**Дипломная работа**

**по теме: Анализ и сравнение написания web-приложений**

**с использованием разных фреймворков**

Выполнил: Долгополов Александр Анатольевич

**Оглавление дипломной работы**

|  |  |  |
| --- | --- | --- |
| 1. | Введение…………………………………………………………. | 2 |
| 1.1 | Обоснование выбора темы……………………………………... | 2 |
| 1.2 | Определение цели и задач исследования……………………… | 3 |
| 2. | Основные понятия и определения……………………………... | 4 |
| 3. | Обзор фреймворков для написания web-приложений……….. | 4 |
| 4. | Разработка web-приложений с использованием Django, Flask и FastAPI………………………………………………………… | 8 |
| 5. | Выводы…………………………………………………………... | 14 |
| 6. | Заключение……………………………………………………… | 16 |
| 7. | Приложение № 1 Файловая структура проектов.………...…... | 17 |
| 8. | Приложение № 2 Список необходимых библиотек…………... | 19 |

**1. Введение**

**1.1 Обоснование выбора темы**

Фреймворк является важным рабочим инструментом любого современного **web**-разработчика. На его изучение может уходить много времени. Кроме того, нужно сделать настройку под конкретный проект и выбрать подходящий под него инструмент.

Фреймворк – это общепринятый каркас и структура кода, то есть шаблон, позволяющий стандартизировать рабочие процессы. Разработчик использует готовые решения, адаптируя их под конкретный проект или задачу. Фреймворк описывает начальную заготовку и свод инструкций, которые упрощают достижение конечной цели: остов сайта проще нарастить, чем собирать все с нуля.

Глобально фреймворки решают три задачи:

увеличение скорости разработки за счет использования готовых модулей и шаблонов;

упрощение работы, так как по своей структуре и принципу работы популярные фреймворки похожи между собой;

обеспечение безопасности. Популярные фреймворки создавались с расчетом на безопасность, поэтому в них уже по умолчанию есть встроенные защиты от популярных методов взлома и кражи данных, например, XSS-атак, SQL-инъекций и т.п.

Фреймворки в зависимости от стоящих перед разработчиком задач разделяются на несколько типов.

Фронтенд-фреймворки – популярные варианты, написанные преимущественно на JavaScript: React, Vue, Angular. Они отвечают за внешний вид интерфейсов, адаптацию под разные устройства, анимацию, примитивные возможности типа открытия вкладок.

Бэкенд-фреймворки – отвечают за реализацию «внутренностей» сайта (хранение данных, их обработка, взаимодействие с базами данных, прием информации от пользователей). Фреймворки для бэкенда преимущественно пишутся на Python, наиболее популярные из них – Django, Flask, FastAPI.

Кроссплатформенные фреймворки позволяют эффективно заниматься разработкой сложных проектов. Отличаются высокой скоростью разработки, но при этом может страдать качество работы. Чтобы этого избежать, нужно хорошо освоить не только фреймворк, но языки и среды разработки, с которыми они лучше всего интегрируются. Кроссплатформенность позволяет запускать исходный код на различных операционных системах (Windows, Linux, macOS) или платформах (в вебе, на десктопных или мобильных устройствах).

Таким образом, фреймворки важны для любого web-разработчика, который планирует работать над проектами сложнее одностраничных сайтов. Они помогают ускорить создание проекта, так как позволяют быстро развернуть шаблонную структуру и редактировать ее при необходимости. В зависимости от потребностей и предпочтений разработчик может выбрать подходящий фреймворк для своего проекта.

**1.2 Определение цели и задач исследования**

Цель исследования – провести сравнительный анализ простых web-приложений, разработанных с использованием Django, Flask и FastAPI.

Задачи исследования:

1. Обзор фреймворков Django, Flask и FastAPI для написания web-приложений.

2. Разработка web-приложений с использованием фреймворков Django, Flask и FastAPI.

3. Сравнительный анализ разработанных web-приложений.

**2. Основные понятия и определения**

Web-приложение – программное приложение, работающее на web-сервере через браузер.

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

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

Фреймворк (framework) – набор инструментов для написания бэкенда web-приложения.

API (Application Programming Interface) – это набор правил, позволяющих одному программному продукту взаимодействовать с другим. В контексте web-разработки API может служить мостом между фронтендом и бэкендом. API облегчает взаимодействие между разными приложениями и платформами. Вместо того чтобы разрабатывать все функции с нуля, разработчики могут использовать готовые API, которые уже предоставляют определённые возможности, такие как авторизация, обработка платежей или получение данных о погоде.

