предоставляет различные акции и специальные предложения для своих пользователей, что делает покупку билетов более выгодной. Например, можно получить скидку на первую покупку или использовать промокод для получения скидки на следующую покупку.

Однако, как и у любого сервиса, у Яндекс.Авиабилеты есть и некоторые недостатки. Например, в некоторых случаях цены на билеты могут отличаться от реальных цен на сайтах авиакомпаний. Также многие пользователи жаловались на то, что при выборе рейса сайт не всегда предоставляет достаточно информации о пересадках и длительности полета.

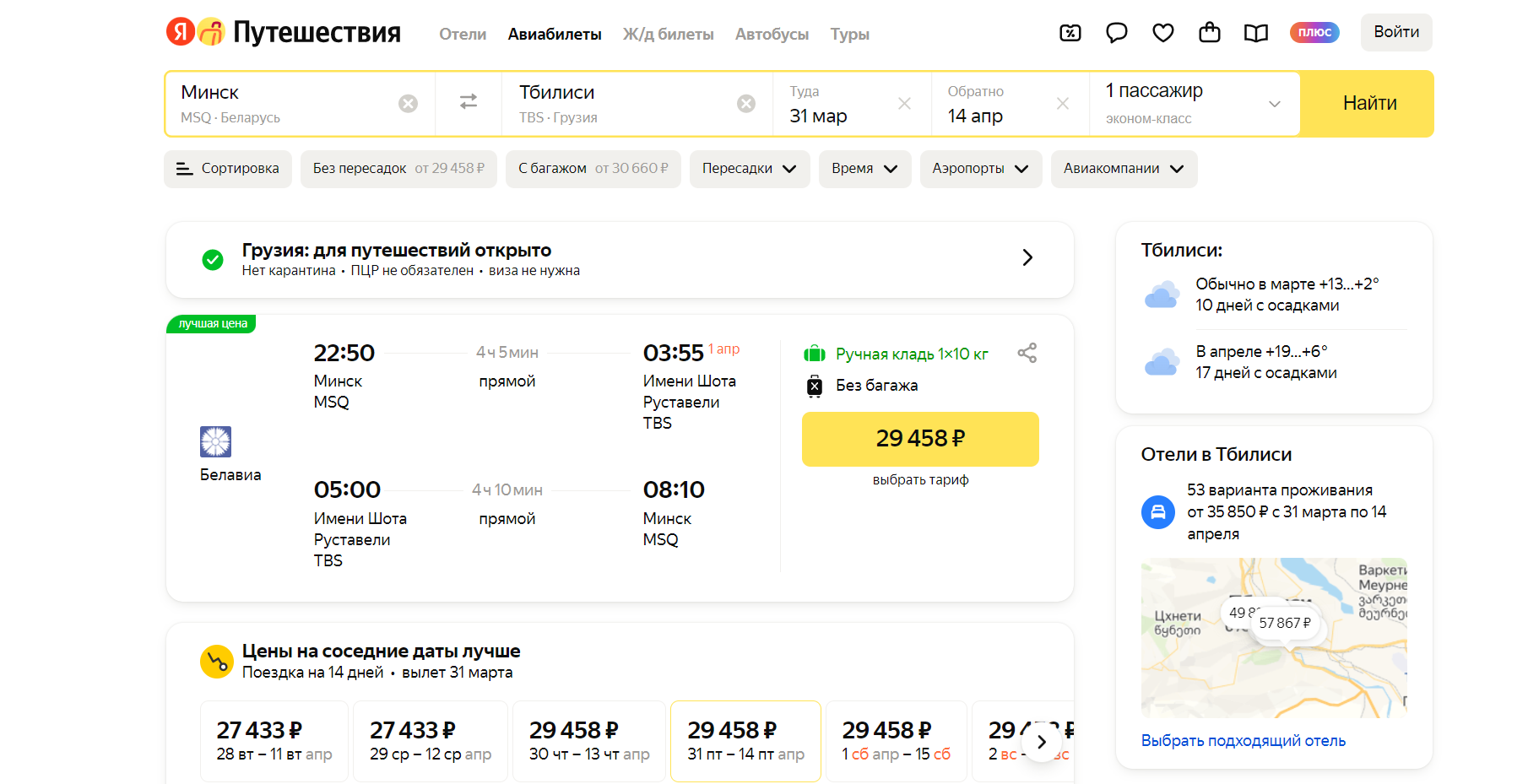


Рисунок 1.6 – Интерфейс Яндекс.Авиабилеты

*Aviasales* – это сервис поиска и сравнения цен на авиабилеты у различных авиакомпаний и агентств. Он имеет простой и понятный интерфейс, позволяющий быстро находить нужные рейсы и выбирать самые выгодные варианты покупки билетов.

Одним из главных преимуществ *Aviasales* является возможность сравнения цен на билеты у многих поставщиков услуг. Это позволяет пользователям выбрать самый выгодный вариант, сэкономить деньги и время на поиске нужного рейса. Кроме того, сервис предоставляет подробную информацию о каждом рейсе, включая даты вылета и прилета, тип самолета, количество пересадок и время в пути.

Однако, у *Aviasales* есть и некоторые недостатки. Во-первых, сервис не является прямым продавцом авиабилетов и не гарантирует наличие билетов на выбранные даты и рейсы. Это может привести к неудобствам, если билет на нужный рейс будет распродан до того, как пользователь успеет его купить.

Во-вторых, *Aviasales* не всегда показывает самые дешевые цены на билеты. Иногда цена, указанная на сайте, может отличаться от той, которую предлагает авиакомпания или агентство при покупке билета. Поэтому, перед покупкой авиабилетов через *Aviasales*, рекомендуется проверить цены на официальных сайтах авиакомпаний.

Несмотря на эти недостатки, *Aviasales* остается популярным сервисом для поиска и сравнения цен на авиабилеты. Его преимущества включают широкий выбор рейсов и направлений, удобный интерфейс и возможность выбрать самый выгодный вариант покупки билетов. Кроме того, пользователи отмечают нестабильную работу сервиса и возможные проблемы с возвратом билетов при отмене рейса.

*KAYAK* - это сервис для поиска и сравнения цен на авиабилеты, аренду автомобилей и бронирование отелей. Он был основан в 2004 году и стал одним из лидеров в индустрии туризма.

Ключевое преимущество *KAYAK* заключается в широком функционале. Пользователи могут не только купить авиабилеты, но и забронировать отель или арендовать автомобиль. Кроме того, *KAYAK* предлагает возможность настройки уведомлений о снижении цен на выбранные направления, что позволяет сэкономить деньги при покупке билетов.

Еще одним преимуществом *KAYAK* является возможность сравнения цен на авиабилеты у разных авиакомпаний и агентств. Это позволяет выбрать самый выгодный вариант и сэкономить деньги.

Однако, как и у других сервисов, у *KAYAK* есть свои недостатки. Например, поиск билетов может занять больше времени, чем на других сайтах, так как *KAYAK* ищет цены у различных авиакомпаний и агентств. Также недостатком *KAYAK* является более сложный интерфейс по сравнению с другими сервисами (см.рисунок 1.5). При бронировании отелей и аренде автомобилей могут возникать дополнительные комиссии и непредвиденные расходы.

Кроме того, на сайте могут появляться рекламные предложения и баннеры, которые могут отвлекать пользователя. Также, при использовании дополнительных функций, таких как бронирование отелей или аренда автомобилей, пользователь может столкнуться с дополнительными комиссиями и сборами.

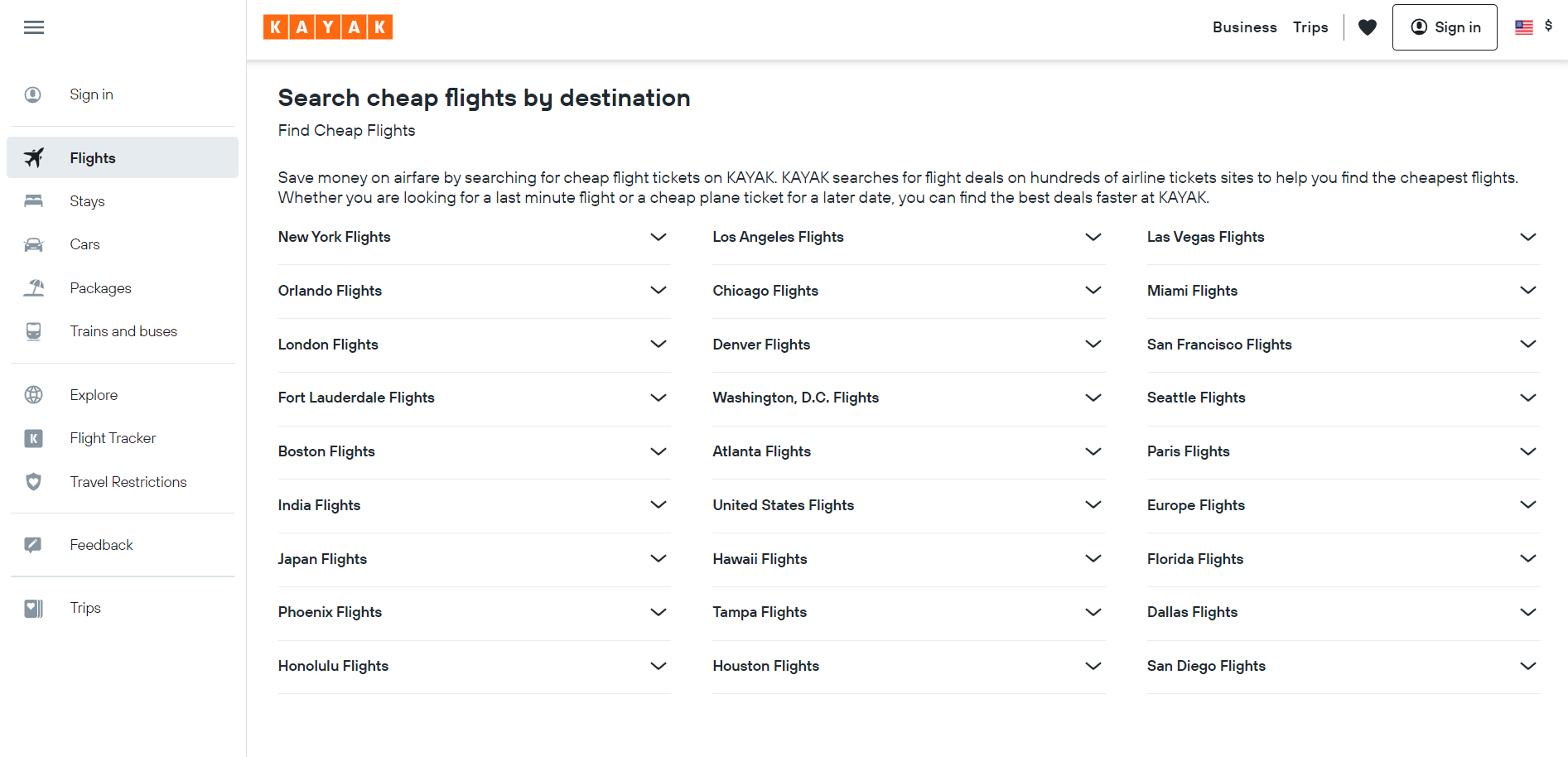


Рисунок 1.5 – Интерфейс *kayak.com*

*Skyscanner* также предоставляет возможность поиска и сравнения цен на билеты у различных авиакомпаний и агентств. Кроме того, этот сервис имеет множество фильтров для удобства пользователя, таких как фильтры по времени вылета и прилета, количество пересадок, авиакомпании и другие. Еще одним достоинством *Skyscanner* является наличие приложения для мобильных устройств, которое упрощает процесс поиска и покупки авиабилетов.

