УДК 004.457

АВТОМАТИЗАЦИЯ ПРОЦЕССА СОСТАВЛЕНИЯ СПИСКА ВОСПРОИЗВЕДЕНИЯ МУЗЫКАЛЬНЫХ ФАЙЛОВ И УЧЕТА ПРОСЛУШИВАЕМЫХ ПЕСЕН С ПОМОЩЬЮ СЕРВИСА LAST.FM

Набережнев П.Е., студент гр. ИТб-132, IV курс
Научный руководитель: Турчин Д.Е., ст. преподаватель
Кузбасский государственный технический университет
имени Т.Ф. Горбачева
г. Кемерово

При прослушивании музыки возникают проблемы составления списка воспроизведения и scrobbling-а песен для группы пользователей с одного устройства.

Scrobbling – термин, придуманный разработчиками сервиса Last.fm для обозначения операции отправки метаданных о каждой песни, которые были проиграны музыкальным приложением, в сервис Last.fm с целью дальнейшей обработки. Существующие системы, поддерживающие scrobbling музыкальных композиций, такие как AIMP и Last.fm Desktop App, дают возможность scrobbling-а песен только для одного пользователя. При прослушивании музыки группой людей с одного устройства, например, компьютера, оснащенного аудио системой, выше упомянутые системы не позволяют вести учет прослушанных песен для каждого пользователя отдельно. Учет ведется только для одного пользователя, который авторизирован на устройстве.

Существующие системы по работе с музыкальными библиотеками дают возможность составления плейлистов из папок, единичное добавление песен или добавления выделенных песен в различные части плейлиста. В приложении AIMP также предусмотрена возможность автоматического составления плейлиста, используя различные условия для песен, которые будут содержаться в нем. Также предусмотрена возможность составления шаблонов сортировки по различным данным песен в плейлисте. Но система не рассматривает более сложных алгоритмов составления списков воспроизведения. Например, пользователь планирует прослушать двух исполнителей, количество песен в которых 26 и 10 соответственно. Порядок воспроизведения должен быть следующим: 2 песни первого исполнителя, 3-тья второго исполнителя, и так далее, пока не закончатся песни второго исполнителя, так как количество его песен меньше чем у первого. Также пользователю необходимо, чтобы каждая 5 песня в плейлисте была какого-то определенного жанра. Составление подобных плейлистов, используя различные шаблоны составления списка воспроизведения, существующими системами не предусмотрено.

Цель данной работы состоит в обеспечении автоматизированного составления списка воспроизведения, используя различные шаблоны его формирования, и учета прослушанных песен с помощью веб-сервиса Last.fm.

К числу основных задач работы относятся:

- определить модели хранения данных, представляющие плейлист;
- определить состав и структуру классов, представляющих шаблоны составления плейлиста;
- определить модели данных для хранения прослушанных песен;
- описание процесса scrobbling-а с помощью веб-сервиса Last.fm.

Необходимые для хранения плейлистов модели данных представлены на рисунке 1.

Рисунок 1. Диаграмма классов, представляющих модель данных хранения плейлистов

Необходимые для реализации шаблонов составления плейлиста абстракции представлены на рисунке 2.

Рисунок 2. Диаграмма классов, представляющих абстракции шаблонов составления плейлистов

Интерфейс **IMixer** инкапсулирует алгоритм «перемешивания» песен, переданных в его метод Mix в качестве параметра. Метод возвращает новый список уже отсортированных в определенном алгоритмом порядке песен. Данный интерфейс предполагает возможность реализации с помощью композиции других реализаций для обеспечения более гибкого составления списка воспроизведения.

Интерфейс **IInsertStrategy** инкапсулирует алгоритм вставки в плейлист песен, переданных в метод Insert. Реализациями данного интерфейса будут являться следующие алгоритмы вставки песен: в конец плейлиста, в начало плейлиста, после текущей воспроизводимой песни и т.д.

При составлении списка воспроизведения, выбирая нужный **IMixer** и их комбинацию, можно получать необходимые наборы песен для прослушивания, прилагая к этому минимум усилий. Благодаря **IInsertStrategy,** полученный набор песен может быть добавлен в существующий плейлист различными способами.

Необходимые для реализации scrobbling-а песен для группы пользователей модели данных представлены на рисунке 3.

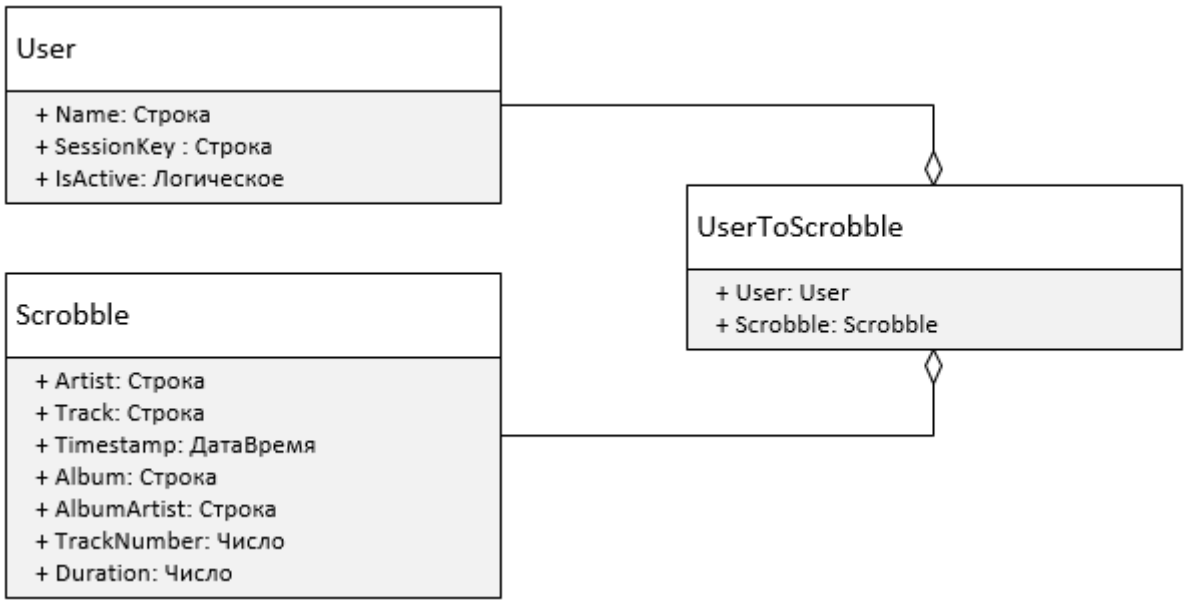

Рисунок 3. Диаграмма классов, представляющих модели данны scrobbling-а песен.

Модель **User** представляет пользователя, который может прослушивать музыку.

Модель **Scrobble** представляет прослушанную песню в определенный момент времени.

Модель **UserToScrobble** представляет прослушанные пользователем песни.

После прослушивания песни формируется запись сущности **Scrobble**, атрибуты которого соответствуют метаданным прослушанной песни, а **Timestamp** – времени начала прослушивания. После чего для всех пользователей, у которых атрибут **IsActive** равен «Да» (обозначает, что пользователь в текущий момент времени прослушивает музыку), формируются сущности **UserToScrobble** со ссылкой на ранее созданный **Scrobble**. Если при прослушивании музыки нет доступа к интернету, то прослушанные песни необходимо сохранять в локальном хранилище. В противном случае нужно отправлять полученные прослушивания на веб-сервис Last.fm используя его API.

API Last.fm включает в себя два основных запроса для scrobbling-а песен:

- отправка метаданных песни, которая прослушивается в данный момент;
- отправка метаданных списка прослушанных песен.

Каждый из запросов расположен по корневому адресу – http://ws.audioscrobbler.com/2.0/.

Для отправки метаданных песни, прослушиваемой в данный момент, необходимо отправить запрос на корневой адрес с параметрами:

- **method** равным «track.updateNowPlaying»;
- **artist** – название исполнителя;
- **track** – название песни;
- **api_key** – Last.fm API ключ;
- **api_sig** – Last.fm MD5 хэш.
- **sk** – ключ сессии, сгенерированный при аутентификации пользователя Last.fm.

Параметр **api_key** представляет собой уникальный ключ, который выдаётся сервисов Last.fm при регистрации приложения. Вместе с ключом выдается «shared secret», использующийся при аутентификации запросов, отправляемых приложением.

Параметр **api_sig** получается путем конкатенации всех параметров запроса в виде «{key}{value}». После чего в результирующую строку добавляется «shared secret», и по полученному результату вычисляется MD5 хэш. Пример построенной строки:

«api_keyxxxxmethodtack.updateNowPlayingxxxxsecret».

Параметр **sk** получается во время первой аутентификации пользователя в приложении. Полученный ключ может быть использован неограниченное количество времени до тех пор, пока пользователь не запретит приложению отправлять данные прослушанных песен. Процедуры аутентификации пользователя различны в зависимости от типа приложения (веб, настольное, мобильное приложения), с которого будут отправляться запросы. Для настольного приложения процесс проходит в 3 этапа:

1. Получение token-а приложения.
2. Запрос авторизации пользователя.
3. Получение ключа сессии.

Для получения token-а на сервис Last.fm отправляется запрос с параметрами**: method=auth.getToken**, **api_key**, **api_sig**.

Для авторизации пользователя ему необходимо открыть страницу браузера, которая расположена по следующему адресу: http://www.last.fm/api/auth/?api_key=xxxx&token=xxxx. На данной странице пользователь может установить соответствующие разрешения на отправку запросов приложением. Далее необходимо, используя тот же token, получить ключ сессии. Для этого необходимо отправить запрос на корневой адрес с параметрами: **method=auth.getSession**, **api_key**, **token**, **api_sig**. Полученный ключ сессии необходимо сохранить с ассоциирующимся пользователям, чтобы в дальнейшем вести учет прослушанных песен для него.

Для отправки метаданных прослушанных песен необходимо отправить запрос на корневой адрес со следующими параметрами:

- **method** равным «track.scrobble»;
- **artist[i]** – название исполнителя;
- **track[i]** – название песни;
- **timestamp[i]** – время начала прослушивания в формате UNIX;
- **api_key** – Last.fm API ключ;
- **api_sig** – Last.fm MD5 хэш;
- **sk** – ключ сессии, сгенерированный при аутентификации пользователя Las.fm.

Где **i** – индекс отправляемой песни. В пределах одного запроса поддерживается отправка не более 50 песен.

Таким образом, используя сформированные модели данных для хранения прослушанных песен и состава пользователей, возможен учет прослушанных песен для группы пользователей в сервисе Last.fm. С помощью различных комбинаций реализаций абстракций **IMixer** и **IInsertStrategy** возможно автоматизировать процесс составления списка воспроизведения.

Список литературы:

1. Last.fm API [Электронный ресурс] – Электрон. текстовые дан. – 2017. – Режим доступа: https://www.last.fm/api/intro