**Анализ и сравнение написания web-приложений с использованием разных фреймворков**

**Выполнил студент 72 потока:**

**Фуников Георгий Дмитриевич**

**2024**

**СОДЕРЖАНИЕ**

1. Введение…………………………………………………………………………..3  
 1.1 Обоснование выбора темы……………………………………….…………..3  
 1.2 Определение цели и задач исследования...………………………………….3  
2. Обзор популярных инструментов для разработки разработки веб-приложений на Python…………………………………………………………………………….4  
 2.1 Django………………………………………………………………………...4  
 2.2 Flask………………………………………………………………………….. 6

2.3FastAPI………………………………………………………………………..8  
3. Проектирование приложения…………………………………………………...12  
4. Разработка в соответствии с созданной документацией…………………….. 13  
5. Анализ и интерпретация результатов…………………………………………..13  
6. Заключение……………………………………………………………………….14

**1.ВВЕДЕНИЕ**

**1.1Обоснование выбора темы:**

Разработка веб-приложений становится все более сложной по мере того, как растут требования к функциональности и производительности. Фреймворки веб-приложений предоставляют структурированные подходы к разработке веб-приложений, что упрощает и ускоряет процесс разработки.  
Этот проект посвящен анализу и сравнению различных фреймворков веб-приложений, таких как Django, FastAPI, Flask.

**1.2Определение цели и задач исследования:**

**Цель:**

Анализировать и сравнивать написание веб-приложений с использованием Django, Flask и FastAPI.

**Задачи:**

1. Разработать простые веб-приложения с использованием Django, Flask и FastAPI.
2. Провести сравнительный анализ этих приложений
3. Сделать выводы о преимуществах и недостатках каждого фреймворка и определить ситуации, в которых они наиболее подходят.

**2. Обзор популярных инструментов для разработки разработки веб-приложений на Python**

**2.1 Django**

Django — это высокоуровневый фреймворк с открытым исходным кодом, написанный на языке программирования Python, который используется для создания веб-приложений и веб-сайтов.

Основные особенности Django:

* **Быстрая разработка**. Фреймворк предоставляет встроенные инструменты и библиотеки, которые позволяют быстро создавать функциональные веб-приложения.
* **Поддержка концепции повторного использования кода**. Уже написанные модули и компоненты можно использовать в разных проектах, что значительно ускоряет процесс разработки.
* **Безопасность**. Фреймворк включает в себя множество встроенных механизмов безопасности, таких как защита от SQL-инъекций, межсайтового скриптинга и подделки межсайтовых запросов.
* **Масштабируемость**. Фреймворк разработан с учётом масштабируемости, что позволяет легко расширять приложение по мере роста его популярности. Он поддерживает различные базы данных и может быть развёрнут на различных платформах.
* **Полный стек инструментов**. Фреймворк предоставляет полный стек инструментов для разработки веб-приложений, включая ORM, систему шаблонов, маршрутизацию URL и админ-панель.
* **Портативность**. Фреймворк позволяет легко переносить приложения между различными платформами — например, с локального сервера разработки на облачный хостинг.
* **Универсальность**. Фреймворк подходит для разработки самых разнообразных проектов, от небольших блогов до крупных корпоративных систем.

Области применения фреймворка Django:

* **Создание новостных сайтов и платформ для дистрибуции контента**.  Фреймворк подходит для создания информационных порталов, онлайн-версий печатных журналов или других цифровых СМИ.
* **Разработка CMS**. На Django создают системы управления контентом, на которых работают админки сайтов.
* **Создание CRM**. С помощью фреймворка разрабатывают системы управления процессами в организации и взаимодействия с клиентами.
* **Разработка сервисов для бронирования**.
* **Создание корпоративных порталов и платформ для ведения документооборота**.
* **Разработка алгоритмических генераторов и платформ для анализа данных**, систем фильтрации и машинного обучения.

**2.2 Flask**

Flask - это микрофреймворк. У него нет инструментов и библиотек, характерных для других фреймворков. Вместо них данное программное обеспечение оснащено рядом базовых возможностей. Расширение функциональности «приложения» the Flask осуществляется за счет установки дополнений. Это минималистичное программное обеспечение для быстрой разработки.

