Титульный лист

МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ

Государственное бюджетное профессиональное образовательное учреждение Московской области «Ногинский колледж»

Курсовой проект

по МДК.09.01 Проектирование и разработка веб-приложений

ПМ.09 Проектирование, разработка и оптимизация веб-приложений

Тема:

ИСПОЛЬЗОВАНИЕ ФРЕЙМВОРКА FLASK В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЯ «БРОНИРОВАНИЕ АВИАБИЛЕТОВ»

Разработчик: студент группы 3ИСПР2

Алимов Д.Ю.

(подпись)

Оценка защиты курсового проекта

« »

Дата защиты

« » 2024

Руководитель проекта:

преподаватель Степанов С.О.

(подпись)

Ногинск, 2024

Содержание

Оглавление

[ВВЕДЕНИЕ 3](#_Toc169539403)

[1. Основная часть 4](#_Toc169539404)

[1.1Flask: Знакомство и возможности области фреймворка 4](#_Toc169539405)

[1.2 Установка Flask 4](#_Toc169539406)

[2. Репозиторий GitHub 6](#_Toc169539407)

[3. База данных и форма регистрации/авторизации 8](#_Toc169539408)

[3.1 Models.py 9](#_Toc169539409)

[3.2 Create\_db.py: 9](#_Toc169539410)

[3.3 Login.html 10](#_Toc169539411)

[3.4 Register.html 10](#_Toc169539412)

[3.5 Base.html 11](#_Toc169539413)

[4. Запросы GET и POST 12](#_Toc169539414)

[4.1 Register 13](#_Toc169539415)

[4.2 Хэширование паролей 14](#_Toc169539416)

[4.3 Index 14](#_Toc169539417)

[4.4 Profile 15](#_Toc169539418)

[5. Стиль 15](#_Toc169539419)

[6. Установка веб-приложения на общедоступный хостинг. 19](#_Toc169539420)

[ЗАКЛЮЧЕНИЕ 20](#_Toc169539421)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 21](#_Toc169539422)

[ПРИЛОЖЕНИЯ 22](#_Toc169539423)

# ВВЕДЕНИЕ

Темой курсового проекта является «Бронирование авиабилетов».

Целью данного проекта было изучение возможностей Flask в создании веб-сайта, а также практическое применение полученных знаний в создании полноценного функционального сайта.

Для достижения поставленной цели мне необходимо выполнить следующие задачи:

1. Изучить фреймворк Flask;

2. Установка Flask и специальных расширений;

3. Практическая разработка сайта с учетом недостатков.

Бронирование авиабилетов является одним из наиболее популярных видов онлайн-сервисов в современном мире. С каждым годом количество путешественников, пользующихся услугами онлайн-платформ для покупки билетов, увеличивается. Это обусловлено удобством и быстротой процесса бронирования, а также возможностью сравнения цен и услуг различных авиакомпаний.

Актуальность данной темы обусловлена постоянным ростом числа путешественников и повышением требований к качеству обслуживания

Современное состояние систем бронирования авиабилетов на рынке характеризуется разнообразием предложений от различных интернет-платформ, но не всегда они удовлетворяют все потребности пользователей. Поэтому разработка новой системы, учитывающей все актуальные требования и тенденции рынка, является важной задачей в сфере онлайн-туризма.

Этот фреймворк оказался для меня в новинку, так как я услышал о нем впервые. Он оказался достаточно популярным для современного мира, и большинство проектов таких как: Netflix, Reddit, Trivago, Samsung использовали Flask в качестве инструмента.

## 1. Основная часть

### 1.1Flask: Знакомство и возможности области фреймворка

Моё знакомство с Flask началось с изучение документации. Итак что же такое Flask? Это легковесный веб-фреймворк для языка Python, который предоставляет минимальный набор инструментов для создания веб-приложений. На нём можно сделать и лендинг, и многостраничный сайт с кучей плагинов и сервисов.

У Flask много преимуществ, которые выделяют его среди других фреймворков:

1. Простой синтаксис — это всё-таки Python;
2. Удобные шаблоны — можно быстро создавать прототипы веб-приложений;
3. Большое количество инструментов для гибкой настройки сайтов под любые нужды.

Первое что мне предстояло это подготовка перед установкой фреймворка.

Это новейшая версия Python и pip, чтобы в последствии не возникало ошибок. Моей средой разработки служила программа Visual Studio Code, так как это был более оптимальный и удобный для меня вариант.

1.2 Установка Flask

Я осуществлял установку через терминал Visual Studio Code, перед этим установив специальное расширение для Python, но перед этим нужно проверить актуальность pip и python а после этого устанавливать Flask.

pip –version

python --version

pip install Flask

Он сразу же начал загрузку и через секунду все было установлено. После этого я искал информацию о том что мне может пригодится, это привело меня к установке специальных расширений, таких как:

1. Flask – login — это расширение для управления сеансами пользователей в Flask. Оно выполняет общие задачи входа в систему, выхода из системы и запоминания сеансов пользователей в течение длительных периодов времени.

Расширение Flask-Login умеет:

* + Сохранять идентификатор активного пользователя в сеансе Flask и легко входить и выходить из него.
  + Ограничивать просмотр защищённых URL-адресов только вошедшим в систему пользователям.
  + Выполнять функцию «запомнить меня».
  + Защищать сеансы пользователей от кражи файлов cookie.

1. Flask – admin - это расширение для микрофреймворка Flask, предназначенное для упрощённого построения административной панели приложения с поддержкой множества различных ORM. Не используйте тег для вопросов, связанных с администрированием flask-приложений в целом.
2. Flask – SQLAlchemy - это расширение для Flask, упрощающее использование SQLAlchemy внутри приложений на основе Flask. SQLAlchemy – мощный фреймворк для работы с реляционными базами данных, поддерживающий множество разных баз данных. Он предлагает высокоуровневые функции объектно-реляционного отображения инизкоуровневый доступ на языке SQL
3. Flask – request - создает объект запроса при каждом обращении к URL приложения, который, в свою очередь, запоминает совпадающую конечную точку endpoint и аргументы, передаваемые в функцию-представление.

Класс flask.Request является подклассом werkzeug.wrappers.Request и предоставляет все атрибуты, определенные Werkzeug, плюс несколько специфичных для Flask.

1. Flask – werkzeug - это набор библиотек, которые можно использовать для создания WSGI-совместимых веб-приложений на Python.

WSGI (Web Server Gateway Interface) — это интерфейс между веб-сервером и веб-приложением на базе Python. Он необходим для веб-приложений на Python, поскольку веб-сервер не может напрямую взаимодействовать с Python.

Иными словами, Werkzeug предоставляет набор утилит для создания приложения на Python, которое может взаимодействовать с сервером WSGI, например, с Gunicorn.

Werkzeug предоставляет следующую функциональность, используемую Flask:

* Обработка запросов;
* Обработка ответов;
* Маршрутизация URL;
* Middleware;
* HTTP-утилиты;
* Обработка исключений.

## 2. Репозиторий GitHub

Далее была поставлена задана создать репозиторий в GitHub что я

успешно сделал и уже там я создал проект в котором хранился и дополнялся код.