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

**3. Обзор фреймворков для написания web-приложений**

Python предлагает несколько мощных web-фреймворков, каждый из которых имеет свои сильные и слабые стороны. Наиболее популярные из них Django, Flask и FastAPI. Они различаются по сложности, возможностям и сфере применения.

Django – это «батарейки включены» фреймворк, предоставляет множество встроенных инструментов и библиотек для решения различных задач. Он является бесплатным с открытым исходным кодом, шаблон его архитектуры – «model-template-views». В настоящее время это один из самых популярных фреймворков в мире из-за его надежного поведения. Django лучше всего подходит для создания сложных web-приложений с большим количеством функционала, где важны скорость разработки и безопасность. Основная цель Django – упростить разработку сложных web-сайтов, управляемых базами данных.

Когда следует выбирать Django:

если нужно создать большое, полнофункциональное web-приложение, например, социальную сеть, интернет-магазин или новостной портал;

если важна безопасность. Django предлагает встроенные механизмы защиты от многих распространенных уязвимостей, таких как SQL-инъекции и XSS-атаки;

если нужно быстро разработать прототип, используя готовые компоненты, такие как система аутентификации, админ-панель и управление базой данных.

Например, необходимо создать сайт для новостного портала. Django предоставит удобные инструменты для управления пользователями, создания административной панели и работы с базой данных, что значительно ускорит процесс разработки.

Flask является микрофреймворком, поскольку для web-разработки с его помощью не требуется какой-либо конкретной библиотеки или инструментов. Он предоставляет разработчику только основные инструменты, а все остальное предлагается добавлять по мере необходимости через сторонние библиотеки. Flask подходит для небольших проектов, где важна гибкость и контроль над каждым аспектом приложения. Цель этого фреймворка – легко и быстро разрабатывать легковесные приложения с меньшим количеством функций.

Когда использовать Flask:

если необходимо создать небольшой или средний проект, где важна гибкость и контроль, например, API для мобильного приложения или простое web-приложение;

если не требуются встроенные компоненты и вы хотите самостоятельно выбирать библиотеки для реализации нужного функционала;

если вы хотите изучить основы web-разработки, не отвлекаясь на сложные встроенные механизмы.

Например, необходимо разработать API для мобильного приложения. Flask позволяет легко и быстро создать небольшой, гибкий сервер, который будет обрабатывать запросы от мобильного клиента.

FastAPI – это современный, быстрый и надежный фреймворк, который помогает в создании API с версиями Python 3.6+. Он считается одним из самых быстрых фреймворков, кодируется с меньшим количеством ошибок, создаваемых разработчиками. Основные преимущества FastAPI: быстрая разработка, минимум ошибок, высокая производительность. Он идеально подходит для проектов, где важны скорость работы и асинхронная обработка запросов.

Когда выбрать FastAPI:

если нужно разработать высокопроизводительное API с поддержкой асинхронной обработки запросов;

если важна скорость разработки и нужно минимизировать количество ошибок в коде, благодаря использованию аннотаций типов;

если создаются микросервисы или системы, где каждый запрос должен обрабатываться как можно быстрее.

Например, необходимо создать высокопроизводительное API для обработки большого объема данных в реальном времени (сервис рекомендаций или аналитическую платформу). FastAPI позволит найти быстрое и надежное решение.

Результаты сравнения фреймворков представлены в таблице.

| **Фреймворк**  **Характеристика** | **Django** | **Flask** | **FastAPI** |
| --- | --- | --- | --- |
| **Тип** | Полнофункциональный | Минималистичный | Фреймворк для API |
| **Сложность** | Средняя | Низкая | Средняя |
| **Производительность** | Умеренная | Высокая | Очень высокая |
| **Гибкость** | Низкая (жесткая структура) | Высокая (выбор компонентов) | Высокая (поддержка асинхронности) |
| **Безопасность** | Встроенные механизмы | Нужно реализовывать самостоятельно | Встроенная поддержка |
| **Лучшие сценарии** | Сложные web-приложения | Небольшие web-приложения и API | Высокопроизво-дительные API |
| **Сообщество и документация** | Очень большое и активное | Большое и активное | Быстро растущее и активное |

Сравнительный анализ Django, Flask и FastAPI показывает, что у Django больше всего модулей, которые позволяют повторно использовать код. Это платформа web-разработки с полным стеком, в отличие от Flask и FastAPI, которые являются минималистичными фреймворками, используемыми для создания быстрых web-сайтов.

