# Тестовое задание по теме “Crawler Frontier для веб-скрапинга данных из социальных сетей”

# API музыкального сервиса

## Описание

Spotify, увы, не доступен больше. А аналоги весьма печальны. Чтож, есть желание разработать собственный музыкальный сервис.

## Ключевые сущности системы

* Исполнитель (персона / группа / фит). Стоит учесть, что хочется искать песни по каждому из участников группы.
* Песня.
* Альбом.

## Предполагаемое поведение

Система подразумевает под собой наличие двух ролей: обычный пользователь и администратор.

Администратор может загружать / удалять / редактировать существующие данные о песнях / альбомах / исполнителях и только. Штатное использование системы администратором не предусмотрено.

Пользователь может лайкать треки / альбомы / исполнителей. На основе лайков, пользователь может запросить рекомендуемые треки (поскольку у нас не машинки, то это будет некий рандом исходя из жанров песен, полюбившихся исполнителей, который включает в себя неизвестные пользователю треки). А так же есть обычный поиск по жанрам / наименованиям всех ключевых сущностей. Также существует фича, при помощи которой пользователь может посмотреть свою активность в разрезе условно года, т.е. топ-5 жанров, исполнителей и песен, которым уделял пользователь время

### Исполнитель

Страницы персоны включает следующую информацию:

* Имя + Фамилия
* Фотообложка (если она существует байты в ответе)
* Кол-во прослушиваний
* Реестр песен
* Реестр выпущенных альбомов

Страницы музыкальных групп включает следующую информацию:

* Наименование группы
* Фотообложка (если она существует байты в ответе)
* Ссылки на участников группы
* Кол-во прослушиваний
* Реестр песен
* Реестр выпущенных альбомов

### Песня

Прослушивание песни подразумевает собой, что сервер отправит следующую информацию:

* Название песни
* Обложка (если она существует байты в ответе)
* Длительность композиции
* Сама композиция (не будем заморачиваться стримингом, просто вернем байты в ответе)

### Альбом

* Наименование альбома
* Обложка (если она существует байты в ответе)
* Общее кол-во прослушиваний
* Реестр песен

Возможно я что-то забыл, и остались тонкие непонятные моменты, но тогда лучше поинтересоваться у меня и согласовать их

## Техническая сторона вопроса

### Используемые технологии

* IntelliJ Idea Ultimate (активация через универ должна получиться)
* Scala 2.12.17
* Play 2.8.18
* Sangria 3.5.2 (придется делать GraphQL API, а не обычный rest)
* Slick
* PostgreSQL
* (\*) Желательно API с базой данных запихнуть внутрь docker-compose.yml, чтоб в случае чего я мог поднять приложение у себя локально. В случае чего могут быть получены подсказки от нас. Желательно вообще не устанавливать на комп PG, а прям сразу поднимать его в докере
* (\*) Может быть хотелось бы хранить файлы и изображения в minio, а не в постгресе. Но делать это рекомендуется только в том случае, когда лаба уже завершена и до дедлайна остается время. Получишь только плюсик в карму

Сложный момент у тебя будет с тем, как при помощи GraphQL загружать файлы в систему. С этим тебе может помочь Altair GraphQL Client для тестирования.

(\*) Поскольку есть несколько ролей у системы, необходимо защитить query/mutation (Пользователь в хэдерах запросов должен передавать токен, а ты проверять, что он действительно валидный, принадлежит этом пользователю и этот пользователь может использовать этот метод. Поскольку над генерацией можно не заморачиваться, то можно его генерировать и запоминать после регистрации пользователя в БД. Поскольку у нас нет UI, то будем считать, что он всегда хранится как куки-файлик и после регистрации всегда известен пользователю)

## Процесс выполнения

1. Проектируешь схему базы данных. Указываешь, какие именно будут таблицы, какие столбцы (с указанием типов), связи между таблицами. Проводим ревью этого этапа. (Сделано за вас, иначе очень много изучать и делать, времени на все не хватит). Скрипты: [create\_db.sql](https://drive.google.com/open?id=10EYVRgoGzh8-Duvm6fDRpk939sqt5Jwv)
2. Проектируешь GraphQL API. В нем указываешь, какие эндпойнты будет содержать система, кому они доступны/недоступны, какой payload будет у каждого из методов
3. Реализация системы. Проводим ревью
4. Разрабатываешь формат загрузки / редактирования песен и прочей информации администратором, если время осталось
5. Можно поделать пункты под звездочкой

## Полезные ссылки

* Scala
  + <https://www.coursera.org/specializations/scala> – курсы, которые помогут вникнуть поверхностно в язык
  + <https://books.underscore.io/essential-scala/essential-scala.html> – так же можно почитать
  + <https://danielwestheide.com/books/the-neophytes-guide-to-scala/> – достаточно продвинутый и короткий онлайн-блог
* Slick
  + <https://underscore.io/books/essential-slick/>
  + <https://github.com/tminglei/slick-pg>
* Play – <https://www.playframework.com/>
* Sangria
  + <https://sangria-graphql.github.io/getting-started/> – оф дока
  + <https://www.howtographql.com/>