Всё началось с создание папок, были созданы такие папки как:

* app.py – в котором хранились все связи и все установленные расширения;
* models.py – для моделей страниц;
* templates – для html страниц;
* static – для стилей html страниц;
* instance – для хранения БД;
* create\_db.py – для создания БД.

В app.py я подключал расширения чтобы они связывались c остальным кодом и правильно функционировали. Большая часть времени уходила на то чтобы понять, а что именно мне нужно писать? В этом мне помогли видео уроки и теоретические материалы взятые из интернета. Первой моей целью было заставить сайт работать, т.к. было множество ошибок в коде которые я упускал из виду, и на которые потребовалось много времени для их решения:

Первой ошибкой с которой я столкнулся был сам Python. У меня был указан неправильный интерпретатор в настройках компьютера. Мне не составило много проблем решить эту проблему.

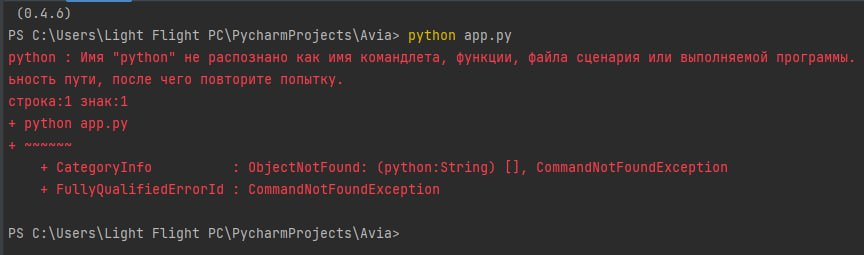


Рисунок 2.1 – Ошибка связанная с отсутствием Python

Второй ошибкой был pip который конфликтовал с версией Python.

Рисунок 2.2 – Ошибка связанная с отсутствием интерпретатора

Третьей ошибкой была неправильная загрузка дополнений Flask из-за который постоянно вылезало оповещение о том, что интерпретатор не подключен. Эта ошибка висела довольно продолжительное время, но с помощью интернета я успешно ее устранил.

Четвертой ошибкой был неправильное подключение кода, из-за этого сайт долго не хотел высвечиваться, и постоянно жаловался на версию pip и Python, хотя в них вообще проблем небыло. Из-за этого ошибка также держалась долго пока я не понял что проблема в коде.

Пятой и последней ошибкой был неправильный синтаксис и некоторые значения кода, где в одном случае я писал user а в другом user, из-за этого код конфликтовал. А я не мог увидеть ошибку.

Наконец-то спустя продолжительное время сайт запустился, но на нем было пусто, так как я ничего еще не сделал.

## 3. База данных и форма регистрации/авторизации

Я использовал базу данных SQLAlchemy, сама БД представляет из себя Python-библиотеку, которая позволяет работать с реляционными базами данных с помощью ORM.

Реляционные базы данных хранят информацию в виде связанных между собой таблиц. К таким базам относят Oracle, Microsoft SQL Server, PostgreSQL и другие. Реляционные базы данных работают по принципам реляционной алгебры, но это тема для отдельной статьи.

Сила SQLAlchemy — в её **ORM**. Расшифровывается как **object relational mapper**, или **«объектно-реляционное отображение»**. ORM позволяет управлять базами данных с помощью методов объектов в коде и при этом не использовать SQL-запросы. На самом деле это очень удобно, так как позволяет писать привычный код, не переключаясь на SQL.

Эта бд показалась мне очень удобной т.к. у нее много хороших особенностей, таких например как то, что мы можем заменить базу данных в любой момент не ломая код. Но с ее подключением также возникло много трудностей. Таких как неправильно поданная информация из интернета которая давала неправильные указания. И мне потребовалось достаточно много времени чтобы понять как работают папки и что в них нужно писать .

Итак чтобы создать функциональную бд мне потребовалось 2 основных файла: models и create\_db.

Файл models отвечал за вид самой БД, что там будет и с какими параметрами.

Файл create\_db отвечал за функцию создания. То есть с каждым запуском из папки models брались данные и автоматически создавали БД. Не надо было ничего прописывать самостоятельно в терминале, это очень удобно. Но чтобы это понять у меня ушло очень много проб и ошибок.

3.1 Models.py:

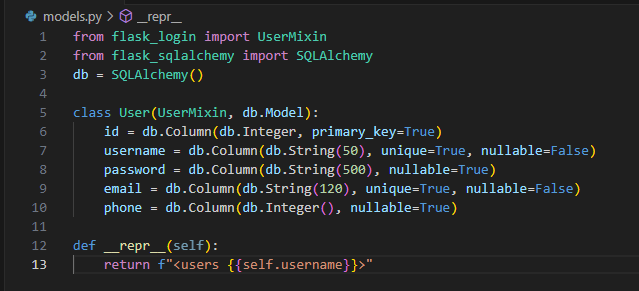


Рисунок 3.1.1 – Модуль models.py

### 3.2 Create\_db.py:

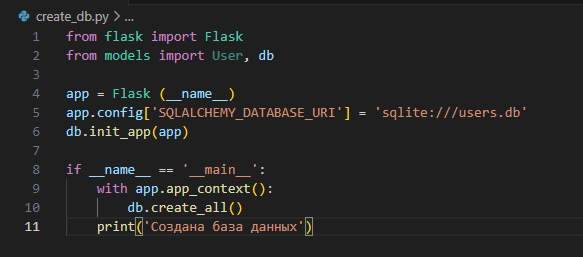


Рисунок 3.2.1 – Модуль create\_db.py

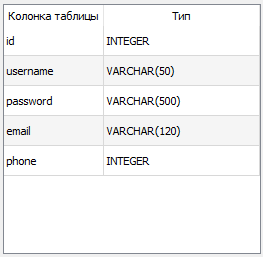


Рисунок 3.2.2 – Строение базы данных.

Далее мне надо было примерно представить как должен будет выглядеть сайт, поэтому я принялся писать шаблоны для сайтов.

Все началось с формы Регистрации/Авторизации. Я четко знал что там будет имя и номер телефона, поэтому я искал информацию как правильно писать на flask соблюдая синтаксис во избежании ошибок. Flask использует не стандартный синтаксис используя {% %}, тем самым осуществляя подключение которое зависит от base.html. Без него сайт не будет видеть шаблоны и сайт будет выдавать ошибки. Что и случалось у меня пару раз, но я сразу исправил все недочеты. Мой шаблон авторизации выглядел так:

3.3 Login.html

Из кода ниже мы можем сделать выводы что на сайте логина у нас должны отображаться ячейки имени, телефона и пароля. Довольно стандартное окно логина.



Рисунок 3.3.1 – Шаблон авторизации.

Теперь давайте перейдем к регистрации, она не менее важна тут так как через нее осуществляется занесение данных в базу данных. Вот так он выглядит:

3.4 Register.html

Как мы можем заметить в обоих фрагментах кода присутствует строчка {% extends "base.html" %}. Именно она осуществляет подключение к base.html, самому важному файлу в папке templates. Именно через этот файл осуществляется подключение к шаблонам html.



