**Дипломная работа по теме:**  
**Анализ и сравнение написания веб-приложений с использованием разных фреймворков.**

**Автор: Дмитриев Евгений Валерьянович**   
**Группа: Python 27**

**Оглавление дипломной работы:**

1. Введение. 3стр.

Обоснование выбранной темы. 3стр.

Определение целей и задач исследования. 4стр.

2.Основные понятия и определения. 5стр.

3.Методы и потходы к разработке. 6стр.

Выбор тематики сайта. 6стр.

Архитектура веб-приложения. 6стр.

4.Обзор популярных инструментов для разработки веб-приложений на Python 7стр.

Django

Flask 8стр.

FastAPI

5.Проектирование приложения. 9стр.

Планирование и анализ требований.

Основные требования.

Технические требования.

6.Разработка на основе созданной документации. 10стр.

Планирование разработки.

Разработка.

7.Анализ и интерпретация результатов. 17стр.

Сравнение фреймворков.

Интерпретация результатов. 18стр.

Рекомендации по выбору фреймворка в зависимости от проекта.

8.Заключение. 19стр.

Обзор выполненной работы.

Дальнейшие планы. 20стр.

Приложение 1. Requirements

**1.Введение**

**Обоснование выбора темы:**

1. В современном мире, где бы ты не находился тебя окружают интернет и разного рода сервисы, которые стали неотъемлемой частью повседневности. Создание веб-приложений способствует развитию сервисов и их наполнения что ведет к повышению качества уровня жизни. Сложно представить себе современный мир без веб-приложений, что говорит о его несомненной актуальности еще на долгие годы вперед.
2. Потребности рынка: существует растущая потребность в специалистах, обладающих навыками разработки и обслуживания веб-приложений. У каждого предприятия в наше время есть свой сайт, и он не стоит на месте, а дорабатывается и ошлифовывается до идеального состояния ведь от этого зависит качество предоставляемых ими услуг.
3. Практическая значимость: Работа в области разработки веб-приложений позволит применить полученные знания на практике и даст возможность улучшить свои навыки в работе с разными видами фреймворков. Это позволит создавать полезные и интересные возможности для пользователей.
4. Личный интерес и потенциальные перспективы: Использование различных фреймворков для разработки веб-приложений позволит развить и отработать опыт создания интересных и практичных проектов. Кроме того, такие навыки востребованы на рынке труда, что открывает перспективы для карьерного роста и развития.

Таким образом, выбор темы “Анализ и сравнение написания веб-приложений с использованием разных фреймворков.” обусловлен ее актуальность, потребностями рынка, практической значимостью и личным интересом, что делает эту тему подходящей для проведения дипломной работы.

**Определение целей и задач исследования:**

Цель исследования: разработать веб-приложения на разных фреймворках и сравнить их.

**Задачи исследования:**

1. Обзор фреймворков для написания веб-приложения: провести анализ существующих инструментов и освоить их функционал.
2. Разработка сценариев сравнения: определить критерии сравнения и сравнить по ним веб-приложения, написанные на целевых фреймворках.
3. Создание веб-приложений для проведения тестирования: реализовать веб-приложения для дальнейшего сравнения работы разных фреймворков.
4. Написание дипломной работы: составить дипломную работу, включающую в себя введение, обзор литературы, методологию и результаты исследования, анализ результатов, выводы и рекомендации.

Цели и задачи исследования направлены на получение практических результатов, которые позволят сформировать рейтинг фреймворков для веб-приложений.

**2.Основные понятия и определения**

**Обзор основных понятий в области разработки веб-приложений:**

1. Фреймворк (Framework): Программная платформа, которая представляет готовые компоненты и инструменты для разработки приложений. В контексте веб-приложений часто используются Django, Flask, FastAPI.
2. Веб-приложение (Web Application): Программное приложение, которое работает на веб-сервере и доступно через браузер. В контексте данного проекта, веб-приложение будет использоваться для сравнения инструментов фреймворков для разработки веб-приложений.
3. CRUD Запросы: это акроним, обозначающий четыре базовые функции, используемые при работе с базами данных:  
   Создание( Create )  
   Чтение( Read )  
   Изменение (Update)  
   Удаление( Delete )

В контексте веб-приложений является одним из инструментов взаимодействия с пользователем например (Get запрос является запросом для чтения( Read ))

1. Базы данных (Database): Программное приложение, которое хранит в себе информацию для вывода и получения информации через CRUD запросы.
2. Frontend (интерфейс пользователя) : это оболочка веб-приложения, с которой взаимодействует пользователь так называемая лицевая сторона проекта.
3. Backend (серверная логика): это начинка веб-приложения, с которой связывается Frontend для обработки запросов, поступающих от пользователя.
4. API ( интерфейс программного приложения): программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими
5. REST API: это архитектурный стиль API, который используется для регламентирования взаимодействия между клиентской и серверной частью программного обеспечения.

