Bachelorproef AMuRate: Project Plan 3e Bachelor Computerwetenschappen

David Sverdlov
dsverdlo@vub.ac.be

May 25, 2013

Contents

1 Inleiding

1.1 Doel

De uitdaging van dit bachelorproject is om een applicatie te ontwikkelen voor het mobiele platform Android. De applicatie moet gebruikers in staat stellen om muzieknummers op te zoeken en een beoordeling te kunnen geven. Die scores worden dan opgestuurd naar een data-collection server met een database, waar later recommendeer algoritmen op zullen werken. (De implementatie daarvan valt niet binnen de scope van dit project.)
De promotor van dit project is Dr. Peter Vranckx en mee begeleidt door Maarten Deville.

1.2 Projectbeschrijving

Gebruikers moeten eerst en vooral muziek kunnen opzoeken. Hier zal een scherm voor zorgen, waar men de zoekgegevens kan invullen. Er is een veld voor de titel van een lied in te vullen, en een veld voor de artiest- of groepsnaam. Beide velden hoeven niet tesamen ingevuld te zijn, het volstaat dat er maar een veld ingevuld is. Er zal een melding gegeven worden wanneer er op de zoek-knop geduwd werd zonder dat er iets van zoekgegevens werd opgegeven.

Een hulp-knop informeert mensen over het (correct) gebruik van de applicatie. Dit gebeurt door een pop up en kan gesloten worden door op OK te klikken.

Wanneer er geen werkende data connectie is, zij dat de data-connectie niet aan staat, of dat er geen beschikbaar netwerk in de buurt is, zal de gebruiker een melding krijgen dat hij een werkende data verbinding nodig heeft eer informatie gedownloaded kan worden.

Wanneer een gebruiker beschikt over een actieve data verbinding en minstens één zoekterm ingevuld heeft, zal de applicatie een HTTP GET oproep vormen en naar de Last.fm API sturen. Op het scherm zal een indicator tevoorschijn komen dat er een oproep behandeld wordt. Eens aangekomen, zal de Last.fm API de oproep analyseren en een antwoord vormen met de opgevraagde informatie. Eens dat antwoord gedownloaded is, zal de applicatie het resultaat van de oproep aan de gebruiker tonen. Zie hoofdstuk GUI voor een visueel voorbeeld van dit proces.

Wanneer men kiest om de resultaten te bekijken, wordt er een lijst getoond met alle opties. Door op een van de resultaten te klikken, kan er naar het volgende scherm gegaan worden: een gedetailleerde pagina met

meer informatie dan de beknopte optie waarop geklikt werd.

Het gedetailleerde scherm van een lied geeft wat triviale informatie over het lied in kwestie, zoals de duratie, album, artiest, ... Daarnaast nog een afbeelding van het album[1]. Indien er meer album informatie beschikbaar is, zullen de velden die informatie geven over het album een andere kleur krijgen en dienen als knop om naar de gedetailleerde pagina voor een album te gaan.
Als laatste is er een deel van het scherm bedeeld aan het score systeem. Naast de mogelijkheid om zelf eenmalig een score te geven (tussen 0 en 5, met stappen van een halve precisie), wordt ook de gemiddelde score weergegeven van alle scores voor dat lied (en het aantal scores waar het gemiddelde van berekend werd).

Het gedetailleerde scherm voor een album toont enkele informatieve zaken van het album en alle liedjes van het album. De gebruiker kan op elk lied klikken om naar het scherm te navigeren met meer informatie en/of een score te geven.

Wanneer een score toegediend wordt en er een werkende data verbinding is, moet de score niet alleen in de lokale database toegevoegd worden, maar ook naar de server gestuurd worden. De scores worden geidentificeerd via het unieke Android device nummer van het GSM toestel. Dit is een 64-bit hex string.

1.3 Afkortingen en definities

1.3.1 API

API staat voor Application Programming Interface en zorgt voor de communicatie tussen programmas, door de scheiding te vormen tussen verschillende lagen van abstracties.