Рисунок 3.4.1 – Шаблон регистрации.

Я не буду показывать полностью код base.html так как он очень большой, но я выделю пару строчек и подробно распишу их. Начнем со строчек

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet"> - эта строчка кода подключает CSS-файл Bootstrap версии 5.2.3 к моему HTML-документу, что позволяет использовать все стили и компоненты, предоставляемые этим CSS-фреймворком.

3.5 Base.html

В этом коде мы можем увидеть подключение которое предоставляет доступ к кнопкам навигации, их всего 4 но функционал выполняют как часы. Это был функционал HTML-шаблонов.

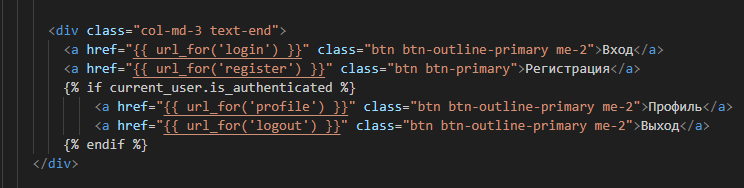


Рисунок 3.5.1 – Кнопки навигации.

Чтобы наш login и register работали нужно было прописать им соответсвующие пути и осуществить подключение к БД. На этом этапе у меня было достаточно много ошибок так как все источники давали разную информацию, приходилось обращаться к видеоурокам которые также были не сильно эффективными. Поэтому я засел на довольно продолжительное время собирая информацию и пробуя различный код. Спустя время я собирая по частичкам понял строение функции и у меня получилась связь которая заработала. Вот так выглядит связь в login:

## 4. Запросы GET и POST

GET — метод для чтения данных с сайта. Например, для доступа к указанной странице. Он говорит серверу, что клиент хочет прочитать указанный документ. На практике этот метод используется чаще всего, например, в интернет-магазинах на странице каталога. Фильтры, которые выбирает пользователь, передаются через метод GET.

POST — метод для отправки данных на сайт. Чаще всего с помощью метода POST передаются

@app.route('/login', methods=['GET', 'POST']) - это декоратор, который определяет маршрут "/login" в веб-приложении. Этот маршрут будет обрабатывать два типа HTTP-запросов: GET и POST.

def login(): - это функция, которая будет вызываться, когда клиент обращается к маршруту "/login". Внутри этой функции будет реализована логика обработки запросов.

if request.method == 'POST': - это условие, которое проверяет, был ли текущий запрос отправлен методом POST. Это означает, что пользователь отправил форму на этот маршрут.

Таким образом, этот код определяет маршрут "/login", который будет обрабатывать как GET-запросы (например, для отображения формы входа), так и POST-запросы (для обработки отправленной формы). Когда пользователь отправляет форму на этот маршрут, код внутри условия if request.method == 'POST': будет выполнен.

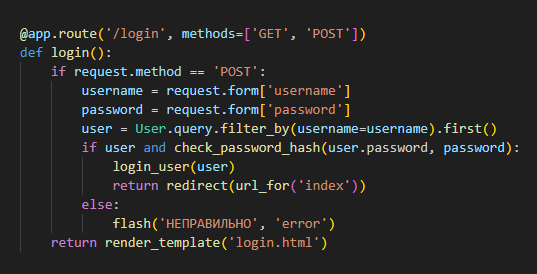


Рисунок 4.1 – Связь авторизации.

### 4.1 Register

username = request.form['username'], password = request.form['password'], email = request.form['email'], phone = request.form['phone']: - Эти строки извлекают данные, введенные пользователем в форму регистрации.

User = User(username=username, password=generate\_password\_hash(password), email=email, phone=phone): - Создается новый объект User с введенными пользователем данными.

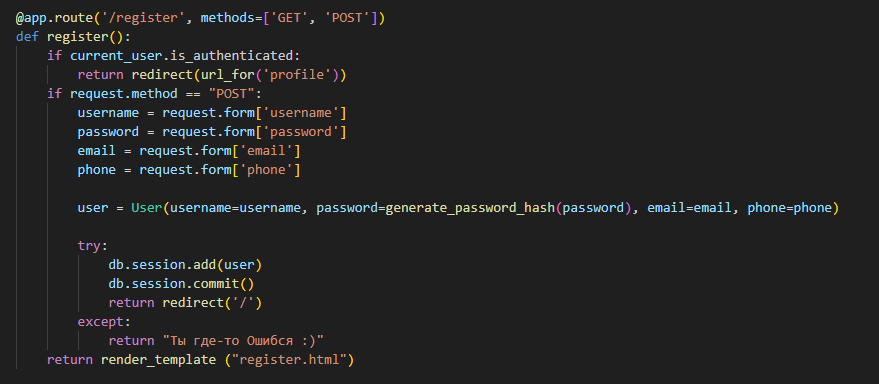


Рисунок 4.1.1 – Связь регистрации.

### 4.2 Хэширование паролей

Пароль пользователя хэшируется с помощью generate\_password\_hash() для безопасного хранения. Пароли никогда не должны храниться в виде чистого текста в базе данных. Если так делать, злоумышленник, способный взломать базу данных, получит возможность узнать и пароли, и электронные адреса. Известно, что люди используют один и тот же пароль для разных сайтов, а это значит, что одна комбинация откроет злоумышленнику доступ к остальным аккаунтам пользователей.

Вместо хранения паролей прямо в базе данных, нужно сохранять их хэши. Хэш — это строка символов, которые смотрятся так, будто бы были подобраны случайно.

“pbkdf2:sha256:50000$Otfe3YgZ$4fc9f1d2de2b6beb0b888278f21a8c0777e8ff980016e043f3eacea9f48f6dea”

Хэш создается с помощью односторонней функции хэширования. Она принимает длину переменной и возвращает вывод фиксированной длины, которую мы и называем хэшем. Безопасным хэш делает тот факт, что его нельзя использовать для получения изначальной строки (поэтому функция и называется односторонней). Тем не менее для одного ввода односторонняя функция хэширования будет возвращать один и тот же результат.

try: ... except:: - Этот блок пытается добавить нового пользователя в базу данных и сохранить изменения. Если возникает ошибка, будет возвращено сообщение "Ты где-то Ошибся :)"

### 4.3 Index

Данная часть кода отвечает за связь с главной страницей.

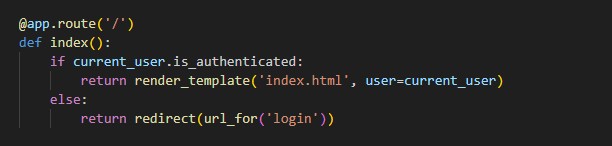


Рисунок 4.3.1 – Связь регистрации.

### 4.4 Profile

Этот код отвечает за связь с профилем пользователя.

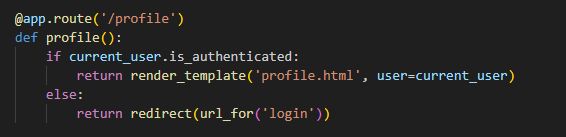


Рисунок 4.4.1 – Связь регистрации.