Данные термины помогают понять ключевые аспекты работы с веб-приложениями.

**3. Методы и подходы к разработке**

**Выбор тематики сайта:**

В связи с заданием по созданию простого веб-приложения определится с тематикой сайта для обзора функциональности фреймворков в одних и тех же условиях.

**Архитектура веб-приложения:**

Frontend и Backend: Фронтенд может быть разработан с помощью HTML, CSS и JavaScript, а бэкенд может быть реализован на Python с использованием фреймворков, таких как Django, Flask и FastAPI.

REST API: Для связи между фронтендом и бэкендом можно использовать REST API. Это позволит фронтенду отправлять данные на сервер для их обработки и отправлять результаты в бэкенд.

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

Основные - Django, Flask, FastAPI, ниже более подробное описание их возможностей.

**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 управление доступом.
* Мощная система маршрутизации URL.
* Шаблонизатор для создания HTML-шаблонов.

Особенности:

Идеально подходит для разработки больших и сложных проектов благодаря своей полной экосистеме.

**Flask**

Flask- фреймворк для веб-разработки на Python. Обеспечивает простоту и гибкость в разработке.

**Основные возможности:**

* Минималистичный подход: только основные компоненты без лишних зависимостей.
* Высокая гибкость для добавления нужных библиотек и модулей.
* Jinja2 для шаблонов и Werkzeug для машрутизации 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 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 аннотации типов данных.
* Интеграция с различными библиотеками и инструментами, такими как Pydantic и SQLAlchemy.

**Особенности:**

Часто используется для создания API и микросервисов.

1. **Проектирование приложения**

**Планирование и анализ требований.**

* Выбор темы веб-приложения для разработки.
* Определение структуры приложения: Разработка схемы архитектуры, включающий бэкенд, фронтенд и базу данных.
* Разработка прототипа приложения на одном из фреймворков.
* Реализация основного функуционала: Создание базовой версии приложения с минимально необходимым функционалом для тестирования.
* Разработка полного функционала.
* Интеграция баз данных.

**Основные требования**

* Отображение информации из базы данных: Пользователь должен спокойно переходить через панель навигации и кнопки.
* Внесение данных через форму Post запроса: Пользователь должен иметь возможность вносить новые данные.

**Технические требования**

* Фронтенд: использовать HTML, CSS, JavaScript, возможно Bootstrap.
* Бэкенд: Python с использованием фреймворков Django, Flask, FastAPI.
* База данных: Использование базы данных (например, SQlite) для хранения данных.

1. **Разработка в соответствии с созданной документацией**

**Планирование разработки.**

Разработка была разделена на несколько основных этапов:

* Проектирование интерфейса.
* Реализация серверной логики.