Основные особенности Flask:

* **Минимализм**. Flask предлагает компактный набор инструментов для разработки сайтов, что делает его идеальным фреймворком для начинающих разработчиков.

* **Гибкость.** Модульность Flask позволяет разработчикам настраивать веб-приложения в соответствии со своими потребностями
* **Расширяемость**. С помощью плагинов в Flask можно интегрировать любые необходимые функции.

* **Связь с базами данных**. Flask-приложения можно связывать с различными базами для работы с датасетами.

* **Микроподход**. Flask позволяет максимально быстро делать веб-приложения на базе одного файла с минимальной функциональностью.

* **Тестирование**. Проверку и отладку решений Flask осуществляет с помощью фреймворка pytest.

* **Профессиональное комьюнити и служба поддержки**. Flask популярен среди программистов, вокруг него большое сообщество Flask-разработчиков, готовых помочь начинающим.
* **Эффективность**. Flask оптимизирован для производительности и может обрабатывать множество запросов быстро и эффективно.

Области применения фреймворка Flask:

Flask – конструктор, используемый разработчиками в самых разных сферах. Официальное сообщество разрабатывает спектры расширений. О том, что они относятся к Фласк, указывает приставка the flask: flask-sqlalchemy, flask-login и так далее.

Данный инструмент лучше всего использовать для небольших проектов и website, работающих со статическим контентом. Соответствующий микрофреймворк отлично подходит для формирования микросервисов.

**2.3 FastAPI**

FastAPI — это фреймворк для создания лаконичных и довольно быстрых HTTP API-серверов со встроенными валидацией, сериализацией и асинхронностью,  
что называется, из коробки. Стоит он на плечах двух других фреймворков: работой с web в FastAPI занимается Starlette, а за валидацию отвечает Pydantic.

Основные особенности FastAPI:

* **Безопасность типов и проверка**. FastAPI использует подсказки по типам Python и модели Pydantic для автоматической проверки данных и сериализации. В результате получаются типобезопасные API, в которых ошибки обнаруживаются во время компиляции, а не во время выполнения.

* **Высокая производительность**. FastAPI работает со скоростью, сопоставимой с фреймворками Node.js и Go, благодаря эффективному использованию современных функций Python.

* **Автоматическая интерактивная документация API**. FastAPI автоматически генерирует документацию API (Swagger UI и ReDoc) на основе подсказок по типу кода и моделей Pydantic.

* **Система внедрения зависимостей**. FastAPI имеет надёжную систему внедрения зависимостей, которая упрощает управление сложными зависимостями, такими как подключения к базе данных, аутентификация и внешние службы.

* **Асинхронная поддержка**. FastAPI изначально поддерживает асинхронное программирование, позволяя писать асинхронные обработчики маршрутов и в полной мере использовать синтаксис async/await в Python для операций ввода-вывода.

* **Аутентификация и авторизация**. FastAPI предоставляет встроенную поддержку различных методов аутентификации, включая токены JWT, OAuth2 и проверку ключа API.

* **Сериализация и синтаксический анализ данных**. FastAPI обрабатывает автоматическую сериализацию объектов Python в ответы JSON. Он также может анализировать данные входящего запроса, включая параметры запроса, тело запроса (JSON, данные формы) и заголовки, на основе подсказок по типу.

**Области применения фреймворка FastAPI**:

* создание высокопроизводительных API и микросервисов;

* разработка высоконагруженных систем, требующих обработки большого количества запросов одновременно;

* создание защищенных API благодаря встроенной поддержке аутентификации и авторизации;

* интеграция с базами данных, такими как SQLAlchemy и ORM, что упрощает взаимодействие с ними и управление данными;

* реализация популярных функций API, таких как загрузка файлов, валидация входных данных, маршрутизация.

