Urban University

Дипломная работа по курсу "Python-разработчик"

Тема: Анализ и сравнение написания web-приложений с использованием разных фреймворков

# Содержание

[Содержание 2](#_Toc185663858)

[1. Обзор проекта 2](#_Toc185663859)

[2. Структура проекта 2](#_Toc185663860)

[3. Домашняя страница 2](#_Toc185663861)

[4. Дашборд 3](#_Toc185663862)

[5. Страница авторизации 3](#_Toc185663863)

[6. Шаблоны страниц 3](#_Toc185663864)

[7. Заключение 3](#_Toc185663865)

[Приложение 1. Пример файловой структуры проекта 3](#_Toc185663866)

[Приложение 2. Список необходимых библиотек 3](#_Toc185663867)

# Введение

Быстрый рост популярности веб-сайтов и приложений в сети Интернет связан с современным уровнем развития информационных технологий. Современные веб-приложения обладают огромными функциональными возможностями, позволяя автоматизировать и усовершенствовать множество выполняемых процессов.

Учитывая достаточно большое количество фреймворков, разработчикам становится достаточно сложно выбрать один из них для разработки конкретного приложения. В дипломной работе проводится анализ и сравнение разработки веб-приложений с использованием трех популярных фреймворков Python: Django, Flask и FastAPI. Работа включает в себя создание простых веб-приложений на каждом из фреймворков, а также оценку их производительности, удобства разработки и доступной документации. Исследуется, как каждый из этих фреймворков справляется с различными задачами, а также рассматриваются реальные примеры использования на практике. В итоге работу завершает обобщение полученных результатов и рекомендации по выбору подходящего инструмента в зависимости от специфики проектов.

Python – это интерпретируемый язык программирования, который становится все более популярным для создания веб-приложений. Тем не менее, существует множество веб-фреймворков на выбор, но все они имеют разные варианты использования.

Django следует архитектурному шаблону model-template-view и принципу DRY (не повторяйтесь).

Flask основан на Werkzeug WSGI toolkit и движке шаблонов Jinja2.

FastAPI это микро-веб-фреймворк, основанный на Pydantic.

В дипломной работе приводится сравнение этих фреймворков на основе различных факторов, таких как эффективность, удобочитаемость, варианты использования и т.д.

Django – это веб-фреймворк с полным стеком с открытым исходным кодом, написанный на Python, который следует архитектурному шаблону MTV (модель-шаблон-представление) и принципу DRY (не повторяйтесь). Оно было сделано Адрианом Головати и Саймоном Уиллисоном и было публично опубликовано в 2005 году. Данное название основано на имени певца Джанго Рейнхардта.

Flask – это удобный для начинающих веб-фреймворк с открытым исходным кодом, написанный на Python, который следует архитектурному шаблону MVC (модель-представление-контроллер) и поддерживает шаблоны Jinja, которые можно использовать для создания представлений. Оно было сделано Армином Ронахером из Pocoo, международной группы энтузиастов Python, созданной в 2004 году. Название "Flask" является отсылкой к более раннему фреймворку Bottle.

FastAPI также является микро-веб-фреймворком Starlette-ASGI (Asynchronous Server Gateway Interface) с открытым исходным кодом, написанным на Python и используемым для создания RESTful API. Оно было сделано Себастьяном Рамиресом в декабре 2018 года. Название основано на скорости разработки API, отсюда и "FastAPI".

*Цель*

Обоснование выбора фреймворка для разработки веб-приложений на Python на основе их сравнительного анализа.

*Задачи*

1. Изучить основные характеристики фреймворков Django, Flask и FastAPI;

2. Разработать аналогичные простые веб-приложения на каждом из фреймворков;

3. Провести сравнение по критериям производительности и удобства разработки.