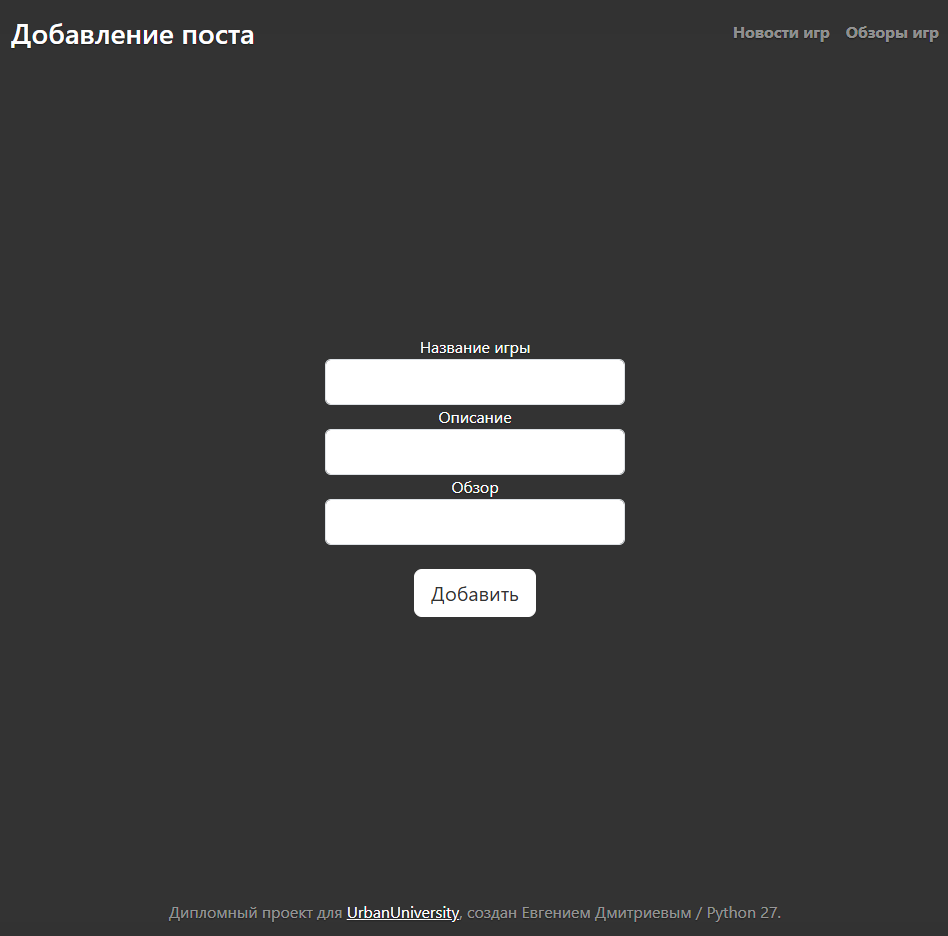
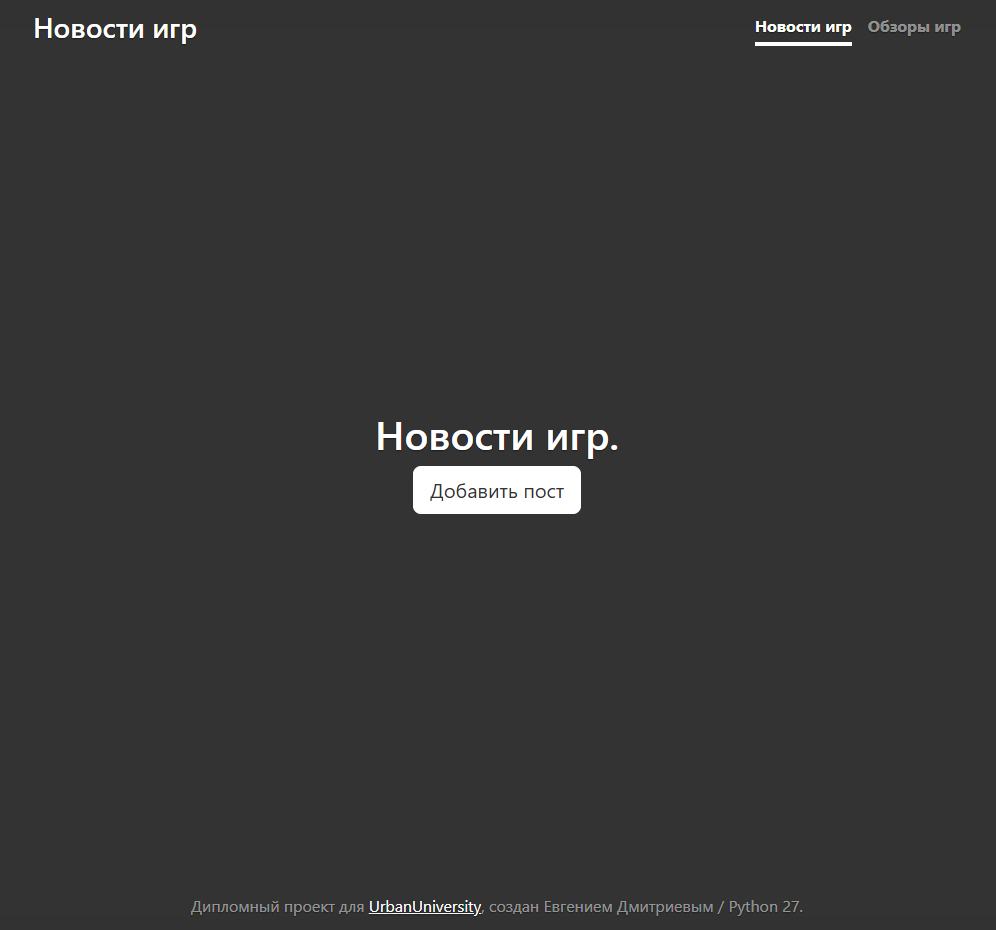
Планирование задач и отслеживание прогресса производилось в Microsoft office.