1.3.2 XML/JSON

XML staat voor Extensible Markup Language en is een van de meest gebruikte opmaaktalen, die gestructureerde gegevens kunnen omzetten in platte tekst. (Om het zo makkelijk(er) door te kunnen sturen.)
JSON is aan afkorting van JavaScript Object Notation en is een alternatieve simpele manier om objecten voor te stellen als platte tekst.

[1] Indien er een afbeeldings URL beschikbaar is, anders een standaard blanco afbeelding.

1.3.3 HTTP

HTTP staat voor HyperText Transfer Protocol en is het medium tussen een webbrowser en een webserver. Die communicatie gebeurt door middel van URLs, die verwijzen naar 'iets' op een of andere webserver.

1.3.4 GUI

GUI staat voor Graphical User Interface en is een visuele vormgeving van een programma, dat door middel van knoppen, afbeeldingen, ... gebruikers toelaat om op een gebruiksvriendelijkere manier met de applicatie om te gaan .

1.4 MBID

MBID staat voor MusicBrainz Identifier. MusicBrainz is een grootschalige onderneming die een universeel, uniek muziek identificerings systeem implementeert. Een ID bestaat uit 36 karakters en kan gebruikt worden voor liedjes, artiesten, albums,...

2 Achtergrond

2.1 Last.fm/API: REST

Om informatie (over muziek in dit geval) op te kunnen zoeken, moet er gebruik gemaakt worden van een online database met een openbare en hanteerbare API. *Last.fm* is een muziek recommendation service met een enorme online muziek database, die een gratis (lees: voor geregistreerde gebruikers) API aanbiedt, die iedereen toelaat om mobiele/desktop programmas of web services te bouwen met hun data.
Die procedure verloopt als volgt: om data uit de database te halen moet er een call (oproep) gestuurd worden. Deze gebeurt via HTTP GET naar de Last.fm server die op zijn beurt antwoordt met een XML object (of JSON op aanvraag). De methode van operatie die hier gebruikt wordt heet 'REST'. De API verschaft tientallen methodes zoals artist.search, artist.getTopAlbums, artist.getTopTracks, track.getInfo, track.search, track.getTags, album.search, album.getBuyLinks, ... De search en getInfo oproepen zullen het meeste gebruikt worden in dit project.

2.2 REST architectuur

REST staat voor Representational State Transfer en is een type van architectuur voor bepaalde gedistribueerde systemen (zoals het World Wide Web), waarin een duidelijke onderscheiding wordt gemaakt tussen client en

server. Clienten kunnen oproepen doen naar de server, die de oproepen analyseert, een antwoord formuleert, en dat antwoord terug naar de klant stuurt. REST maakt gebruik van werkwoorden en zelfstandige naamwoorden om de oproepen ook leesbaar voor mensen te maken. De standaard werkwoorden zijn GET, POST, PUT en DELETE, om data te kunnen verkrijgen, wijzigen of opsturen.

2.3 Website en repository

Voor dit project werd er gebruik gemaakt van Git als versiebeheersysteem, omdat het een gratis, eenvoudige en betrouwbare manier is om de broncode te beheren. Om updates in het project naar de repository te sturen wordt er gebruikt gemaakt van Github voor Windows. Nog een voordeel van deze dienst is dat Github de mogelijkheid biedt om snel en simpel websites te maken voor de bestaande projecten. Zo werd er een kleine website voor dit project opgesteld, die terug te vinden is op: `http://dsverdlo.github.com/AMuRate/`. Hier kan men de open-source broncode bekijken, het logboek of dit document raadplegen en de applicatie (.apk) downloaden voor Android 4.0+ toestellen.

