Задание номер 4

У компании "Рога и копыта" есть сервис "RED", который работает по REST и определяет количество красного цвета на изображении. Сервис стал очень популярным и компания решила добавить ещё один сервис "RED_STATS", в который "RED" скидывает сообщения о том что у такого-то клиента пришла картинка с таким-то количеством красного цвета. Для клиентов у "RED_STATS" должны появиться две следующие функции: возможность подписаться на оповещения, что пришла картинка, у которой красного больше, чем заданное клиентом значение. Получать статистику по времени, сколько вообще за данный промежуток было послано изображений, сколько изображений, у которых красного было больше заданного значения. Ваша задача спроектировать работу "RED_STATS", как он будет общаться с клиентами, как хранить данные, как масштабироваться и т.д

Попрошу учитывать, что в схемах находится pseudo_code позволяющий кратко описать нужный блок функционала и в реальности там могут находится несколько другие конструкции

Для выполнения поставленной цели проектирования сервиса RED_STATS, необходимо решить вытекающие из описания задания задачи, а именно:

1. Определить, как базово работает сервис RED
2. Сформулировать какой функционал необходим сервису RED_STATS
3. Спроектировать хранение данных и обновить базовый функционал сервиса RED для корректной работы RED_STATS

Решение задачи 1

Предположим, что сервис RED это веб сервис, в базовом функционале которого имеется один API endpoint /red, содержащий в себе следующий функционал.

- Функция извлечения изображения из http запроса. Способ передачи изображения по http может быть как multipart, так и просто бинарный файл. Actor может передавать множество изображений, может одно в данном контексте это не столь важно.
- Функция расчёта количества красного цвета на изображении (технически данный функционал в рамках архитектуры не имеет особого значения и относится к бизнес-логике приложения)

Сервис в ответ на запрос возвращает количество красного на переданном ему изображении.

Схематично базовый функционал сервиса RED показан на рисунке 1
Такого абстрактного функционала в целом достаточно, чтобы сервис мог работать. Однако для работоспособности сервиса RED_STATS необходимо его улучшение

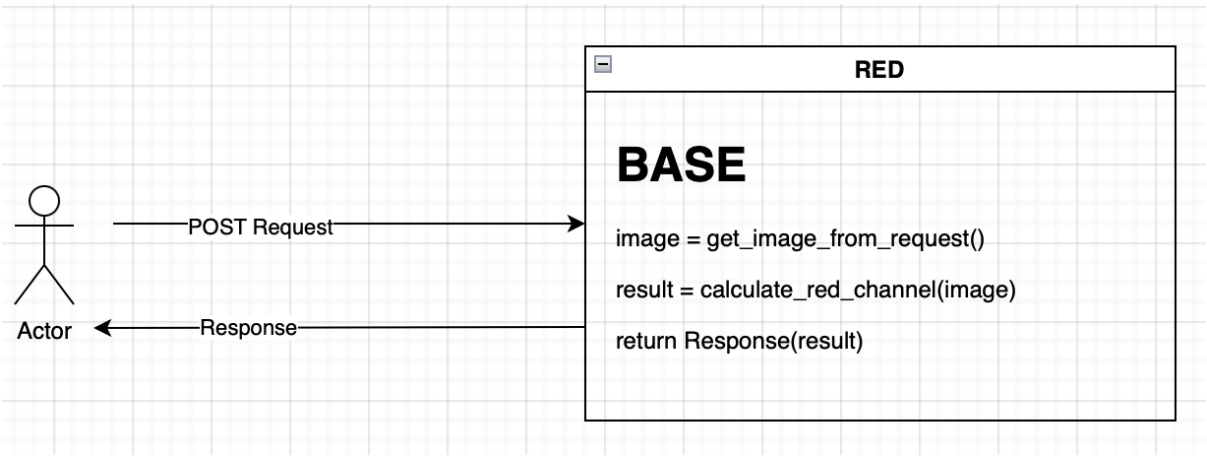

Рисунок 1: Базовая схема RED

Решение задачи 2:

Базовый функционал RED_STATS:

1. возможность подписаться на оповещения, что пришла картинка, у которой красного больше, чем заданное клиентом значение.
2. Получать статистику по времени, сколько вообще за данный промежуток было послано изображений, сколько изображений, у которых красного было больше заданного значения

Сперва разберем каким образом можно подписаться на оповещение, в рамках функционала сервиса RED_STATS. Один из способов — это реализация через webhook, когда клиент отправляет нам сообщение о том, что он хочет получать оповещения на указанный им адрес.

Также если в рамках сервиса RED не было острой необходимости производить авторизацию и аутентификацию клиента, то для работы описанного функционала её необходимо производить, так как без привязки к конкретному клиенту будет невозможно реализовать вторую часть базового функционала RED_STATS.

Далее рассмотрим функционал со статистикой:
Для его реализации в рамках сервиса должна храниться следующая информация:

1. Рассчитанное количество красного цвета для каждого изображения
2. В какой момент времени был произведен расчёт
3. Идентификатор клиента, для которого был произведён расчёт

Базовая схема для функционала 1 и функционала 2 представлена на рисунке 2

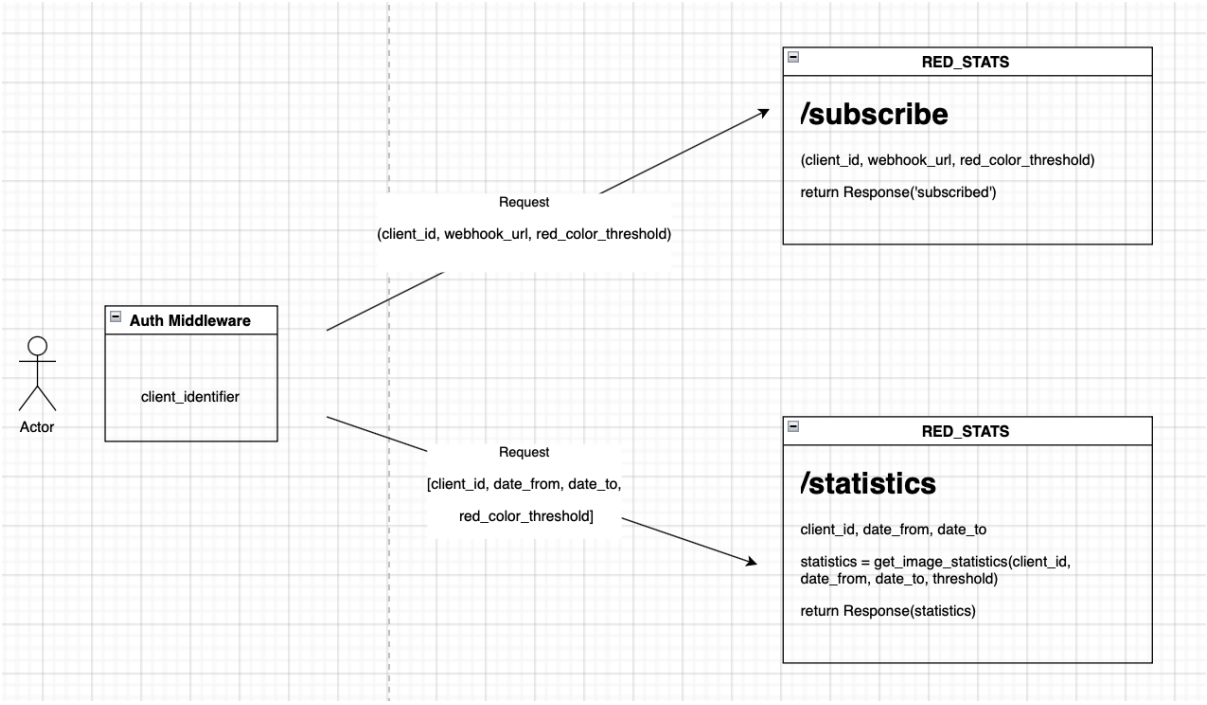

Рисунок 2 Схема базового функционала

Решение задачи 3:

Определившись с тем, как должен работать базовый функционал, необходимо определиться с тем, где и каким образом будут храниться данные, обновить сервис RED, и продумать систему доставки для подписки пользователем.

