Онлайн-университет Urban

Дипломная работа

тема:

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

# Дипломную работу выполнил:

# Шабанов Александр Владимирович

# 

Санкт-Петербург

2024 г.

Содержание**:**

1. ВВЕДЕНИЕ 3

* Обоснование выбора темы 3
* Определение цели и задач исследования 4

1. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ 5
2. ОБЗОР ПОПУЛЯРНЫХ ИНСТРУМЕНТОВ ДЛЯ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ НА PYTHON 6

* Django 6
* Flask 7
* FastAPI 7

4. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 9

* Основные требования 9
* Требования к архитектуре веб-приложения 10

5. РАЗРАБОТКА В СООТВЕТСТВИИ С ТРЕБОВАНИЯМИ. 10

* Фронтенд-разработка 10
* Бэкенд-разработка 11
* Интеграция базы данных 11

6. АНАЛИЗ И ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ 13

* Сравнение фреймворков 13
* Интерпретация результатов 14

7. ЗАКЛЮЧЕНИЕ 15

* Обзор выполненной работы 15
* Планы по развитию веб-приложения 15

1. ИСПОЛЬЗОВАННЫЕ В РАБОТЕ ИСТОЧНИКИ 16

**1. ВВЕДЕНИЕ**  
  **Обоснование выбора темы:**  
  
1.В современном мире глобальная сеть Интернет стала неотъемлемой частью повседневной жизни человека. По данным отчета Digital 2024: Global Overview Report от [DataReportal](https://datareportal.com/).com, более 66% всех жителей нашей планеты пользуются интернетом, а общее число интернет-пользователей в мире составляет 5,35 млрд. За 2023 год аудитория интернета прибавила 1,8% (97 млн новых пользователей), а количество профилей пользователей соцсетей превысило отметку в 5 млрд, что эквивалентно 62,3% населения мира. Во многом это было обусловлено широким распространением сотовых сетей с доступом в Интернет, развитием социальных сетей и удешевлением стоимости интернет-трафика. Для взаимодействия пользователей друг с другом, с базами данных, маркетплейсами и прочими сервисами используются различные веб-приложения, написанные в том числе и на фреймворках языка Python, таких как Django, FastAPI или Flask.

2. Потребности рынка: любая организация, коммерческая или некоммерческая, в эпоху информационного общества зависит от информационных ресурсов, важнейшим из которых сейчас является веб-сайт. Наличие сайта является важным элементом развития в современном информационном обществе и знание технологий создания сайтов является необходимым условием при планировании, проектировании и разработке веб-систем. Поэтому существует потребность в специалистах, обладающих навыками разработки различного программного обеспечения во многих отраслях производства, торговли, банковской сферы, соцмедиа, рекламной индустрии, маркетинга, образования и досуга и т.д.

3. Практическая значимость: знания в области разработки веб приложений позволят применить их на практике и предоставят опыт взаимодействия с базами данных. Это позволит создавать интересные и полезные для пользователей возможности, предоставляя им обширную аналитическую информацию и ценный функционал.

4**.** Личный интерес и потенциальные перспективы: использование различных инструментов и создание продуктов, сочетающих в себе разные технологии (например, веб-приложение и реляционные базы данных) позволяет развить и отработать опыт создания интересных и разносторонних проектов. Кроме того, такие навыки востребованы на рынке труда, что открывает перспективы для трудоустройства и карьерного развития.

Таким образом, выбор темы "Анализ и сравнение написания web-приложений с использованием разных фреймворков" обусловлен её актуальностью, потребностями рынка, практической значимостью и личным интересом, что делает эту тему подходящей для проведения дипломной работы.

**Определение цели и задач исследования:**

В своей работе я рассмотрю фреймворки Python для разработки веб-приложений.

**Цель исследования:** создание веб-приложения для работы с реляционной базой данных на трех различных фреймворках: Django, Flask и FastAPI; сравнение их возможностей и оценка эффективности.  
  