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

Flask более гибкий, чем Django. С другой стороны, FastAPI является гибким с точки зрения кода и не ограничивает компоновку кода. В результате можно сказать, что Flask является наиболее гибким среди всех трех.

Django имеет самое крупное сообщество из-за его широкого использования и популярности.

Таким образом, Django идеально подходит для создания надежных web-сайтов, потому что он имеет множество функциональных возможностей и очень хорошо работает в производстве. Flask идеально подходит для инженеров по машинному обучению или разработчиков, которые хотят быстро создать прототип web-приложения или API. В то же время FastAPI идеально подходит, если основными критериями для разработчика являются скорость или масштабируемость.

**4. Разработка web-приложений с использованием Django, Flask и FastAPI**

В настоящей дипломной работе были разработаны три web-приложения с использованием фреймворков Django, Flask и FastAPI.

Приложение на Django для магазина по продаже компьютерных игр реализована файловой структурой, показанной на рис. 1.

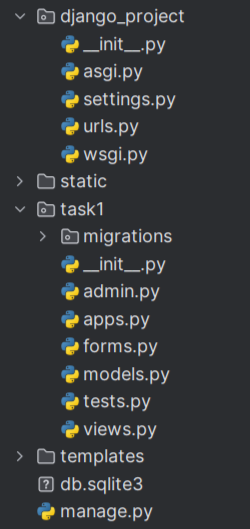


Рис. 1 Файловая структура проекта на Django

Общий вид главной страницы магазина представлен на рис. 2, страницы магазина с перечнем компьютерных игр – рис. 3.

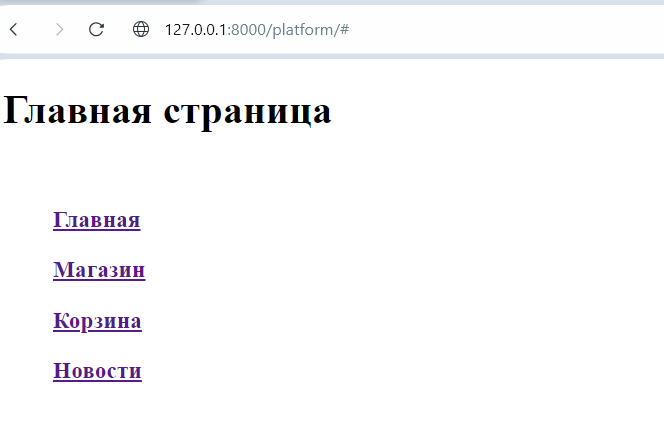


Рис. 2 Главная страница проекта на Django

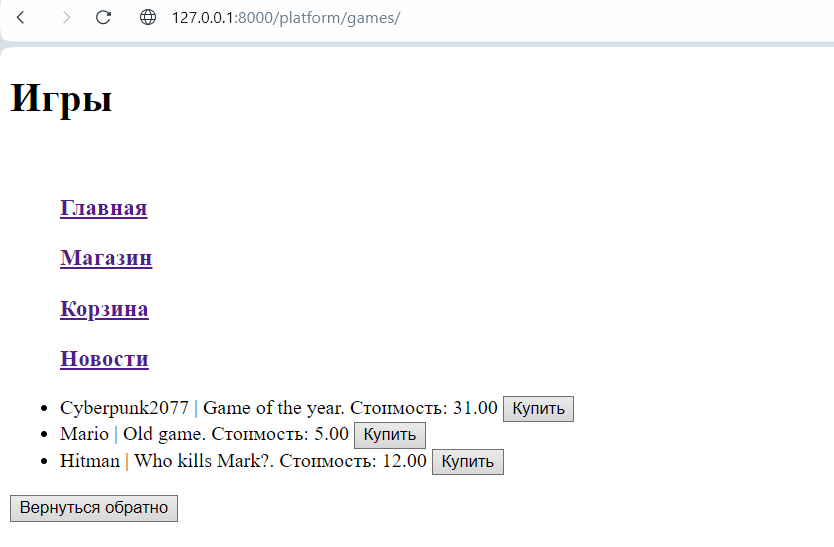


Рис. 3 Страница магазина с перечнем игр

С использованием Flask разработано простое web-приложение на Python с базой данных для авторизации пользователей, файловая структура проекта представлена на рис. 4.