**3. Проектирование приложения**

Пример работы приложения на Django

Для начала нам нужно установить Django, его можно установить в командной строке с помощью команды pip install Django (рис.1.1)



(рис.1.1)

Чтобы начать работу с проектом нужно прописать в командной строке следующую команду django-admin startproject Django (рис.1.2)



(рис.1.2)

Дальше нам надо создать приложение в директории Django. Воспользовавшись командой cd (имя вашей директории) переходим в нее и прописываем в командной строке следующую команду py manage.py startapp (имя вашей директории) (рис.1.3)

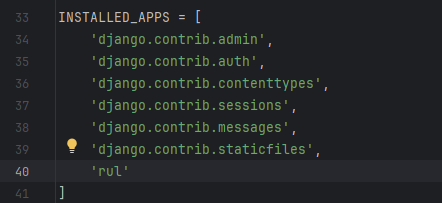


(рис.1.3)

После этого заходим в директорию Django в файл под названием

settings.py и добавим наше приложение в лист INSTALLED\_APPS

(рис.1.4)

(рис.1.4)

Также в этом же файле в листе TEMPLATES внутри листа DIRS

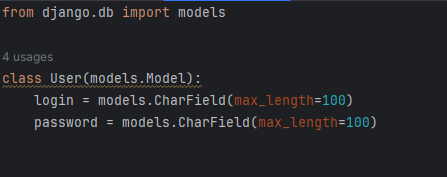
пропишем папку с нашим шаблоном с помощью BASE\_DIR (рис.1.5)



(рис.1.5)

Теперь заходим в директорию, которую ранее создали, у меня это директория url. Дальше переходим в файл под названием models.py,

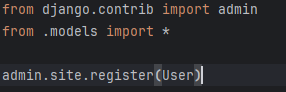
здесь мы создаем модель (рис.1.6)



(рис.1.6)

Дальше в файле admin.py регистрируем модель в админ-панели

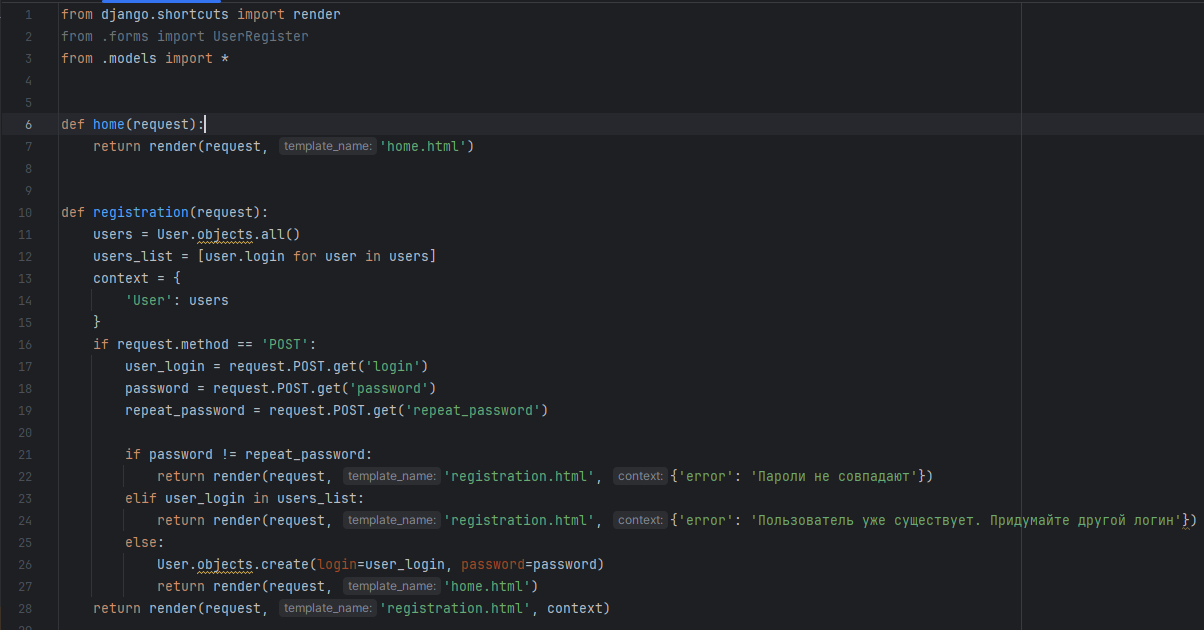
Django (рис.1.7)



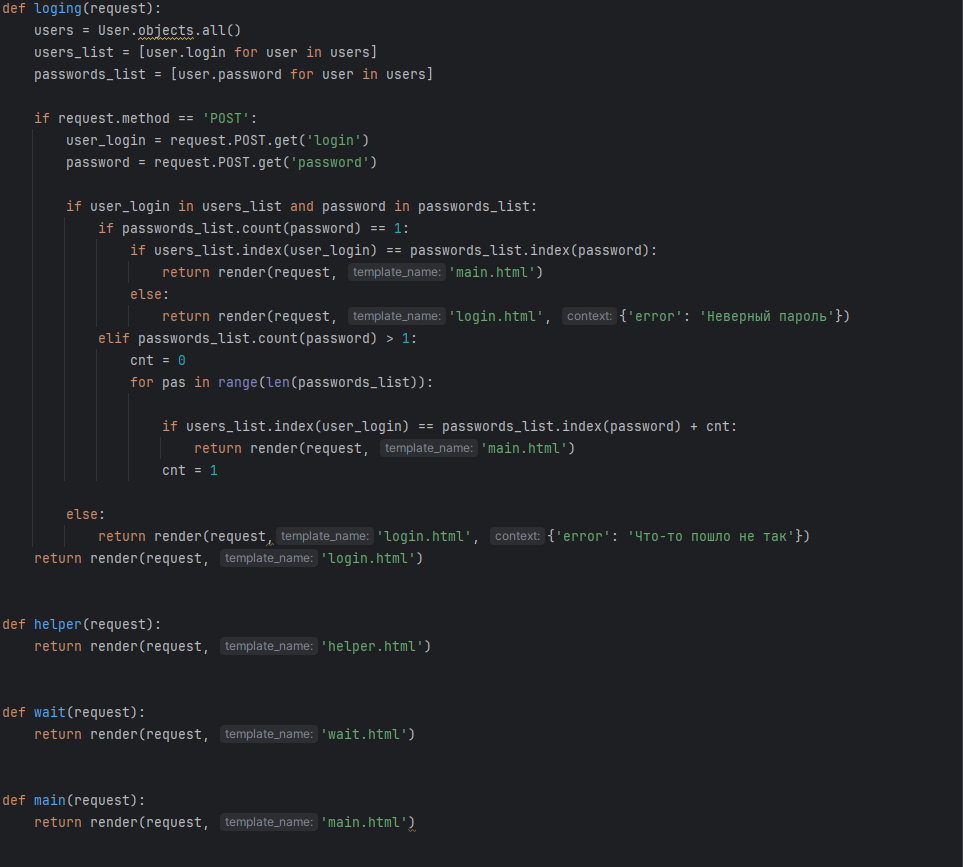
(рис.1.7)

Теперь переходим в файл views.py. В этом файле мы будем создавать функции. В начале мы должны импортировать render (from django.shortcuts import render) также нужно импортировать нашу модель из models (from .models import \*). Дальше создаем функции

home, helper, wait, main для этих функций передаем переменную request и возвращаем render с заготовленным шаблоном. Также создаем функцию registration и вносим туда логику регистрации, в которую будут поступать запросы GET и POST. И создаем функцию loging, в котором будем проверять верность логина и пароля. (рис.1.8) (рис.1.9)

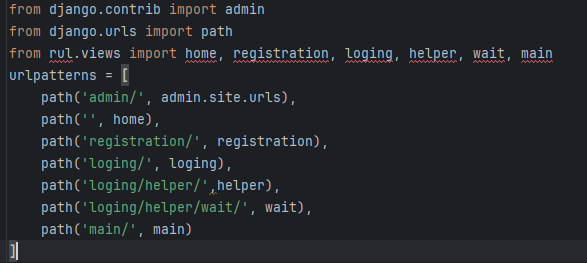


(рис.1.8)



(рис.1.9)

Далее мы переходим в папку urls.py и там импортируем функции и прописываем к ним путь с помощью path (рис.1.10)



(рис.1.10)

После написания всех функций и шаблонов мы можем запустить сервер, введя команду py manage.py runserver (рис.1.11)



(рис.1.11)