  **Задачи исследования:** 1. Сделать обзор фреймворков для написания веб приложения, провести их сравнительный анализ, выбрать наиболее подходящие под имеющуюся задачу.  
  
 2. Разработка сценариев сравнения: разработать алгоритм взаимодействия с БД, определить критерии сравнения.  
  
 3. Реализовать веб-приложение, применяя три различных фреймворка в backend с единообразной структурой frontend’а с доступом к базе данных SQLite.

4. Написать дипломную работу, включающую в себя введение, основную часть, заключение, список использованных источников информации. Разработать методологию исследования, провести анализ полученных результатов, обобщить выводы и сделать рекомендации.  
 Цели и задачи исследования направлены на получение практических результатов, которые позволят продемонстрировать возможности применения Django, Flask и FastAPI.

**2. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ**

Обзор основных понятий в области разработки и программирования веб-приложений для взаимодействия с реляционной базой данных:

1. Веб-приложение (Web Application): программное приложение, которое работает на веб-сервере и доступно через браузер. В контексте данного проекта, веб-приложение будет использоваться для работы с реляционной базой данных SQLite.

2. Фреймворк (Framework): программная платформа, которая предоставляет готовые компоненты и инструменты для разработки приложений. В контексте веб-приложений часто используются Django, FastAPI или Flask.

3. ORM (Object-Relational Mapping) - технология программирования, которая связывает [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) с концепциями [объектно-ориентированных языков программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), создавая «виртуальную [объектную базу](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) данных».

4. Реляционная база данных (RDB, РБД): Реляционная база данных — это тип базы данных, в которой данные организованы в таблицы, и эти таблицы связаны между собой на основе данных, общих для каждой из них. Эта структура позволяет выполнять гибкие, сложные запросы одновременно ко многим таблицам. Реляционные базы данных лучше всего подходят для обработки структурированных данных и сложных запросов. Наиболее известными реляционными базами данных являются Open Source проекты PostgreSQL, MySQL и SQLite, а также проприетарные решения Oracle, Microsoft SQL Server и IBM Db2Relational и др.

5. CRUD-операции: это основные методы работы с базами данных. Включают в себя четыре функции: Create (создание), Read (чтение), Update (редактирование) и Delete (удаление).

Данные термины помогают понять ключевые аспекты проверки эффективности применения различных фреймворков для программирования веб-приложений.

**3. ОБЗОР ПОПУЛЯРНЫХ ИНСТРУМЕНТОВ ДЛЯ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ НА PYTHON.**

В настоящее время любое web-приложение состоит из множества технологий, при этом серверная часть является основным элементом разработки, реализующим внутри себя все логические и бизнес процессы. Поэтому выбор средства разработки является существенной частью web-разработки.

Основными фреймворками, использующимся в современной backend разработке на Python являются Django, Flask и FastApi. Рассмотрим подробнее каждую из перечисленных технологий.

**Django**

Django - это бесплатная платформа веб-разработки на Python с открытым исходным кодом, используемая при создании веб-сайтов. Он был создан в 2003 году c использованием архитектуры MVT (Model-View-Template). Благодаря своей надежной и простой природе, это один из самых популярных фреймворков во всем мире. Основными плюсами Django являются:

* эффективная структура кода, позволяющая разработчикам добавлять большое количество функций на веб-сайты;
* 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;
* высокая степень безопасности, включающая защиту от SQL-инъекций и DDos-атак.

Особенности применения:

Идеально подходит для разработки веб-проектов любого размера, но особенно - для масштабных и сложных проектов благодаря своей полной экосистеме, например: агрегаторы бронирования, торговые платформы, сайты с расширенным функционалом (админ-панель, отправка писем и т.д). В связи с громоздкой архитектурой Django проекта, скорость разработки может быть ограничена из-за множественного повторного использования одних и тех же модулей.