Однако, некоторые недостатки *Skyscanner* включают в себя не всегда точные цены на билеты, которые могут измениться на последних этапах покупки, а также ограничения в планировании маршрутов с пересадками. Кроме того, *Skyscanner* не предоставляет информацию о местах на самолете и услугах, предоставляемых авиакомпаниями, таких как бесплатные перекусы или выбор места.

## Постановка задачи

Цель проекта: Создание веб-приложения для покупки билетов на авиарейсы.

Задачи проекта:

1. Разработка домашней страницы с возможностью регистрации и входа в систему, корзиной для оплаты покупок.
2. Создание базы данных для хранения информации о авиарейсах и билетах.
3. Добавление функционала для пользователя-администратора, включающего возможность удаления, редактирования и добавления информации о авиарейсах и билетах.
4. Создание интерфейса для поиска доступных авиарейсов и покупки билетов.
5. Разработка системы управления заказами.
6. Реализация функционала экспорта данных авиарейсов из БД в файл формата *json*.

Аппаратные требования:

1. Компьютер с доступом в Интернет
2. Браузер, поддерживающий *HTML*, *CSS* и *JavaScript*

Программные требования:

1. Операционная система: любая, поддерживающая серверные приложения
2. СУБД: *PostgreSQL*

Для разработки веб-приложения можно использовать следующие технологии:

Языки программирования: *HTML*, *CSS*, *JavaScript*, *Python*

Библиотеки и фреймворки: *FastAPI* и др.

# Проектирование системы ПОКУПКИ БИЛЕТОВ НА АВИАРЕЙСЫ

## Проектирование базы данных

В данном разделе представлено проектирование базы данных для системы покупки билетов на авиарейсы. База данных является фундаментальным элементом любой информационной системы, и эффективное ее проектирование является ключевым для обеспечения надежности, гибкости и эффективности работы системы.

Целью этого раздела является представление описания сущностей, атрибутов и доменов базы данных, а также обсуждение процесса нормализации, который поможет устранить избыточность и аномалии данных.

Для начала рассмотрим сущности, их атрибуты и домены, которые будут присутствовать в нашей базе данных. Атрибуты являются характеристиками сущности, которые хранятся в базе данных. Домен определяет множество значений, которые может принимать атрибут.

1. Сущность "Пользователь":

Атрибуты:

* Идентификатор: представляет идентификатор пользователя.
* Имя: представляет имя пользователя.
* Фамилия: представляет фамилию пользователя.
* Электронная почта: представляет адрес электронной почты пользователя.
* Пароль: представляет пароль для аутентификации пользователя.
* Роль: представляет роль пользователя.
* Дата создания: представляет дату регистрации пользователя.

Домены:

* Идентификатор пользователя представляет собой уникальное число.
* Имя и Фамилия могут иметь домен типа строка с определенной максимальной длиной.
* Электронная почта имеет домен типа строка с проверкой на соответствие формату электронной почты.
* Пароль может иметь домен типа строка или храниться в зашифрованном виде для обеспечения безопасности.
* Дата создания имеет домен типа дата (значение по умолчанию текущая дата).
* Роль имеет домен типа целое число (по умолчанию значение 1 – пользователь, 2 - администратор)

1. Сущность "Рейс":

Атрибуты:

* Идентификатор: представляет идентификатор рейса.
* Номер рейса: представляет номер рейса.
* Цена: представляет стоимость рейса.
* Дата создания: представляет дату создания рейса.

Домены:

* Идентификатор рейса представляет собой уникальное число.
* Номер рейса может иметь домен типа строка с определенной максимальной длиной(16 символов).
* Цена имеет домен типа число с плавающей точкой.
* Дата создания имеет домен типа дата (значение по умолчанию текущая дата).

1. Сущность "Аэропорт":

Атрибуты:

* Идентификатор: представляет идентификатор аэропорта.
* Код: представляет уникальный код аэропорта.
* Страна: представляет страну аэропорта.

Домены:

* Идентификатор рейса представляет собой уникальное число.
* Код может иметь домен типа строка с определенной максимальной длиной (10 символов).
* Страна имеет домен типа строка с определенной максимальной длиной (64 символа).

1. Сущность "Паспортные данные":

Атрибуты:

* Идентификатор: представляет идентификатор паспортных данных пользователя.
* Номер паспорта: представляет номер паспорта пользователя.
* Имя пассажира: представляет имя пассажира.
* Фамилия пассажира: представляет фамилию пассажира.
* Дата рождения: представляет дату рождения пассажира.

Домены:

* Идентификатор пассажира представляет собой уникальное число.
* Номер паспорта может иметь домен типа строка с определенной максимальной длиной (32 символа).
* Имя пассажира имеет домен типа строка с определенной максимальной длиной (64 символа).
* Фамилия пассажира имеет домен типа строка с определенной максимальной длиной (64 символа).
* Дата рождения имеет домен типа дата.

1. Сущность "Бронь":

Атрибуты:

* Идентификатор: представляет идентификатор рейса.
* Номер рейса: представляет номер рейса.
* Цена: представляет стоимость рейса.
* Дата создания: представляет дату создания рейса.

Домены:

* Идентификатор рейса представляет собой уникальное число.
* Номер рейса может иметь домен типа строка с определенной максимальной длиной (16 символов).
* Цена имеет домен типа число с плавающей точкой.
* Дата создания имеет домен типа дата (значение по умолчанию текущая дата).

1. Сущность "Бронь-Пассажир":

Атрибуты:

* Идентификатор брони: представляет идентификатор брони.
* Идентификатор пассажира: представляет идентификатор пассажира.

Домены:

* Идентификатор брони имеет домен целого уникального числа.
* Идентификатор пассажира имеет домен целого уникального числа.

1. Сущность "Трансфер":

Атрибуты:

* Идентификатор рейса: представляет идентификатор рейса.
* Идентификатор аэропорта вылета: представляет идентификатор аэропорта вылета.
* Идентификатор аэропорта прибытия: представляет идентификатор аэропорта прибытия.

Домены:

* Идентификатор рейса представляет собой уникальное число.
* Идентификатор аэропорта вылета может иметь домен типа целое число.
* Идентификатор аэропорта прибытия может иметь домен типа целое число.

Каждая сущность представляет конкретный объект в системе и имеет набор связанных с ней атрибутов, которые описывают ее свойства.

Один из важных аспектов проектирования базы данных - это нормализация данных. Нормализация помогает устранить избыточность, неоднозначность и аномалии в данных, а также обеспечивает эффективность запросов и обновлений базы данных. В процессе нормализации данные структурируются в различные нормальные формы (НФ), начиная с первой нормальной формы (1НФ) и до более высоких нормальных форм, таких как вторая нормальная форма (2НФ) и третья нормальная форма (3НФ).

При проектировании базы данных для нашей системы покупки билетов на авиарейсы мы уделяем особое внимание нормализации данных, чтобы обеспечить эффективность и целостность информации.

Для достижения первой нормальной формы (1НФ) необходимо убедиться, что каждый атрибут в базе данных является атомарным, то есть не делится на более мелкие составляющие. В нашем проекте мы обеспечили, что каждый атрибут сущностей является атомарным, и он представляет только одно значение.

Для достижения второй нормальной формы (2НФ) необходимо убедиться, что каждый неключевой атрибут полностью зависит от всего первичного ключа сущности. В нашем случае, например, атрибуты сущности "Билет" (Номер билета, Цена, Статус билета) полностью зависят от первичного ключа "Номер билета". Таким образом, мы удовлетворяем требованиям 2НФ.

Для достижения третьей нормальной формы (3НФ) необходимо убедиться, что каждый неключевой атрибут не зависит от других неключевых атрибутов, а зависит только от первичного ключа сущности. В нашем проекте мы обеспечили, что все атрибуты каждой сущности непротиворечивы и не зависят от других неключевых атрибутов.

При проектировании базы данных мы учли возможность связей между сущностями, чтобы установить связь и взаимодействие между ними. На основе нашего проекта "система покупки билетов на авиарейсы", мы определили следующие связи:

1. Связь "Пользователь делает Бронь билета":

* Эта связь указывает, что каждый пользователь может совершать покупку билетов.
* Связь может быть реализована с использованием внешнего ключа, который связывает пользователя с конкретным билетом, который он приобретает.
* Эта связь позволяет отслеживать и контролировать покупки, сделанные каждым пользователем, и обеспечивает связь между сущностями "Пользователь" и "Бронь".

1. Связь "Бронь относится к Рейсу":

* Эта связь указывает, что каждый билет связан с определенным авиарейсом.
* Связь может быть реализована с использованием внешнего ключа, который связывает билет с конкретным авиарейсом.
* Эта связь позволяет отслеживать, какой билет относится к какому авиарейсу, и обеспечивает связь между сущностями "Бронь" и "Рейс".

Эти связи позволяют нам строить запросы и операции, которые связывают данные между сущностями. Например, мы можем получить список билетов, купленных конкретным пользователем, используя связь "Пользователь делает Бронь билета". Мы также можем получить информацию о конкретном авиарейсе, связанном с определенным билетом, с помощью связи "Бронь относится к Рейсу".

Важно проектировать связи между сущностями таким образом, чтобы они отражали реальные взаимосвязи и операции, которые выполняются в системе. Это обеспечивает эффективность работы с данными, целостность базы данных и точность хранения информации о покупках билетов на авиарейсы.

Логическая схема и *ER*-диаграмма в нотации Питера Чена представляют визуальное представление структуры базы данных и ее сущностей, атрибутов и связей. Эти схемы помогают вам лучше понять и представить организацию данных, а также обеспечивают основу для создания физической базы данных.

Логическая схема базы данных включает в себя таблицы, поля и связи между таблицами. Она описывает структуру базы данных без учета конкретных аспектов физической реализации, таких как типы данных и индексы (рис.2.1.1).

*ER*-диаграмма (сокращение от *Entity*-*Relationship* *diagram*) в нотации Питера Чена представляет сущности, атрибуты и связи между ними в графическом виде. Она позволяет легко визуализировать структуру данных и связи между сущностями (рис.2.1.2).

