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

# 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.

# ОСНОВНАЯ ЧАСТЬ

## ГЛАВА 1. FLASK: 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.

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:

* Flask – login;
* Flask – admin;
* Flask – SQLAlchemy
* Flask – request;
* Flask – werkzeug.

И пр.

Далее была поставлена задана создать репозиторий в GitHub что я

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

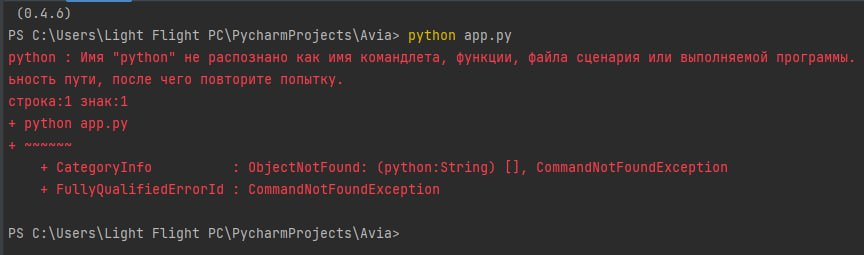
Всё началось с создание папок, были созданы такие папки как:

* 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.



Второй ошибкой был pip который конфликтовал с версией Python. 

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.

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.

models.py:

from flask\_login import UserMixin

from flask\_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

class User(UserMixin, db.Model):

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(50), unique=True, nullable=False)

password = db.Column(db.String(500), nullable=True)

email = db.Column(db.String(120), unique=True, nullable=False)

def \_\_repr\_\_(self):

return f"<users {{self.username}}>"

create\_db.py:

from flask import Flask

from models import User, db

app = Flask (\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///users.db'

db.init\_app(app)

if \_\_name\_\_ == '\_\_main\_\_':

with app.app\_context():

db.create\_all()

print('Создана база данных')

Заключение

Список используемых источников

Приложения