**Flask**

Flask - это микро-веб-фреймворк, написанный на Python. Первый релиз вышел в 2010 году, но уже по состоянию на октябрь 2020 года Flask занимал второе место по количеству звезд на GitHub среди фреймворков веб-разработки Python, лишь немного отставая от Django.

Flask поставляется с такими опциями, как механизмы шаблонов, ORM (Flask SQLAlchemy). Основной целью разработки данного фреймворка было создание простого и быстрого средства масштабирования сложных приложений и микросервисов.

К преимуществам данного фреймворка следует отнести:

* минималистичный подход, только основные компоненты без лишних зависимостей;
* высокая степень гибкости. Большинство структурных частей Flask могут быть изменены;
* Jinja2 для шаблонов и Werkzeug для маршрутизации и обработки запросов;
* Данный фреймворк очень удобен для начинающих, ввиду своей простоты.

Особенности применения:

Подходит для небольших и средних проектов, где требуется простота и большая гибкость, например, таких как: статические сайты, боты, системы электронной коммерции и т. д.

**FastAPI**

Это современный, высокопроизводительный фреймворк для создания API с Python 3.7+ на основе стандартов OpenAPI и JSON Schema. К основным достоинствам можно отнести:

* высокая производительность благодаря использованию асинхронного программирования (async/await) на ASGI-серверах (Starlette и Uvicorn);
* автоматическая генерация интерактивной документации API;
* поддержка валидирования и аннотаций типов данных;
* интеграция с различными библиотеками и инструментами, такими как Pydantic и SQLAlchemy.

Особенности применения:

Часто используется для создания API и микросервисов, также отлично масштабируется при развертывании моделей машинного обучения. Несмотря на подробную, качественную документацию, образовательных материалов очень мало, т.к. сообщество FastAPI разработчиков невелико по сравнению с другими фреймворками.

Таким образом, данные фреймворки можно сравнить с помощью следующих критериев:

- количество модулей: среди Django, Flask и FastAPI у Django больше всего модулей, которые позволяют повторно использовать код. Это платформа веб-разработки с полным стеком, в отличие от Flask и FastAPI, которые являются минималистичными фреймворками, используемыми для создания быстрых веб-сайтов.

- размер сообщества: у Django самое крупное сообщество из-за его широкого использования и популярности. Flask также имеет большое сообщество по числу разработчиков.

- производительность: по производительности FastAPI является лидером, потому что он ориентирован на скорость, затем следует Flask и, наконец, Django, который не очень быстрый.

- гибкость: Flask более гибкий, чем Django. С другой стороны, Fast API является гибким с точки зрения кода и не ограничивает компоновку кода. Таким образом, мы можем сказать, что Flask является наиболее гибким среди всех трех.

- доступные вакансии: для разработчиков на Python есть больше всего вакансий с требованиями Django, далее идет Flask и, наконец, Fast API, у которого гораздо меньше вакансий.

- образование: Django более сложен в освоении, но имеет много онлайн-материалов и ресурсов. Flask прост и понятен, а также содержит множество онлайн-материалов и ресурсов, в то время как Fast API также является относительно простым фреймворком, если вы начинаете с веб-разработки, но у вас наименьшее количество онлайн-ресурсов.

**4. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ**

В качестве темы для проведения исследования я выбрал разработку и реализацию типового веб-приложения “Салон красоты”.

Разработанный сайт веб-приложения должен содержать несколько пользовательских разделов, таких как: “О нас”, “Наши услуги”, “Наши работы”, “Статьи”, “Контакты”. Каждая страница должна содержать актуальную информацию, источником которой является подключенная база данных. Контент отдельных страниц должен поддерживать динамическое обновление информации из БД (“Наши услуги”, “Наши работы”, “Статьи”, “Контакты”).