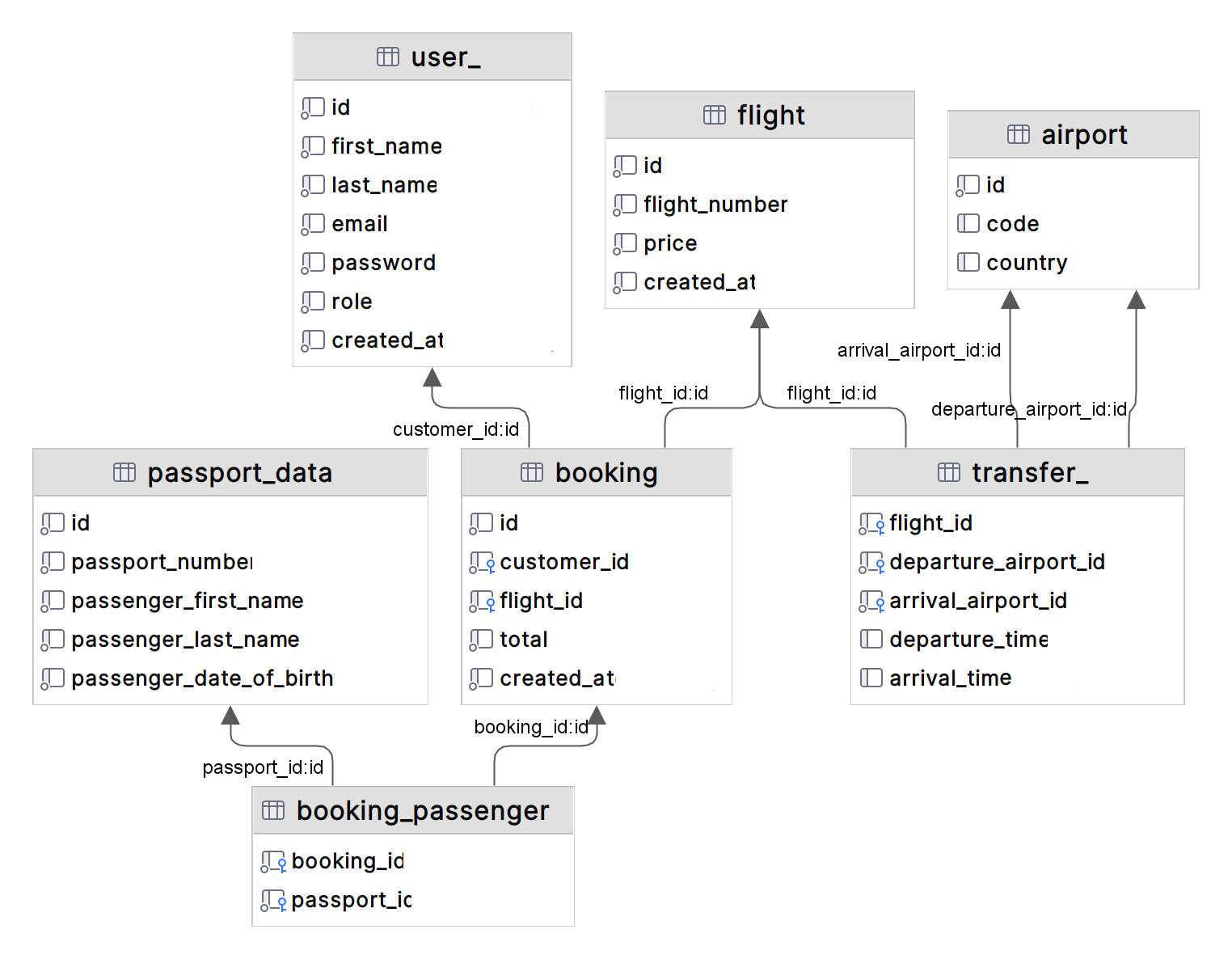


Рисунок 2.1.1 – Логическая схема базы данных

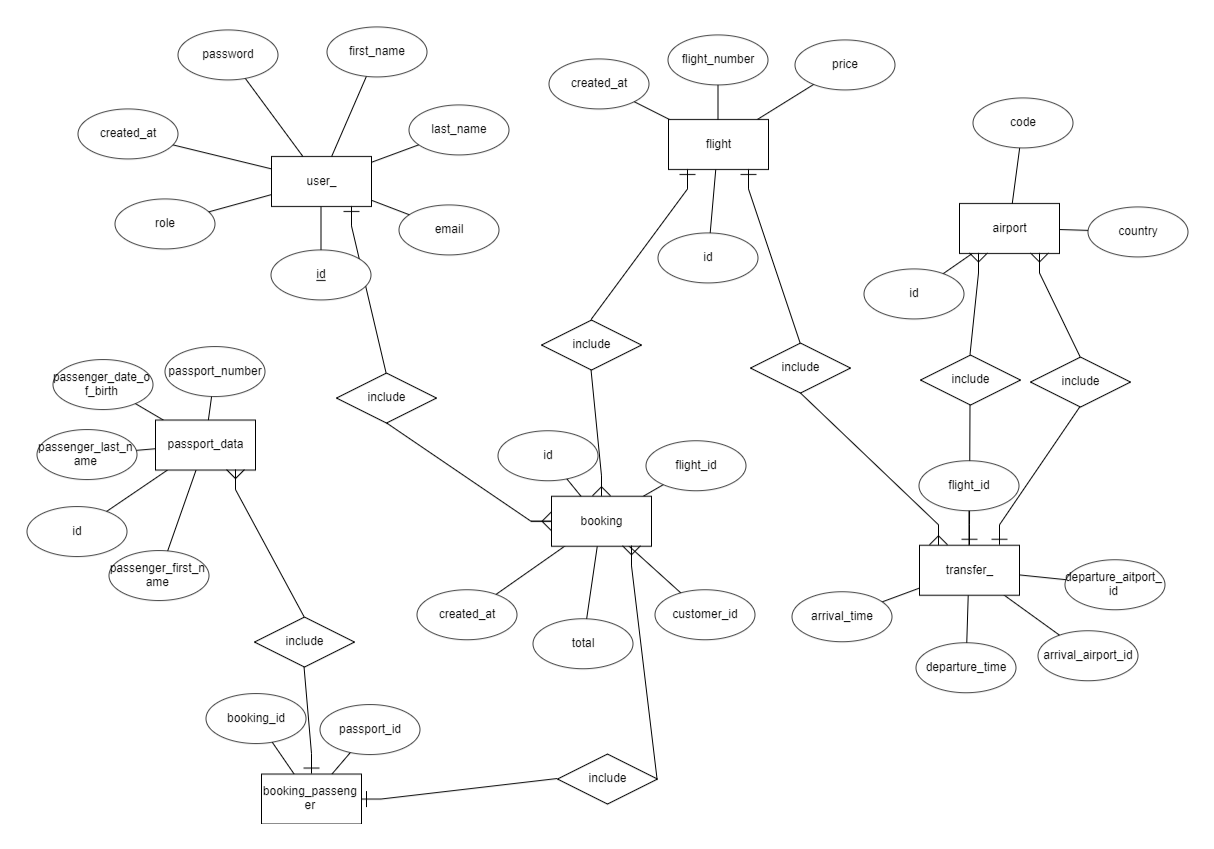


Рисунок 2.1.2 – *ER*-диаграмма в нотации Питера Чена

## Проектирование приложения

В данном разделе рассмотрим проектирование самого приложения, которое обеспечивает функциональность системы покупки билетов на авиарейсы.

При проектировании приложения для системы покупки билетов на авиарейсы мы стремимся создать удобный и интуитивно понятный интерфейс для пользователей, который обеспечит следующую функциональность:

* Регистрация и аутентификация пользователей.
* Поиск и просмотр доступных авиарейсов и билетов.
* Покупка билетов на выбранный авиарейс.
* Управление профилем пользователя и просмотр истории покупок.
* Управление и обновление информации об авиарейсах.

Для наглядного представления структуры и функциональности приложения используются следующие диаграммы:

Диаграмма классов (рис.2.2.1):

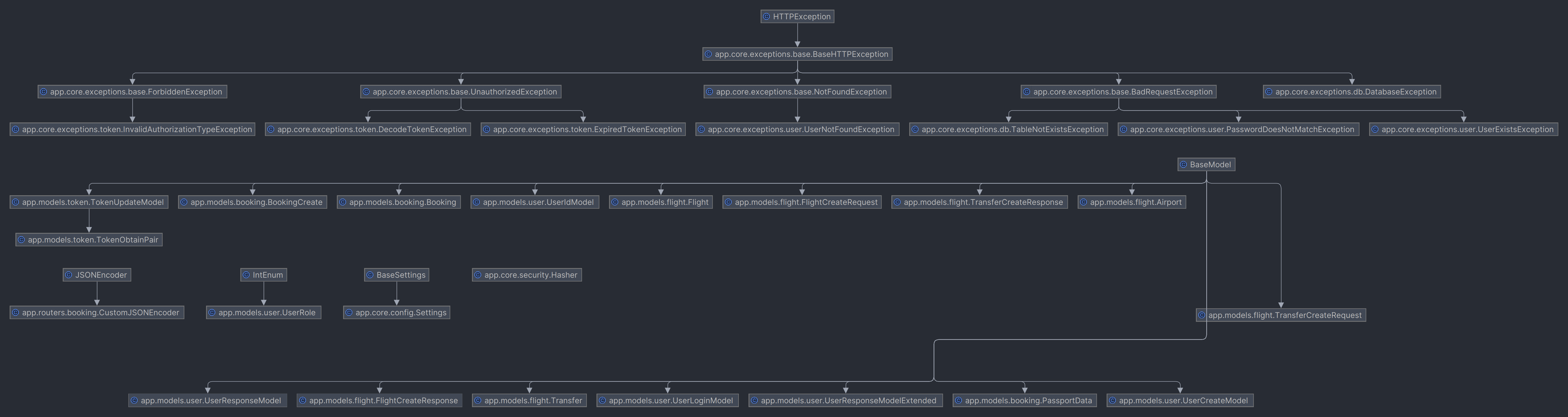


Рисунок 2.2.1 – Диаграмма классов

Диаграмма вариантов использования (рис. 2.2.2):

* Диаграмма вариантов использования описывает основные сценарии использования приложения и его функциональность с точки зрения пользователя.
* На диаграмме будут представлены различные варианты использования, такие как "Просмотр информации об авиарейсе", "Бронь билета", "Регистрация пользователя" и другие.

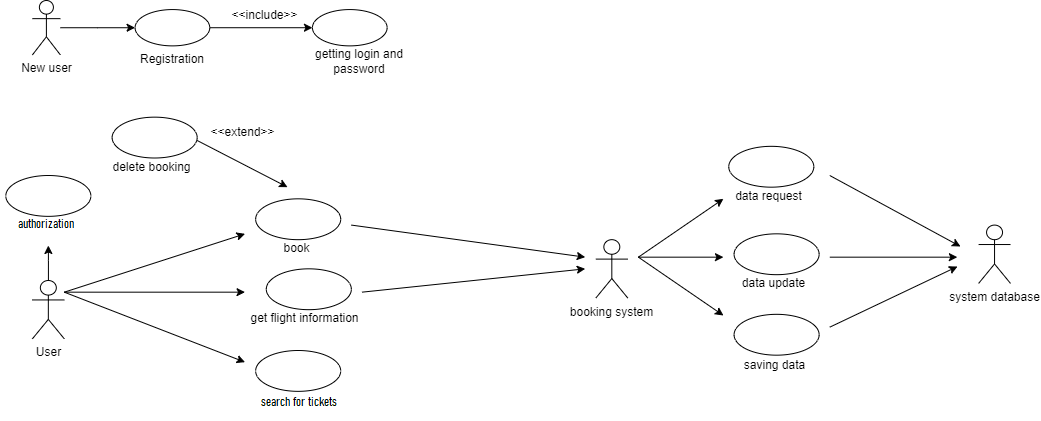


Рисунок 2.2.2 – Диаграмма вариантов использования

Схема алгоритма регистрации (рис. 2.2.3):

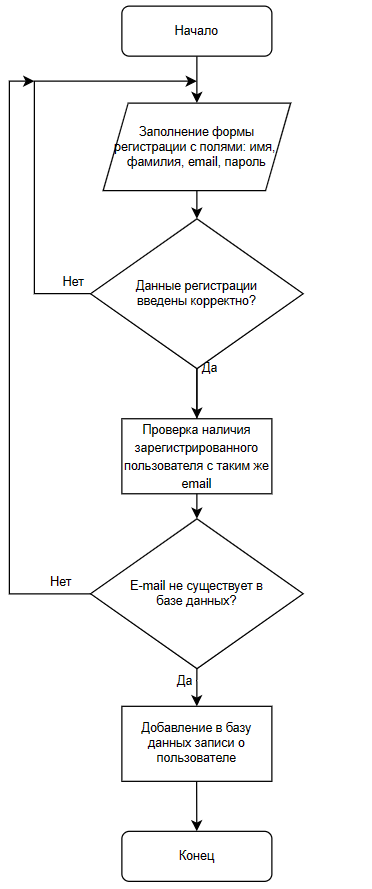


Рисунок 2.2.3 – Схема алгоритма регистрации

Схема алгоритма аутентификации (рис. 2.2.4):

