Описание

В данном задании вам предлагается реализовать бэкенд для веб-сервиса хранения файлов, аналогичный сервису Яндекс Диск. Обычно взаимодействие с такими сервисами происходит следующим образом:

1. Пользователь загружает и структурирует файлы в предложенном ему облачном пространстве.
2. Пользователь может скачивать файлы и фиксировать историю их изменений.

Ваша задача - разработать REST API сервис, который позволяет пользователям загружать и обновлять информацию о файлах и папках.

Технические требования

Реализуйте сервис на Python, Java или C++ в зависимости от выбранного направления школы. Сервис должен удовлетворять следующим требованиям:

- реализует спецификацию API, описанную в файле `openapi.yaml`, и корректно отвечает на запросы проверяющей системы
- некоторые обработчики из них являются необязательными, их реализация позволит вам набрать дополнительное количество баллов
- сервис должен быть развернут в контейнере на адресе `0.0.0.0:80`
- сервис должен обеспечивать персистентность данных (должен сохранять состояние данных при перезапуске)
- сервис должен обладать возможностью автоматического перезапуска при рестарте контейнера, в котором работает ваш бэкенд (этого можно достичь настройкой контейнера)
- после запуска сервиса время ответа сервиса на все методы API не должно превышать 1 секунду
- время полного старта сервиса не должно превышать 1 минуту
- импорт и удаление данных не превосходит 1000 элементов в 1 минуту
- RPS (Request per second) получения истории, недавних изменений и информации об элементе суммарно не превосходит 100 запросов в секунду

Тестирование

В качестве предварительного тестирования мы подготовили для вас юнит-тест `unit_test.py`, написанный на Python. Он позволит проверить минимальную работоспособность вашего бэкенда до отправки решения на проверку.

Для прохождения проверки обратите внимание на следующее:

- Коды ответа HTTP.
- Корректность JSON структуры запроса и ответа.
- Типы данных (строки, числа).
- Формат даты.
- Проведение необходмых валидаций входных данных.
- Краевые случаи.
- Формат и коды ошибок.

Рекомендуется написать свои тесты для проверки разработанной функциональности.

Развёртывание приложения

На выделенном контейнере вы можете:

1. Работать с вашим приватным репозиторием.
2. Использовать средства контейнеризации, например Docker, он уже установлен.
3. Устанавливать программное обеспечение для сборки и запуска вашего приложения, используя Интернет. Например, для реализации задания на языке Java вы можете использовать Maven или Gradle. Явных требований по сборке и развертыванию приложения нет.
4. Настраивать контейнер по своему усмотрению (например, настройки автозапуска или версии Java).

Оценивание решения

Оценивание решения будет проходить после отправки решения кандидатом на проверку в несколько этапов.

1. Автоматическое тестирование. Проверяющей системой будут выполнятся запросы к вашему бэкенду. Будет проверятся корректность ответов, их коды ответа HTTP и тела.
2. Ручная проверка решения проверяющими Школы бэкенд-разработки. Будут учитываться различные особенности вашего решения:
   - Способ решения.
   - Качество решения.
   - Возможность обработки нескольких запросов сервисом одновременно.
   - Покрытие тестами.
   - Документация описывающая код, сборку, запуск и работу приложения.

Полезные материалы

1. Подробнее про спецификацию OpenAPI вы можете узнать здесь Спецификация OpenAPI
2. Практическое руководство по разработке бэкенд-сервиса на Python
3. Визуализация файла спецификации Open API
4. Автозапуск сервера при рестарте контейнера

FAQ

Как обратиться к моему приложению с рабочего компьютера?

Как и проверяющая система, вы можете сделать запрос по адресу `https://hostname.usr.yandex-academy.ru` для проверки своего решения, где hostname - это название вашего контейнера (можно получить войдя в контейнер).

Как подключиться к базе данных внутри моего контейнера?

Вы можете развернуть требуемую базу данных внутри Docker (или другой программы для контейнеризации) или на выданном вам контейнере. В качестве примера вы можете изучить статью на Habr , рассказывающую о развёртывании PostgreSQL в Docker контейнере.

Почему не использовали oneOf в спецификации OpenAPI?

Поскольку у OpenAPI в последних версиях есть нерешенные проблемы с кодогенерацией oneOf классов, мы решили отказаться от использования такой функциональности.

Правки

Здесь будут описаны внесенные изменения/правки в задание.