Тестовое задание (digest job tracker v2)

Общие требования

Необходимо разработать веб-приложение, которое выполняет следующие задачи:

1) Пользователь при первом посещении получает уникальный идентификатор, далее все действия, которые он производит ассоциированы с этим идентификатором. Уникальный идентификатор имеет формат UUID и должен отображаться на каждой странице в footer’e.
2) Пользователь может запускать обработку задач, задача состоит в расчете хэш-суммы некоторого объекта. Задача имеет аттрибуты
   a) src - URI объекта, может быть указателем на локальный файл (file:///etc/hosts), так и удаленный объект (https://www.linkedin.com/robots.txt)
   b) algo – тип алгоритма хэширования, возможные значения – md5, sha-1, sha-256

   При выполнении задачи происходит получение объкта по адресу src и подсчет хэш-суммы с использованием алгоритма algo.

3) Обработка задачи происходит на бэкенде в фоновом режиме, пользователь может просматривать список задач, которые он запускал на обработку с указанием статуса, который может быть следующим:

a) Ожидает обработки (обработка еще не начата, так как в системе нет ресурсов)
b) Обработка в процессе (с указанием потраченного времени в секундах)
c) Обработка завершена успешно (с указанием результата выполнения задачи – дайджест в виде hex, например d8e8fca2dc0f896fd7cb4cb0031ba249)
d) Обработка завершена с ошибкой (с указанием текста stack trace для возможности анализа проблемы)

4) Пользователь может удалять задачи из списка, в случае если их обработка завершена. Также пользователь может отменять обработку задач, в случае если они ожидают обработку, либо если обработка в процессе.
5) Пользователь может загрузить ссылки файлом (одна строка - одна ссылка)

   Реализация очереди должна подразумевать настраиваемые ограничения (не более N одновременных закачек)

   Реализация очереди должна контролировать количество обращений к ресурсу (не более N раз в M секунд)

Нефункциональные требования

Обязательные требования:

1) Java8/scala.
2) Система сборки – maven/gradle/sbt.
3) Наличие тестов.
4) REST на бэкенде.

5) На фронтенде – knockout,angular, react или любой другой современный JS фреймворк.

Дополнительные требования (можно не реализовывать, но нужно описание как это можно будет сделать, и заложить возможность реализации в архитектуру):

1) Наличие возможности масштабировать обработку на бэкенде.
2) Запуск и развертывание через docker.
3) Персистентность обрабатываемых задач.
4) Мониторинг процесса обработки.