Дипломная работа по теме: Анализ и сравнение написания web-приложений с использованием разных фреймворков

Автор: Кольцова Валерия Дмитриевна

Содержание

[1. Введение 3](#_Toc186210871)

[Обоснование выбора темы 3](#_Toc186210872)

[Определение цели и задачи исследования 3](#_Toc186210873)

[2. Основные понятие и определения 4](#_Toc186210874)

[Обзор основных понятий в области разработки простых приложений 4](#_Toc186210875)

[3. Обзор популярных инструментов для разработки разработки веб-приложений на Python 5](#_Toc186210876)

[Django 5](#_Toc186210877)

[Flask 5](#_Toc186210878)

[FastAPI 6](#_Toc186210879)

[4. Проектирование приложения 7](#_Toc186210880)

[5. Разработка в соответствии с созданной документацией 8](#_Toc186210881)

[Django 8](#_Toc186210882)

[FastAPI 11](#_Toc186210883)

[Flask 12](#_Toc186210884)

[6. Заключение 16](#_Toc186210885)

# Введение

## Обоснование выбора темы

Использование фреймворка Python – верный способ ускорить и оптимизировать разработку. Фреймворки Python не являются обязательными, но также было бы неразумно пренебрегать возможностью улучшить процесс разработки там, где это возможно.

Фреймворки повышают скорость написания кода и позволяют сосредоточиться на логике приложения, а не на написании базовых функций. Это наборы различных пакетов и модулей, которые помогают программисту. Главная задача фреймворка – дать возможность сэкономить время на написании простых команд, что существенно повышает скорость разработки.

Существует огромное количество фреймворков для веб-разработки на Python. Выбор конкретного решения зависит от стоящих перед разработчиком задач и личных предпочтений. Необходимо каждый раз оценивать выбираемый фреймворк на соответствие требованиям конкретной задачи.

## Определение цели и задачи исследования

Цель исследования: провести сравнение популярных фреймворков Django, Flask и FastAPI.

Задачи исследования:

1. Обзор фреймворков для написания веб приложений. Провести анализ существующих инструментов, выбрать наиболее подходящие под имеющуюся задачу.  
2. Разработка простых веб-приложений на данных фреймворках для сравнения.  
3. Написание дипломной работы.

# Основные понятие и определения

## Обзор основных понятий в области разработки простых приложений

**Веб-разработка** – это процесс создания и поддержки веб-сайтов и веб-приложений.

**Веб-приложение (Web Application)**: Программное приложение, которое работает на веб-сервере и доступно через браузер.

**Фреймворк** – это набор инструментов, компонентов и методов, которые облегчают разработку программного обеспечения.

**Backend** – это часть веб-разработки, которая отвечает за обработку данных и взаимодействие с базами данных и серверами.

**Frontend** – это часть веб-разработки, которая отвечает за внешний вид и взаимодействие с пользователем. Frontend-разработчики используют HTML, CSS и JavaScript для создания интерфейсов.

**HTML** (HyperText Markup Language) – основной язык разметки веб-страниц. С помощью HTML разработчик определяет структуру и содержимое страницы.

**CSS** (Cascading Style Sheets) – язык описания внешнего вида веб-страниц. CSS позволяет определить цвета, шрифты, отступы, фоны и другие стилевые элементы.

Object-Relational Mapping **(ORM)** – это технология, которая позволяет разработчикам взаимодействовать с базами данных.

# 3. Обзор популярных инструментов для разработки веб-приложений на Python

Основные фреймворки – Django, Flask, FastAPI, ниже более подробное описание.

## Django

[Django](https://www.djangoproject.com/) – это фреймворк с открытым исходным кодом, который позволяет разработчикам создавать веб-приложения практически любого уровня. Он входит в число лучших фреймворков Python и заслуженно пользуется популярностью.

Для Django доступен колоссальный набор дополнительных библиотек и поддержка огромного сообщества разработчиков, что существенно упрощает и ускоряет разработку.

Ключевые особенности Django:

* наличие собственного ORM;
* встроенный административный интерфейс);
* шаблонизатор;
* библиотека работы с формами;
* система кэширования и интернализации;
* система авторизации и аутентификации.

## Flask

Flask – это легкий фреймворк для создания веб-приложений на Python. В отличие от Django, Flask предоставляет только минимальный набор инструментов, что позволяет разработчикам выбирать дополнительные библиотеки и расширения по мере необходимости. Это делает Flask идеальным выбором для проектов меньшего размера или для тех, кто хочет больше контроля над архитектурой приложения.

Ключевые особенности Flask:

* встроенный сервер разработки и отладчик;
* использование шаблонизатора Jinja2;
* множество расширений, предоставляемых сообществом.

## FastAPI

[FastAPI](https://fastapi.tiangolo.com/) – это асинхронный веб-фреймворк, который появился на свет в конце 2018 года. Он предназначен для быстрой разработки API-приложений и основан на использовании стандартной аннотации типов Python.

FastAPI обычно применяется для разработки API, без фронтенда, но при желании его можно дополнить стандартными шаблонами на основе шаблонизатора, например Jinja2, который применяется во Flask.

Ключевые особенности FastAPI:

* 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.

# Проектирование приложения

Планирование и анализ требований:

1. определение структуры приложения: разработка схемы архитектуры, включающей фронтенд, бэкенд, базу данных;
2. реализация основного функционала: создание базовой версии приложения с минимально необходимым функционалом для тестирования и демонстрации.
3. разработка ключевого функционала;
4. реализация пользовательского интерфейса.

Технические требования:

1. фронтенд: использовать HTML, CSS;
2. бэкенд: Python с использованием фреймворков Django, FastAPI, Flask.
3. база данных: использование базы данных для хранения данных о статьях;
4. API: Разработка API для взаимодействия между фронтендом и бэкендом.

# 5. Разработка в соответствии с созданной документацией

## Django

Реализована серверная логика с использованием фреймворка Django.  
На рисунке 5.1 представлена файловая структура приложения.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 5.1 – Файловая структура приложения

На рисунке 5.2 представлены маршруты для вывода данных.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 5.2 – Маршруты

Создание и редактирование элементов базы данных происходит через панель администратора (рисунок 5.3). В данном проекте через нее можно редактировать меню и новости.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 5.3 – Панель администратора

На рисунке 5.4 показан раздел меню. Выбор категории реализован с помощью кнопок, которые ведут в необходимый раздел (рисунок 5.5). Также реализовано переключение страниц, так как для страницы задано значение: не более двух записей.

Изображение выглядит как текст, салат, овощи, блюдо

Автоматически созданное описание

Рисунок 5.4 – Раздел меню

Изображение выглядит как текст, снимок экрана, посуда, еда

Автоматически созданное описание

Рисунок 5.5 – Раздел с салатами

## FastAPI

На рисунке 5.6 представлена файловая структура приложения.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 5.6 – Файловая структура приложения.

В отличии от Django маршруты в FastAPI маршруты прописываются сразу к функциям (рисунок 5.7).

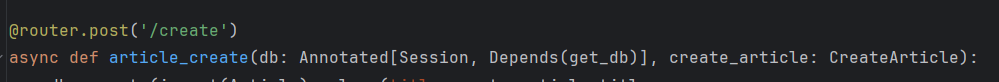


Рисунок 5.7 – Маршруты

Для просмотра логики приложения можно воспользоваться Swagger UI (рисунок 5.8). Пользовательский интерфейс на основе браузера, который в интерактивном режиме документирует API.

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Рисунок 5.8 – Swagger UI

## Flask

На рисунке 5.9 представлена файловая структура приложения.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5.9 – Файловая структура приложения.

Также, как и FastAPI маршруты в Flask маршруты прописываются сразу к функциям (рисунок 5.10)

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Рисунок 5.10 – Маршруты

Реализована форма создания статьи. Показанная на рисунке 5.11. После создания пользователя перенаправляет на страницу со всеми статьями (рисунок 5.12).

Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Автоматически созданное описание

Рисунок 5.11 – Форма создания статьи

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5.12 – Страница со всеми статьями

Статьи можно редактировать и удалять (рисунок 5.13). При удалении спрашивается согласие на удаление (рисунок 5.14).

Изображение выглядит как текст, снимок экрана, Шрифт, Графика

Автоматически созданное описание

Рисунок 5.13 – Выбор действий со статьей

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок 5.14 – Удаление статьи

Таже реализован поиск по названию статьи (рисунок 5.15) однако для того, чтобы статья была найдена, запрос должен быть идентичным названию (чувствителен к регистру и отсутствию слов).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 5.15 – Поиск

# 6. Заключение

В данной работе было реализованы простые веб-приложения для работы со статьями на трех фреймворках.

Для разработки простых веб-приложений как в данном дипломном проекте отлично подходит фреймворк Flask, однако если в дальнейшем развивать данные приложения, то предпочтительнее будут фреймворки Django и FastAPI.

Фреймворк Django удобен, благодаря встроенного административного интерфейса, также реализована удобная работа с базой данных.

FastAPI как следует из названия делает упор на скорость. Фреймворк поддерживает асинхронность. Благодаря асинхронному коду запросы выполняются независимо друг от друга и могут запускаться параллельно. Также FastAPI предлагает чрезвычайно удобную систему автоматического документирования. Он предоставляет пользовательский интерфейс на основе браузера, который в интерактивном режиме документирует API на основе графического пользовательского интерфейса Swagger UI .