



Vrije
Universiteit
Brussel

Bachelorproef AMuRate: Project Plan 3e Bachelor Computerwetenschappen

David Sverdlov
dsverdlo@vub.ac.be

31 mei 2013



Inhoudsopgave

1	Inleiding	4
1.1	Doel	4
1.2	Projectbeschrijving	4
1.3	Afkortingen en definities	6
1.3.1	API	6
1.3.2	XML/JSON	6
1.3.3	HTTP	6
1.3.4	GUI	6
1.4	MBID	6
1.5	REST	7
1.6	JDBC	7
2	Achtergrond	7
2.1	Last.fm API	7
2.2	Codebeheer	7
2.3	Naam en logo	8
3	Vereisten	8
4	Implementatie	9
4.1	Android applicatie	9
4.2	Java server	10
4.3	Data flow	10
4.4	Databases	11
4.4.1	Lokale Database	11
4.4.2	Server Database	13
5	Technische beschrijving	14
5.1	Netwerkverbinding	14
5.2	Uniek datamodel	14
5.3	Verschillende zoekopdrachten	15
5.4	Last.fm en hun data	16
5.5	Beveiliging	18
5.6	Database Syncing	18
5.7	Uitbreidbaarheid	19
5.8	Installatie	19
6	Graphical User Interface	19
7	Tegengekomen problemen	28
7.1	Synchronous tasks	28
7.2	Veranderende MBID's	28

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 muziknummers op te zoeken en een beoordeling te kunnen geven. Die scores worden dan opgestuurd naar een data-collection server met een database, waar later recommender 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 wordt geassisteerd door Maarten Deville.

1.2 Projectbeschrijving

Hier gaan we het gebruik van de applicatie in dieper detail bespreken. Voor een visuele voorstelling refereer ik door naar sectie 6: GUI.

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. Wanneer een gebruiker op de zoekknop duwt zonder enige gegevens ingevuld te hebben, zal hij/zij een melding krijgen dat er eerst een zoekterm opgegeven moet worden.

Een hulpknop informeert de gebruiker 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 gedownload kan worden.

Wanneer een gebruiker beschikt over een actieve data verbinding en minstens één zoekterm ingevuld heeft, zal de applicatie een verbinding maken met een grote externe database met muziekgegevens. De database waar we gegevens van halen in dit project is die van Last.fm. Op het scherm zal een indicator tevoorschijn komen dat er een oproep behandeld wordt. Eens aangekomen, zal de Last.fm API de oproep behandelen en een antwoord vormen met de opgevraagde informatie. Eens dat antwoord door de applicatie ontvangen 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.

De gebruiker kan liedjes zoeken door het titelveld in te vullen, maar ook artiesten/groepen opzoeken door enkel iets in het artiestveld te schrijven. Wanneer de gebruiker enkel op artiest zoekt, gaan de gevonden resultaten doorverwijzen naar een gedetailleerde artiestenpagina. Op dit scherm is een kleine biografie te lezen van de artiest, een link naar een wiki-pagina, een afbeelding en een lijst van de top tracks en top albums van die artiest. De gebruiker kan scrollen in beide lijsten en op elk item klikken om naar de respectievelijke gedetailleerde pagina's te gaan.

Het gedetailleerde scherm van een lied geeft meer informatie over het lied in kwestie, zoals de duratie, albumnaam, artiestnaam, een afbeelding van het album¹. Indien er meer album informatie beschikbaar is, zal er een knop zijn waar gebruikers op kunnen klikken om naar de gedetailleerde pagina voor een album te gaan.

Op het track scherm is er een deel van de pagina voorzien voor het scoresysteem. Naast de mogelijkheid om zelf eenmalig een score te geven (tussen 0 en 5 in sprongen van een halve ster), wordt ook de gemiddelde score weergegeven van alle scores voor dat lied (en het aantal scores waar het gemiddelde van berekend werd). Een boodschap informeert de gebruiker of hij al een score heeft gegeven voor dat lied, en wat die score was. Als er geen verbinding gemaakt kan worden met de server, zal de gebruiker hier ook een melding over krijgen.

Het gedetailleerde scherm voor een album toont informatie over het album en alle liedjes ervan. De gebruiker kan op elk lied klikken om naar het scherm te navigeren met meer informatie om een score te geven. Met een X-knop kan er terug gegaan worden naar het hoofdmenu.

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. In de externe database worden de scores opgeslagen samen met het tijdstip van het geven van de score, de mbid, de titel, de artiest en de gebruiker. De gebruiker slagen we op via het unieke Android device nummer van het GSM toestel. Een gebruiker mag maar 1 score hebben per MBID in de databank.