## 5. Стиль

Функции у сайта готовы, но я не могу оставить их в таком виде. Поэтому я посчитал нужным обратится к стилю.

Я использовал Bootstrap как фундамент. Что же такое Bootstrap и где он используется?

Bootstrap — это открытый и бесплатный HTML-, CSS- и JS-фреймворк, который используют веб-разработчики для быстрой верстки адаптивных дизайнов сайтов и веб-приложений. Включает в себя CSS- и HTML-шаблоны оформления для веб-форм, меток, типографики, кнопок, блоков навигации и других компонентов веб-интерфейса.

Фреймворк используют frontend-, fullstack-разработчики. Для работы с инструментом нужны минимальные знания верстки, поэтому он подходит новичкам.

Bootstrap используется, когда:

* у сайта много страниц;
* страницы собраны из простых базовых элементов — кнопок или таблиц;
* не будет глобального редизайна;
* шаблонность страниц окупается скоростью внедрения.

По сути, Bootstrap — это набор файлов. После их подключения к странице для верстки станет доступно большое количество готовых компонентов и классов. Они позволяют быстро и качественно создавать адаптивный дизайн сайта.

Классы в Bootstrap делятся на 3 большие группы:

* для создания сетки — адаптивного макета страницы;
* для стилизации контента — текста, изображений, кода, таблиц и прочей информации;
* служебные — для решения популярных вспомогательных задач, таких как отображение и скрытие элементов, выравнивание текста на странице, настройка цвета фона, отступов и пр.

Кроме классов, в Bootstrap есть компоненты (готовые объекты интерфейса).

Фреймворк дает готовые решения, которые позволяют создавать макеты сайтов быстрее. Для быстрого запуска проектов есть множество примеров. Для верстки любых прототипов (альбомов, слайдеров, панели входа и т. д.) потребуются файлы index.html, которые находятся в каждой папке с примерами, и соответствующий CSS-файл. Например, base.css. Изменять код можно по своему усмотрению, внося правки в текстовом редакторе. Bootstrap позволяет скопировать код из примера и вставить его в свой проект, к которому подключен фреймворк.

Чтобы внедрить Bootstrap в свой проект я использовал строчку:

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">

Она осуществляет подключение стиля к файлу base.css и создает готовый шаблон насайте по которому пользователю будет удобно ориентироваться на сайте.

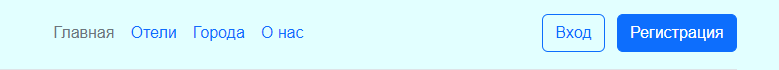


Рисунок 5.1 – Навигация на сайте.

С этим разобрался, теперь мне надо было придать внешний вид авторизации. Для начала я сделал тело страницы монотонным цветом чтобы различать где какие обьекты находятся.

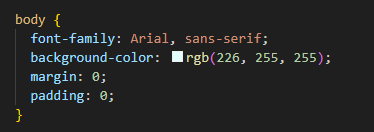


Рисунок 5.2 – Тело сайта.

Выглядела страница вполне привлекательно, но это было только началом, потому что мне надо было придумать и правильно расписать вход. С этим возникли некоторые трудности так как обьекты не хотели вставать так как я задумал, но спустя время у меня получилось более точная картинка.

Особенность этого кода в том что он подстраивается сразу на все страницы, что упростило проект

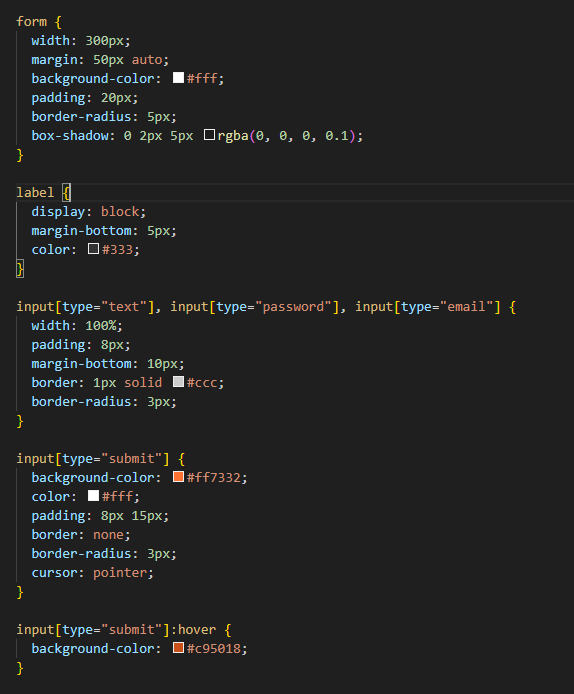


Рисунок 5.3 – Стиль сайта.

В итоге у меня получились данные страницы логина и входа.

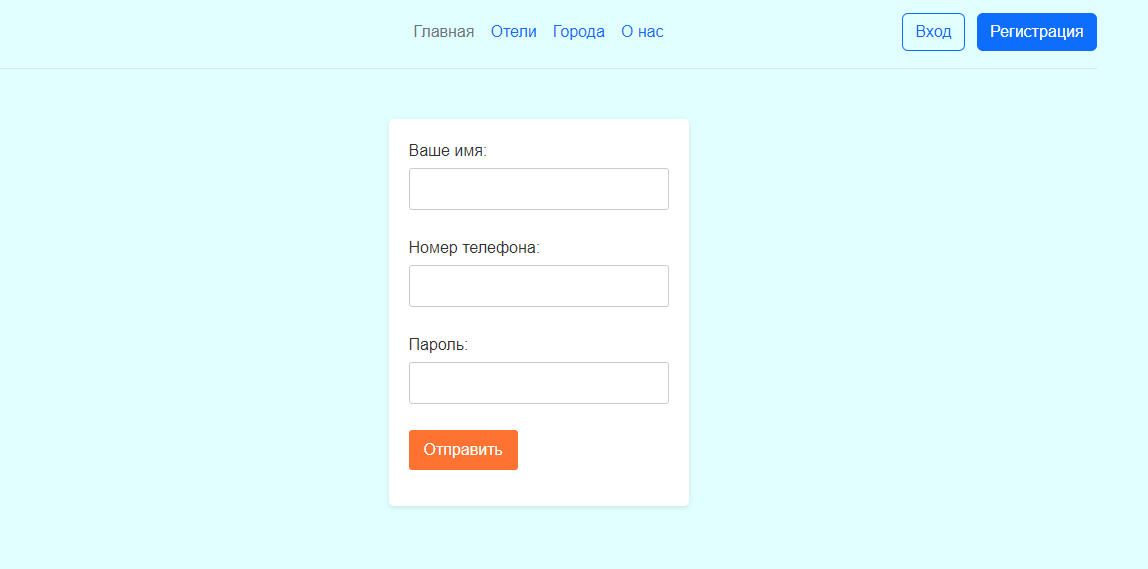


Рисунок 5.4 – Страница входа.