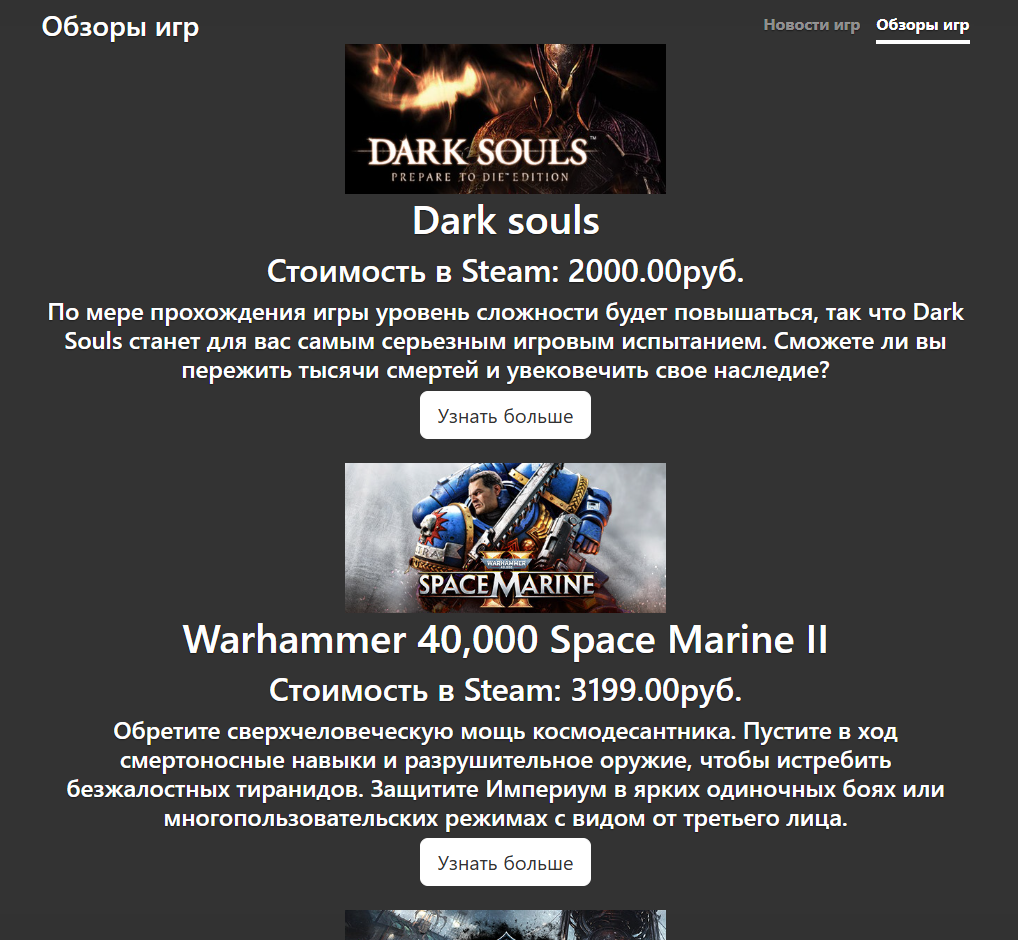
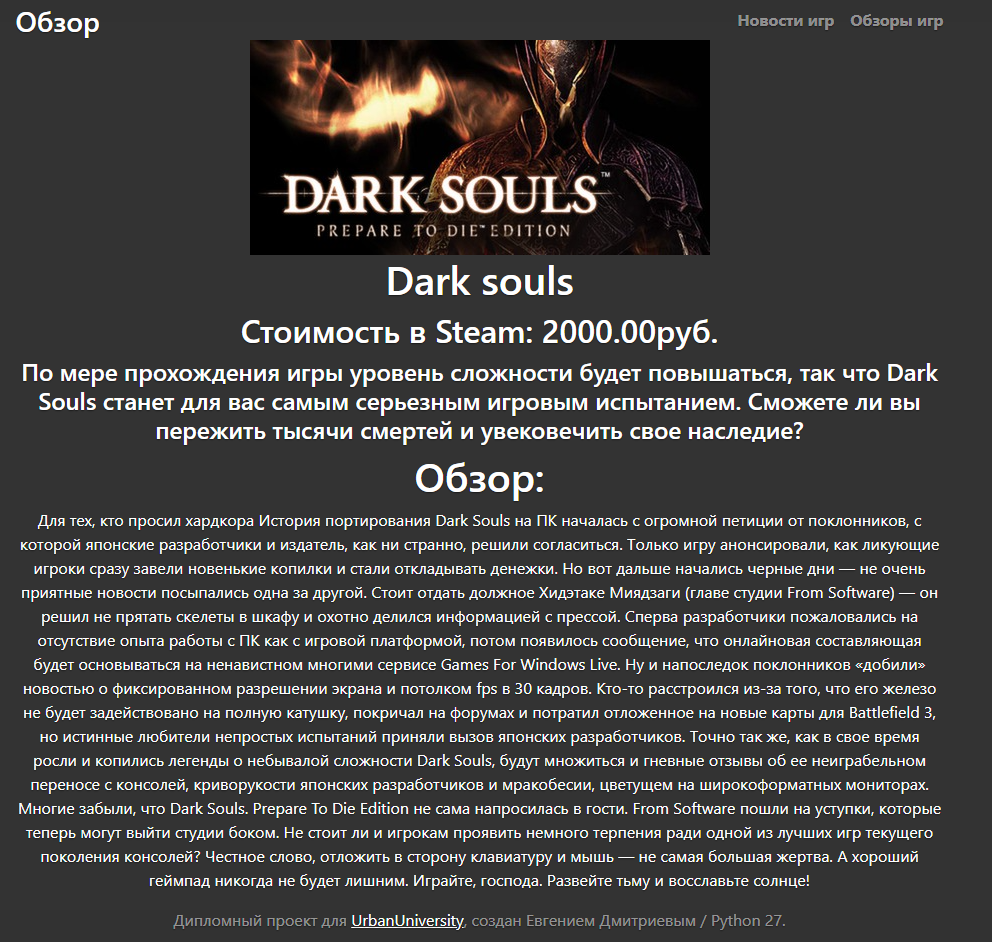
**Разработка.**