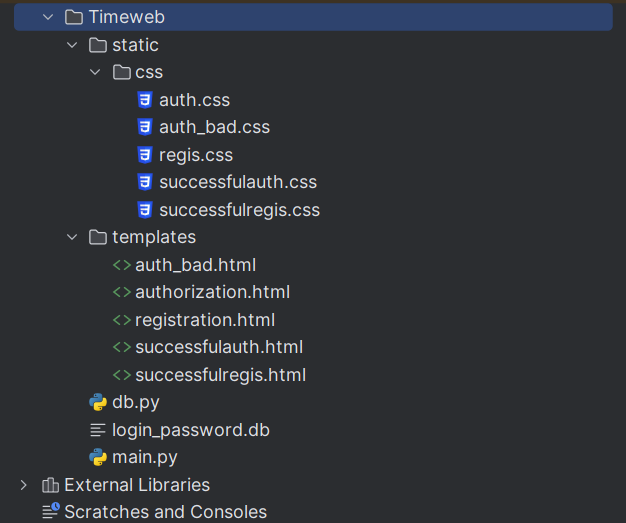


Рис. 4 Файловая структура проекта с использованием Flask

Общий вид начальной страницы приложения показан на рис. 5.

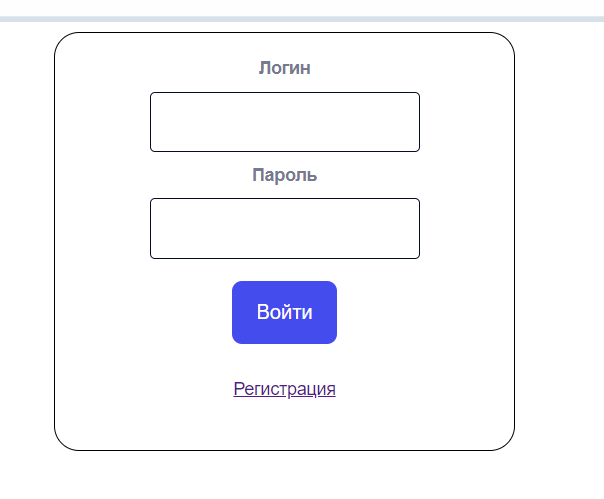


Рис.5 Начальная страница приложения

Имена (логины) и значение паролей по мере регистрации пользователей сохраняются в базе данных (рис. 6).

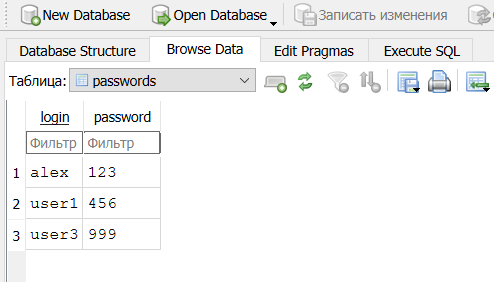


Рис. 6 База данных с именами и паролями пользователей

Третий проект создан с использованием FastAPI и базы данных SQLite, он фиксирует список задач. Файловая структура данного проекта представлена на рис. 7.

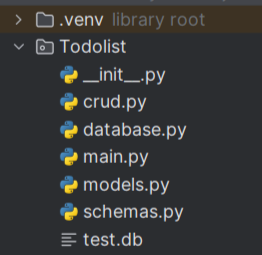


Рис. 7 Файловая структура проекта «Список задач»

После запуска сервера API будет доступно по адресу <http://127.0.0.1:8000>. FastAPI предоставляет встроенную документацию, которую можно открыть в браузере по адресу – Swagger UI: <http://127.0.0.1:8000/docs> (рис.8)

