**Этап 2**

В результате поиска лучших фреймворков, а так же проведенного анализа были выведены топ 3 лучших, а так же используемых фреймворков, это -

**1) Flask**

**2) Django**

**3) FastAPI**

Каждый фреймворк уникален и у каждого из них есть плюсы и минусы. Начнем постепенно каждый разбирать фреймворк

**Flask**

**Плюсы:**

1) **Flask** довольно легковесный и гибкий благодаря чему он идеально подходит для малых и средних проектов

2) **Flask** довольно легкий в использовании и на настройки

3) **Flask** имеет обширное сообщество, а также множество плагинов и документации

**Минусы:**

1) **Flask** не содержит **ORM** по умолчанию, а также требует подключения сторонних библиотек

2) У **Flask** довольно маленькая производительность по сравнению с **FastAPI**

3) **Flask** может не подходить для крупных проектов. Все из-за ограниченной функциональности “из коробки”

**Django**

**Плюсы:**

1) У **Django** есть довольно большое количество встроенных функции

2) **Django** имеет административную панель и систему авторизации, а так же содержит **ORM (Object-Relational Mapping)**

3) **Django** как и **Flask** имеет обширное сообщество, а так же множество плагинов и документации

**Минусы:**

1) **Django** довольно сложный в освоении по сравнению с Flask

2) Из-за большой функциональности **Django**, может привести к медленной работе приложения

3) **Django** по сравнению с **Flask** и **FastAPI** менее гибкий

**FastAPI**

**Плюсы:**

1) **FastAPI** имеет более высокую производительность благодаря асинхронному программированию и использованию **Pydantic** и **Starlette**

2) **FastAPI** поддерживает современные стандарты API, таких как **OpenAPI** и **JSON** **Schema**

3) **FastAPI** довольно гибкий и прост в использовании

**Минусы:**

1) **FastAPI** относительно молодой фреймворк, из-за чего возможны проблемы с совместимости и стабильности

2) У **FastAPI** мало дополнительных расширений и плагинов по сравнению **Flask** и **Django**

3) **FastAPI** не содержит по умолчанию **ORM**, из-за чего требуется подключения сторонних библиотек

“Из коробки” — это фраза, которая означает функциональность или возможности, которые предоставляются с продуктом сразу после установки, без необходимости в дополнительной настройке или интеграции сторонних компонентов. В контексте веб-фреймворков это может включать встроенные инструменты для роутинга, обработки запросов, авторизации и других задач, связанных с разработкой веб-приложений.

ORM (Object-Relational Mapping) - это техника программирования, которая позволяет представлять структуры данных и отношения между ними в виде объектов и классов в объектно-ориентированном языке программирования, таком как Python. ORM позволяет разработчикам взаимодействовать с базами данных с использованием объектов и методов языка программирования, что упрощает работу с базами данных и обеспечивает лучшую абстракцию данных.

Pydantic - это библиотека для Python, которая позволяет выполнять валидацию и сериализацию данных. Pydantic использует аннотации типов Python для определения структуры и ожидаемых типов данных. Она облегчает разработку веб-приложений и API, так как автоматически проверяет корректность входных данных и преобразует их в соответствующие объекты Python.

Starlette - это асинхронный веб-фреймворк для Python, который обеспечивает высокую производительность и масштабируемость. Starlette предоставляет базовые компоненты для разработки веб-приложений, такие как роутинг, обработка запросов и ответов, поддержка WebSockets и интеграция с GraphQL. Starlette является основой для более высокоуровневых фреймворков, таких как FastAPI, и обеспечивает производительность и асинхронные возможности, которые делают FastAPI привлекательным для разработчиков.

**Пример кода каждого из предоставленных ранее фреймворков**

Перед написанием кода я рекомендую сначала убедиться, что они у вас установленный. Проверить довольно просто, напишите в терминале команду **pip list.** Данная команда выведет вам список всех установленных ранее фреймворков. Так же с помощью команды **pip freeze | grep name\_framework,** можно узнать установлен ли у вас данный фреймворк.

Для начала установим все необходимые нам фреймворков. Установить их довольно просто, напишите терминале команду pip install name\_framework.

Установка фреймворков **Flask, Django, FastAPI**

**1) pip install flask**

**2) pip install django**

**3) pip install fastapi uvicorn**

**Создаем проект в любом приложения IDE, в моем случае PyCharm**

**1)Пример Flask**

****

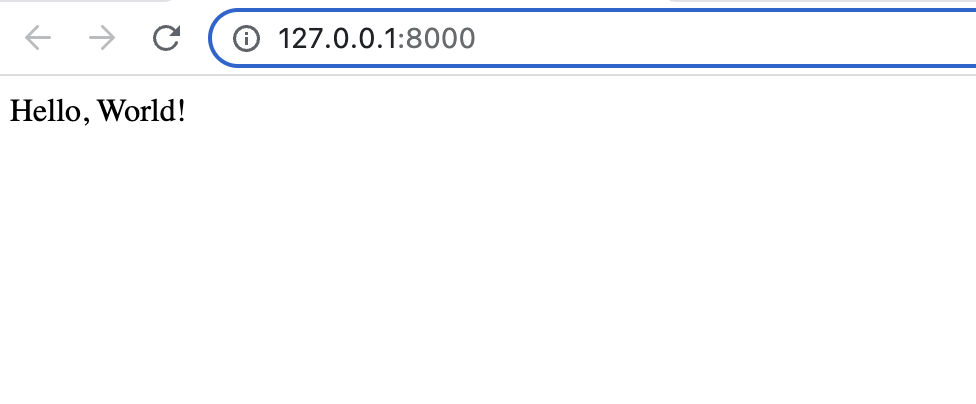
**Вывод:**

****

**2) Пример Django**

****

**Вывод:**

****

**3)Пример FastAPI**

****

**Вывод:**

****

**Примечание:**

Для запуска Flask:

**Запустите его с помощью кнопки “run”**

**Для запуска Django:**

Первое этап: что вам нужно сделать это создать проект: **“django-admin startproject myproject”**. Так же учтите что вы находитесь в папке проекта. Это можно сделать с помощью команды **“cd name\_project”.** Если не сделать это, то ваш проект не запуститься

Второй этап: вам нужно перейти в файл **“urls.py”**,и добавить код.

Третий этап: в консоли нужно написать команду **“python manage.py runserver** **”**

**Для запуска FastAPI:**

В терминале напишите команду **“uvicorn main:app --reload”**

**Итог:**

Не смотря на преимущество фреймворков FastpAPI и Flask, я все равно выбираю Django. С Django я работал, и мне сам фреймворк сам понравился в плане его функциональности и работоспособности.