**Дипломная работа**

**Тема: Анализ и сравнение написания web-приложений с использованием разных фреймворков:**разработка простых веб-приложений с использованием Django, Flask и FastAPI, их сравнение.

Автор: Сухарев Евгений Николаевич

**Оглавление:**

1. [Введение](#Введение)
2. [Основные понятия и определения](#Основные)
3. [Методы и подходы к разработке](#Методы)
4. [Обзор фреймворков, используемых для разработки  
   приложения](#ОбзорФреймворков)
5. [Проектирование приложения](#ПроектирваниеПриложения)
6. [Разработка в соответствии с документацией](#Разработка)
7. [Анализ и интерпретация результатов](#АнализРезультатов)
8. [Заключение](#Заключение)
9. **Введение**

Разработка веб-приложений — это процесс создания интерактивных программ, которые работают в веб-браузере. Веб-приложения могут быть простыми, как одностраничные сайты, или сложными, как социальные сети или интернет-магазины.

Веб-приложения играют важную роль в современном мире, предоставляя пользователям доступ к разнообразным сервисам и информации через интернет. Они могут быть использованы для различных целей, таких как онлайн-торговля, социальные сети, управление проектами, обучение и многое другое. Веб-приложения обладают рядом преимуществ, включая доступность с любого устройства с интернет-соединением, возможность обновления контента в реальном времени и интеграцию с другими сервисами и платформами.

Фреймворки и библиотеки упрощают разработку веб-приложений, предоставляя готовые решения для распространенных задач. В качестве примера рассмотрим 3 популярных фреймворка: Django, FastAPI и Flask.

Цель исследования: создание веб-приложения для сравнения методов написания бэкенд части с использованием вышеописанных инструментов.

Задачи работы:

1. Обзор фреймворков для написания веб приложений и работы с шаблонами и БД;

2. Разработка структуры и алгоритмов работы приложения;

3. Изучение методов написания запросов в БД с использованием различных ORM;

4. Написание дипломной работы, включающей в себя введение, методологию, результаты исследования, анализ результатов и выводы.

Цели и задачи исследования направлены на получение практических навыков использования фреймворков, формирования структуры приложения и самостоятельного изучения документации.

1. **Основные понятия и определения**
2. **Фреймворк (Framework)**: Программная платформа, которая предоставляет готовые компоненты и инструменты для разработки приложений.
3. **Веб-приложение (Web Application)**: Программное приложение, которое работает на веб-сервере и доступно через браузер
4. **База данных:** Абстракция над файловой системой операционной системы, которая значительно упрощает создание приложений, создающих, читающих, обновляющих и удаляющих различные данные.
5. **Система управления базами данных (СУБД)**: Программное обеспечение, которое позволяет сохранять, организовывать, изменять, получать и анализировать данные.
6. **ORM**: Библиотека кода, которая автоматизирует перенос данных, хранящихся в таблицах реляционной базы данных, в объекты, которые используются в коде приложения.
7. **Фронтенд (frontend)** — это разработка пользовательских функций и интерфейса. К ним относится всё, что пользователи видят на сайте или в приложении, и с чем можно взаимодействовать: картинки, выпадающие списки, меню, анимация, карточки товаров, кнопки, чекбоксы, интерактивные элементы.
8. **Бэкенд (backend)** — это логика работы сайта, скрытая от пользователя. Именно там происходит то, что можно назвать работой сайта.
9. **Django Template Language (DTL)** — это язык шаблонов для генерации динамических частей веб-страницы в Django.
10. **JWT (JSON Web Tokens)** — это простой и безопасный способ передачи информации между клиентом и сервером с помощью шифрования.
11. **Деплой (deploy)** — это развертывание и запуск веб-приложения или сайта в его рабочей среде, то есть на сервере или хостинге. Разработчик загружает приложение, написанное на локальном компьютере, в специальное пространство, из которого оно доступно в интернете.
12. **Методы и подходы к разработке**

В качестве примера простого веб-приложения будет разработан TODO лист. Список задач будет отображаться для конкретного пользователя. В проекте будет реализован следующий функционал рассматриваемых фреймворков:

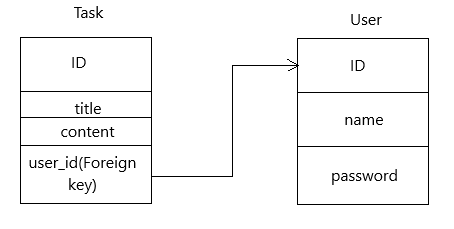
- работа с html шаблонами;

- работа с url и маршрутами;

- работа с СУБД (формирование запросов в БД, создание таблиц, осуществление связи между ними);

- организация системы регистрации и авторизации пользователей;

Фронтенд приложения будет реализован с использованием HTML и CSS. В качестве СУБД выбрана SQLite, т.к. она наиболее проста и встроена в стандартный пакет python. Для осуществления связи между фронтендом и бэкендом будем использовать рассматриваемые в теме проекта фреймворки для сравнения методов написания кода и способов реализации задачи.