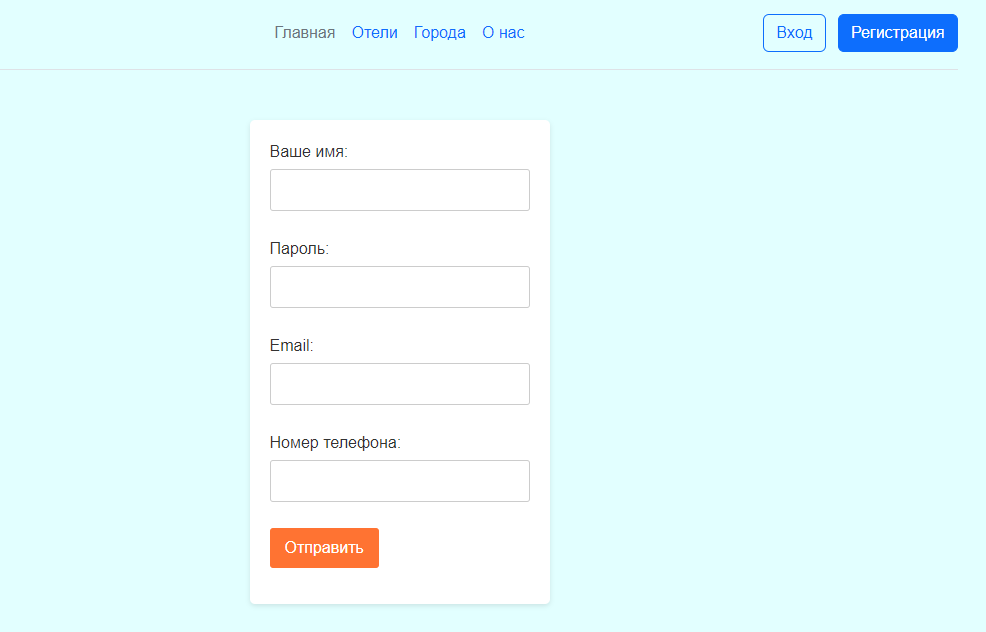


Рисунок 5.5 – Страница регистрации.

Для меня это показалось прекрасным результатом, поэтому я не решил не останавливаться и сделать весь сайт с похожим видом, чтобы глаза не уставали от неожиданного изменения стиля.

Я решил продолжить писать стиль для главной страницы и как с логином у меня возникли трудности в позиционировании и с формой даты, так как я прежде не затрагивал Flask меня немного поставила эта задача в ступор, потому что я не знал как правильно поставить форму. Мне выдавало ошибки с тем что я неправильно указываю синтаксис и он не может синхронизировать данные. Но спустя время у меня получилась вполне привлекательная форма даты.

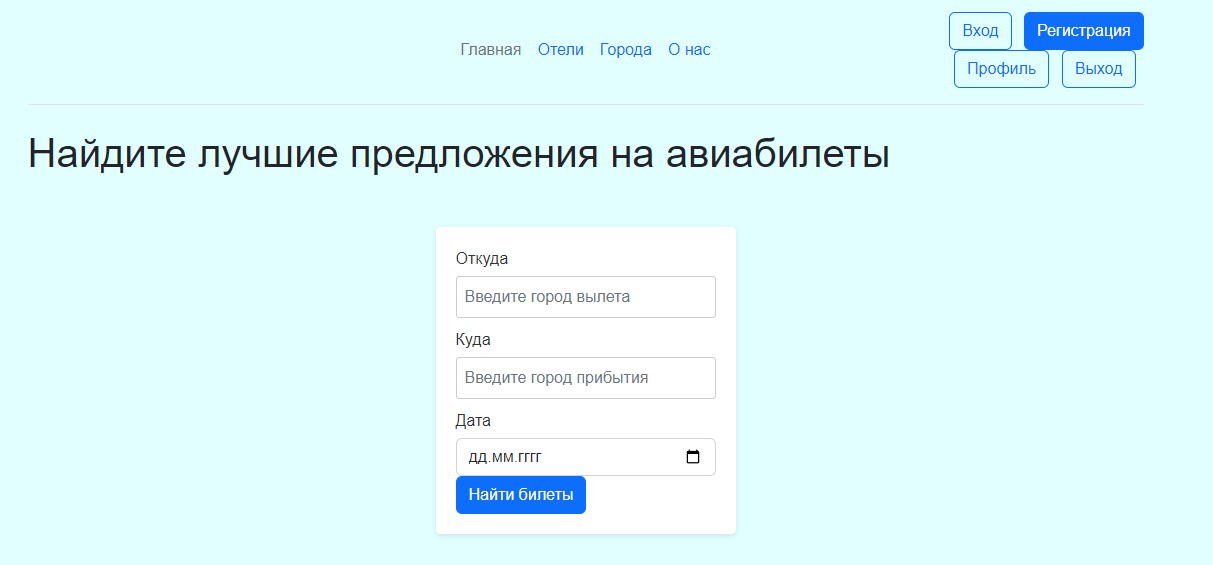


Рисунок 5.6 – Главная страница.

Как вы можете заметить надпись стоит немного не ровно, у меня были проблемы с этим которые так и не получилось исправить. Но это не проблема, ведь мое изучение стилей на этом не заканчивается и я обязательно смогу воплотить сайты именно так как хочу.

Далее страница профиля. Тут у меня возникала куча проблем с тем как их правильно расположить и с отображением купленных билетов, это было проблематично с тем что функции иногда конфликтовали со стилем, но спустя время у меня получилось придать странице адекватный вид.

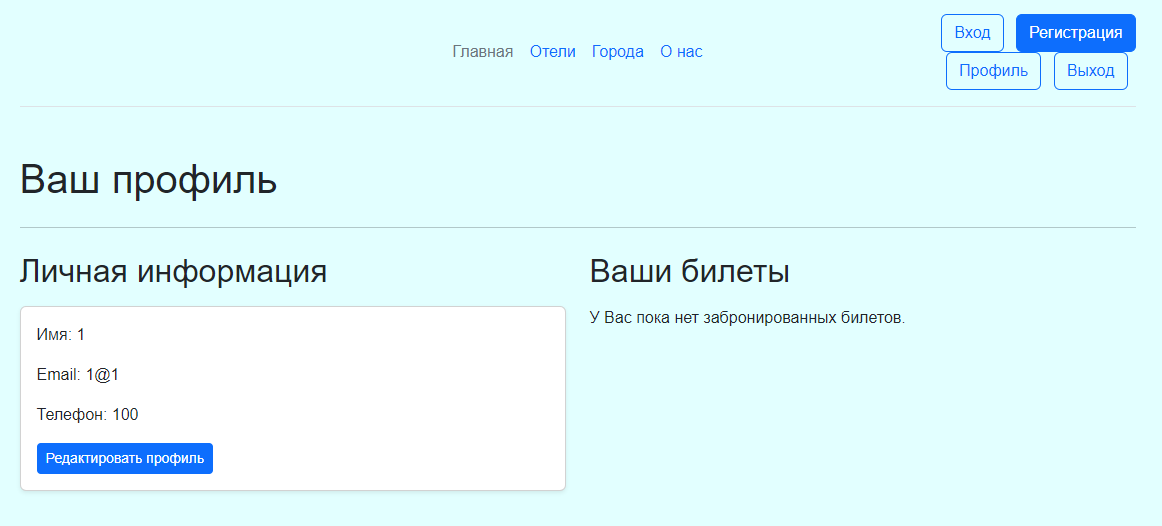


Рисунок 5.7 – Страница профиля.