Проектирование приложения в соответствии с его описанием было разбито на несколько этапов:

* выбор базы данных. Определение наиболее подходящей БД (например, SQLite, MySQL или PostgreSQL) и инструментов для разработки;
* определение структуры приложения: разработка архитектурной схемы, включающей фронтенд, бэкенд, базу данных и их интеграцию;
* разработка прототипа;
* реализация основного функционала: создание базовой версии приложения с минимально необходимым функционалом для тестирования и демонстрации;
* интеграция компонентов приложения: подключение и настройка выбранной базы данных для хранения объектов;
* создание веб-сервиса для взаимодействия с фронтендом. Разработка логики для обработки запросов от пользовательского интерфейса;
* реализация пользовательского интерфейса. Создание форм и страниц для отображения результатов их взаимодействия с пользователем.

**Основные требования**

Для разрабатываемого приложения были сформулированы следующие критерии:

* актуализация: пользователь должен видеть актуальную информацию на страницах веб-приложения, которая по его запросу поступает из базы данных;
* управление данными: в БД приложения должна быть реализована возможность управления данными, в т. ч. загружать, обновлять, удалять файлы, хранящиеся на сервере (на основе CRUD методов);
* отображение результатов: приложение должно отображать результаты выборки по запросам из базы данных и оперировать ими.

**Требования к архитектуре веб-приложения**

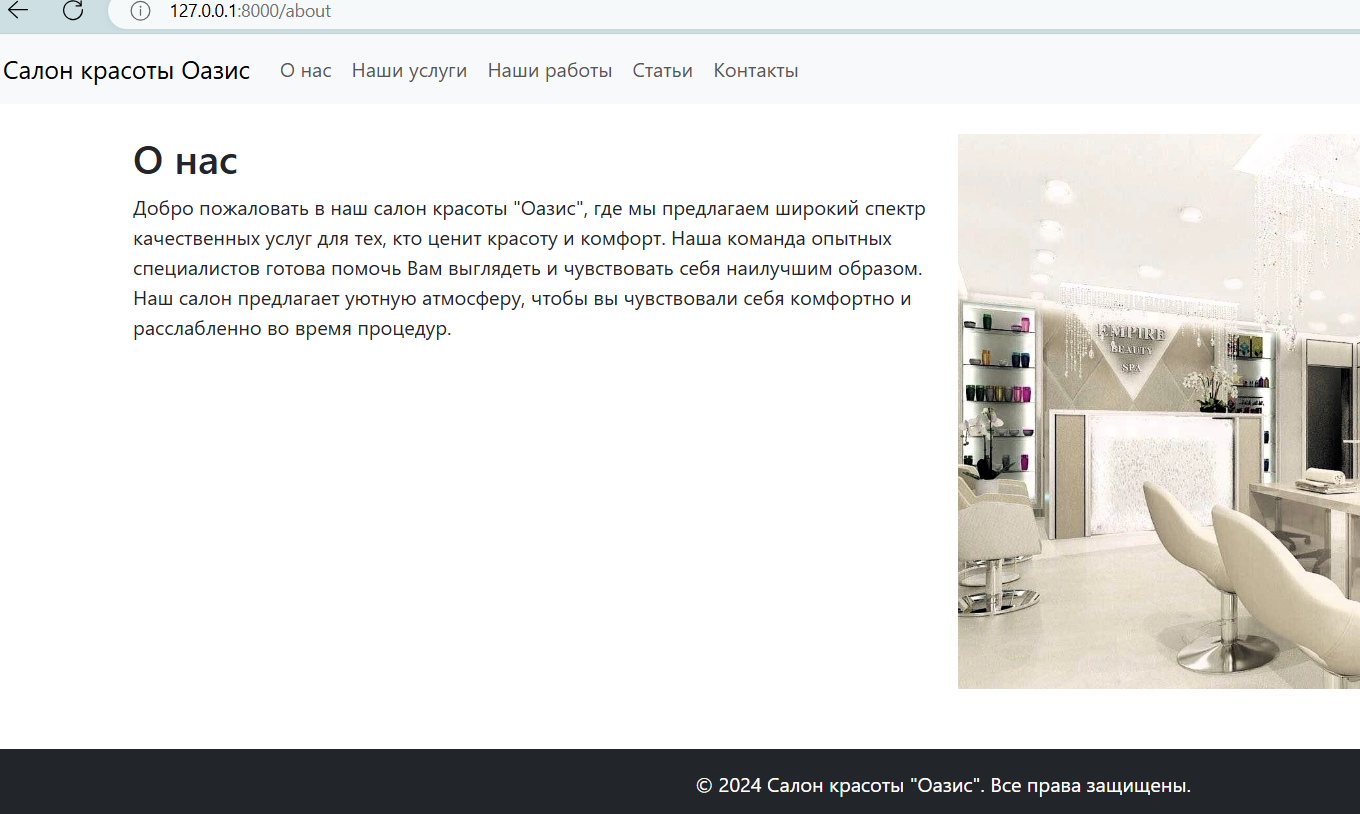
Frontend и Backend: фронтенд (интерфейс пользователя) и бэкенд (серверная логика). Фронтенд может быть разработан с использованием HTML, CSS и JavaScript, а бэкенд будет реализован на Python с использованием фреймворков FastAPI, Flask и Django. База данных: использование базы данных SQLite для хранения данных о пользователях и объектах учета.