Рисунок 1 Структура таблиц БД

Структура базы данных приложения представляет собой две таблицы: одна содержит в себе всех зарегистрированных пользователей, другая – задачи. Таблицы связаны друг с другом по принципу «один ко многим (one to many)». Т.е. с одним пользователем может быть связано много задач.

1. **Обзор фреймворков, используемых для разработки приложения**

**Django**  
Django —фреймворк для веб-разработки на Python.  
Основные возможности:  
· Встроенная страница администрирования для управления данными.  
· ORM (Object-Relational Mapping) для работы с базами данных.  
· Поддержка аутентификации пользователей и управления доступом.  
· Мощная система маршрутизации URL.  
· Шаблонизатор для создания HTML-шаблонов.  
Особенности:  
Идеально подходит для больших и сложных проектов благодаря своей полной экосистеме.  
**Flask**  
Это фреймворк для веб-разработки на Python. Обеспечивает простоту и гибкость.  
Основные возможности:  
Минималистичный подход: только основные компоненты без лишних зависимостей.  
Высокая гибкость для добавления нужных библиотек и модулей.  
Jinja2 для шаблонов и Werkzeug для маршрутизации и обработки запросов.  
Особенности:  
Подходит для небольших и средних проектов, где требуется большая гибкость.  
**FastAPI**  
Это современный, высокопроизводительный фреймворк для создания API с Python 3.7+ на основе стандартов OpenAPI и JSON Schema.  
Основные возможности:  
· Высокая производительность благодаря использованию асинхронного программирования (async/await).  
· Автоматическая генерация интерактивной документации API.  
· Поддержка валидирования и аннотаций типов данных.  
· Интеграция с различными библиотеками и инструментами, такими как Pydantic и SQLAlchemy.  
Особенности:  
Часто используется для создания API и микросервисов.

1. **Проектирование приложения**

Определение структуры приложения:

Разработка схемы архитектуры, включающей фронтенд, бэкенд, базу данных и при необходимости миграции.

Разработка функционала:

Написание логики работы приложения, маршрутов, базового шаблона, моделей таблиц и функций для создания и осуществления подключения к БД

Реализация пользовательского интерфейса:

Создание html страниц на основе базового шаблона, создание форм для заполнения при создании задач, авторизации и регистрации пользователей, создание ссылок для перехода между страницами и выполнения команд.

**Основные требования**

Основное меню с использованием шаблона:

Меню должно отображаться на всех страницах при осуществлении переходов. При этом меню должно иметь различные разделы, доступные авторизованным и анонимным пользователям.

Авторизация и регистрация:

В приложении должны осуществляться авторизация и регистрация пользователей с использованием различных форм. Так же должен быть функционал, доступный только авторизованным пользователям.

Взаимодействие с БД:

Создание с помощью ORM запросов в БД для создания и удаления задач, создания пользователей.

**Технические требования**

Фронтенд: Простой интерфейс. Использовать HTML и CSS

Бэкенд: Python с использованием фреймворков Django, FastAPI и Flask

База данных: Использование БД (SQLite) для хранения данных о пользователях и связанных с ними задачах. Использование ORM (Sqlalchemy).

1. **Разработка в соответствии с документацией**

Разработка была разделена на несколько этапов:

- проектирование интерфейса;

- проектирование структуры будущей БД;

- реализация функционала на фреймворке Django;

- реализация функционала на фреймворке FastAPI;

- реализация функционала на фреймворке Flask.

**Разработка**

**Фронтенд разработка:**

Создан простейший базовый шаблон для пользовательского интерфейса с использованием CSS для стилизации.

