Задание

С использованием связки SQLAlchemy+Redis+FastAPI необходимо реализовать:

- REST API (FastAPI):
- службу обработки изображений (в обработке использовать библиотеку pil, либо pillow);
- службу сохранения обработанной информации (в качестве БД использовать Mysql, MariaDB, PostgreSQL, либо SQLite);

и две очереди сообщений (Redis):

- очередь сообщений для обмена между REST API и службой обработки изображений;
- очередь сообщений для обмена между службой обработки изображений и службой сохранения обработанной информации;

REST API должен принимать на вход изображение (формат JPEG, размер 640x480 или близкий к этому) и его текстовое описание (кодировка UTF-8, не более 200 символов), далее отправлять полученную информацию, а также текущие дату и время в очередь сообщений (самостоятельно реализовать очередь с использованием Redis, формат сохранения информации определяется самостоятельно).

Служба обработки изображений извлекает из очереди сообщений изображение и его текстовое описание. Далее, с использованием библиотеки pil, либо pillow в нижней части изображения служба создаёт черное поле и в нём отображает текстовое описание (если длина текста превышает ширину изображения необходимо предусмотреть корректный перенос текста по словам). Высота чёрного поля должна соответствовать высоте выводимого текста. Цвет текста белый, шрифт можно взять любой распространённый (например arial), приблизительный размер шрифта 12px (лучше указать). Алгоритм формирующий чёрное поле и отображающий текст должен учитывать размер шрифта. По завершению отрисовки сообщение, состоящее из обработанного изображения, текстового описания и ранее полученных даты и времени отправляется в очередь на обработку службе сохранения обработанной информации.

Служба сохранения обработанной информации извлекает из очереди сообщение, сохраняет изображение в специально выделенную директорию (определить самостоятельно), сохраняет в БД текстовую информацию (дата/время приёма изображения, текстовое описание изображения, путь к сохранённому изображению).

Кроме того, REST API должен возвращать:

- список информации об обработанных изображениях (ID, дата/время, описание изображения) в формате JSON;
- возможность получения самого обработанного изображения по его ID.

В качестве транспортного протокола REST API использует HTTP.

Список вызовов REST API

URL	HTTP метод	Описание
/api/images	POST	загрузить изображение с описанием
/api/images	GET	получить список информации об обработанных изображениях
/api/images/<id>	GET	получить обработанное изображение по его ID

Создаваемый python-код должен содержать обработку ошибок, в том числе возврат кодов HTTP ошибок для вызовов REST API с при передаче некорректной информации (валидировать JPEG не надо, только факт его передачи). Ошибки, возникающие в работе служб необходимо обрабатывать и выводить в файловый журнал стандартной библиотекой logging, для необработанных ошибок записывать в вывод журнала их traceback.

Проект необходимо развернуть в Docker, сделать docker compose сценарий запуска. БД должна разворачиваться автоматически при запуске контейнера. Redis, PostgreSQL и т.д. могут быть отдельными контейнерами, полученными с hub.docker.com .

Результат выполнения задания разместить на любом git хостинге (github, gitlab, bitbucket и т. д.), либо выслать в архиве HR-у.

Результат выполнения задания должен содержать Markdown инструкцию по установке, запуску и проверке работы ПО.

Пример входного* и выходного изображений:

* описание входного изображения содержит фразу «С начала фазы запрещающего сигнала прошло: 7 секунд».

Разрабатываемое ПО должно:

- использовать конфигурационные файлы (на выбор: формат ini, json, yaml), либо переменные окружения для указания директории хранения обработанных изображений, названия, размера используемого шрифта;
- использовать Swagger.

Будет плюсом:

- покрытие кода тестами, в том числе интеграционными;
- развёртывание БД с использованием инструмента для миграции базы данных Alembic (для Mysql, MariaDB, PostgreSQL).