Документация по работе сервиса

Разработанное веб-приложение предназначено для обработки изображений. На вход приложению подаётся одно или несколько изображений. Результатом работы программы является одно изображение, либо текстовый результат обработки изображения.

Приложение предоставляет следующую функциональность:

* увеличение яркости изображения;
* обрезание изображения;
* соединение нескольких изображений в панораму;
* обнаружение объектов на изображении.

Функция увеличения яркости изображения позволяет увеличить яркость исходного изображения на заданное значение.

Функция обрезания изображения позволяет вырезать из исходного изображения изображение меньшего размера.

Функция соединения нескольких изображений в панораму позволяет склеить части панорамного изображения в одно изображение.

Функция обнаружения объектов на изображении позволяет обнаружить и выделить на изображении заранее известные объекты. Результатом работы метода являются прямоугольные координаты найденных объектов, название классов объектов и вероятность того, что классы объектов были определены правильно.

Взаимодействие с приложением происходит по сети Интернет посредством протокола http.

Ниже представлен стек используемых технологий:

* python3
* fastapi
* uvicorn
* mongodb
* docker.

Python используется в качестве языка программирования. FastAPI - веб-фреймворк для создания API, написанный на Python. uvicorn представляет собой реализацию ASGI веб сервера, написанного на Python. MongoDB используется в качестве базы данных. Docker - инструмент контейнеризации, облегчающий развёртывание приложений.

Приложение состоит из 3х docker контейнеров (рисунок 1): базы данных mongo, контейнера с приложением и контейнера с моделью распознавания объектов на изображении. Контейнеры состоят в общей локальной docker сети.

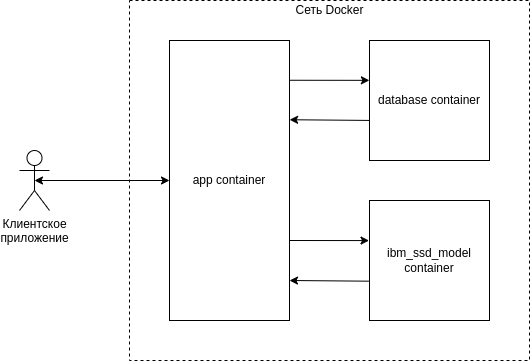


Рисунок 1 - Схема Docker контейнеров приложения

Рассмотрим подробнее структуру приложения, которая представлена на рисунке 2.

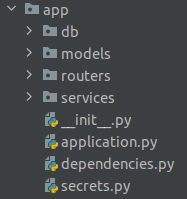


Рисунок 2 - Структура приложения

Приложение написано на языке python с использованием фреймворка FastAPI. Для хранения данных используется нереляционная база данных MongoDB. Она легка в освоении, гибка и удобна в использовании.

Вернёмся к рассмотрению структуры приложения. Приложение состоит из нескольких модулей и файлов.

Модуль db содержит код подключения к базе данных и вспомогательные методы для манипулирования данными (методы авторизации, создания пользователя и получения списка пользователей и другие).

Модуль models содержит классы, представляющие собой модели данных приложения, которые используется для описания входных данных и их валидации.

Модуль routers содержит контроллеры приложения - методы, которые обрабатывают запросы клиентов.

Модуль services содержит бизнес логику приложения - классы и функции для обработки изображений. Каждый класс представляет собой определённый сервис, выполняющий ровно одну вещь. Например, увеличение яркости изображения, определение объектов на изображении и тому подобное.

Рассмотрим, как происходит обработка изображения (рисунок 3). Клиент отправляет изображение и входные параметры на определённый метод контроллера. Контроллер принимает и валидирует входные данные, затем создаёт новую запись в таблице jobs. Получает созданный идентификатор - job\_id. Добавляет соответствующую задачу в очередь на фоновое выполнение, передавая функцию, которая будет выполняться и входные параметры. Возвращает клиенту идентификатор созданной операции job\_id.

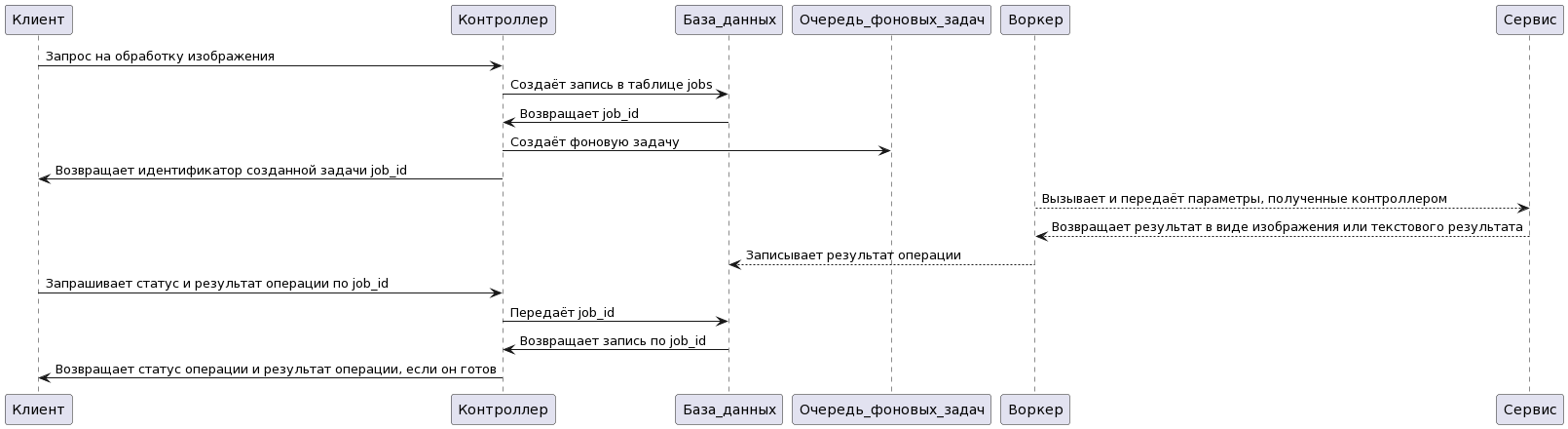


Рисунок 3 - Схема обработки изображения

Затем воркер - процесс, который занимается выполнением фоновых задач, достаёт задачу из очереди и запускает переданную функцию, передавая соответствующие параметры. Запущенная функция меняет статус операции на “в работе” и запускает соответствующий сервис (то есть вызывается класс, составляющий непосредственно бизнес логику приложения). Сервис выполняется и возвращает результат в виде модифицированного изображения, либо текстового результата (координаты распознанных на изображении объектов). Затем воркер записывает результат выполнения сервиса в базу данных (изображения на жёсткий диск). На этом выполнение фоновой задачи завершается.

Далее клиент запрашивает результат операции, отправляя запросы на специальный url. Если операция завершена, клиенту возвращается результат и статус операции.

Возвратимся к структуре приложения (рисунок 2). В файле application.py создаётся объект приложения FastAPI, подключаются контроллеры, содержатся основные настройки.

В файле dependencies.py содержатся общие функции, которые могут применяется во всех контроллерах. Например, методы авторизации.

В файле secrets.py содержатся секреты приложения, которые загружаются из env файла.