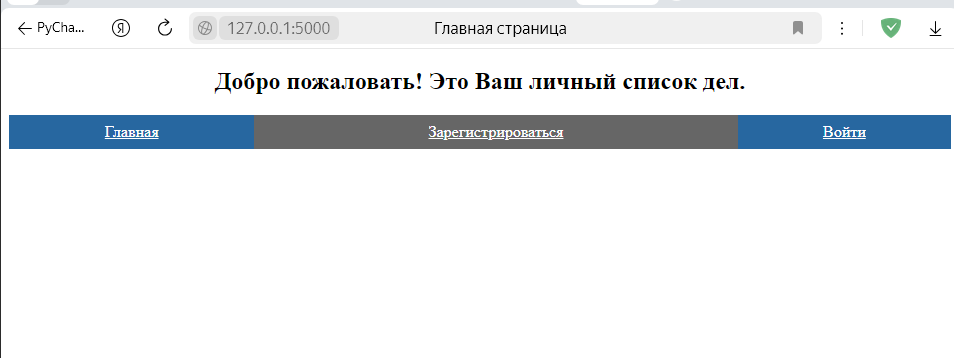


Рисунок 2 Главная страница

Разработаны формы для создания задач, регистрации и авторизации пользователей.

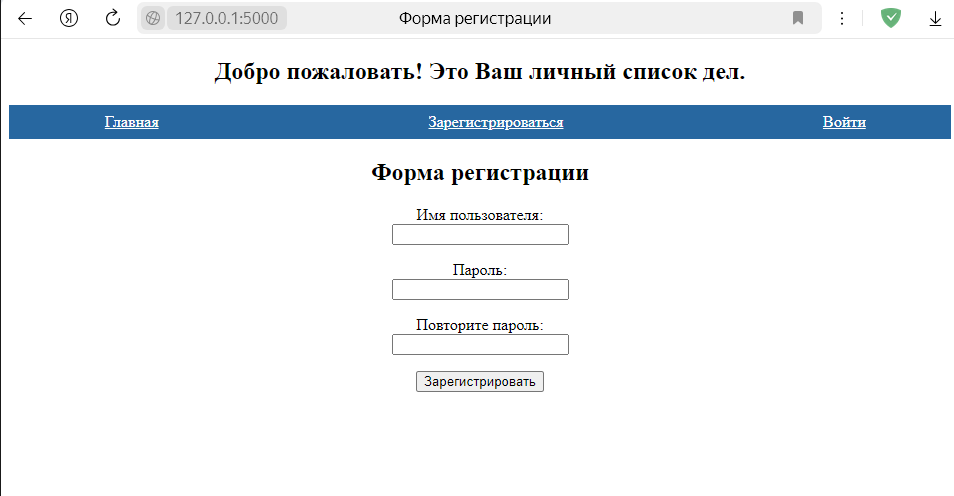


Рисунок 3 Форма регистрации

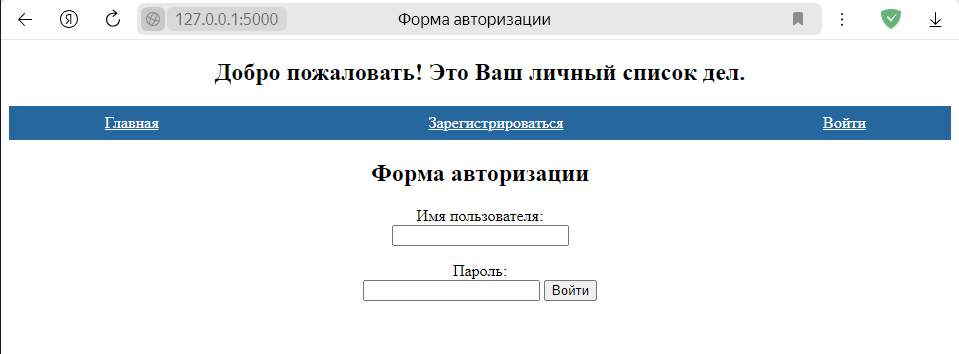


Рисунок 4 Форма авторизации

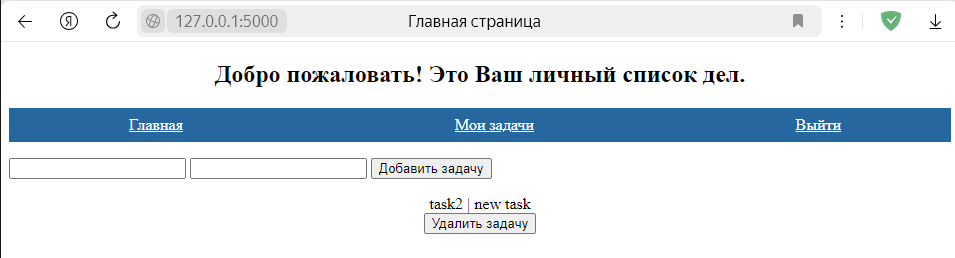


Рисунок 5 Форма для создания задач

**Бэкенд-разработка**

**Реализована серверная логика с использованием фреймворка Django.**

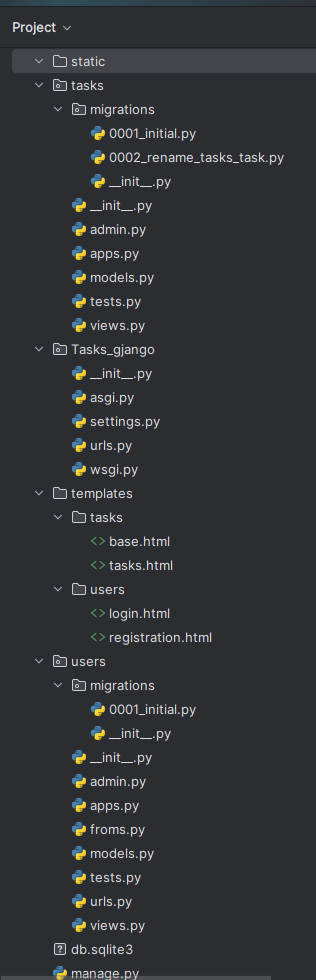
1. Создана структура проекта и приложений. Структура создается автоматически командами в консоли: django-admin startproject и python manage.py startapp.
2. Созданы два приложения: для пользователей и задач.
3. Написаны модели для будущих таблиц БД
4. Произведена настройка файла settings.py с указанием пути к БД, используемой СУБД, пути к шаблонам
5. Произведены миграции командами python manage makemigrations и python manage migrate. В результате чего создана БД с моделями и техническими таблицами Django.
6. Настроены маршруты в файле urls.py и функции отображения в соответствующих приложениях пользователей и задач.
7. Внесены изменения в шаблоны с использованием синтаксиса DTL.
8. Проверена работа всего функционала веб-приложения.

Рисунок 6 Файловая структура проекта на Django

**Реализована серверная логика с использованием фреймворка FastAPI.**

1. Создана файловая структура проекта