**5. РАЗРАБОТКА В СООТВЕТСТВИИ С ТРЕБОВАНИЯМИ**

Разработка была разделена на несколько основных этапов: проектирование интерфейса, реализация серверной логики и проектирование структуры базы данных. Планирование задач, отслеживание этапов разработки и сама разработка осуществлялось в программе Pycharm.

**Фронтенд-разработка**:

Создан пользовательский интерфейс с использованием DTL/Jinja2 для шаблонов и Bootstrap для стилизации.

****

**Бэкенд-разработка:**  
Реализована серверная логика с использованием фреймворков Django, FastAPI и Flask. Сделано 3 веб-приложения:

[Django](https://github.com/LifeInputter/D_work/tree/main/beauty_salon) <https://github.com/LifeInputter/D_work/tree/main/beauty_salon>

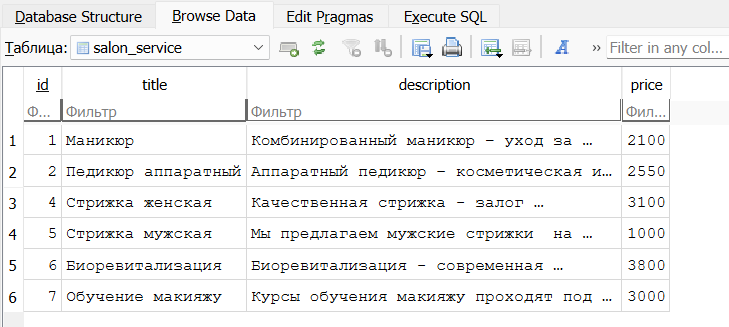
[FastAPI](https://github.com/LifeInputter/D_work/tree/main/FastAPI/beauty_salon) <https://github.com/LifeInputter/D_work/tree/main/FastAPI/beauty_salon>

[Flask](https://github.com/LifeInputter/D_work/tree/main/Flask/beauty_salon) <https://github.com/LifeInputter/D_work/tree/main/Flask/beauty_salon>

Настроены маршруты для обработки взаимодействия с реляционной базой данных SQLite и передача результатов на фронтенд.