Фронтенд разработка:

Создан пользовательский интерфейс с использованием Bootstrap для стилезации.

Разработаны форма приема постов для новостного блока, и страницы отображения общей информации и информации по объекту.

Рис. 1. Форма отправки постов для новостного блока.Рис. 2. Страница отображения всех постов, созданных пользователем.

Рис.3. Страница отображения информации о играх.Рис. 4. Страница полного отображения игры из базы данных.

**Бэкенд разработка:**

Реализована серверная логика по фреймворка:Django, Flask и FastAPI.

Настроены все маршруты, модели и представления.  
Подключена база данных db.sqlite3

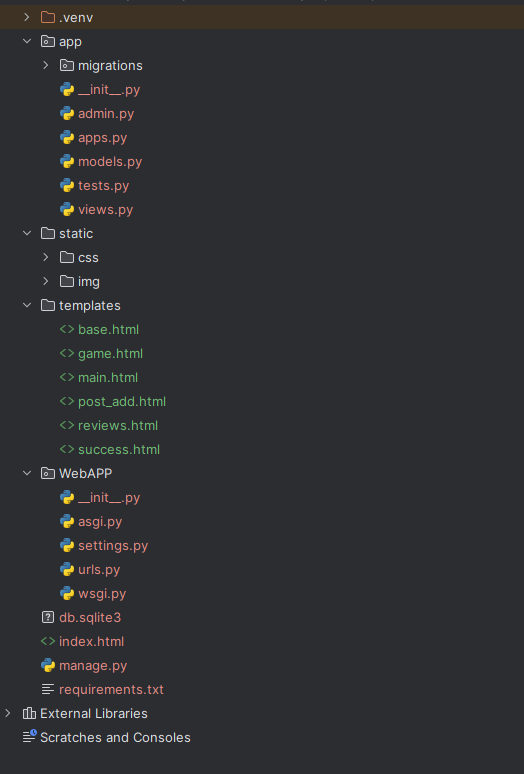


Рис.5. Файловая структура Django

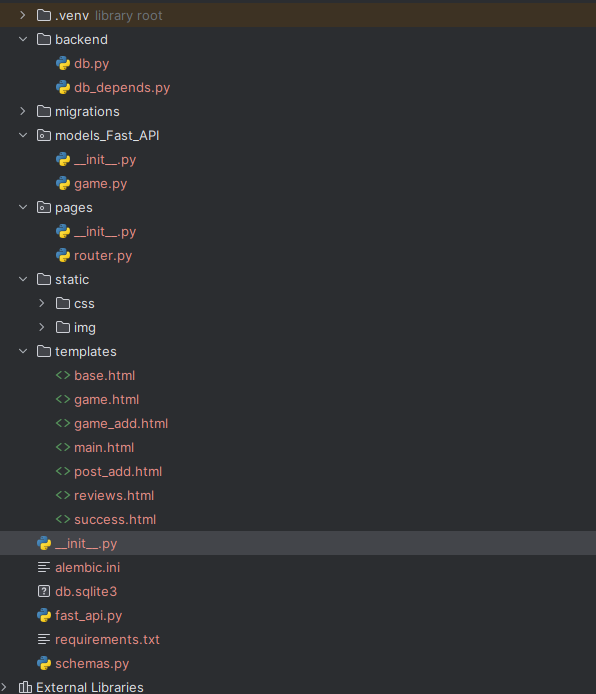


Рис. 6. Файловая структура FastAPI

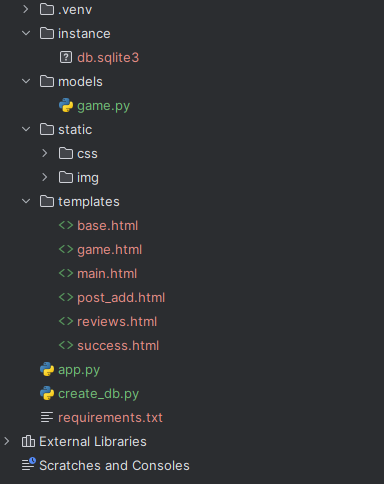


Рис. 7. Файловая структура Flask

**7.Анализ и интерпретация результатов**

**Сравнение фреймворков.**

**Django**

* Хорошо структурированный фреймворк
* Много настроек для работы с проектами
* Легкий в освоении, но занимает больше времени для выстраивания проекта
* Панель администратора очень сильно помогает в работе

**Flask**

* Интуитивно понятный
* Легко строится проект
* Весьма пластичный и легко настраиваемый
* Легко работать с базами данных

**FastAPI**

* Асинхронность очень удобно встроена в фреймворк
* Встроенный Swagger дает понимание о ваших действующих функциях
* Оптимальный выбор для малых проектов
* Не очень удобный доступ к миграциям базы данных

**Интерпретация результатов**

Django лучше показал себя для работы с проектами за счет своей структурности и понятным Traceback-ом

Flask легко подстраивался в уже написанные веб-приложения и показал хорошие результаты