2.4 Naam en logo

De naam van een project geeft meestal een kleine beschrijving van wat de applicatie doet/ waar hij voor dient. AMuRate komt van "Android Music Rating". Omdat het project vooral geïmplementeerd zou worden via een Android emulator en simpelweg "AMR" te kort en onduidelijk zou zijn, werd er gekozen voor AMuRate. Het logo (zoals te zien op de voorpagina) is een compositie van een 5-ster in een bol die iets weg heeft van een "dragon ball" (uit de Japanse animatieserie DragonBall Z). In DBZ bestaan er maar 7 bollen op de hele wereld en wanneer die allemaal verzameld zijn, mag de beheerder een wens doen. Een beetje gelijkaardig aan hoe elke artiest hoopt 5 sterren (/ 7 dragon balls) te verzamelen. In de ster zelf zijn drie gekleurde letters ("AMR") te onderscheiden, naar de naam van de applicatie.

3 Vereisten

Voor de voortgang van dit project besproken wordt, eerst een opsomming van alle vereisten en huidige status.

ID	Side	Beschrijving	Status
1	Client	Titel en artiest kunnen invullen	done
2	Client	Enkel op titel zoeken	done
3	Client	Enkel op artiest opzoeken	done
4	Client	Gedetailleerd lied weergeven	done
5	Client	Gedetailleerd artiest weergeven	done
6	Client	Gedetailleerd album weergeven	done
7	Client	Scores beheren in lokale database	done
8	Client	Geschiedenis beheren in database	done
9	Client	Geschiedenis weergeven op scherm	done
10	Server	Scores opsturen naar externe db	done
11	Server	Scores ophalen uit externe db	done
12	Server	Multithreading op de server	done
13	Both	Ongesyncte scores naar ext. db	done
14	Both	Check of er al een score bestaat	done

Tabel 1: Vereisten van het project met uitleg en implementatiestatus.

Zoals in de projectbeschrijving reeds gezegd werd, bestaat het project uit twee afzonderlijke delen, die verbonden kunnen worden door een internetverbinding. Aan de client-kant is er de applicatie op Android, en aan de server-kant een data-collection server die scores verzameld en geregeld gebruikt voor recommendeer algoritmen. De server-kant is een kleiner pakket dan de client-kant, maar niet minder belangrijk.

text

4 Implementatie

4.1 Package onderverdeling

De broncode van het project is opgedeeld in 3 packages: *gui, objects en services*. Hier een voorlopig overzicht van de packages en hun inhoud:

com.dsverdlo.AMuRate.gui
> AlbumActivity.java
> AnimationView.java

> ArtistActivity.java
> HistoryActivity.java
> MainActivity.java
> SearchArtistActivity.java
> SearchResultsActivity.java
> TrackActivity.java

com.dsverdlo.AMuRate.objects

> Album.java
> Artist.java
> History.java
> HistoryAdapter.java
> Rating.java
> RatingAdapter.java
> Track.java

com.dsverdlo.AMuRate.services

> Client.java
> DatabaseManager.java
> DatabaseSyncer.java
> MyConnection.java

In het *GUI* package zitten alle activities[2] en alle klassen die te maken hebben met het uiterlijk van de applicatie. Elk scherm om iets op weer te geven heeft een eigen activity nodig.
In het *objects* package zitten de objecten/units/simpele java klassen, die abstractie brengen in het project. Bijvoorbeeld wanneer een track opgezocht werd en klaar is met downloaden, zal de JSON omgezet worden in een klasse Track. Zo moet de GUI zich niets aantrekken van hoe de representatie eruit zag toen het van de server kwam.
Package *services* bevat de klassen die instaan voor verschillende diensten die moeten gebeuren in de applicatie. Zo is er de MyConnection klasse, die de oproepen doet en antwoorden download van de Last.fm API. De database beheerder klasse is ook een dienst die verleend wordt aan het project, en bijgevolg ook in dit package zit.

4.2 Database

4.2.1 Lokale Database

De lokale database moet data bijhouden die offline geraadpleegd zou willen worden. Hiermee bedoelen we onder andere geschiedenis van de ingevoerde zoekopdrachten, bekeken liedjes en de laatst gegeven scores. Aangezien deze

[2]Dit is de naam van een scherm in Android

applicatie vooral beroep doet op een actieve internetverbinding, zal er niet meer dan dat opgeslagen worden.
De implementatie van deze database gebeurt in SQLite.