2. Установлены дополнительные библиотеки:  
   - Jinja2 – для работы с html шаблонами;  
   - SQLalchemy – для работы с БД;  
   - Alembic – для произведения миграций и создания таблиц.
3. Создан базовый класс Base, от которого будут наследоваться модели таблиц
4. Создана локальная сессия для осуществления подключения к БД
5. Написаны модели для будущих таблиц БД
6. Произведены миграции с помощью Alembic
7. В main.py прописаны маршруты и функции отображения
8. Внесены изменения в html шаблоны с использованием синтаксиса Jinja2
9. Проверена работа всего функционала веб-приложения.

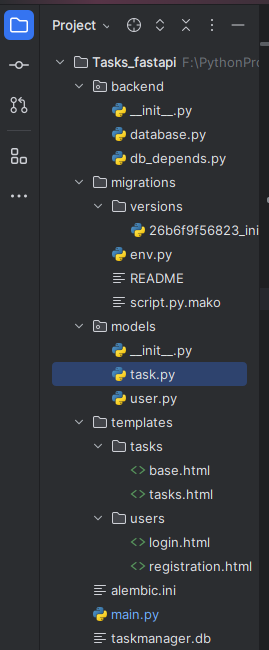


Рисунок 7 Файловая структура проекта на FastAPI

**Реализована серверная логика с использованием фреймворка Flask.**

1. Создана файловая структура проекта
2. Установлены дополнительные библиотеки:  
   - Jinja2 – для работы с html шаблонами;  
   - SQLalchemy – для работы с БД;  
   - Flask-SQLalchemy – для обеспечения работы Flask совместно с SQLalchemy.
3. Создан базовый класс Base, от которого будут наследоваться модели таблиц
4. Создан экземпляр класса SQLAlchemy из расширения flask\_sqlalchemy для осуществления подключения к БД
5. Написаны модели для будущих таблиц БД
6. В main.py прописаны маршруты и функции отображения
7. Внесены изменения в html шаблоны с использованием синтаксиса Jinja2
8. Проверена работа всего функционала веб-приложения.