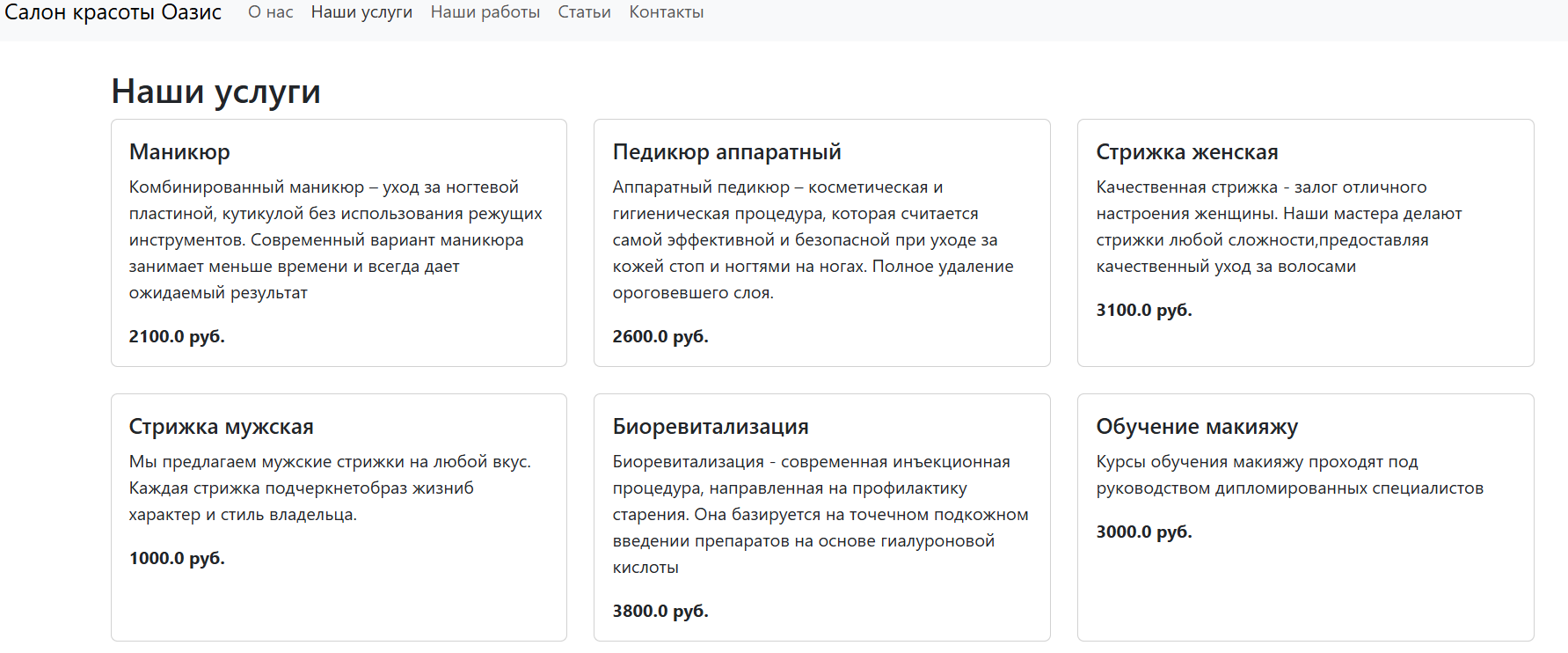
**Интеграция базы данных:**

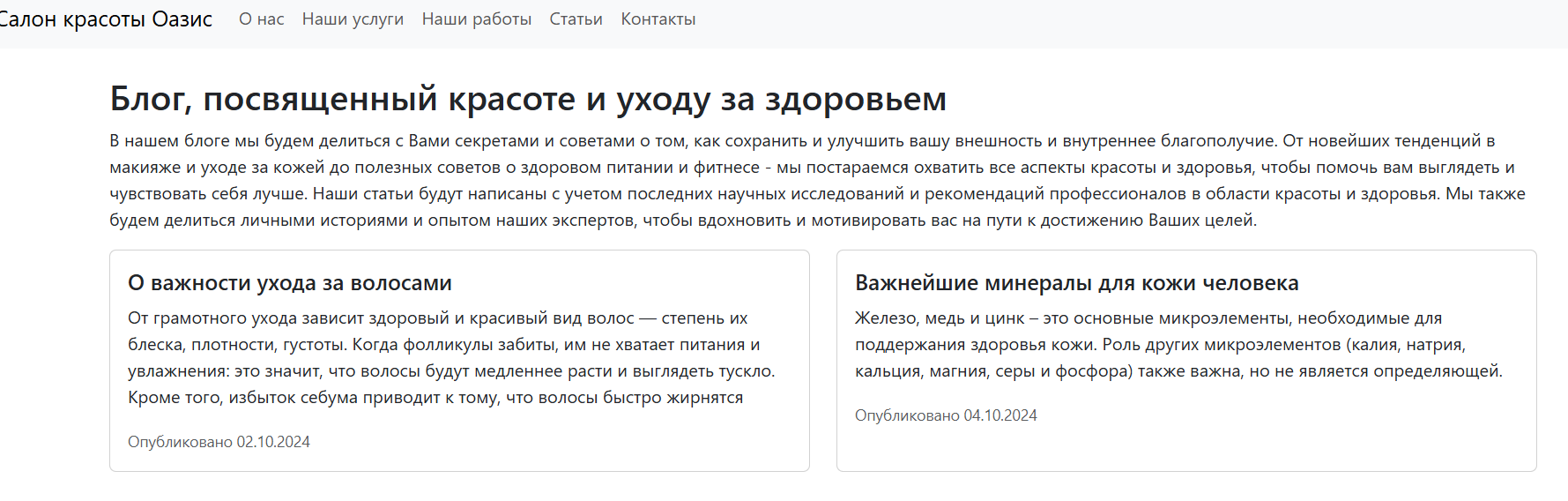
Разработаны модели с необходимыми полями для записи в базу данных, осуществлена интеграция базы в приложение и ее инициализация.   
 Создание и миграция базы данных проводилась в Django встроенными инструментами (Django ORM, query set), для FastAPI использовалась библиотека Alembic и Swagger, для Flask использовалась база данных которая применяется и в приложении на FastAPI.

****

****

Реализована обработка результатов запросов и их отображение на веб-страницах:

****

****

**6. АНАЛИЗ И ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ**

**Сравнение фреймворков**

По завершении разработки приложений на разных фреймворках проведено их сравнение. Были протестированы три фреймворка языка Python на предмет работы с реляционной базой данных SQLite.

Основные метрики, такие как скорость разработки, скорость работы, удобство в использовании, были использованы для оценки развертывания и производительности каждого фреймворка на субъективном уровне.

Обобщим параметры всех используемых фреймворков для анализа:

|  |  |  |  |
| --- | --- | --- | --- |
| Framework | Django | Flask | FastAPI |
| Тип БД | SQLite | SQLite | SQLite |