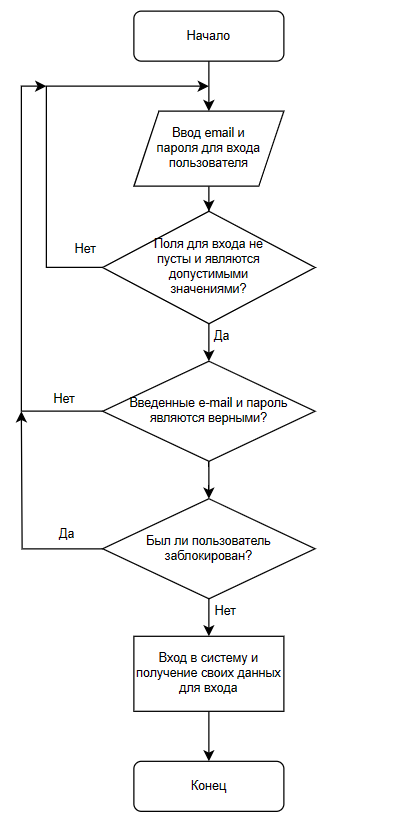


Рисунок 2.2.4 – Схема алгоритма аутентификации

Схема алгоритма авторизации (рис. 2.2.5):

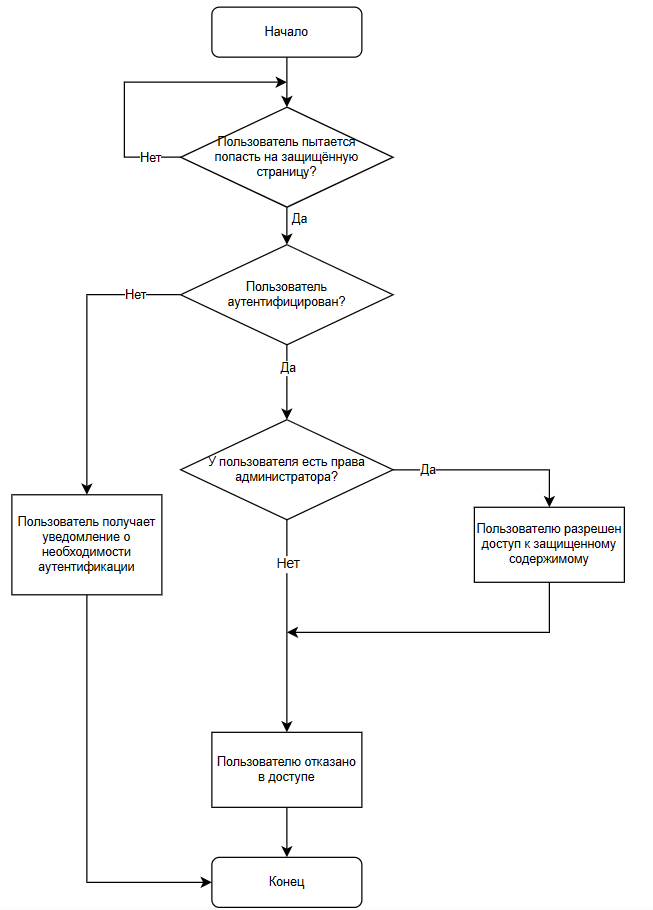


Рисунок 2.2.5 – Схема алгоритма авторизации

Схема алгоритма поиска товаров (рис. 2.2.6):

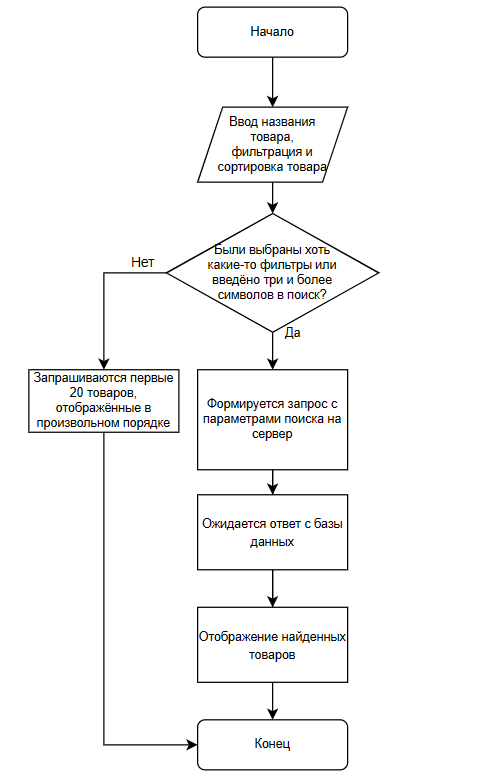


Рисунок 2.2.4 – Схема алгоритма поиска товаров

Схема алгоритма оформления заказа(рис. 2.2.7):

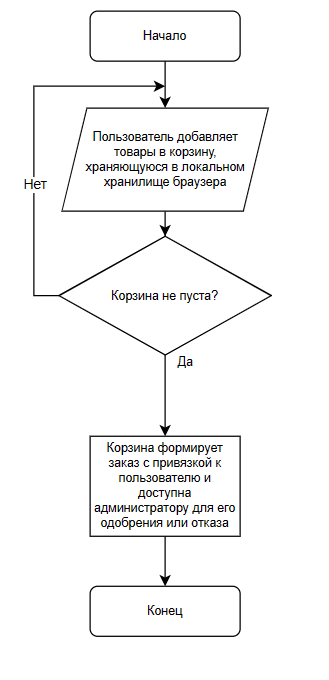


Рисунок 2.2.7 – Схема алгоритма оформления заказа

Архитектура приложения для системы покупки билетов на авиарейсы будет основана на модель-представление-контроллер (*Model-View-Controller, MVC). MVC* является шаблоном проектирования, который разделяет компоненты приложения на три основных уровня: модель, представление и контроллер. Это позволяет логически разделить функциональность и упростить разработку, тестирование и поддержку приложения.

В данной архитектуре приложения будут присутствовать следующие компоненты:

1. Модель:

* Модель представляет бизнес-логику и данные приложения.
* В данном случае модель будет содержать классы, отвечающие за пользователей, билеты, авиарейсы и другие сущности системы.
* Модель также обеспечивает доступ к базе данных, выполнение операций чтения и записи данных.

1. Представление:

* Представление отвечает за отображение данных и интерфейс пользователя.
* В нашей системе представление будет состоять из веб-интерфейса, который позволит пользователям просматривать доступные авиарейсы, покупать билеты и управлять своим профилем.
* Представление будет отображать данные из модели и обрабатывать действия пользователя.

1. Контроллер:

* Контроллер управляет взаимодействием между моделью и представлением.
* Контроллер принимает запросы от пользователя через представление, обрабатывает их и обновляет состояние модели.
* Контроллер также отвечает за передачу данных между моделью и представлением.

Архитектура приложения будет поддерживать разделение логики и данных, что позволит легко расширять функциональность, вносить изменения и поддерживать систему. Кроме того, использование *MVC* позволяет повысить масштабируемость и переиспользование компонентов.

Для обеспечения безопасности и защиты данных пользователей, важно реализовать механизм аутентификации и авторизации, а также использовать соответствующие механизмы хэширования паролей и защиты от атак.

В данном курсовом проекте мы будем использовать алгоритм хеширования паролей *SHA-256 (Secure Hash Algorithm 256 bit). SHA-256* является одним из наиболее распространенных и безопасных алгоритмов хеширования, который генерирует фиксированную длину хэш-значения, равную 256 бит.

Процесс хеширования пароля с использованием *SHA-256* включает следующие шаги:

1. Получение пароля от пользователя.
2. Преобразование пароля в байтовую строку с помощью соответствующей кодировки, например, *UTF-8.*
3. Вычисление хэш-значения с использованием алгоритма *SHA-256*. В языках программирования обычно есть встроенные функции или библиотеки, которые реализуют этот алгоритм.
4. Получение результирующего хэш-значения, которое будет представлено в виде строки или байтового массива.
5. Сохранение хэш-значения пароля в базе данных или другом хранилище.

Важно отметить, что *SHA-256* является односторонней функцией хеширования, что означает, что невозможно восстановить исходный пароль из его хэш-значения. При аутентификации пользователя процесс проверки пароля состоит в сравнении хэш-значения введенного пользователем пароля с сохраненным хэш-значением в базе данных.

Преимущества использования алгоритма *SHA-256* для хеширования паролей включают:

* Безопасность: *SHA-256* считается криптографически стойким алгоритмом хеширования, что обеспечивает высокую степень безопасности паролей пользователей.
* Уникальность хэш-значений: Хэш-значения, генерируемые *SHA-256*, имеют высокую вероятность быть уникальными для каждого пароля, даже если исходные пароли очень похожи.
* Простота реализации: Алгоритм *SHA-256* широко поддерживается в различных языках программирования и имеет готовые реализации, что облегчает его внедрение в проект.

# разработка системы покупки билетов на авиарейсы

## Выбор средств реализации (СУБД)

В данном разделе рассматривается важный аспект разработки системы покупки билетов на авиарейсы - выбор языка программирования и системы управления базами данных (СУБД), которые будут использоваться в проекте. Выбор этих средств играет значительную роль в эффективной реализации системы, обеспечении ее функциональности, надежности и расширяемости.

Основными требованиями к системе покупки билетов на авиарейсы являются высокая производительность, удобство использования, безопасность. Важно выбрать такие язык программирования и СУБД, которые максимально соответствуют данным требованиям и обеспечат успешное выполнение поставленных задач.

В данной работе для реализации системы покупки билетов на авиарейсы был выбран язык программирования *Python* и система управления базами данных *PostgreSQL*. Ниже будет представлено обоснование данного выбора, а также описание преимуществ, которые делают *Python* и *PostgreSQL* оптимальными средствами реализации данной системы. Также будет проведен анализ альтернативных вариантов языков программирования и СУБД, сравнение их с выбранными средствами на основе ключевых критериев.

Выбор правильных языка программирования и СУБД является важным этапом разработки системы покупки билетов на авиарейсы, и их сочетание должно быть обоснованным и оптимальным для успешной реализации проекта.

* + 1. ***MySQL***

*MySQL* - это система управления реляционными базами данных, которая широко используется для хранения и управления данными веб-приложений и других типов приложений. *MySQL* является одним из наиболее популярных и распространенных СУБД в мире, и многие компании и организации выбирают его для хранения своих данных.

*MySQL* поддерживает стандарт языка *SQL (Structured Query Language),* который позволяет программистам легко выполнять запросы к базе данных и извлекать нужные данные. *MySQL* также поддерживает множество расширений и дополнительных функций, которые делают его гибкой и мощной СУБД.

Одним из основных преимуществ *MySQL* является его открытый исходный код, что делает его доступным для всех, кто хочет использовать его в своих проектах. Благодаря этому большое сообщество разработчиков работает над дальнейшим развитием и улучшением *MySQL*, в том числе добавляет новые функции и исправляет ошибки.

*MySQL* также хорошо масштабируется и может работать с большими объемами данных. Он поддерживает репликацию и кластеризацию, что позволяет распределять нагрузку на несколько серверов и обеспечивать высокую доступность данных.

Однако, как и любая другая СУБД, *MySQL* имеет и свои недостатки. Например, он не поддерживает некоторые функции, которые есть в других СУБД, таких как *PostgreSQL*. Также *MySQL* может иметь проблемы с производительностью при работе с большими объемами данных, хотя это зависит от конкретной реализации и настройки сервера.