FastAPI был несколько топорным и сложно перестраивался для того же функционала в веб-приложении, но имел неоспоримый плюс в создании страниц под отдельные объекты

**Рекомендации по выбору фреймворка в зависимости от проекта**

Для стабильного и качественного проекта рекомендуется использовать Django так как его структура будет направлять вас в правильном направлении. Плюс он интуитивно понятен и освоить его на базовом уровне не составит труда так как есть много литературы и гайдов по работе с данным фреймворком.  
  
Flask же позволит вам сделать небольшой стабильный проект с наименьшей время затратой без проблем с подключением баз данных и их миграцией.

FastAPI я бы поставил на третье место по удобству, но его неоспоримым плюсом является встроенная асинхронность и возможность без дополнительных модулей использовать Swagger.

**8. Заключение**

**Обзор выполненной работы**

Проектирование и разработка веб-приложения на фреймворках: Django, Flask и FastAPI были успешно завершены в соответствии с изначально созданной документацией. Приложение включает в себя функционал получение и оправки CRUD запросов.

Реализованное приложение соответствует требованиям и демонстрирует стабильную работу на всех вышеперечисленных фреймворках.

Отдельно отмечу фреймворк Django как наиболее удобный и стабильный среди прочих.

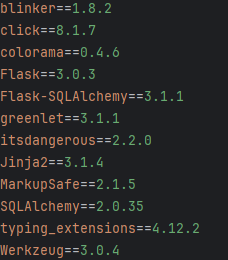
**Дальнейшие планы**

Планируется совершенствование имеющегося приложении и введение туда системы аутентификации, регистрации и взаимодействия пользователя с объектами базы данных.

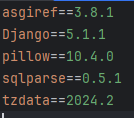
Так же планируется расширить функционал добавления постов.

**Приложение 1. Requirements**

Flask:



Django:



FastAPI:

