***Дипломная работа на тему:***

**1. Анализ и сравнение написания web-приложений с использованием разных фреймворков: Разработать простые веб-приложения с использованием Django, Flask и FastAPI, провести их сравнение.**

**Проект: интернет-магазин**

Автор: Турченко Алексей

**Обоснование выбора темы:**

1. В современном мире любая компания, которая хочет быть “глобальной” или просто повысить узнаваемость и привлечь клиентов из нового канала трафика, обязана иметь веб-сайт.

2. **Потребности рынка:** веб-разработчики всегда будут иметь высокий спрос на рынке так как сайт требуется всем видам бизнеса, от малого, до корпораций.

3. **Практическая значимость:** разработка интернет-магазина позволяет прокачать себя в нескольких фреймворках на Python, а так же затронуть сопутствующие технологии, такие как: HTML, CSS, JS

4. **Личный интерес и потенциальные перспективы**: Личный интерес вызван тем, что в прошлом я много обрабатывал заявок с сайтов, а так же руководил обработкой этих заявок, а теперь хочу помочь бизнесу привлекать эти заявки.

Таким образом, выбор проекта "**Создание интернет-магазина**" обусловлен ее актуальностью, потребностями рынка, практической значимостью и личным интересом, что делает эту тему подходящей для проведения дипломной работы.

**Определение цели и задач исследования:**

**Цель исследования:** Написать интернет-магазин на трех фреймворках и сравнить фреймворки для его написания

**Задачи исследования:**

1. **Обзор фреймворков** для разработки интернет магазина

2. **Разработка сценариев сравнения:** Определить критерии сравнения и на основании их провести аналитику

3. **Создание платформы и проведение тестирования:** реализовать интернет-магазин и протестировать его функционал

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 технологий для написания интернет-магазина

**2. Основные понятия и определения**

**Обзор основных понятий в интернет-магазине:**

1. Система управления базами данных (СУБД): Программное средство, предназначенное для организации хранения и обработки структурированных данных, таких как информация о продуктах, заказах и клиентах интернет-магазина.

2. Корзина покупок (Shopping Cart): Функционал интернет-магазина, позволяющий пользователям добавлять товары для последующего оформления заказа и покупки.

3. Веб-приложение (Web Application): Программное обеспечение, обеспечивающее доступ к функционалу интернет-магазина через веб-браузер с помощью веб-сервера.

4.Товары в интернет-магазине представляют собой физические или цифровые продукты, которые предлагаются для продажи на веб-сайте. Товары могут включать в себя различные категории, такие как одежда, обувь, электроника, косметика, товары для дома, книги, игрушки и многие другие. Каждый товар обычно имеет свою уникальную страницу с описанием, изображениями, ценой, характеристиками, возможностью добавления в корзину и оформления заказа. Интернет-магазины предлагают широкий ассортимент товаров, позволяя покупателям выбирать и покупать необходимые продукты онлайн, а также предоставляют удобные способы оплаты и доставки.

**3. Методы и подходы к разработке**

При разработке веб-приложения с использованием технологий Django, Flask и FastAPI возможно применение различных методов и подходов, архитектурных концепций и паттернов, которые определяют структуру, организацию и функциональность приложения.

**1. Django**

Django - это высокоуровневый веб-фреймворк на языке программирования Python, который предлагает полноценный набор инструментов для разработки веб-приложений. Основные методы и подходы к разработке с использованием Django включают:

- Модель-Представление-Контроллер (Model-View-Controller, MVC) - Django следует паттерну проектирования MVC, где модели представляют данные приложения, представления обрабатывают отображение данных и контроллеры управляют бизнес-логикой.

- ORM (Object-Relational Mapping) - Django предлагает ORM для удобной работы с базой данных, позволяя взаимодействовать с базой данных через объекты Python, а не через SQL запросы.

- Архитектура проекта - Django рекомендует использование шаблона проекта, который включает разделение приложения на модули, настройку маршрутизации URL, работу с шаблонами и статическими файлами.

**2. Flask**

Flask - минималистичный фреймворк для создания веб-приложений на Python, который предоставляет базовые инструменты для разработки. Методы и подходы к разработке с использованием Flask включают:

- Микрофреймворк - Flask предлагает минимальный набор инструментов, что позволяет разработчику выбирать необходимые библиотеки и компоненты для проекта.

- RESTful API - с помощью расширений, таких как Flask-RESTful, Flask позволяет легко создавать RESTful API для взаимодействия с клиентскими приложениями.

- Расширяемость - Flask поддерживает модульность и позволяет легко интегрировать различные библиотеки и расширения для расширения функциональности приложения.

**3. FastAPI**

FastAPI - современный веб-фреймворк на языке Python, который отличается высокой производительностью и мощным функционалом. Методы и подходы к разработке с использованием FastAPI включают:

- Асинхронное программирование - FastAPI поддерживает асинхронные запросы и может эффективно обрабатывать большие нагрузки данных.

- Проверка типов данных - FastAPI использует Pydantic для проверки и валидации данных, что упрощает разработку и обеспечивает безопасность приложения.

- Автодокументация API - FastAPI автоматически генерирует документацию API на основе аннотаций типов данных, что упрощает понимание и использование API.

**Организация БД:**

Token: id, token, user

User: id, username, password, is\_admin

Product: id, name, article, description, price, photo, date

**Пути на сайте:**

/catalog - страница с каталогом

/products/add - добавление продукта

/product/delete/{id} - удаление продукта по id

/product/{id} - страница конкретного товара

/delete\_by\_art - удаление по артикулу

/authorize - авторизация

/login - регистрация

Особенности фреймворков:

Django:

- Полнофункциональный веб-фреймворк: Django предоставляет все необходимые компоненты для создания веб-приложений, включая модели данных, маршрутизацию, представления, шаблоны и административный интерфейс.

- Акцент на разработку быстрыми темпами: Django ориентирован на быструю разработку приложений, предоставляя множество "готовых к использованию" компонентов и инструментов.

- Объектно-реляционное отображение (ORM): Django включает в себя мощную ORM-систему, позволяющую работать с базами данных на высоком уровне абстракции, без необходимости использовать SQL напрямую.

- Высокая масштабируемость и безопасность: Django обеспечивает высокую производительность и безопасность приложений "из коробки", что делает его отличным выбором для крупных проектов.

- Обширное сообщество и обширная экосистема: Django имеет большое и активное сообщество разработчиков, а также множество готовых модулей и расширений, упрощающих разработку.

Flask:

- Микрофреймворк: Flask - это "микрофреймворк", который предоставляет только основные функциональные возможности, оставляя разработчику больше свободы в выборе дополнительных библиотек и инструментов.

- Простота и гибкость: Благодаря своей минималистичности, Flask отличается простотой в освоении и настройке, предоставляя разработчикам больше свободы в выборе архитектуры и компонентов.

- Легковесность и быстрота: Из-за своей простоты и отсутствия "тяжелой" функциональности "из коробки", Flask отличается высокой производительностью и быстротой работы.

- Модульность и расширяемость: Flask построен на основе модульного дизайна, что позволяет легко расширять его функциональность с помощью сторонних расширений.

- Подходит для небольших и средних проектов: Flask идеально подходит для небольших и средних веб-приложений, где требуется быстрая разработка и гибкая настройка.

FastAPI:

- Быстрый и производительный: FastAPI основан на асинхронном фреймворке Starlette и библиотеке Pydantic, что обеспечивает высокую производительность и быстроту работы приложений.

- Автоматическая документация API: FastAPI автоматически генерирует интерактивную документацию API (Swagger/OpenAPI) без дополнительных усилий со стороны разработчика.

- Поддержка типизации Python: FastAPI использует возможности статической типизации Python, что позволяет избежать ошибок и улучшает читаемость кода.

- Простота и удобство использования: FastAPI ориентирован на быструю разработку и интуитивно понятен для разработчиков, знакомых с Python.

- Кроссплатформенность и совместимость с ASGI: FastAPI работает поверх ASGI-серверов, таких как Uvicorn, Deta и Gunicorn, что обеспечивает его кроссплатформенность и возможность масштабирования.

**5. Проектирование приложения**

**Планирование и анализ требований**

Выбор фреймворка и инструментов: Django, Flask и FastAPI

**План действий:**

1. Структура проекта
2. БД
3. Реализация основного функционала: Создание базовой версии приложения с минимально необходимым функционалом бэкенда
4. Разработка фронтенд части проекта

**Требования проекта:**

**Разработка на трех фреймворках:**

Django, Flask и FastAPI

**Функционал:**

/catalog - страница с каталогом

/products/add - добавление продукта

/product/delete/{id} - удаление продукта по id

/product/{id} - страница конкретного товара

/delete\_by\_art - удаление по артикулу

/authorize - авторизация

/login - регистрация

**Бэкенд**: Python.фреймворки:

Django, Flask и FastAPI

**СУБД**: SQlite

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

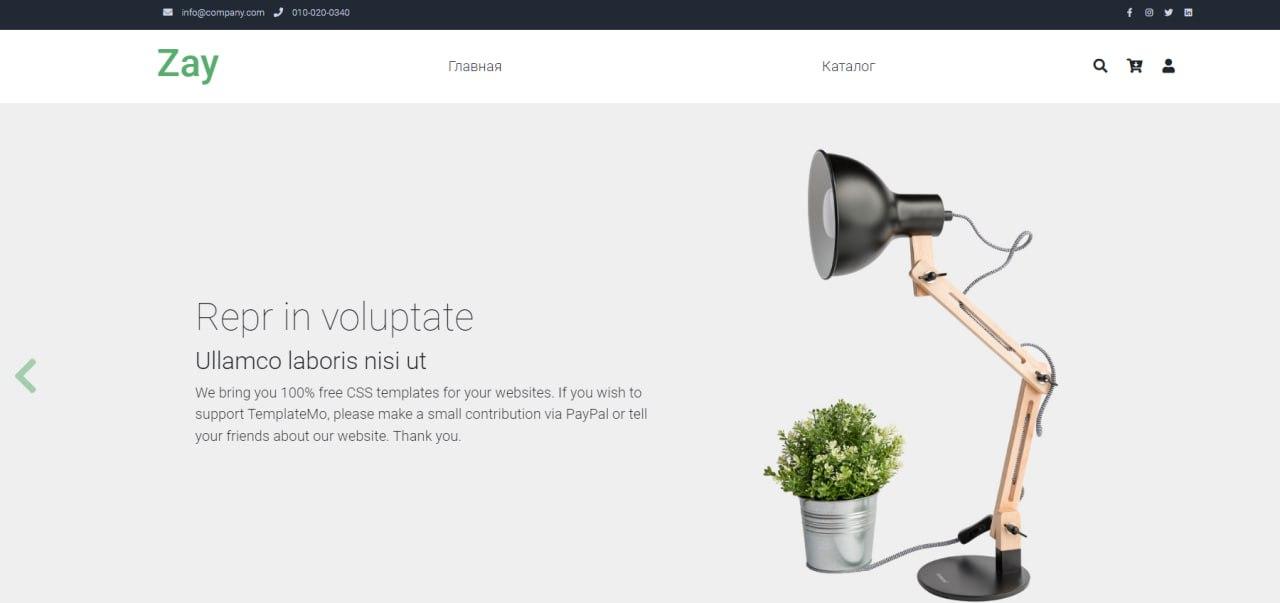
**Планирование разработки**

**Разработка была разделена на несколько основных этапов:**

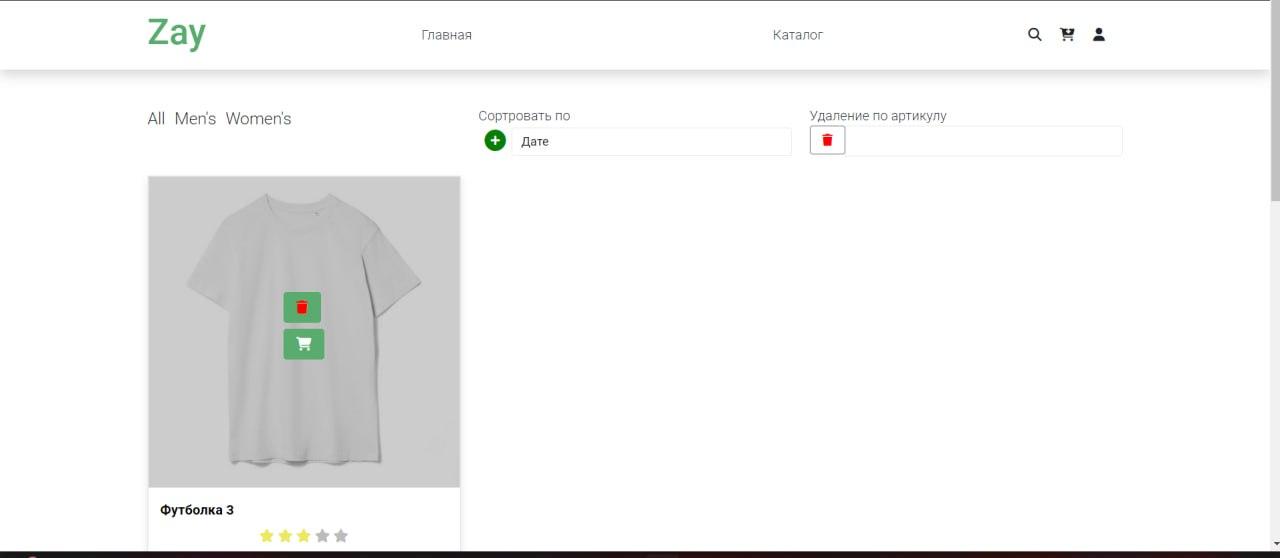
1. Проектирование
2. Бэкенд
3. Фронтенд

**Реализация продукта:**

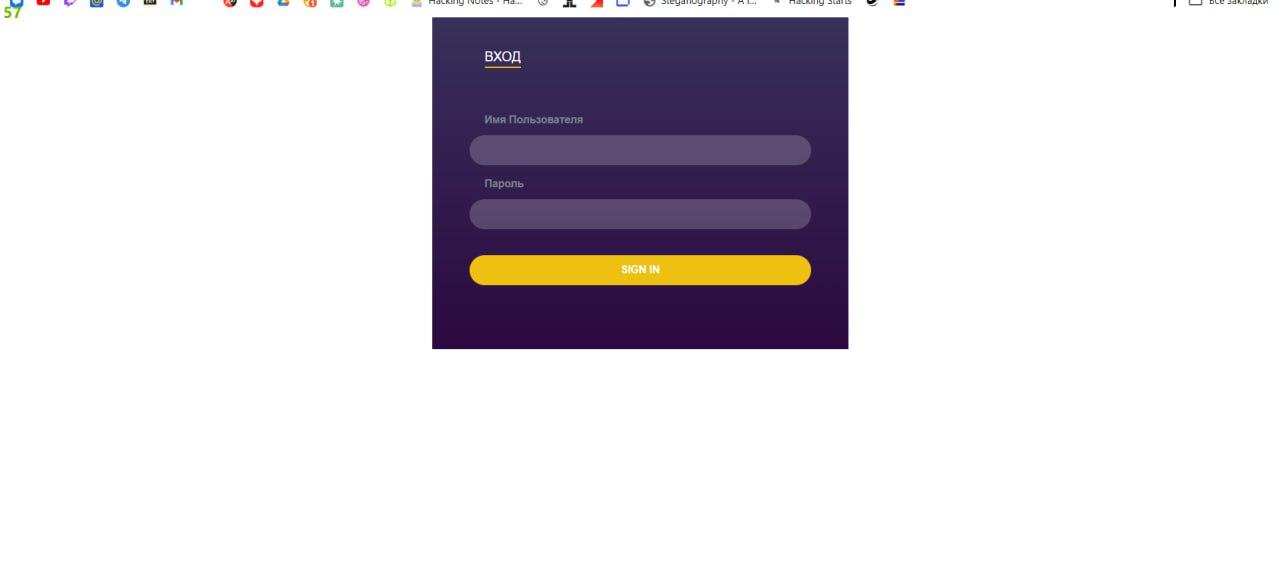
Главная страница



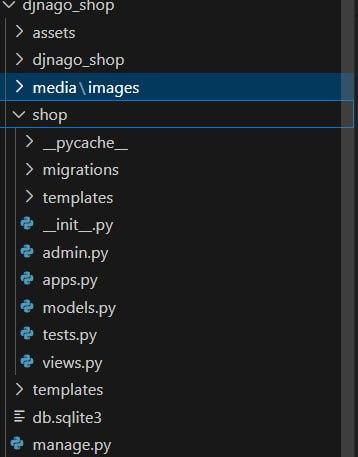
Редактирование товаров: Админ-панель

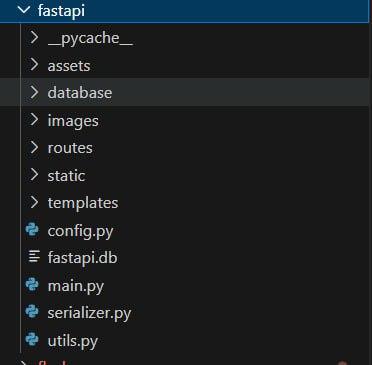


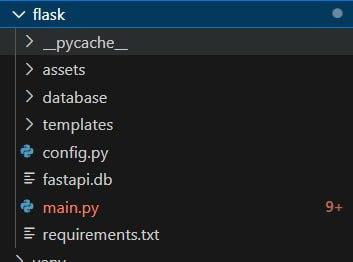
Авторизация



Структура проекта







**Сравнения технологий**

**Критерии:**

1. Сложность освоения:

- Насколько сложно новичку начать работать с данным фреймворком.

2. Производительность:

- Скорость работы приложений, построенных на основе данного фреймворка.

3. Функциональность:

- Набор встроенных возможностей, предоставляемых фреймворком.

4. Модульность:

- Возможность расширения функциональности с помощью сторонних модулей/библиотек.

5. Сообщество и документация:

- Активность сообщества разработчиков и качество документации.

6. Сферы применения:

- Для каких задач лучше всего подходит данный фреймворк.

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий** | **Django** | **Flask** | **FastAPI** |
| **Сложность освоения** | Средняя | Низкая | Средняя |
| **Производительность** | Средняя | Высокая | Высокая |
| **Функциональность** | Высокая | Низкая | Высокая |
| **Модульность** | Высокая | Высокая | Высокая |
| **Сообщество и документация** | Высокая | Средняя | Высокая |
| **Сферы применения** | Веб-приложения, API | Микросервисы, API | Быстрые API |

**Вывод:**

1. Django является наиболее полнофункциональным фреймворком из представленных, подходящим для создания полноценных веб-приложений и API. Он имеет большое и активное сообщество, обширную документацию, но при этом требует некоторых усилий для освоения.

2. Flask является более легким и простым в освоении фреймворком, обладающим высокой производительностью. Он подходит для создания микросервисов и API, но имеет меньшую функциональность по сравнению с Django.

3. FastAPI - новый, быстрый и высокопроизводительный фреймворк для создания API. Он сочетает в себе простоту Flask и функциональность Django, предоставляя современные инструменты для построения эффективных API-сервисов. FastAPI активно развивается и имеет растущее сообщество.

Выбор фреймворка будет зависеть от конкретных требований проекта и предпочтений разработчиков. Django подойдет для комплексных веб-приложений, Flask - для небольших API и микросервисов, а FastAPI - для быстрых и эффективных API-решений.