* + 1. ***PostgreSQL***

*PostgreSQL* - это одна из самых мощных и расширяемых реляционных СУБД, которая имеет открытый исходный код и поддерживается сообществом разработчиков. Она предоставляет надежное, эффективное и масштабируемое решение для хранения и управления большими объемами структурированных данных. *PostgreSQL* используется во многих крупных и сложных проектах, таких как социальные сети, банковские системы, электронная коммерция, телекоммуникации и др.

Основными преимуществами *PostgreSQL* являются:

1. Надежность и стабильность. *PostgreSQL* имеет высокую степень устойчивости к сбоям, обеспечивая сохранность данных и минимизацию времени простоя. Кроме того, *PostgreSQL* обладает механизмом восстановления после сбоев, который позволяет быстро восстанавливать данные в случае потери.
2. Масштабируемость. *PostgreSQL* может обрабатывать большое количество данных и поддерживает горизонтальное масштабирование, что позволяет расширять базу данных без значительного снижения производительности.
3. Гибкость и расширяемость. *PostgreSQL* предлагает различные возможности расширения, такие как написание пользовательских функций и хранимых процедур на разных языках программирования, а также создание пользовательских типов данных.
4. Поддержка *SQL*. *PostgreSQL* поддерживает полный набор стандартов *SQL*, а также многие дополнительные функции, такие как массивы, геометрические типы данных, *JSON* и др.
5. Безопасность. *PostgreSQL* предлагает широкий набор функций безопасности, включая аутентификацию, авторизацию, шифрование и т.д.

В курсовом проекте выбрана *PostgreSQL* как СУБД для хранения и управления данными, связанными с покупкой билетов на авиарейсы. Этот выбор был обоснован высокой надежностью, масштабируемостью и поддержкой *SQL*, что позволит эффективно обрабатывать большой объем структурированных данных. Кроме того, *PostgreSQL* имеет множество библиотек и инструментов для упрощения работы с данными, а также поддерживает различные языки программирования, что позволит быстро и гибко разрабатывать и изменять функциональность системы покупки билетов на авиарейсы. Благодаря открытому исходному коду *PostgreSQL*, мы также можем быть уверены в поддержке и активной разработке этой СУБД сообществом разработчиков.

Другим важным фактором, который подтверждает выбор *PostgreSQL* для данного курсового проекта, является его совместимость с языком программирования *Python*. *Python* обладает множеством библиотек и фреймворков, которые интегрируются легко и эффективно с *PostgreSQL*. Такое сочетание языка программирования и СУБД обеспечит беспроблемное взаимодействие между различными компонентами системы, а также облегчит разработку и сопровождение проекта.

* + 1. ***MongoDB***

*MongoDB* - это документоориентированная система управления базами данных (СУБД), которая отличается от традиционных реляционных СУБД, таких как *PostgreSQL*, *MySQL* или *Oracle*. *MongoDB* предлагает гибкую и масштабируемую модель данных, основанную на хранении и обработке документов в формате *BSON* (*Binary* *JSON*).

Преимущества использования:

* Гибкая модель данных: *MongoDB* позволяет хранить сложные структуры данных в формате документов, что идеально подходит для представления информации о билетах, пассажирах, рейсах и других сущностях в системе. Документы могут содержать вложенные поля, массивы и другие типы данных, что облегчает работу с различными атрибутами билетов.
* Масштабируемость: *MongoDB* обладает горизонтальной масштабируемостью, позволяющей распределить данные на несколько серверов (узлов) в кластере. Это позволяет обеспечить высокую производительность и отказоустойчивость системы, особенно при большом объеме данных или интенсивной нагрузке.
* Быстрая скорость чтения и записи: *MongoDB* предлагает эффективные механизмы чтения и записи данных. Благодаря своей архитектуре, *MongoDB* может быть очень быстрой при выполнении запросов, особенно когда данные хранятся в документах, которые непосредственно соответствуют модели предметной области.
* Гибкий язык запросов: *MongoDB* предоставляет мощный язык запросов, называемый *Query Language (QL),* который позволяет выполнять разнообразные операции с данными, включая поиск, фильтрацию, агрегацию и сортировку. Это обеспечивает возможность эффективной выборки и анализа данных.
  + 1. ***SQL Server***

СУБД *SQL Server* - это мощная система управления базами данных, разработанная компанией *Microsoft*. Она предлагает широкий спектр функциональных возможностей и высокую производительность, что делает ее привлекательным вариантом для использования в различных проектах, включая систему покупки билетов на авиарейсы.

*SQL Server* обеспечивает надежное хранение, организацию и доступ к данным, а также поддерживает множество продвинутых функций для работы с базами данных.

Некоторые из основных преимуществ *SQL Server* включают:

* Масштабируемость: *SQL* *Server* позволяет эффективно масштабировать систему с увеличением объема данных и нагрузки. Он поддерживает кластеризацию серверов, репликацию данных и другие техники для обеспечения высокой доступности и производительности системы.
* Транзакционная поддержка: *SQL* *Server* обладает мощными механизмами управления транзакциями, что особенно важно для системы покупки билетов, где необходимо обеспечить целостность данных и поддержку конкурентного доступа.
* Поддержка языка *SQL*: *SQL Server* полностью поддерживает язык *SQL*, что упрощает разработку и выполнение запросов к базе данных. Это позволяет легко формировать сложные запросы для получения необходимых данных.
* Безопасность: *SQL Server* предоставляет различные механизмы безопасности, такие как роли, разрешения и шифрование данных, что обеспечивает защиту конфиденциальности и целостности информации в системе покупки билетов.
* Инструменты разработки и администрирования: *SQL Server* поставляется с разнообразными инструментами для разработки и администрирования баз данных. Это включает в себя Microsoft *SQL Server Management Studio (SSMS)* - мощное средство для создания, изменения и управления базами данных.
* Интеграция с другими продуктами *Microsoft: SQL Server* хорошо интегрируется с другими продуктами и технологиями от *Microsoft*, такими как *.NET Framework, Azure, Business Intelligence* и многими другими. Это может быть полезно для расширения функциональности системы покупки билетов и интеграции с другими системами.

## Выбор средств реализации (ЯП)

**3.2.1 *Java***

***Java* является объектно-ориентированным языком программирования, разработанным компанией *Sun Microsystems* (в настоящее время владеется компанией *Oracle*). Он известен своей платформенной независимостью и широким применением в различных областях разработки программного обеспечения.**

**Одной из ключевых особенностей *Java* является его возможность работать на различных операционных системах и архитектурах благодаря виртуальной машине *Java (JVM)*. Это позволяет разработчикам писать программы на *Java* один раз и запускать их на разных платформах без необходимости перекомпиляции. Такая платформенная независимость делает *Java* привлекательным выбором для создания кросс-платформенных приложений.**

***Java* имеет широкую поддержку различных библиотек и фреймворков, что делает его очень гибким и мощным инструментом разработки. Он позволяет создавать разнообразные типы приложений, включая веб-приложения, мобильные приложения, настольные приложения, игры и многое другое.**

***Java* также известен своей простотой и читаемостью. Синтаксис языка похож на синтаксис языка *C++,* что делает его относительно легким в изучении и использовании. *Java* также обеспечивает автоматическое управление памятью через механизм сборки мусора, что облегчает работу с памятью и предотвращает утечки памяти.**

**Другим важным аспектом *Java* является его обширная стандартная библиотека. Эта библиотека включает в себя множество классов и методов, которые предоставляют различные функциональности, такие как работа с вводом-выводом, сетевое программирование, многопоточность, работа с базами данных и другие. Благодаря этой стандартной библиотеке разработчики могут сократить время разработки и повысить производительность своих приложений.**

***Java* также активно используется в сфере предприятийской разработки. Среда выполнения *Java (JRE)* и *Java Enterprise Edition (Java EE)* предоставляют мощные средства и инфраструктуру для создания масштабируемых и надежных приложений для предприятий. *Java EE* включает в себя различные компоненты, такие как сервлеты, *JSP (JavaServer Pages)*, *EJB (Enterprise* *JavaBeans*), *JPA (Java Persistence API)* и многое другое, которые обеспечивают функциональность распределенных и масштабируемых систем.**

**В целом, *Java* является мощным и универсальным языком программирования, который предлагает широкий набор функциональности, надежность, платформенную независимость и разнообразие инструментов и библиотек. Благодаря этим особенностям *Java* является привлекательным выбором для создания масштабируемых и надежных приложений, в том числе в сфере предприятийской разработки.**

**3.2.2 *JavaScript***

***JavaScript* - это высокоуровневый интерпретируемый язык программирования, широко применяемый для разработки динамических веб-приложений. Он был создан для обеспечения интерактивности на веб-страницах и в настоящее время является одним из наиболее распространенных и популярных языков программирования веб-разработки.**

***JavaScript* предоставляет возможности для создания интерактивного поведения на веб-страницах, включая манипуляции с элементами *HTML*, обработку событий, анимации, валидацию данных, отправку запросов на сервер и динамическое обновление содержимого страницы без ее полной перезагрузки. Благодаря своей простоте и мощным возможностям, *JavaScript* стал важным инструментом для разработчиков, действующих в веб-среде.**

**Одним из основных преимуществ *JavaScript* является его широкая поддержка браузерами. Практически все популярные веб-браузеры, такие как *Chrome*, *Firefox*, *Safari* и *Edge*, предоставляют среду выполнения *JavaScript*, что позволяет разработчикам создавать веб-приложения, которые могут работать во всех современных браузерах без необходимости установки дополнительных плагинов или расширений.**

***JavaScript* является языком со слабой типизацией, что означает, что переменные могут содержать значения различных типов данных (строки, числа, объекты и т.д.). Это дает разработчикам большую гибкость при работе с данными, однако требует внимательности при обработке и проверке типов данных для избежания ошибок.**

**Благодаря активному сообществу разработчиков и огромному количеству доступных библиотек и фреймворков, *JavaScript* обладает обширной экосистемой. Это позволяет разработчикам использовать готовые решения для ускорения разработки и реализации сложных функциональностей в веб-приложениях. Некоторые из популярных библиотек и фреймворков, построенных на *JavaScript*, включают *React*, *Angular*, *Vue*.*js* и *Node.js*.**

**С появлением *JavaScript* на серверной стороне (с использованием *Node.js*), он стал также эффективным языком для разработки полноценных веб-серверов и микросервисов. *Node.js* позволяет разработчикам использовать *JavaScript* как язык программирования как на клиентской, так и на серверной стороне, что способствует созданию полного стека *JavaScript*-приложений.**

**Одним из ключевых преимуществ *JavaScript* является его простота и обучаемость. Синтаксис *JavaScript* является интуитивным и легким для понимания даже начинающим разработчикам. Это делает язык доступным для широкого круга разработчиков и способствует его популярности.**

***JavaScript* также является поддерживаемым языком со стороны сообщества и разработчиков. Существует огромное количество документации, учебных ресурсов, форумов и сообществ, где разработчики могут получить помощь, делиться знаниями и участвовать в развитии языка.**

**Благодаря своей универсальности, *JavaScript* может использоваться не только для разработки веб-приложений, но и для создания приложений для мобильных устройств, настольных приложений, игр и даже программирования интернет-вещей (*IoT*). Это расширяет возможности разработчиков и позволяет им использовать свои навыки *JavaScript* в различных областях.**

**Однако, как и у любого языка программирования, у *JavaScript* есть и свои недостатки. Один из них - это проблемы с безопасностью. Поскольку *JavaScript* выполняется на стороне клиента, существует потенциальная возможность для злоумышленников внедрять вредоносный код и осуществлять атаки на веб-приложения. Однако, с использованием соответствующих практик разработки и безопасности, эти риски могут быть сведены к минимуму.**