| ORM | Django ORM | нет | SQLAlchemy |
| Миграция БД | Django | нет | alembic |
| Шаблонизатор | DTL | Jinja2 | Jinja2 |
| Админ.панель | есть | нет | нет |
| Асинхронность[[1]](#footnote-1) | нет | нет | есть |

Субъективные оценки, полученные в процессе разработки по шкале от 1 до 5, где 1 - худшая, 5 - лучшая оценка:

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | Django | Flask | FastAPI |
| Скорость разработки | 4 | 4 | 3 |
| Удобство добавления данных | 5 | 3 | 4 |
| Скорость работы приложения | 4 | 4 | 5 |

**Интерпретация результатов**

Все три фреймворка языка Python были успешно протестированы на предмет работы с базой данных SQLite, определены сильные и слабые стороны каждого из них.

Django является популярным и полнофункциональным серверным веб-фреймворком. Удобен своей строгой структуризацией. Его полный стек предлагает все, что нужно для разработки приложения, включая ORM, админку, аутентификацию и т.д. Админка Django позволяет легко управлять данными в БД, что крайне удобно в плане администрирования разрабатываемого или действующего сайта. Использует в своей работе множество модулей и библиотек, которые ускоряют разработку и тестирование проектов. Благодаря своей модульности позволяет легко масштабировать создаваемые приложения.