## 6. Установка веб-приложения на общедоступный хостинг.

В web-разработке процесс деплоя играет важную роль. Это ответственный момент, когда все усилия, потраченные на написание и тестирование кода, воплощаются в “живое” приложение, доступное пользователям. Ведь для этого приложение и делается, чтобы им кто-то пользовался. Каждый разработчик, независимо от уровня и специализации, регулярно сталкивается с задачей deploy.

Deploy - это не просто загрузка кода на сервер. Прогресс не стоит на месте, приложения стали сложнее и сейчас deploy - это сложный и многоступенчатый процесс, который обычно включает в себя подготовку окружения, установку зависимостей, миграцию базы данных, минимизацию и компиляцию ассетов, проверку работоспособности и, наконец, переключение трафика на новую версию приложения.

Важность этого процесса не может быть недооценена. От качества и эффективности деплоя зависит стабильность работы приложения, удовлетворенность пользователей и, в конечном итоге, успех всего проекта.

Особенно важным является настройка deploy для проектов, которые часто обновляются, и требуют частого развёртывания на сервере.

# ЗАКЛЮЧЕНИЕ

Этот курсовой проект принес немало опыта для меня. Я впервые попробывал писать на языке Flask о котором раньше вообще ничего не знал. Я надолго запомнил этот язык из-за его специфичного синтаксиса без которого код не будет действовать. Мне понравилось пробывать новое и разбираться в этом чтобы в конце предоставить функциональный сайт в котором имеется форма авторизации, регистрации а также главная страницы с бронированием авиабилетов, профиль в котором отображается заполененные вами данные при регистрации, а также купленные билеты.

Для меня было интересно изучать работу с базой данных SQLite, создавать и работать с различными запросами для получения и обновления информации. Также было приятно узнать, как работать с шаблонизатором Jinja2 для создания красивого и динамического контента на страницах сайта.

В процессе создания проекта я столкнулся с некоторыми сложностями, такими как правильное организация кода и файлов, обработка и проверка данных из форм, а также работа с сессиями пользователей. Но благодаря стараниям и усердной работе, я смог преодолеть все трудности и завершить проект успешно.

Я уверен, что полученный опыт и навыки пригодятся мне в дальнейшем при работе над другими проектами и задачами. Этот курсовой проект был не только хорошим упражнением, но и отличной возможностью применить полученные знания на практике.

В процессе работы над проектом я не только улучшил свои навыки программирования на языке Python, но и научился планировать и структурировать проекты более эффективно. Проект также помог мне освоить принципы работы с веб-приложениями и понять, как взаимодействуют сервер и клиент при обработке запросов.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Natkaida, Самоучитель по Python для начинающих. Часть 23: Основы веб-разработки на Flask / Natkaida [Электронный ресурс] // proglib.io : [сайт]. —  URL: https://proglib-io.turbopages.org/proglib.io/s/p/samouchitel-po-python-dlya-nachinayushchih-chast-23-osnovy-veb-razrabotki-na-flask-2023-06-27 (дата обращения:23.04.2024).
2. Дударь Г. Изучение Flask для начинающих / Дударь Г. [Электронный ресурс] // Youtube.com : [сайт]. — URL: https://www.youtube.com/watch?v=jgAVGtkk03Q&list=PL0lO\_mIqDDFXiIQYjLbncE9Lb6sx8elKA (дата обращения: 25.04.2024).
3. Зверев Д. Фреймворк Flask: как он работает и зачем нужен / Зверев Д. [Электронный ресурс] // Skillbox.ru : [сайт]. — URL: https://skillbox.ru/media/code/freymvork-flask-kak-on-rabotaet-i-zachem-nuzhen/ (дата обращения: 28.04.2024).
4. Шкляр Е. Чем отличаются HTTP-методы GET и POST / Шкляр Е. [Электронный ресурс] // Htmlacademy.ru : [сайт]. — URL: https://htmlacademy.ru/blog/php/get-vs-post (дата обращения: 11.05.2024).
5. Ронахер А. Руководство пользователя / Ронахер А. [Электронный ресурс] // flask.palletsprojects.com : [сайт]. — URL: https://flask.palletsprojects.com/en/3.0.x/ (дата обращения: 14.05.2024).
6. Панов И. Интерфейс управления файлами / Панов И. [Электронный ресурс] // stackoverflow.com : [сайт]. — URL: https://ru.stackoverflow.com/tags/flask-admin/info (дата обращения: 15.05.2024).
7. Джигир Н. Управление базой данных с помощью Flask-SQLAlchemy / Джигир Н. [Электронный ресурс] // github.io : [сайт]. — URL: https://nikitadzhigir.github.io/Lessons/lesson\_4.html (дата обращения: 19.05.2024).
8. Алдрисон А. Что такое Werkzeug? / Алдрисон А.Н. [Электронный ресурс] // pythonist.ru : [сайт]. — URL: https://pythonist.ru/chto-takoe-werkzeug/ (дата обращения: 20.05.2024).
9. Bootstrap / [Электронный ресурс] // skillfactory : [сайт]. — URL: https://blog.skillfactory.ru/glossary/bootstrap/ (дата обращения: 28.05.2024).

# ПРИЛОЖЕНИЯ

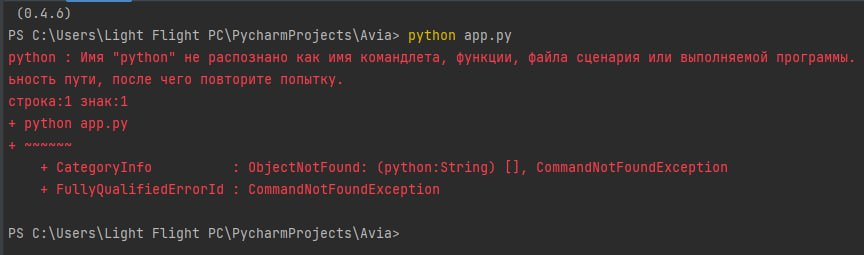


Рисунок 2.1 – Ошибка связанная с отсутствием Python

Рисунок 2.2 – Ошибка связанная с отсутствием интерпретатора

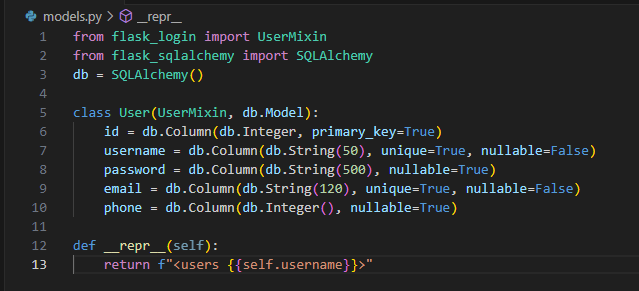


Рисунок 3.1.1 – Модуль models.py

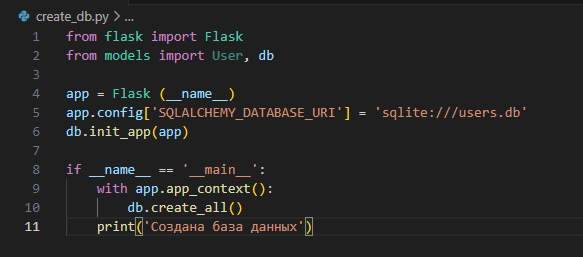


Рисунок 3.2.1 – Модуль create\_db.py



Рисунок 3.3.1 – Шаблон авторизации.