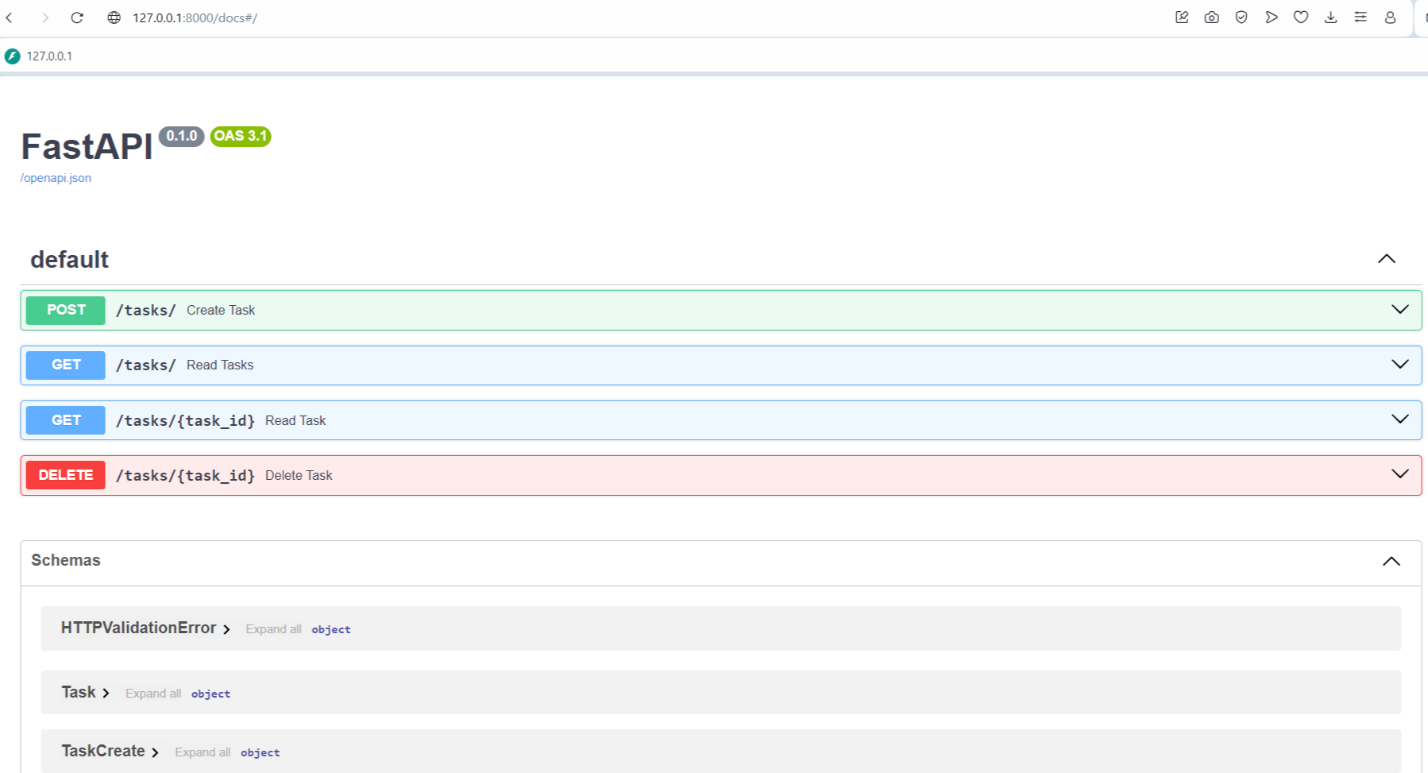


Рис. 8 Вид страницы FastAPI - Swagger UI

Непосредственно список задач отражается в базе данных (рис. 9), таблица имеет четыре столбца:

id – уникальный идентификатор задачи;

title – заголовок задачи;

description – описание задачи;

completed – флаг, указывающий, завершена ли задача.

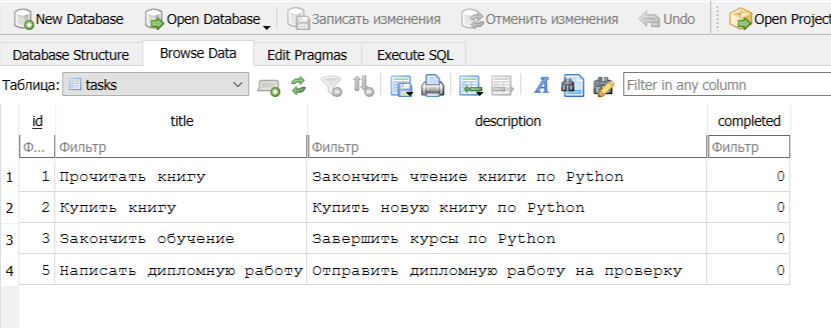


Рис. 9 База данных со списком задач

**5. Выводы**

Сравнительный анализ web-приложений, разработанных на трех разных фреймворках, показывает, что универсального ответа на вопрос «Какой web-фреймворк лучше?» – нет. Каждый из рассмотренных фреймворков имеет сильные и слабые стороны.