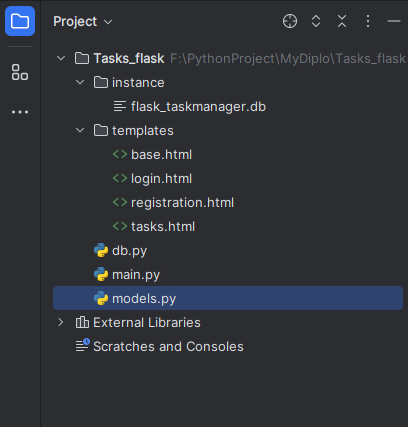


Рисунок 8 Файловая структура проекта на Flask

1. **Анализ и интерпретация результатов**

**Сравнение фреймворков**

**Django** – это высокоуровневый, полнофункциональный веб-фреймворк, придерживающийся философии “батарейки в комплекте”.

Выявленные достоинства:

- При создании проекта и приложений мы получаем готовую развернутую структуру.

- При разработке приложения не пришлось устанавливать дополнительных библиотек.

- Интуитивный и более простой по сравнению с другими фреймворками синтаксис ORM.

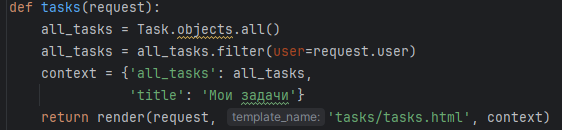


Рисунок 9 Пример запроса в БД

- Наличие встроенной и настраиваемой админ-панели.

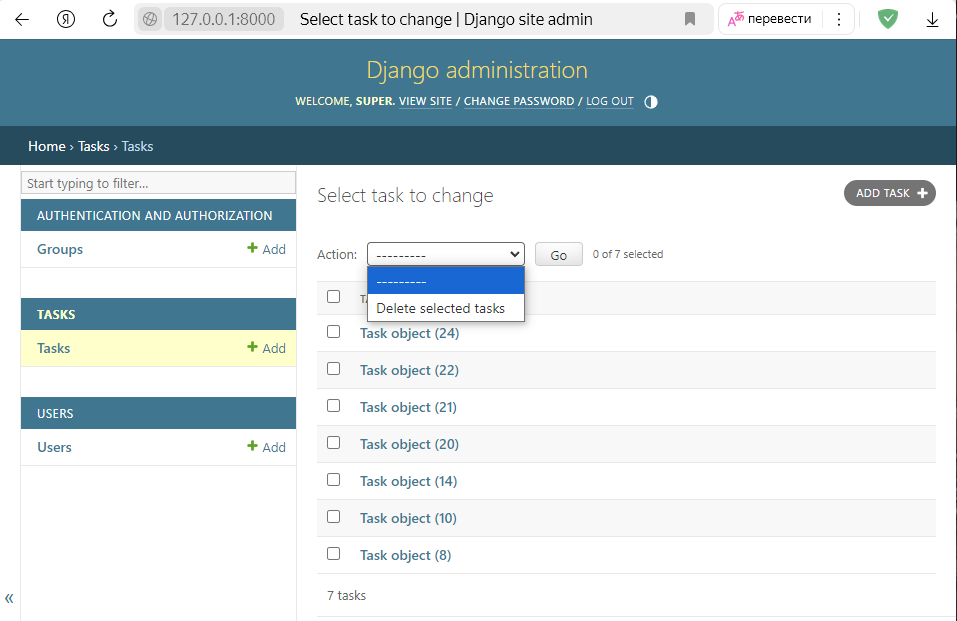


Рисунок 10 Админ панель

- Встроенная система авторизации и аутентификации.

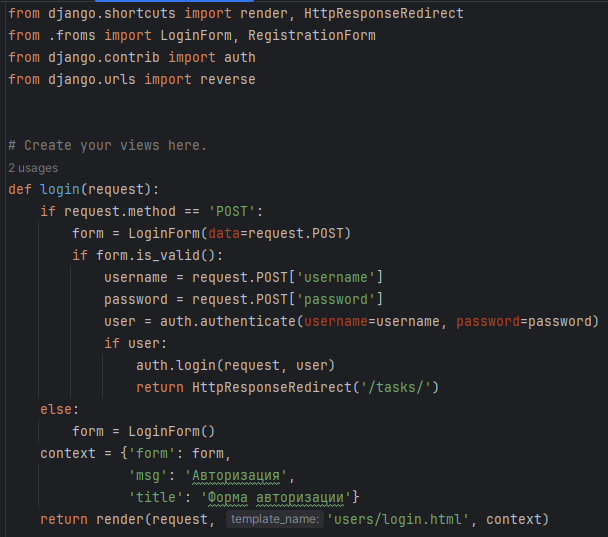


Рисунок 11 Пример использования встроенной системой авторизации и аутентификации

- Встроенный шаблонизатор и шаблоны форм.