- Ratings table

id	integer	primary key
date	integer	
rating	float	
mbid	text	
title	text	
artist	text	
sync	integer	

Tabel 2: Het model van de Rating tabel in de lokale database.

- Geschiedenis table

id	integer	primary key
key	integer	
date	integer	
artist	text	
title	text	
mbid	text	

Tabel 3: De Geschiedenis tabel in de lokale database.

4.2.2 Externe Database

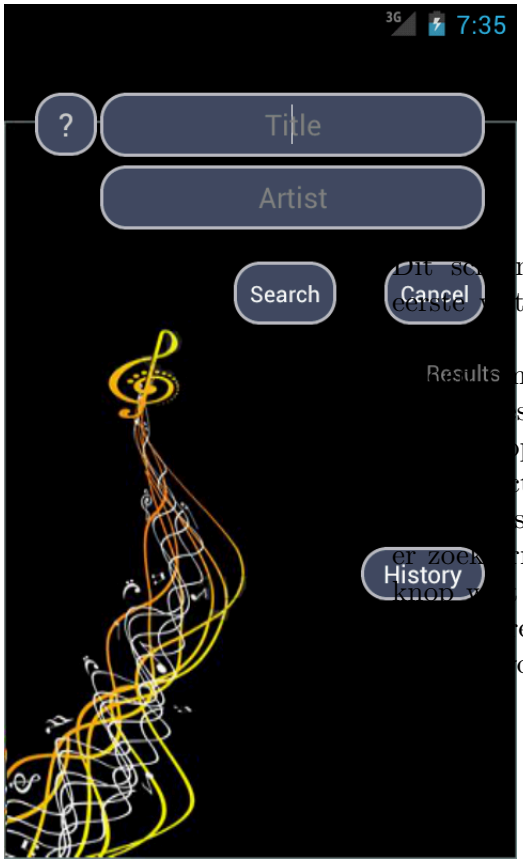
De externe database moet enkel scores van gebruikers verzamelen. Om verschillende gebruikers te onderscheiden zal de er een user ID bijgehouden worden. De tabel zal er dan zo uitzien:

idRatings	integer	primary key
MBID	text	
Artist	text	
Title	text	
Rating	float	
Date	integer	
User	text	

Tabel 4: De Rating tabel layout van de database op de server.

5 GUI

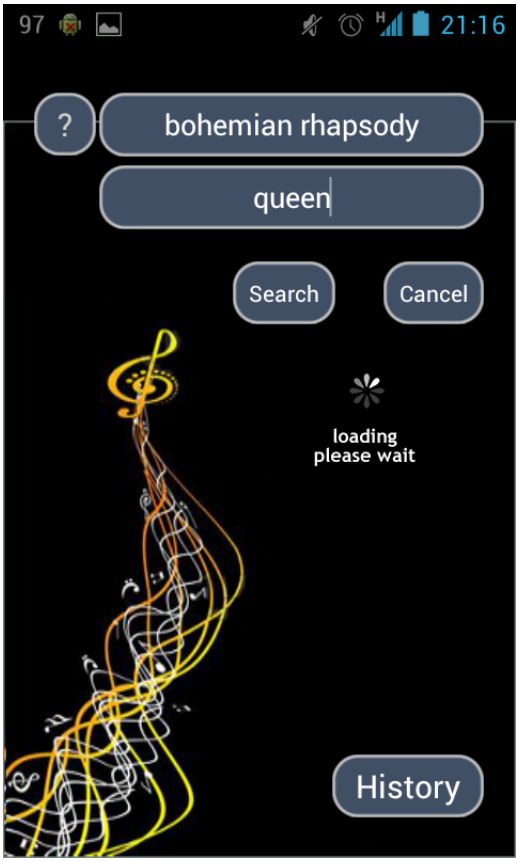
Hieronder volgen enkele schermopnames van de uiteindelijke GUI. Deze opnames zijn van een echte Android telefoon genomen.



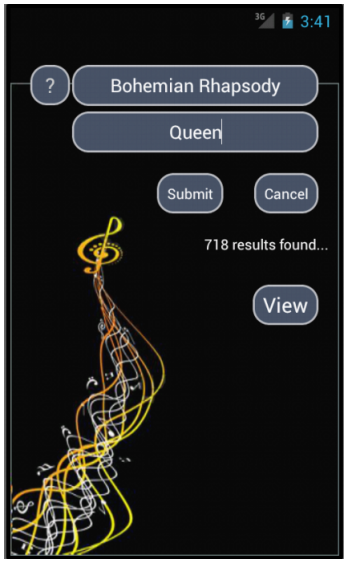
Afbeelding 1: Start scherm.

m is het start scherm en het
de gebruiker ziet wanneer de
gestart wordt. Er zijn twee
zoektermen in te geven (een
st en een voor titel). De linker-
p met het vraagteken erop toont
ties van de applicatie. De sub-
start een zoekopdracht, wanneer
men opgegeven zijn. De cancel
alle ingevulde gegevens (zodat
e zoekopdracht sneller uitgevo-
orden.)

In het naaststaande scherm werden de zoektermen "Bohemian Rhapsody" en "Queen" ingevuld, en op de knop Submit geklikt. De applicatie heeft de zoektermen verwerkt en start het informatie download proces. De text "Loading..." verschijnt op het scherm om de gebruiker te laten weten dat de applicatie ermee bezig is.



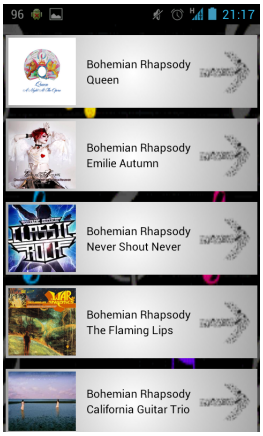
Afbeelding 2: Zoek- en downloadproces.



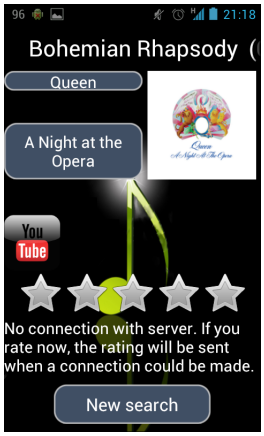
Afbeelding 3: Klaar met downloaden.

Wanneer de applicatie klaar is met het antwoord van de server te downloaden, verandert de text "Loading..." in het resultaat van de oproep, en het aantal gevonden resultaten. Wanneer er een positief aantal resultaten gevonden werd, verschijnt de knop View, waar de gebruiker op kan klikken om de resultaten te bekijken en het gewenste item te selecteren.

De lijst die resultaten weergeeft. Men kan op elke optie klikken om naar een pagina te gaan met meer informatie. Opties die niet beschikken over een afbeelding krijgen de standaard "Niet beschikbaar" afbeelding. De anderen worden asynchroon gedownloaded.



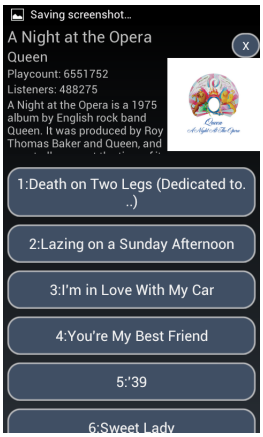
Afbeelding 4: Lijst met zoekresultaten.



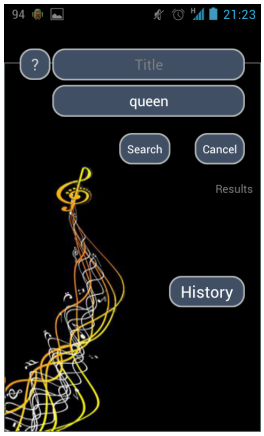
Afbeelding 5: Gedetailleerd scherm voor een lied.

Dit scherm geeft de gedetailleerde pagina van een individueel lied weer. De duratie van het lied wordt getoond, het album van het lied, een afbeelding en het score systeem. Het gemiddelde en het aantal van alle stemmen voor dat lied wordt getoond, en 5 sterren bieden de mogelijkheid aan de gebruiker om een eigen score aan het lied toe te dienen. Helemaal onderaan de knop om een nieuwe zoekopdracht te starten.

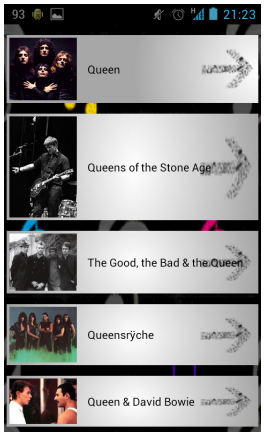
Wanneer er meer album informatie beschikbaar is en de gebruiker klikt op het album, komt hij op dit scherm terecht. Het bevat informatie over een gegeven album zoals artiest, albumafbeelding, playcount, listeners, een scrollbare album informatie tekst en alle liedjes die op dat album staan. De gebruiker kan op elk lied klikken om naar de gedetailleerde liedjes pagina te gaan.



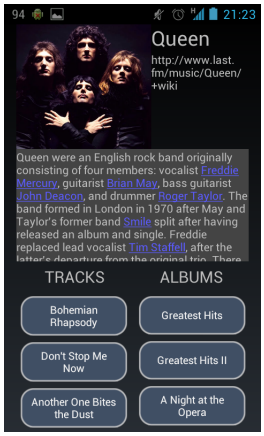
Afbeelding 6: Gedetailleerd scherm voor een album.



Afbeelding 7: We willen hier enkel op artiestnaam zoeken.



Afbeelding 8: Gevonden artiest resultaten voor querie.



Afbeelding 9: Gevonden artiest resultaten voor querie.

6 Technische details

6.1 Data collection

De AMuRate data collection server is een combinatie van een Java server en een MySQL database, die momenteel lokaal op mijn laptop draaien. Die externe database bevat één grote tabel, zoals hier 4.2.2 te zien is. Wanneer een score gegeven zou worden op de applicatie, stuurt die de score via sockets[3] naar de Java server. Wanneer die een verbinding binnen krijgt, wordt er een nieuwe thread aangemaakt (multithreading) die ook een socket verbinding gaat maken met de MySQL database op poort 3306. De database slaagt alle records bij op, tenzij er al een unieke combinatie bestaat (mbid+user).

[3]Standaard communicatiemiddel tussen programmas en computerprocessen

6.2 Unieke records

Gebruikers mogen maar 1 score per lied geven. Records in de database hebben dus 'MBID' en 'User' als primaire sleutel. Dit is deels om SPAM te voorkomen en deels om een eerlijke gemiddeldes te kunnen maken van scores. Wanneer een lied wordt geladen in de applicatie, wordt er in de databases gekeken of de gebruiker al een score heeft gegeven voor de MBID geassocieerd met het gekozen lied. Wanneer de gebruiker al een score heeft gegeven, zal hij/zij geinformeerd worden met de gegeven score. Als er geen verbinding is met de externe database, kunnen we op die moment niet achterhalen of de gebruiker al eens een score heeft gegeven op dat lied. We slagen de score dan even lokaal op tot er terug een verbinding op stand kan komen en updateten dan de gegeven score. Zie sectie Database Syncing.

6.3 Verschillende zoekopdrachten

Op de homepagina zijn er twee inputvelden; 'artist' en 'title'. Gebruikers kunnen kiezen welke velden ze wel en niet willen/kunnen invullen om te vinden wat ze zoeken. Dat levert ons 4 opties;

6.3.1 Geen artiest en geen titel

Aangezien er niets is ingevuld, kan er ook niets opgezocht worden. De gebruiker krijgt een boodschap te zien dat er minstens 1 veld ingevuld moet zijn.

6.3.2 Wel artiest en geen titel

Wanneer enkel de artiestnaam (of groepsnaam) ingevuld wordt, zal er een zoekopdracht gedaan worden naar alle artiesten/groepen met een gelijkaardige naam als de querie. Als er resultaten gevonden zijn, kan de gebruiker kiezen uit een lijst van resultaten. Wanneer een resultaat met een afbeeldingsurl kwam, kunnen we er een afbeelding bij tonen om het selectieproces te vergemakkelijken. Wanneer de gebruiker op een van de opties klikt zal hij doorgewezen worden naar de ArtistActivity pagina van die artiest/groep.

6.3.3 Geen artiest en wel titel

Er wordt gezocht naar alle liedjes met een gelijkaardige titel als die van de querie. De artiesten kunnen varieren en worden niet gefilterd. Als er resultaten gevonden worden, kan de gebruiker op eender welke klikken om naar de TrackActivity te gaan, waar hij/zij een score kan geven.

6.3.4 Wel artiest en wel titel

Wanneer beide velden ingevuld worden wordt er dezelfde aanvraag als hier net boven gestuurd naar de Last.fm API, maar in de querie voegen we de artiestnaam toe. De filtering gebeurt op de Last.fm API zelf, en indien er matches zijn voor beide gegevens, kan de gebruiker erop klikken om naar TrackActivity te gaan.

6.4 Database Syncing

Het kan voorkomen dat de Java server of de MySQL database offline zijn voor een bepaalde onverwacthe periode/reden. Als de gebruiker een score wilt geven en de applicatie ziet dat er geen verbinding tot stand kan komen, wordt de score tijdelijk opgeslagen in de lokale database. Hier komt de kolom 'Sync' handig in voor, want we zetten die sync bit dan op 0 (unsynced. 1=synced). Telkens wanneer de gebruiker verwisselt tussen schermen in de applicatie (redelijk vaak dus), zal de DatabaseSyncer kijken of er unsynced scores in de lokale database zitten. (Dit gebeurt natuurlijk volledig op de achtergrond.) Wanneer er unsynced scores zijn, zal er gekeken of er nu wel een verbinding gemaakt kan worden. Zoniet proberen we laten wel opnieuw, zowel gaan we ze een voor een opsturen naar de externe database. Wanneer ze allemaal zijn aangekomen zetten we hun sync bits op 1 (synced).

6.5 Uitbreidbaarheid

6.6 Installatie

De applicatie staat nog niet in de Android Play Store, maar is wel gratis te downloaden en installeren op de AMuRate website (zie sectie Bronnen, tools en referenties).

- Navigeer met de Android telefoon naar de AMuRate website
- Klik op "Download .apk"
- Navigeer naar de Downloads map en dubbel klik op "AMuRate.apk"
- Op het AMuRate installatiescherm klik rechtsonder op "Install"
- Wanneer "Application installed" tevoorschijn komt kan je klikken op "Done" om het proces te sluiten, of "Open" om de applicatie direct te starten

7 Problemen

7.1 Synchronious tasks

Wanneer data binnengehaald moet worden van de API, mag die operatie niet gebeuren op de main thread van het programma, anders zal de applicatie zeer traag/onbruikbaar zijn elke keer dat er iets gedownloaded moet worden. Hierom werd een klasse "MyConnection" gemaakt, die async[4] data begint binnen te laden terwijl de applicatie vloeiend kan blijven voortdraaien.

7.2 Tracks omzetten van verschillende representaties

Een tegengekomen probleem in dit project, is dat de Last.fm API oproepen (zoals track.getinfo en track.search) antwoorden met verschillende representaties. Om een voorbeeld te geven: hieronder bevinden zich de resultaten van twee oproepen naar methoden voor het lied 'Bohemian Rhapsody' van Queen. Er kan opgemerkt worden dat in het eerste resultaat 'artist' een top level attribuut is van 'track'. In het tweede snippet is 'artist' opnieuw een object met zijn eigen velden. Om zo een object[5] om te zetten in een 'Track' klasse, moet de Track klasse een onderscheid maken in wat voor soort object gebruikt wordt om de klasse te initialiseren. De beste oplossing hiervoor is om per representatie een loadFromX functie te maken die weet hoe X eruit ziet en waar alle informatie staat.

Track.search resultaat:

```
▼<track>
   <name>Bohemian Rhapsody</name>
   <artist>Queen</artist>
   <url>http://www.last.fm/music/Queen/_/Bohemian+Rhapsody</url>
   <streamable fulltrack="0">0</streamable>
   <listeners>1048163</listeners>
   <image size="small">http://userserve-ak.last.fm/serve/34s/84536633.png</image>
   <image size="medium">http://userserve-ak.last.fm/serve/64s/84536633.png</image>
   <image size="large">http://userserve-ak.last.fm/serve/126/84536633.png</image>
  ▼<image size="extralarge">
     http://userserve-ak.last.fm/serve/300x300/84536633.png
   </image>
   <mbid>050349b6-7879-4eb3-81db-5345f51a6c1f</mbid>
 </track>
```

Afbeelding 10: Resultaat van Track.search GET voor Bohemian Rhapsody - Queen.

Track.getinfo resultaat:

[4] Asynchronous betekent dat een process tegelijkertijd naast een ander process kan werken.

[5] In dit document gebruik ik XML notatie omdat deze makkelijker leesbaar is, maar in de broncode wordt er enkel met JSON gewerkt.

```
<track>
  <id>12198</id>
  <name>Bohemian Rhapsody</name>
  <mbid>050349b6-7879-4eb3-81db-5345f51a6c1f</mbid>
  <url>http://www.last.fm/music/Queen/_/Bohemian+Rhapsody</url>
  <duration>356000</duration>
  <streamable fulltrack="0">0</streamable>
  <listeners>1048163</listeners>
  <playcount>6554308</playcount>
  <artist>
    <name>Queen</name>
    <mbid>0383dadf-2a4e-4d10-a46a-e9e041da8eb3</mbid>
    <url>http://www.last.fm/music/Queen</url>
  </artist>
  <album position="11">
    <artist>Queen</artist>
    <title>A Night at the Opera</title>
    <mbid>525a40fc-b3c2-4537-8359-6849467cf79a</mbid>
    <url>
      http://www.last.fm/music/Queen/A+Night+at+the+Opera
    </url>
    <image size="small">http://userserve-ak.last.fm/serve/64s/84536633.png</image>
    <image size="medium">http://userserve-ak.last.fm/serve/126/84536633.png</image>
    <image size="large">
      http://userserve-ak.last.fm/serve/174s/84536633.png
    </image>
    <image size="extralarge">
      http://userserve-ak.last.fm/serve/300x300/84536633.png
    </image>
  </album>
  <toptags>...</toptags>
  <wiki>...</wiki>
</track>
```


Afbeelding 11: Resultaat van Track.getinfo GET voor Bohemian Rhapsody - Queen.

8 Bronnen, tools en referenties

- **Eclipse**
  Programmeerplatform Eclipse (Juno) werd gebruikt voor de implementatie en emulatie.
  `http://www.eclipse.org/`
- **Android Development Toolkit (ADT)**
  Deze plug-in bevat alle benodigdheden om programma's voor Android in Eclipse te bouwen en testen.
  `http://developer.android.com/sdk/index.html`
- **AMuRate homepagina**
  `http://dsverdlo.github.com/AMuRate`
- **Android**
  `http://www.android.com/`
- **Ice Cream Sandwich**
  `http://www.android.com/about/ice-cream-sandwich/`

- **Last.fm**
  `http://www.last.fm/home`
- **Github**
  `https://github.com/`