Обновим функционал RED:
Так как в сервисе появилась привязка к клиенту, при выводе статистики, необходимо чтобы сервис RED записывал куда-то информацию о расчёте красного цвета, сохранял изображения, делая всё это с привязкой к client_id. В качестве хранения meta информации о красном цвете я выбрал связку kafka и postgresql.

Kafka позволит построить систему оповещения, а **PostgresQL** будет хранить информацию для выгрузки статистики. Однако сервис RED будет выгружать само изображение в файловое хранилище, а мета информацию о нём посылать в брокер сообщений kafka.

Также для хранения информации вида client_id -> webhook_url и client_id -> red_channel_threshold будет использоваться **REDIS** в качестве простой NoSQL базы данных. Ведь в будущем возможно будет расширение функционала до нескольких адресов подписки для клиента.

Схема взаимодействия RED и хранилищ данных представлена на рисунке 3

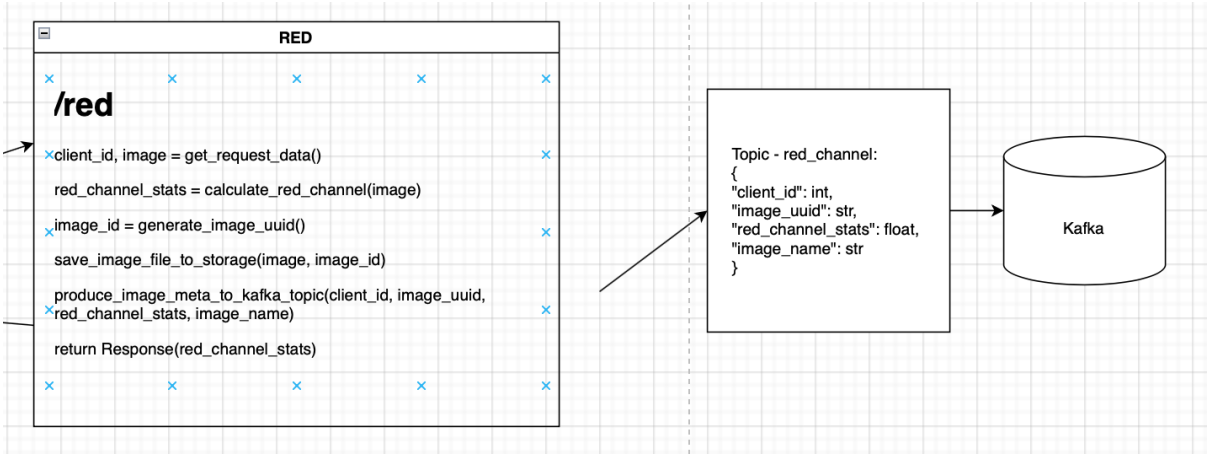

Рисунок 3 – RED и взаимодействие с KAFKA.

Теперь необходимо решить задачу, связанную с доставкой сообщений до клиента, для этого будет добавлен сервис **RED_STATS_CONSUMER** основным функционалом которого будет являться:

- Потребление сообщений из брокера **KAFKA**
- Отправка сообщений клиенту по webhook_url
- Сохранение сообщения в базу данных **PostgresQL**

Прошу заметить, что вместо **KAFKA** может быть использован брокер сообщений **RabbitMQ**, а настройка брокера должна быть - один потребитель одно сообщение.

В результате должна получится система, отвечающая требованиям указанным в задании 4, полная схема представлена в репозитории на рисунке task_4_scheme.png.