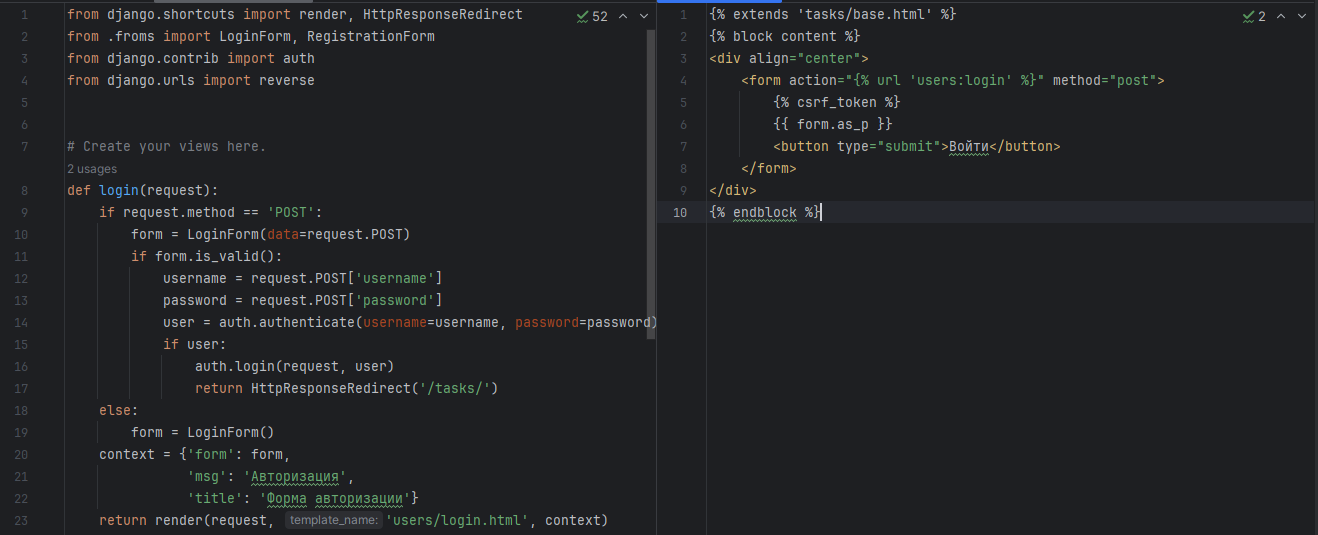


Рисунок 12 Применение шаблонов формы в html документе

Выявленные недостатки:

- Из-за обилия встроенных инструментов Django требует больше мощностей для запуска, чем нативный код на Python.

- Слишком большая сложность для простых проектов. Нет необходимости в такой развернутой и сложной файловой структуре простого проекта.

Вывод: Наличие множества готовых инструментов, большое сообщество и удобная документация делают данный фреймворк отличным выбором для создания сложных приложений и масштабных проектов, таких как системы управления контентом, приложений для социальных сетей и сайтов электронной коммерции.

**FastAPI** – это фреймворк, предназначенный для быстрого и эффективного создания API. Он сочетает в себе подсказки типов Python и асинхронное программирование для создания надежных и автодокументированных API.

Выявленные достоинства:

- Асинхронное программирование.

- Высокая производительность.

- Отсутствие утвержденной архитектуры дает больше свободы в использовании различных методов разработки.

- Качественная документация (на русском языке)

