**План действий для реализации проекта с нуля**

**1. Подготовка окружения**

1.1 Установить необходимые инструменты:

* Убедиться в наличии Docker и Docker Compose.
* Установить Python 3.9 и pip.

1.2 Создать структуру проекта:

* Создать основной каталог проекта, например, booking\_service.
* Внутри каталога создать подкаталоги: api, tests, docker, scripts.

**2. Разработка Mock-сервера**

2.1 Создать Mock-сервер:

* В каталоге api создать файл locust\_mock.py, который будет содержать FastAPI-приложение для имитации API сервиса бронирования.

2.2 Реализовать API эндпоинты:

* **POST /booking/add**: Добавление бронирования (аналог добавления товара в корзину).
* **GET /login/{username}**: Авторизация пользователя и генерация токена.

2.3 Написать requirements.txt:

* Включить необходимые зависимости для FastAPI и Uvicorn.

2.4 Создать Dockerfile для Mock-сервера:

* В каталоге docker создать файл Dockerfile, который будет содержать инструкции для сборки образа mock-сервера на основе FastAPI.

**3. Настройка инфраструктуры с Docker Compose**

3.1 Создать docker-compose.yml:

* Настроить сервисы Kafka, Zookeeper, InfluxDB и Mock-сервер в файле docker-compose.yml в каталоге docker.
* Убедиться, что все сервисы правильно связаны друг с другом через сеть Docker.

3.2 Запустить Docker Compose:

* В корневом каталоге проекта выполнить команду docker-compose up -d --build.
* Убедиться, что все контейнеры запущены и работают без ошибок.

**4. Разработка и тестирование API**

4.1 Реализовать API:

* В каталоге api создать эндпоинты для управления бронированиями и авторизацией пользователя.
* Убедиться, что API обрабатывает запросы корректно и возвращает правильные ответы.

4.2 Провести тестирование API:

* Проверить работу mock-сервера, сделав запросы с помощью инструментов, таких как Postman или curl.

**5. Разработка сценариев нагрузочного тестирования**

5.1 Написать скрипты для Locust:

* В каталоге tests создать сценарии для нагрузочного тестирования, используя Locust.
* Реализовать тесты для проверки производительности следующих операций:
  + Авторизация пользователя.
  + Добавление услуги в корзину бронирования.
  + Оформление бронирования и оплата.

5.2 Интеграция с Kafka и InfluxDB:

* Внедрить Kafka для обработки сообщений о бронировании.
* Записывать метрики производительности в InfluxDB.

**6. Выполнение нагрузочного тестирования**

6.1 Запуск тестов:

* Запустить Locust и выполнить нагрузочные тесты.
* Собрать метрики производительности с помощью InfluxDB.

6.2 Провести анализ результатов:

* Изучить результаты тестирования, выявить узкие места в производительности.

**7. Написание методики и отчетов**

7.1 Разработать методику нагрузочного тестирования:

* Описать методику проведения тестов, включая:
  + Описание тестируемого приложения и его API.
  + Цели нагрузочного тестирования.
  + Метрики, которые будут собираться.
  + Сценарии тестирования.

7.2 Написать отчет:

* Создать отчет по результатам тестирования, включающий:
  + Описание выполненных тестов.
  + Графики и диаграммы производительности.
  + Выводы и рекомендации по оптимизации.

**8. Завершение проекта**

8.1 Провести заключительные проверки:

* Проверить, что все компоненты проекта корректно работают.
* Убедиться, что созданные Docker-образы работают на других машинах без необходимости дополнительной настройки.

8.2 Архивирование и документация:

* Собрать все файлы проекта в единый архив.
* Добавить подробную документацию по развертыванию и использованию.