Рисунок 3.4.1 – Шаблон регистрации.

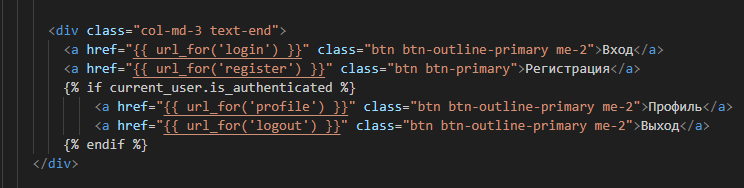


Рисунок 3.5.1 – Кнопки навигации.

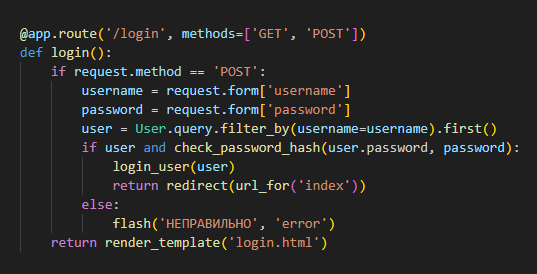


Рисунок 4.1 – Связь авторизации.

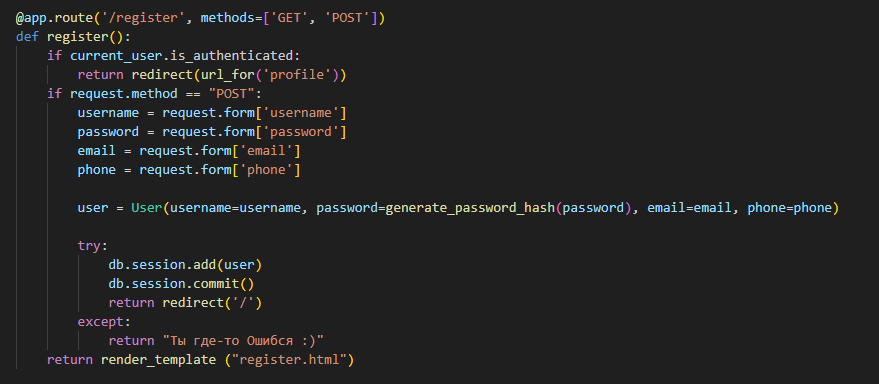


Рисунок 4.1.1 – Связь регистрации.

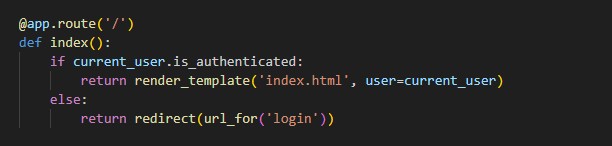


Рисунок 4.3.1 – Связь регистрации.

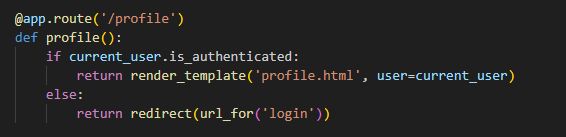


Рисунок 4.4.1 – Связь регистрации.

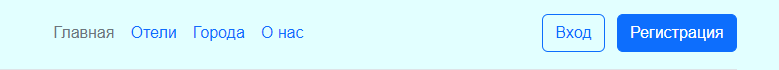


Рисунок 5.1 – Навигация на сайте.

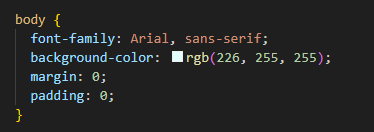


Рисунок 5.2 – Тело сайта.

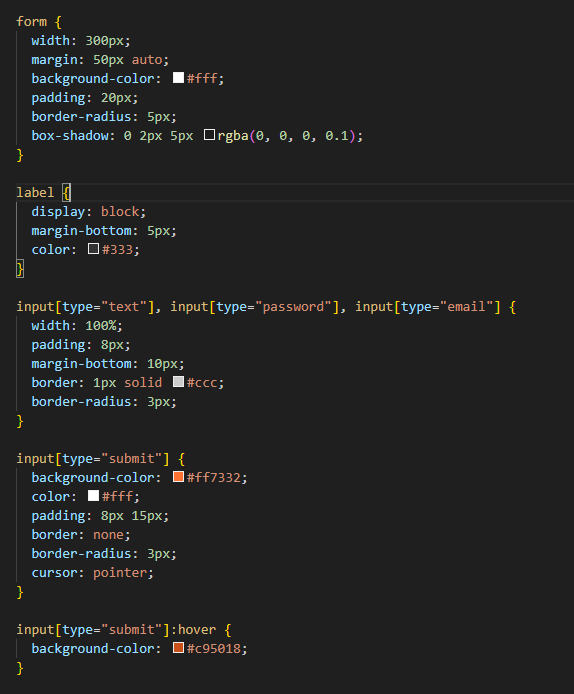


Рисунок 5.3 – Стиль сайта.

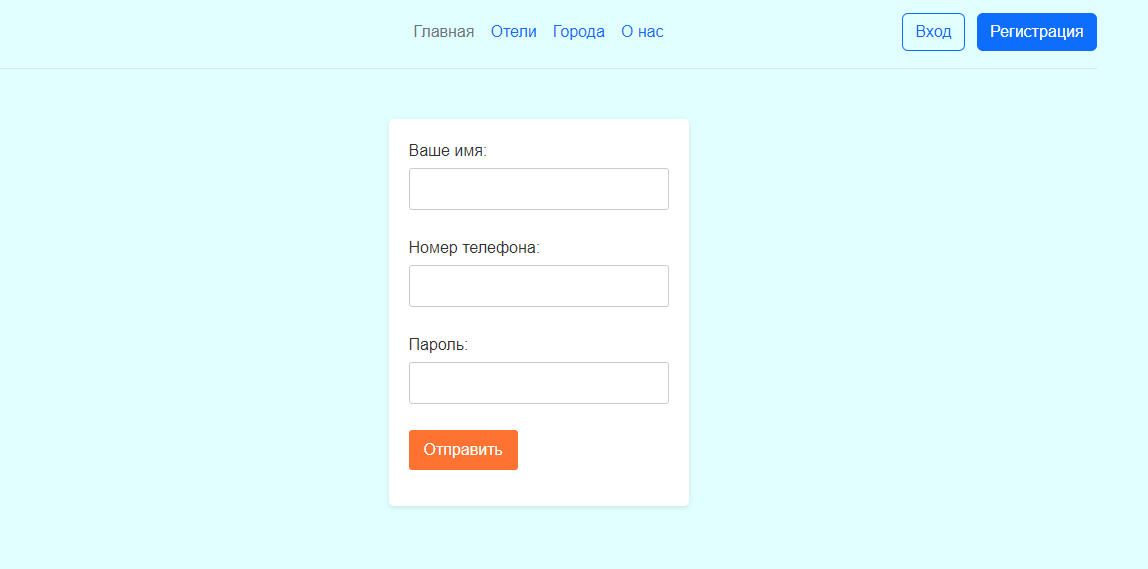


Рисунок 5.4 – Страница входа.

