Конструирование программного обеспечения

Домашняя работа №3
Синхронное межсервисное взаимодействие

Антиплагиат

В преддверии сессии, в связи с необходимостью проверки присланных контрольных работ, у профессорско-преподавательского состава увеличивается нагрузка, и преподаватели не всегда успевают выставить оценки вовремя. В связи с этим вас просят разработать информационную систему, которая будет организовывать хранение работ, присланных студентами, а также формировать отчет по каждой работе по результатам проверки на заимствования (антиплагиат).

Ваш код — единственное оружие против плагиата.

User flow

В ходе общения с преподавателями было составлено словесное описание последовательности действий пользователя, переходов между этапами взаимодействия с информационной системой. В первой версии (MVP) было принято не вводить роли (студент, преподаватель), но для большей прозрачности они упоминаются в описанном user flow.

1. Пользователь (студент) должен иметь возможность отправлять работу на проверку с указанием информации о том, кто отправляет данную работу и по какому заданию.
    a. система должна фиксировать факт сдачи работы в СУБД (кто/когда/по какому заданию) и сохранять присланный файл на сервере (по желанию сохранять можно в S3 хранилище).
    b. после загрузки файла запускается анализ присланной работы и создается отчет, который также сохраняется локально на сервере.
2. Пользователь (преподаватель) может запросить аналитику по той или иной контрольной работе (пример: GET/works/{work_id}/reports → получаем json со статусами и флагом плагиата по всем соответствующим контрольной работе отчетам).

Вам необходимо самостоятельно продумать и описать в README алгоритм определения признаков плагиата. В самом простом варианте можно считать, что плагиат присутствует, если существует более ранняя сдача (другим студентом) присланной работы.

Вы можете изменить предложенный user flow (расширить) если это не стыкуется с вашим видением.

Требуется: разработать микросервисную архитектуру серверной части веб-приложения

Архитектурный комитет предлагает следующее решение серверной части веб-приложения с четким разделением ответственности микросервисов:

1. File Storing Service – отвечает только за хранение и выдачу файлов.
2. File Analisys Service – отвечает только за проведение анализа, хранение результатов (отчетов) и выдачу отчетов.
   Формат запросов и ответов между сервисами определяется на ваше усмотрение.

А также API Gateway, который будет являться центральным сервисом-посредником, принимающим все запросы от клиентов и маршрутизирующим их к соответствующим микросервисам.

Вы в праве предложить иную организацию микросервисов.

Критерии оценки

1. Реализация основных требований **– 2 балла**
2. Распределение функциональности по микросервисам – **2 балла**
    a. Серверная часть веб-приложения состоит минимум из двух бизнес-микросервисов, а также сервиса-посредника.
    b. Обработка ошибок (один из микросервисов упал).
3. Использование контейнеризации – **2 балл**
    a. Корректность Dockerfile и docker-compose.yml
    b. Все микросервисы должны быть упакованы в Docker-контейнеры.
    c. Вся система должна разворачиваться с помощью docker-compose.yml (запуск командой docker compose up)
4. Реализация коллекции Postman / Swagger, которая должна демонстрировать функциональность реализованных микросервисов, охватывая все API **– 1 балл**
5. Качество кода и документация **– 2 балла**
    a. Хорошо организованный, модульный и “чистый” код.
    b. Документация должна включать:
        i. краткое описание архитектуры системы
        ii. пользовательские сценарии микросервисов: технические сценарии обмена данными и взаимодействия между сервисами для обеспечения работы системы.

    Требования на 10 баллов (прежде должны быть выполнены все предыдущие критерии)
6. Визуализация присланной работы в виде облака слов (документация к API построения облака слов: https://quickchart.io/documentation/word-cloud-api/) **– 1 балл**

Штрафы

1. До – 2 баллов за наличие ошибки во время выполнения кода.
2. – 1 балл за каждый день просрочки дедлайна.