**В целом, *JavaScript* является мощным языком программирования, который предоставляет разработчикам средства для создания динамических и интерактивных веб-приложений. Благодаря своей популярности, обширной экосистеме и простоте использования, *JavaScript* остается одним из основных языков программирования в сфере веб-разработки и предлагает множество возможностей для инноваций и творчества.**

**3.2.3 *Python***

***Python* - это высокоуровневый, интерпретируемый язык программирования, который славится своей простотой, элегантностью и читаемостью кода. Созданный Гвидо ван Россумом в конце 1980-х годов, *Python* быстро набрал популярность благодаря своей простоте в использовании и мощным функциональным возможностям.**

**Одним из главных преимуществ *Python* является его простота в изучении и понимании. Синтаксис *Python* подобен английскому языку, что делает его читаемым и интуитивно понятным даже для новичков. Это позволяет программистам быстро начать разрабатывать программы и продуктивно работать над проектами.**

***Python* также обладает широкой стандартной библиотекой, которая включает в себя большое количество полезных модулей и функций, облегчающих разработку. Благодаря этой библиотеке, *Python* предоставляет обширный набор инструментов для различных задач, включая работу с сетью, базами данных, графикой, веб-разработку, научные вычисления и многое другое.**

**Другим значительным преимуществом *Python* является его кросс-платформенность. *Python* может работать на различных операционных системах, таких как *Windows, macOS* и *Linux*, что делает его универсальным инструментом разработки.**

***Python* также известен своей поддержкой объектно-ориентированного программирования (ООП). ООП позволяет разбивать программу на более мелкие и управляемые части, что упрощает разработку, тестирование и поддержку кода.**

**Еще одним преимуществом *Python* является его обширное сообщество разработчиков. Существует огромное количество ресурсов, форумов, библиотек и фреймворков, созданных сообществом *Python*, которые помогают разработчикам решать разнообразные задачи и ускорять процесс разработки.**

***Python* также активно используется в области науки о данных, машинного обучения и искусственного интеллекта. Библиотеки, такие как *NumPy, Pandas, TensorFlow* и *PyTorch*, предоставляют мощные инструменты для работы с данными и реализации сложных алгоритмов машинного обучения.**

**В заключение, *Python* - это мощный язык программирования, который обладает простым и понятным синтаксисом, обширной стандартной библиотекой, поддержкой объектно-ориентированного программирования и кросс-платформенностью. Он позволяет разработчикам быстро создавать функциональные и эффективные программы.**

## Реализация базы данных

В данном разделе представлена схема базы данных, которая включает в себя таблицы, поля, типы данных и диапазон допустимых значений для каждого поля.

Таблица *user\_:*

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Диапазон допустимых значений |
| *id* | *SERIAL* | Уникальные целые числа |
| *first\_name* | *VARCHAR(64)* | Строка до 64 символов |
| *last\_name* | *VARCHAR(64)* | Строка до 64 символов |
| *email* | *TEXT* | Неограниченный |
| *password* | *TEXT* | Неограниченный |
| *role* | *INTEGER* | Целые числа |
| *created\_at* | *TIMESTAMP* | Дата и время |

Таблица *airport*:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Диапазон допустимых значений |
| *id* | *SERIAL* | Уникальные целые числа |
| *code* | *VARCHAR(10)* | Строка до 10 символов |
| *country* | *VARCHAR(64)* | Строка до 64 символов |

Таблица *flight*:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Диапазон допустимых значений |
| *id* | *SERIAL* | Уникальные целые числа |
| *flight\_number* | *VARCHAR(16)* | Строка до 16 символов |
| *price* | *FLOAT* | Числа с плавающей точкой |
| *created\_at* | *TIMESTAMP* | Дата и время, не пустое |

Таблица *booking:*

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Диапазон допустимых значений |
| *id* | *SERIAL* | Уникальные целые числа |
| *customer\_id* | *INTEGER* | Целое число |
| *flight\_id* | *INTEGER* | Целое число |
| *total* | *FLOAT* | Числа с плавающей точкой |
| *created\_at* | *TIMESTAMP* | Дата и время, не пустое |

Таблица *passport\_data:*

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Диапазон допустимых значений |
| *id* | *SERIAL* | Уникальные целые числа |
| *passport\_number* | *VARCHAR(32)* | Строка до 32 символов |
| *passenger\_first\_name* | *VARCHAR(64)* | Строка до 64 символов |
| *passenger\_last\_name* | *VARCHAR(64)* | Строка до 64 символов |
| *passenger\_date\_of\_birth* | *DATE* | Дата, не пустое |

Таблица *booking\_passenger:*

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Диапазон допустимых значений |
| *booking\_id* | *INTEGER* | Уникальные целые числа |
| *passport\_id* | *INTEGER* | Уникальные целые числа |

Таблица *transfer\_:*

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Диапазон допустимых значений |
| *flight\_id* | *INTEGER* | Уникальное целое число |
| *departure\_airport\_id* | *INTEGER* | Целое число |
| *arrival\_airport\_id* | *INTEGER* | Целое число |
| *departure\_time* | *TIMESTAMP* | Дата и время, не пустое |
| *arrival\_time* | *TIMESTAMP* | Дата и время, не пустое |

## Программная реализация приложения

Функция вывода рейсов:

*def \_get\_flights(cursor: psycopg2.extensions.cursor, sql, params: tuple = tuple()) -> list[Flight]:*

# Выполнение *SQL*-запроса с указанными параметрами

*cursor.execute(sql, params)*

# Вывод информации о выполняемом *SQL*-запросе и переданных параметрах

*print(f"*Выполняется *SQL*-запрос: {*sql*} с параметрами {*params*}")

# Получение всех полученных записей из выполненного запроса

*book\_items = cursor.fetchall()*

# Вывод полученных рейсов

*print(f"*Получено рейсов: *{book\_items}*")

# Преобразование полученных записей в список объектов *Flight*, если записи существуют; иначе возвращается пустой список

*return list(map(get\_flight\_object, book\_items)) if book\_items else []*

Функция получения информации о пересадках для указанного идентификатора рейса:

*def get\_transfers\_by\_flight(cursor: psycopg2.extensions.cursor, flight\_id: int) -> list[Transfer]:*

*#* Получение информации о пересадках для указанного идентификатора рейса

*transfer\_items = cursor.fetchall()*

*#* Вывод полученных пересадок

*print(f"*Получены пересадки*: {transfer\_items}")*

*#* Преобразование полученных записей в список объектов *Transfer,* если записи существуют; иначе возвращается пустой список

*return list(map(get\_transfer\_object, transfer\_items)) if transfer\_items else []*

Функция создания нового рейса:

*def create\_one\_flight(cursor: psycopg2.extensions.cursor, flight: FlightCreateRequest) -> int:*

# Выполнение SQL-запроса для создания нового рейса с указанными параметрами

*cursor.execute(FLIGHT\_CREATE\_SQL, (flight.flight\_number, flight.price))*

# Вывод информации о выполняемом SQL-запросе и переданных параметрах

*print(f"*Выполняется *SQL*-запрос: *{FLIGHT\_CREATE\_SQL}* с параметрами *{(flight.flight\_number, flight.price)}")*

# Получение одной записи после выполнения запроса

*record = cursor.fetchone()*

# Если запись не получена, вызывается исключение *DatabaseException*

*if not record:*

*raise* *DatabaseException*

# Получение идентификатора созданного рейса из полученной записи

*flight\_id = record[0]*

# Возврат идентификатора созданного рейса

*return flight\_id*

Функция создания трансфера:

*def create\_transfers(cursor: psycopg2.extensions.cursor, flight: FlightCreateRequest, flight\_id: int) -> None:*

*#* Для каждой пересадки в списке пересадок рейса

*for transfer in flight.transfers:*

*#* Формирование параметров для SQL-запроса создания пересадки

*params = (flight\_id, transfer.departure\_airport, transfer.arrival\_airport, transfer.departure\_time, transfer.arrival\_time)*

*#* Выполнение *SQL*-запроса для создания пересадки с указанными параметрами

*cursor.execute(TRANSFER\_CREATE\_SQL, params)*

*#* Вывод информации о выполняемом *SQL*-запросе и переданных параметрах

*print(f"*Выполняется *SQL-*запрос*: {TRANSFER\_CREATE\_SQL}* спараметрами *{params}")*

*#* Возвращается *None*, так как функция не возвращает значения

*return None*

Функция обновления рейса:

*def update\_one\_flight(cursor: psycopg2.extensions.cursor, flight: Flight, flight\_id: int) -> None:*

*#* Формирование параметров для *SQL*-запроса обновления рейса

*params = (flight.flight\_number, flight.price, flight\_id)*

*#* Выполнение *SQL*-запроса для обновления рейса с указанными параметрами

*cursor.execute(FLIGHT\_UPDATE\_SQL, params)*

*#* Вывод информации о выполняемом SQL-запросе и переданных параметрах

*print(f"*Выполняется *SQL-*запрос*: {FLIGHT\_UPDATE\_SQL} с параметрами {params}")*

## Инструкция пользователя

Ниже приведен алгоритм действий пользователя при работе с готовым продуктом.

Попадая на сайт, пользователь видит главную страницу:

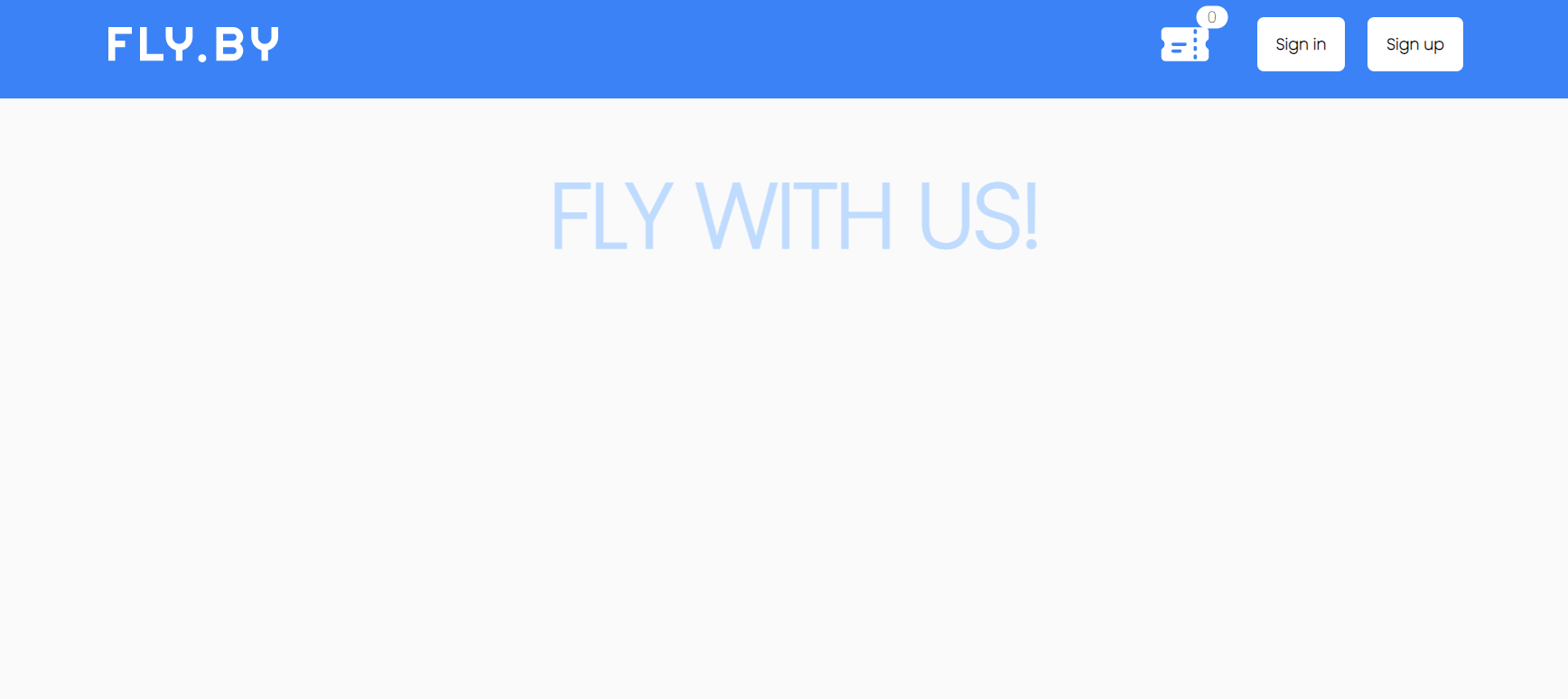


Рисунок 3.5.1 – Главная страница

Панель регистрации нового пользователя:

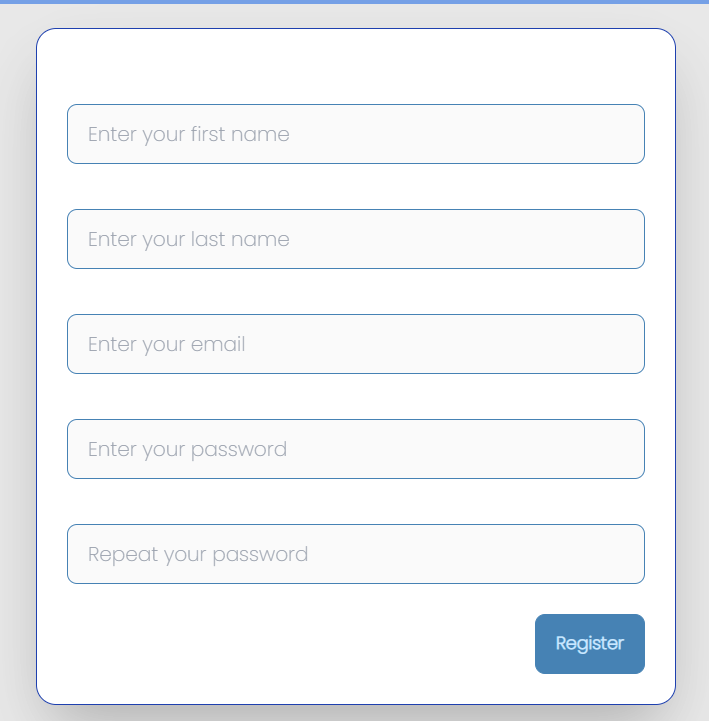


Рисунок 3.5.2 – Панель регистрации

После регистрации пользователь может аутентифицироваться используя почту и пароль:

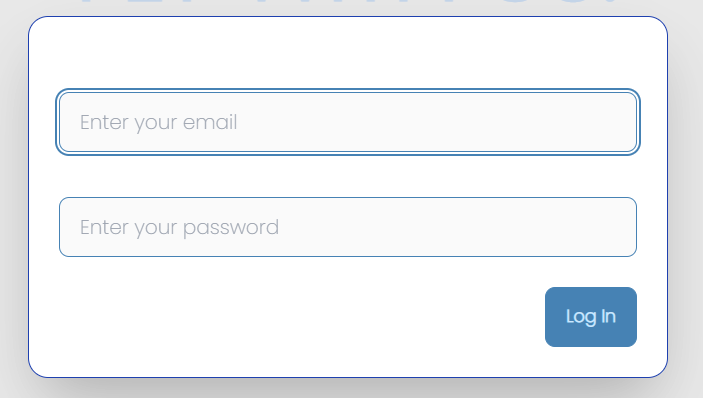


Рисунок 3.5.3 – Панель аутентификации

Если пользователь является администратором, ему доступна панель администратора:

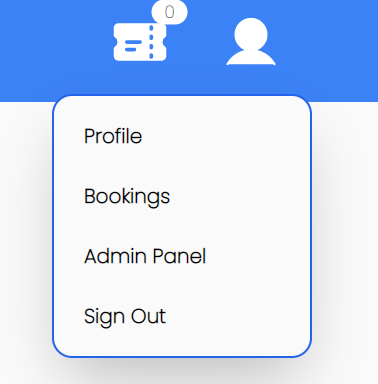


Рисунок 3.5.4 – Панель администратора

Администратор имеет доступ к просмотру всех пользователей и их броням:

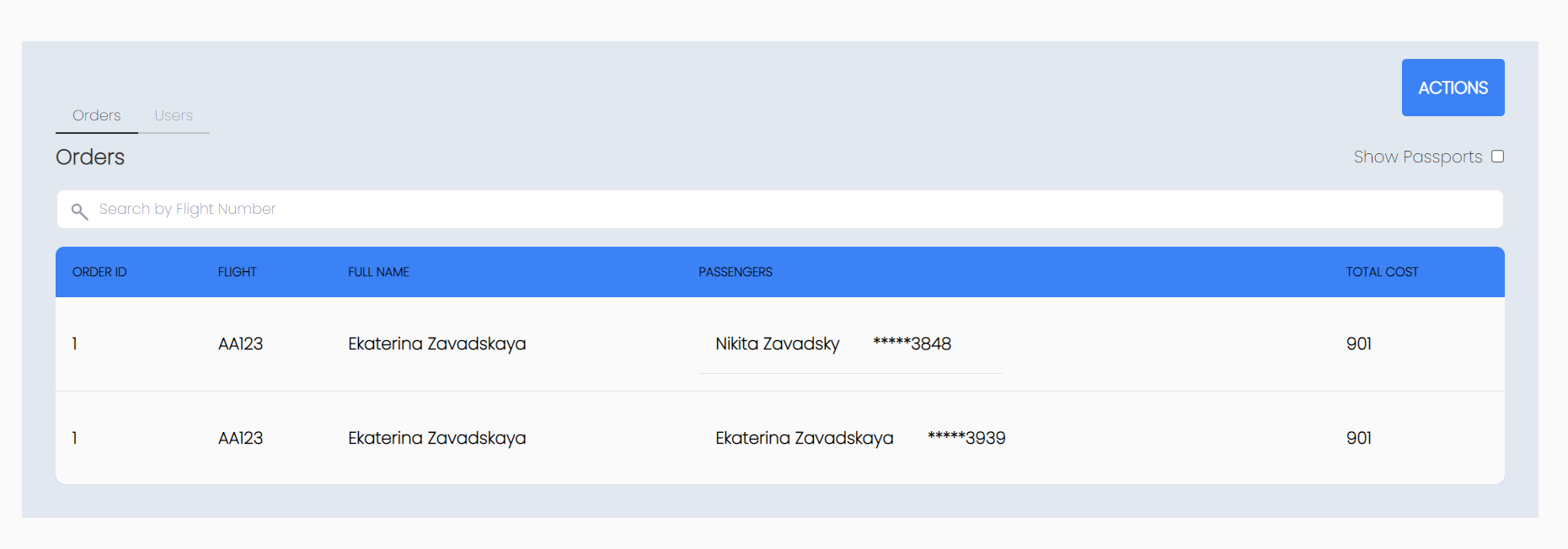


Рисунок 3.5.5 – История броней пользователей

Также администратору доступные следующие функции – скачивание аналитики, экспорт заказов и загрузка информации о рейсах:

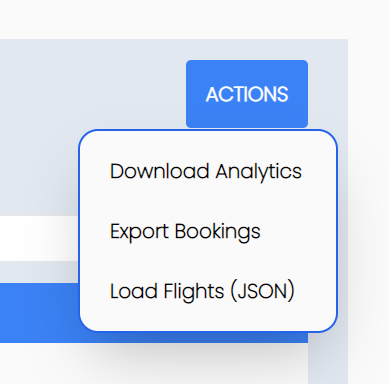


Рисунок 3.5.6 – Функционал администратора

Если же пользователь не обладает правами администратора, его функционал будет таким:

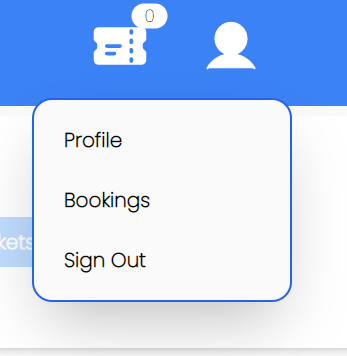


Рисунок 3.5.7 – Функционал обычного пользователя

Пользователь может просмотреть свой профиль и отредактировать данные:

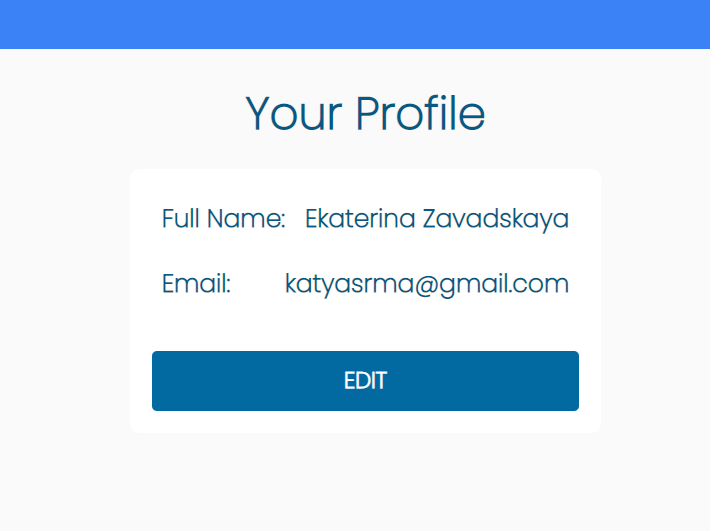


Рисунок 3.5.8 – Личный кабинет пользователя

Пользователь может искать авиабилеты, указав код страны отправления и код страны прибытия, также нужно указать дату:

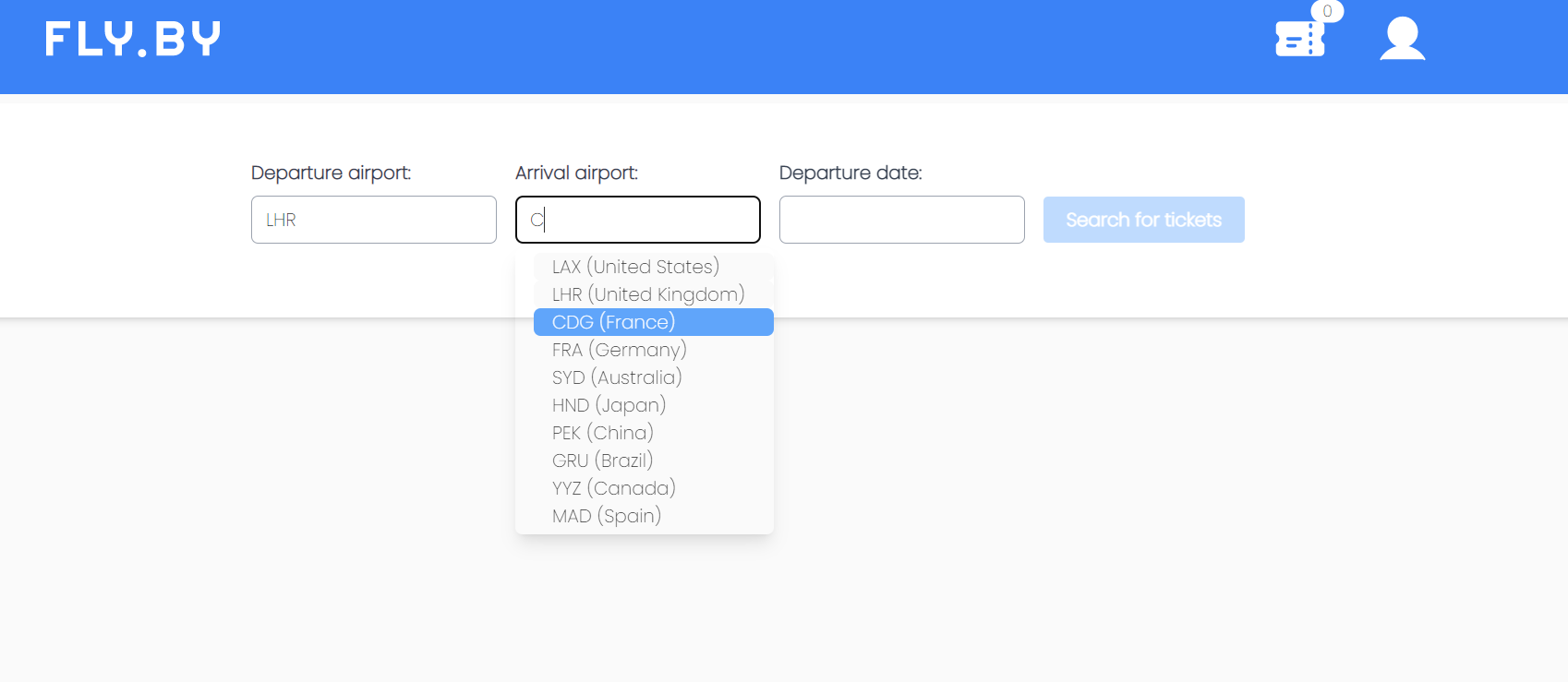


Рисунок 3.5.9 – Панель поиска авиабилетов

После выбора нужных критериев пользователю отображаются авиарейсы, максимально близкие к выбранным датам. Пользователь может добавить в корзину выбранный билет и просмотреть трансферы этого рейса, если рейс не является прямым (т.е. будут пересадки):

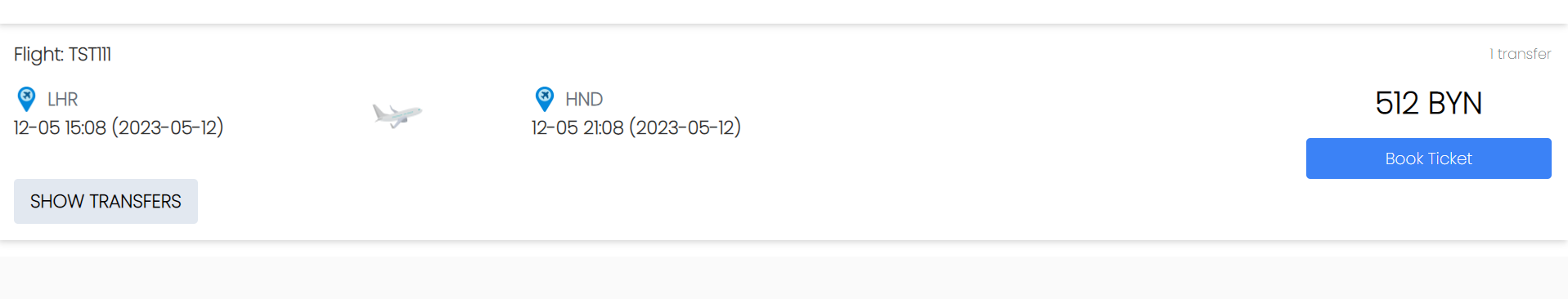


Рисунок 3.5.10 – Результат поиска авиабилетов

После нажатия кнопки *Book Ticket* пользователю нужно ввести данные пассажира. Пользователь может добавлять несколько пассажиров:

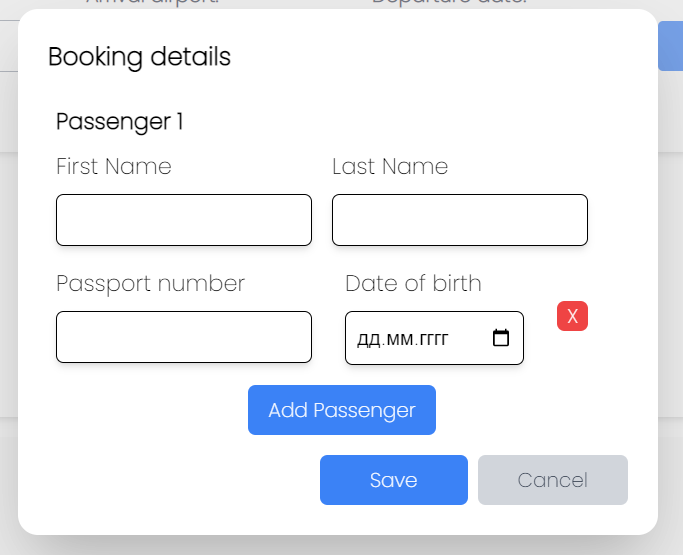


Рисунок 3.5.11 – Процесс бронирования билетов

После перехода в корзину, пользователь можно оформить заказ, удалить билет, либо полностью очистить корзину:

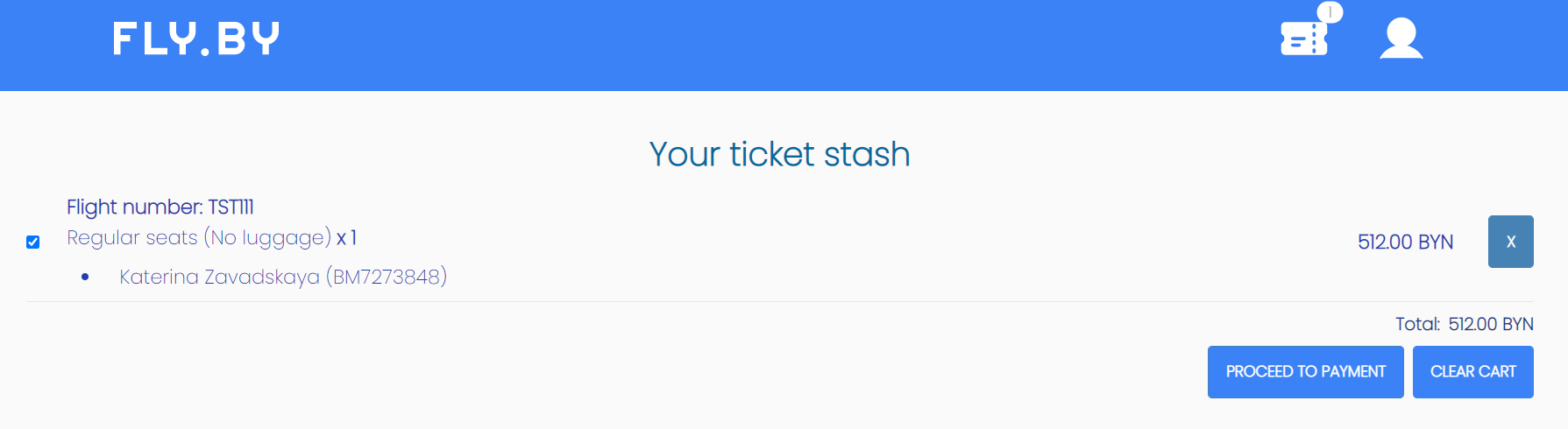


Рисунок 3.5.12 - Корзина

# Заключение

В рамках данного курсового проекта разработана система покупки билетов на авиарейсы, предназначенная для удобного и эффективного осуществления процесса бронирования авиабилетов. В ходе работы рассмотрены различные аспекты проектирования и реализации системы, включая выбор средств реализации, архитектуру, функциональность и безопасность.

Для разработки системы выбран язык программирования *Python* и система управления базами данных *PostgreSQL*. Этот выбор обоснован рядом преимуществ, которые обеспечивают эффективную реализацию системы.

В процессе разработки системы реализованы основные функциональные возможности, включая поиск и выборку доступных рейсов, бронирование билетов, управление информацией о пассажирах и рейсах. Система спроектирована с учетом удобства использования и интуитивно понятного интерфейса для пользователей.

В результате успешной реализации проекта, система покупки билетов на авиарейсы предоставляет удобный, быстрый и безопасный способ осуществления покупки и бронирования авиабилетов. Она может быть использована авиакомпаниями, агентствами по продаже билетов и другими участниками авиационного рынка для улучшения клиентского опыта и оптимизации процесса бронирования.

Пояснительная записка оформлена в соответствии с требованиями стандарта СТП 01-2017.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. PostgreSQL Documentation (https://www.postgresql.org/docs) - Документация по PostgreSQL, мощной объектно-реляционной системе управления базами данных.
2. Святослав Куликов. Реляционные базы данных в примерах – *EPAM Systems, RD Dep*, 2021. – 420 с.
3. К. Дж. Дейт. Введение в системы баз данных/ К. Дж. Дейт - Вильямс, 2017. – 1328 с.
4. Mark Lutz - "Learning Python" - Основы языка Python и его применение в различных областях.
5. Вендров А. М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000.
6. Обзор программных средств для реализации БД [Электронный ресурс] – Режим доступа: [*https://car-tesla.ru/kompjutery/obzor-programmnyh-sredstv-dlia-realizacii-bazy-dannyh*/](https://car-tesla.ru/kompjutery/obzor-programmnyh-sredstv-dlia-realizacii-bazy-dannyh/), свободный.
7. СТП 01-2017. Стандарт предприятия. Дипломные проекты(работы). Общие требования. – Минск: БГУИР, 2017. – 169 с.

# ПРИЛОЖЕНИЕ А

**(справочное)**

**Выдержка из кода**

Реализация функции регистрации нового пользователя:

*def insert\_user(cursor: psycopg2.extensions.cursor, user: UserCreateModel, role: int = UserRole.USER):*

*cursor.execute(*

*"""INSERT INTO user\_ (email, first\_name, last\_name, password, role) VALUES (%s, %s, %s, %s, %s) RETURNING id""",*

*(user.email, user.first\_name, user.last\_name, Hasher.get\_password\_hash(user.password), role),*

*)*

*if not (user\_item := cursor.fetchone()):*

*raise DatabaseException*

*print(user\_item)*

*return user\_item.id # type: ignore*

# ВЕДОМОСТЬ КУРСОВОГО ПРОЕКТА

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | *Дополнительные сведения* | | |
|  | | | | | *Текстовые документы* |  | | |
|  | | | | |  |  | | |
| *БГУИР КП 1-53 01 02 06 015 ПЗ* | | | | | *Пояснительная записка* | *50c.* | | |
|  | | | | |  |  | | |
|  | | | | | *Графический документ* |  | | |
|  | | | | |  |  | | |
|  | | | | | *ER-диаграмма в нотации Питера Чена* | *Формат А3* | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  | | | | |  |  | | |
|  |  |  |  |  | *БГУИР КП 1-53 01 02 06 015 Д1* | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Изм.* | *Л* | *№ докум.* | *Подп* | *Дата* | *Система покупки билетов на авиарейсы*  *Ведомость курсового проекта* | *Лит* | *Лист* | *Листов* |
| *Разраб.* | | *Сурма* |  |  | *Кафедра ИТАС*  *гр. 020605* | | |
| *Провер.* | | *Трофимович* |  |  |
| *Т.контр.* | |  |  |  |
| *Н.контр.* | |  |  |  |
| *Утв.* | |  |  |  |
| *Реценз.* | |  |  |  |