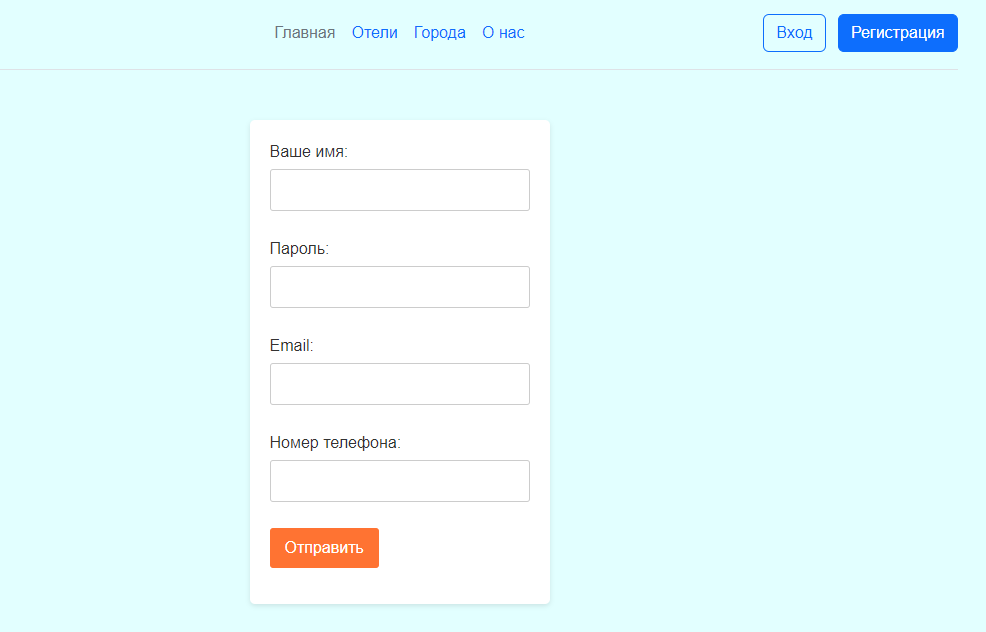


Рисунок 5.5 – Страница регистрации.

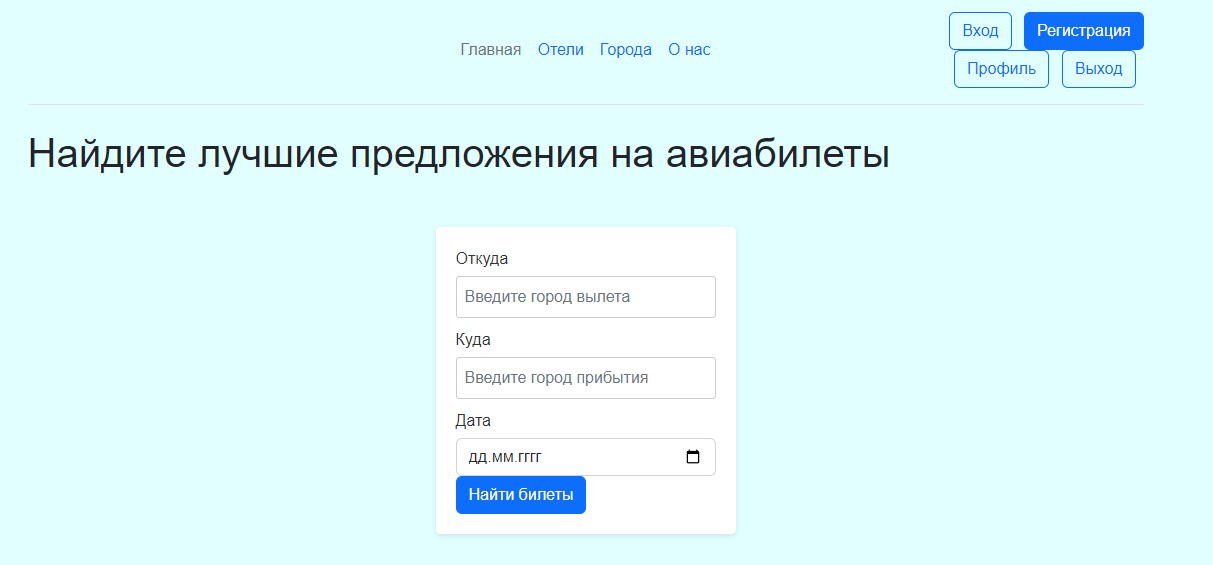


Рисунок 5.6 – Главная страница.

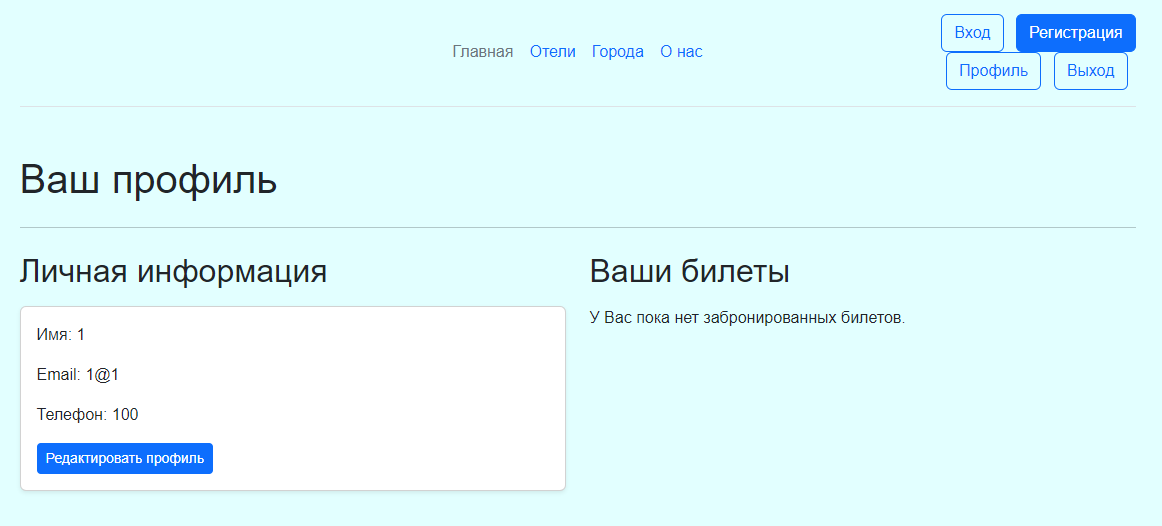


Рисунок 5.7 – Страница профиля.