Flask самый «легковесный» из рассматриваемых фреймворков. Предоставляет разработчику полный контроль над структурой приложения (минимализм), но требует больше работы по настройке. Легко интегрируется с любыми сторонними библиотеками (гибкость). На нем удобно быстро разрабатывать простые веб-страницы, сайты-визитки и т. д. Не имеет строгой структуры. Для написания кода подходит как программирование в одном файле, так и через создание структуры каталогов проекта. Скорость работы в сети - высокая. Отсутствуют встроенные механизмы, такие как админка или аутентификация.

FastAPI — фреймворк, использующий асинхронное программирование. Благодаря чему позволяет выполнятся приложению с высокой скоростью. Хорошо использовать для организации микросервисов на сервере. Также не имеет строгой структуризации: базовый код может быть написан как в одном файле, так и через структуру из нескольких файлов. Предоставляет удобный функционал документирования Swagger, широкие возможности аннотаций типов данных (Pydentic).

Выбор фреймворка зависит от потребностей проекта: Django подойдет для крупных проектов с множеством связанных компонентов. Flask хорош для небольших или средних приложений, где важна гибкость. FastAPI оптимален для приложений, где критична производительность и асинхронная работа.

**7. ЗАКЛЮЧЕНИЕ**  
  
 **Обзор выполненной работы**

Проектирование и разработка веб-приложений для работы с реляционной базой данных на примере веб-приложения “Салон красоты” были успешно завершены в соответствии с изначальными требованиями, предявляемыми к проекту. Развернутые приложения включают функционал интерактивного управления отображаемого контента на сайте за счет подключения к БД и обработки запросов к ней: четыре из пяти разделов сайта имеют динамическое отображение контента, одна - статическое. К шаблонам HTML добавлена стилизация из библиотеки Bootstrap и JavaSsript. Реализованные приложения соответствуют требованиям и демонстрирует высокую скорость работы с БД.

Все три фреймворка языка Python были успешно применены на практике при реализации веб-приложений, подтверждены сильные и слабые стороны каждого из них. БД SQLite зарекомендовала себя как простой и удобный способ хранений и администрирования данных для повседневных задач, однако для улучшения функциональности, безопасности и надежности хранения данных возможно использование других РБД.

Django, Flask и FastAPI предлагают разные подходы к веб-разработке, каждый со своими уникальными преимуществами и ограничениями. Важно изначально определить на входе, что лучше всего соответствует вашим требованиям, чтобы выбрать подходящий инструмент для своего проекта.

**Планы по развитию веб-приложения**

Разработанные веб-приложения можно в дальнейшем усовершенствовать за счет добавления новых функций, таких как: добавление блока для аутентификации пользователей, расширения пользовательской функциональности за счет добавления сведений о программе лояльности, добавление пагинации на странице блога “Статьи”.

**ИСПОЛЬЗОВАННЫЕ В РАБОТЕ ИСТОЧНИКИ:**

1. Digital 2024: Global Overview Report

<https://datareportal.com/reports/digital-2024-global-overview-report>

2. Виды баз данных. Большой обзор типов СУБД. Автор - kirillkosolapov

https://habr.com/ru/companies/amvera/articles/754702/

3. Шубенкин Д. А., Уваров А. М. СРАВНИТЕЛЬНЫЙ АНАЛИЗ BACK-END РАЗРАБОТКИ НА PYTHON // Журнал «Научный лидер», выпуск  №28 (73), 2022

<https://scilead.ru/article/2734-sravnitelnij-analiz-back-end-razrabotki-na-py>

4. FastAPI vs Django: какой Python фреймворк выбрать для веб-приложения?

https://worksolutions.ru/useful/fastapi-vs-django/

5. Сравнительный анализ популярных Python фреймворков: Django, Flask, FastAPI.

<https://vc.ru/u/1950202-perezegor-blog-programmista/1092433-sravnitelnyi-analiz-populyarnyh-python-freimvorkov-django-flask-fastapi?ysclid=m1xgblu13s720727363>

1. под асинхронностью принимается асинхронность из коробки. [↑](#footnote-ref-1)