Сравнение FastAPI с Django

| **Параметры** | **Django** | **FastAPI** | **Flask** |
| --- | --- | --- | --- |
| **Тип** | Веб-фреймворк с полным стеком | Микро-веб-фреймворк | Микро-веб-фреймворк |
| **Пример использования** | Создание сложных веб-приложений и API | Создание API и микросервисов | Идеально подходит для создания небольших веб-приложений и простых API |
| **Производительность** | Быстрый способ создания большого веб-приложения | Очень быстрый для создания API и микросервисов | Медленнее из-за ручной проверки и синхронизированного программирования. |
| **Масштабируемость** | Масштабируемый, но его ORM и шаблонизатор могут замедлить масштабирование | Высокая масштабируемость, поскольку в нем используется асинхронизация кода и изменение типов. | Сложно масштабировать, поскольку отсутствует встроенная поддержка ORM или кэширования. |
| **Кривая обучения** | Сложный для начинающих | Просто для начинающих | Умеренный для начинающих |
| **Инструменты для работы с базами данных** | Полный набор | Ограничено, встроенной поддержки нет | Ограничено, встроенной поддержки нет |
| **Асинхронное программирование** | Да, это можно сделать с помощью Asyncio, но немного медленнее. | Да, но быстрее благодаря Pydantic. | Нет, но можем ли мы использовать другие библиотеки. |
| **ORM (Объектно-реляционное отображение)** | ДА | НЕТ | НЕТ |
| **Сообщество** | Большой и активный | Небольшой, но растущий | Большой и активный |
| **Документация** | Огромный | Небольшой, но все еще растущий | Большой |
| **Преимущества** | Это отличный выбор для тех, кто хочет создавать безопасные, масштабируемые, гибкие и простые в обслуживании веб-приложения | Это отличный выбор для создания высокопроизводительных API и микросервисов | Это отличный выбор для создания веб-приложений малого и среднего уровня, где производительность не является большой проблемой и разработчикам требуется гибкость. |
| **Недостатки** | Это может быть сложным для начинающих, сложным в отладке и не подходящим для небольших проектов | Основные файлы могут быть перегружены и отсутствует встроенная защита. | Отсутствует встроенная поддержка кэширования, ORM, асинхронизации и т. Д |

# Основные понятия и определения

Обзор основных понятий в области написания web-приложений с использованием разных фреймворков:

1. *Фреймворк (Framework):* Программная платформа, которая предоставляет готовые компоненты и инструменты для разработки приложений. В дипломной работе используются Django, Flask и 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 и демонстрации работы фреймворков Django, Flask и FastAPI.

3. *Бэкенд (backend)* – серверная часть веб-приложения, обрабатывающая запросы от клиента, выполняющая логику приложения и взаимодействующая с базой данных.

4. *Фронтенд (frontend)* – клиентская часть веб-приложения, взаимодействующая непосредственно с пользователем, отвечающая за отображение данных и предоставления пользователю интерфейса взаимодействия с приложением.

5. *ORM (Object-Relational Mapping, объектно-реляционное отображение)* – технология взаимодействия с базой данных через объектно-ориентированный интерфейс, скрывая детали SQL-запросов.

*6. Шаблонизатор* – программное обеспечение, позволяющее использовать шаблоны для генерации конечных документов с помощью декларативного языка разметки.

7. *Web-сокет* – протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и web-сервером, используя постоянное соединение.

8. *Диспетчеризация* – это процесс поиска и вызова необходимой функции или метода для уже известного типа данных.

9. *API (application programming interface) –* программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими (в противоположность пользовательскому интерфейсу, используемому для взаимодействия конечного пользователя с программой). Проще говоря, это способ взаимодействия какого-то программного кода с набором каких-то программных компонентов, с помощью которых одна компьютерная программа (например, бот или сайт) может использовать другую программу.

Данные термины помогают понять ключевые аспекты в области написания web-приложений с использованием разных фреймворков.

# Обзор проекта

# Структура проекта

# Домашняя страница

# Дашборд

# Страница авторизации

# Шаблоны страниц

# Заключение

Таким образом, теперь мы можем сделать вывод, что FastAPI можно использовать из-за его высокой производительности при разработке API, Flask можно использовать для создания небольших и средних проектов с большой гибкостью, а Django - для создания многофункциональных крупномасштабных приложений.

# Приложение 1. Пример файловой структуры проекта

# Приложение 2. Список необходимых библиотек