BostonGene. Тестовое задание

Сервис подсчета MD5 хеша для файлов.

Задание

1. Backend - необходимо реализовать минимальное REST API (FastAPI) в связке с очередью (Celery, можно использовать стандартную связку Celery+Redis+RabbitMQ), минимальной базой данных (Postgres + sqlalchemy для взаимодействия) и файлом логов с программно-реализованной последовательной записью. В API должны быть реализованы эндпоинты для загрузки файла и для доступа к результатам в базе.
Порядок действий API:
   * API принимает на вход файл
   * Файл отправляется в очередь (реализовать механизм Promise: создаем id хранения в API, возвращаем его после того, как положили файл в очередь (вместе с id))
   * Worker забирает данные из очереди, вычисляет MD5 хеш, записывает результат в базу с id, созданным как Promise, в конце исполнения результат должен быть записан в файл логов.
   * Файл логов - обычный файл в папке, которая монтируется к контейнерам worker-ов. Необходимо реализовать механизм последовательного взаимодействия worker-ов с файлом (например при помощи семафора).

2. Frontend - минимальное веб приложение, в котором можно загрузить файл и в ответ получить id. Также должно присутствовать поле, в котором можно получить результат по id.

Требования

1. Для backend части использовать python + fastapi + celery + sqlalchemy, не использовать django. Брокера для очереди и базу для результатов можно выбрать любые.

2. Должна присутствовать инструкция по запуску.

3. Готовое решение должно быть полностью докеризованным и подниматься с помощью docker-compose.

4. Все части должны быть докеризованы отдельно (в том числе worker и api).

5. Должна быть реализована возможность добавления дополнительных worker-ов путем добавления сервисов в docker-compose.yml.