Django – это гигант с полным стеком, уже много лет является фаворитом, предлагая универсальное решение, включающее:

встроенный ORM и панель администратора;

готовые функции безопасности;

большое и активное сообщество для поддержки.

Но по мере развития приложений некоторые разработчики задаются вопросом, сможет ли монолитный подход Django удовлетворить спрос на микросервисы и современные API.

Flask – чемпион в легком весе, отличается простотой, предлагая при этом:

минимальные накладные расходы и максимальную гибкость;

архитектуру «подключи и работай», позволяющую выбирать библиотеки по мере необходимости;

высокую эффективность для микросервисов и бережливых проектов.

Однако гибкость Flask может стать и его недостатком – отсутствие стандартов означает, что нужно принимать больше решений, а это может замедлить разработку для больших команд.

FastAPI – восходящая звезда, набирает обороты благодаря таким функциям, как:

асинхронная поддержка, обеспечивающая молниеносную производительность;

автоматическая генерация документов OpenAPI.

Особое внимание уделяется безопасности и производительности разработчика.

FastAPI быстро становится популярным выбором для создания API, но достаточно ли он развит, чтобы конкурировать с Django и Flask в ближайшие пару лет?

В любом случае выбор фреймворка для создания web-приложения зависит от конкретных требований проекта, сложности задач и предпочтений разработчика.

**6. Заключение**

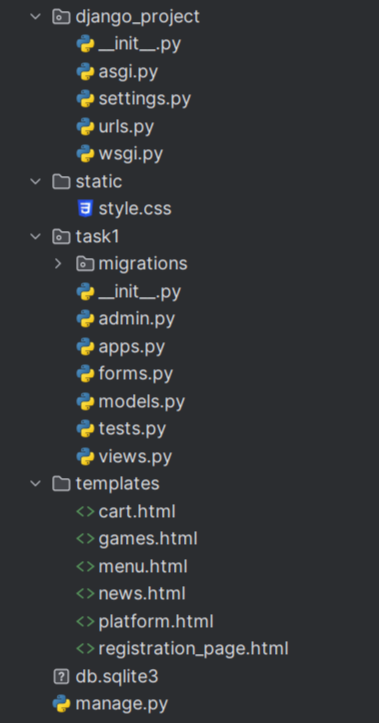
В процессе выполнения настоящей дипломной работы были разработаны три простых web-приложения с использованием фреймворков Django, Flask и FastAPI. Кроме того, проведен сравнительный анализ указанных фреймворков, а также подробно описаны их преимущества и недостатки.

Таким образом, поставленные перед написанием дипломной работы задачи решены, цель – достигнута.

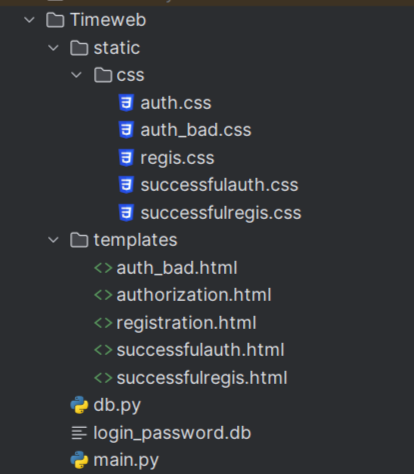
Приложение № 1

Файловая структура проектов

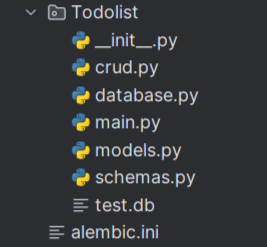
Web-приложение на фреймворке Django



Web-приложение на фреймворке Flask



Web-приложение на фреймворке FastAPI



Приложение № 2

Список необходимых библиотек

Flask==3.1.0

Werkzeug==3.1.3

Django==5.1.4

uvicorn==0.32.1

fastapi==0.115.6

jinja2==3.1.5

starlette==0.41.3

SQLAlchemy==2.0.36

pydantic==2.10.3

asgiref==3.8.1

sqlparse==0.5.3

tzdata==2024.2

markupsafe==3.0.2

blinker==1.9.0

click==8.1.8

colorama==0.4.6

importlib-metadata==8.5.0

itsdangerous==2.2.0

setuptools==68.2.0

zipp==3.21.0