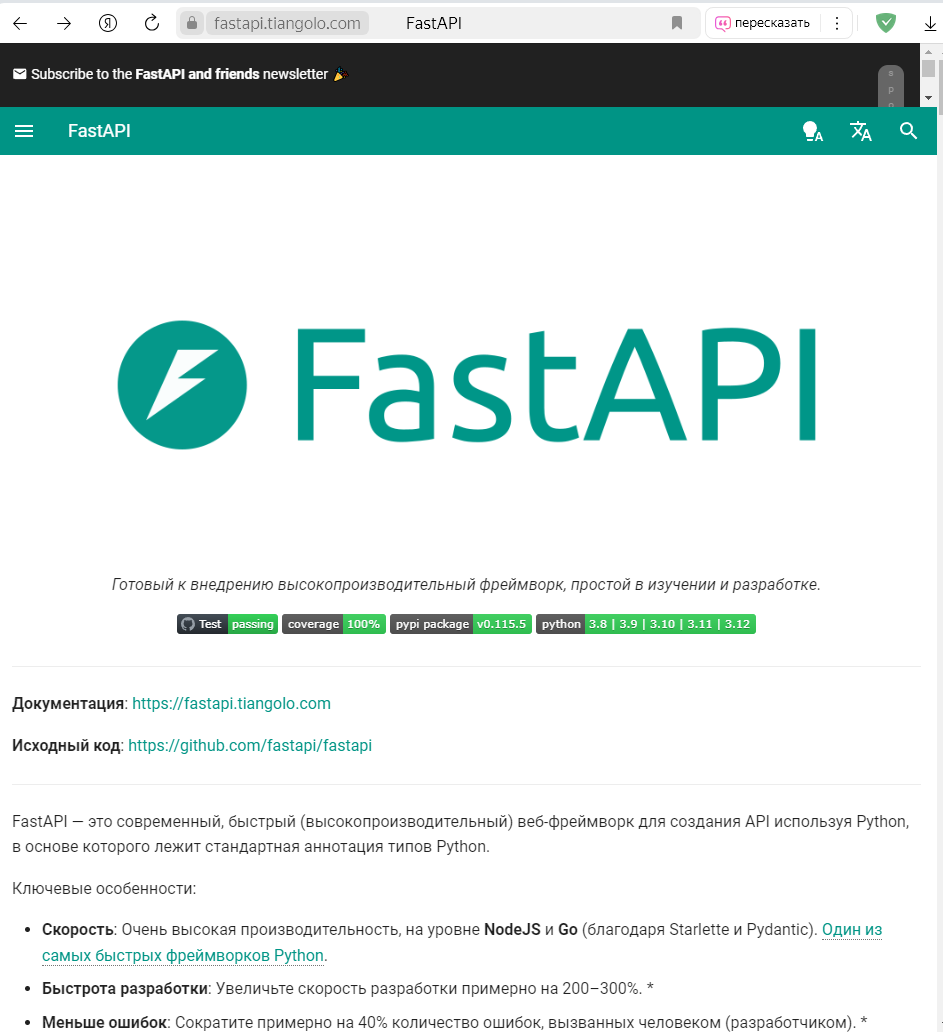


Рисунок 13 Официальная документация Fast API

- Встроенная интерактивная документация

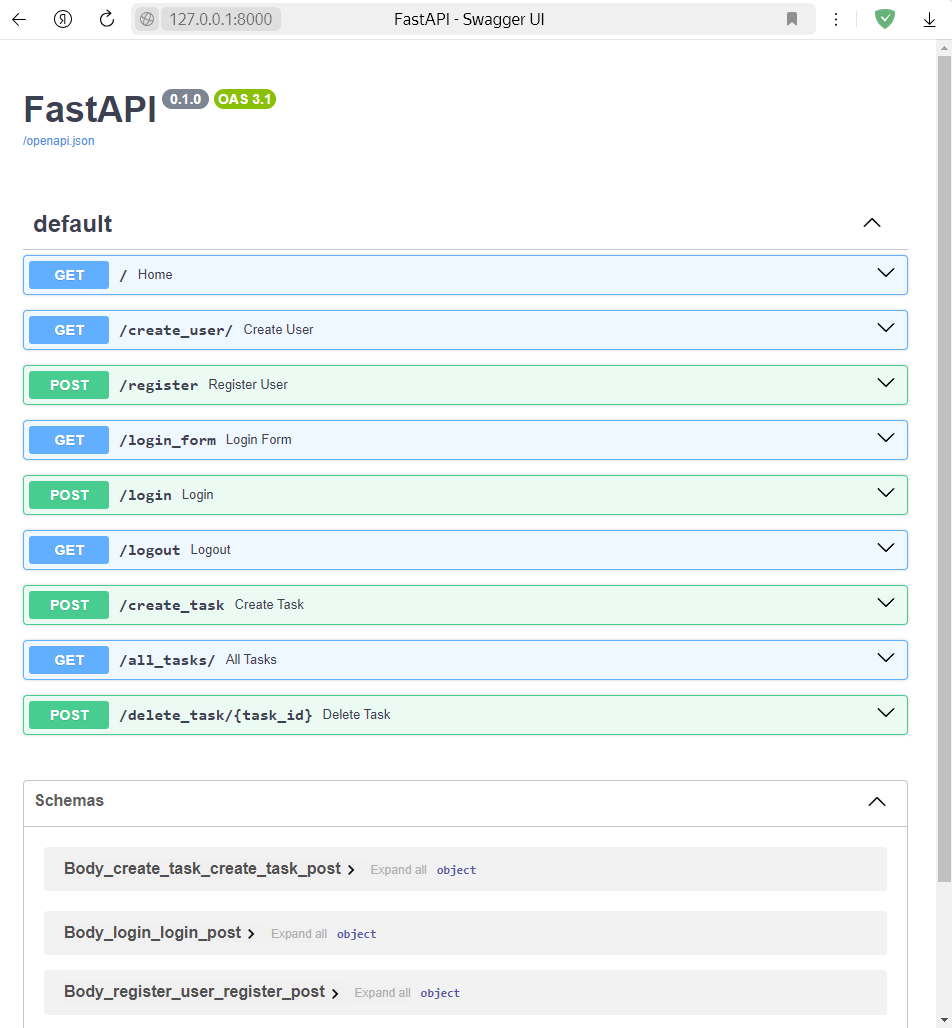


Рисунок 14 Swagger

Выявленные недостатки:

- Необходимость установки дополнительных библиотек для реализации даже простого проекта

- Приспособлен больше под создание API чем под написание небольших сайтов.

- Отсутствие системы авторизации. Так же мало информации в документации по авторизации и применению JWT токенов.

**Flask** – это микрофреймворк, в котором приоритет отдается простоте и гибкости, а не встроенной функциональности. Это означает, что разработчики могут выбирать инструменты и расширения по мере необходимости.

Выявленные достоинства:

- Минималистичная структура проекта.

- Отсутствие утвержденной архитектуры дает больше свободы в использовании различных методов разработки.

- Удобная обработка запросов и гибкость.

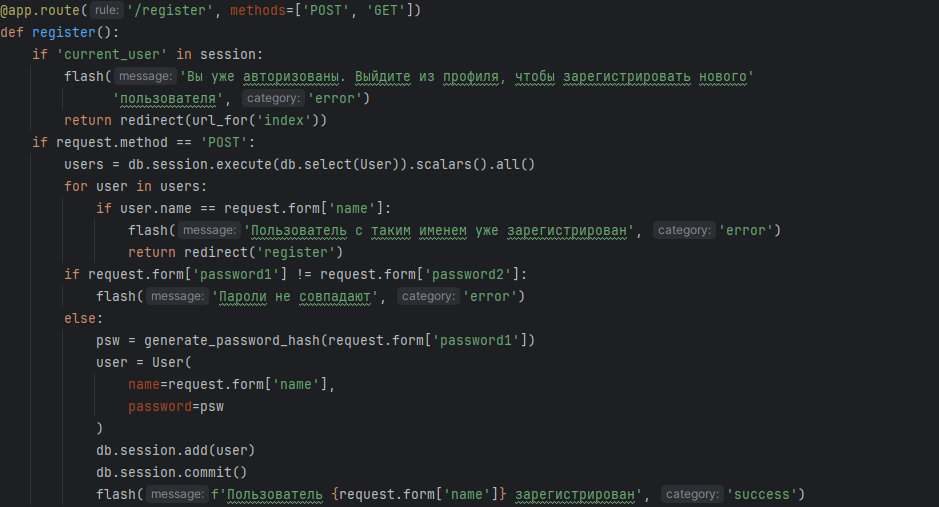


Рисунок 15 Пример гибкой обработки запроса

Выявленные недостатки:

- Необходимость установки дополнительных библиотек для реализации даже простого проекта

- Синхронное программирование. С увеличением числа запросов, обработка занимает больше времени.

**8. Заключение**

**Обзор выполненной работы**

Планирование и разработка простого веб приложения для изучения функционала рассматриваемых в теме работы фреймворков были успешно завершены в соответствии с изначально созданной документацией. Приложение включает функционал перехода между страницами, авторизации и регистрации пользователей, взаимодействия с базой данных. Так же применены стили на html страницах и различные шаблонизаторы.

Реализованное приложение позволяет произвести сравнение функционала, работы и синтаксиса фреймворков Django, FastAPI и Flask.

**Дальнейшие планы**

Планируется дальнейшее расширение функционала приложения, усовершенствование БД, применение большего количества стилей.

Так же будет полезным усовершенствование системы безопасности приложения (jwt токены, шифрование паролей), чтобы обезопасить данные пользователей.

Следующим этапом будет изучение методики деплоя приложения на сервер.