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

Vanaf het beginscherm kan de gebruiker ook op de knop 'History' klikken om zijn geschiedenis te bekijken. Op het geschiedenis scherm zijn er vanboven 3 tabs te zien: 'Search', 'Tracks' en 'Ratings'. De tabs staan in een horizontale scrollview (er kunnen dus makkelijk bijgezet worden). De geschiedenis wordt dus onderverdeeld in zoekgeschiedenis (opgegeven zoektermen), trackgeschiedenis (bekeken tracks) en ratinggeschiedenis die alle gegevens scores bijhoudt. Elke onderverdeling heeft de optie om zijn geschiedenis te wissen.

1.3 Afkortingen en definities

1.3.1 API

API staat voor Application Programming Interface en zorgt voor de communicatie tussen programma's, 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 door te kunnen sturen.)

JSON is een afkorting van JavaScript Object Notation en is een alternatieve simpele manier om objecten voor te stellen als platte tekst.

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,...

1.5 REST

REST staat voor Representational State Transfer en is een type van architectuur voor gedistribueerde systemen (zoals het World Wide Web), waarin een duidelijke onderscheiding wordt gemaakt tussen client en server. Clients kunnen oproepen doen naar de server, die de oproepen analyseert, een antwoord formuleert, en dat antwoord terug naar de client stuurt. De standaard sleutelwoorden voor oproepen in een RESTful architectuur zijn GET, POST, PUT en DELETE. Deze dienen om data te kunnen verkrijgen, aanmaken, wijzigen of opsturen.

1.6 JDBC

JDBC staat voor Java Database Connectivity is een API van Oracle die abstractie brengt in het verbinden en beheren van databases. De API maakt het ook eenvoudig om informatie uit databases te filteren en de resultaten te gebruiken voor eender welke operaties.

2 Achtergrond

2.1 Last.fm API

Om informatie (over muziek in dit geval) op te kunnen zoeken, moet er gebruik gemaakt worden van een online database met een openbare API. *Last.fm* is een muziek recommendation service met een enorme online muziek database, die geregistreerde gebruikers een gratis API aanbiedt. Zo mogen programmeurs mobiele/desktop programma's of web services bouwen, die beroep doen op hun data. De werking en gebruik hiervan wordt in detail besproken in sectie 5.4.

2.2 Codebeheer

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. (Zie sectie 5.8:Installatie voor meer informatie over het installeren)

2.3 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 'dragonball' (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

Hieronder volgt een tabel die alle vereisten opsomt, met hun geïmplementeerde status. Deze tabel werd in de loop van het project meermaals bijgewerkt met verschillende statuskleuren, naargelang de level van afwerking per vereiste. Momenteel zijn ze allemaal groen omdat elke onderstaande functionaliteit geïmplementeerd is.

ID	Categorie	Beschrijving	Status
1	Client	Titel en artiest kunnen invullen	Afgewerkt
2	Client	Enkel op titel zoeken	Afgewerkt
3	Client	Enkel op artiest opzoeken	Afgewerkt
4	Client	Gedetailleerd lied weergeven	Afgewerkt
5	Client	Gedetailleerd artiest weergeven	Afgewerkt
6	Client	Gedetailleerd album weergeven	Afgewerkt
7	Client	Scores beheren in lokale database	Afgewerkt
8	Client	Geschiedenis beheren in database	Afgewerkt
9	Client	Geschiedenis weergeven op scherm	Afgewerkt
10	Client	Scores opsturen naar externe db	Afgewerkt
11	Server	Scores ontvangen op externe db	Afgewerkt
12	Server	Scores ophalen uit externe db	Afgewerkt
13	Server	Met meerdere klanten tegelijk verbinden	Afgewerkt
14	Beide	Ongesyncte scores naar ext. db	Afgewerkt
15	Beide	Kijk na of gebruiker al een score heeft gegeven	Afgewerkt

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 internet-verbinding. 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 wordt voor recommendeer algoritmen. De server-kant is een kleiner pakket dan de client-kant, maar niet minder belangrijk.

4 Implementatie

4.1 Android applicatie

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
- > BlankActivity.java
- > HistoryActivity.java
- > MainActivity.java
- > SearchArtistActivity.java
- > SearchResultsActivity.java
- > TrackActivity.java

com.dsverdlo.AMuRate.objects

- > Album.java
- > AMuRate.java
- > Artist.java
- > History.java
- > Rating.java
- > Track.java

com.dsverdlo.AMuRate.services

- > DatabaseSyncer.java
- > DownloadImageTask.java
- > DownloadLastFM.java
- > InternalDatabaseHistoryAdapter.java
- > InternalDatabaseManager.java
- > InternalDatabaseRatingAdapter.java
- > ServerConnect.java

In het *GUI* package zitten alle activiteiten² 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. Het is ook hier dat de globale applicatie 'AMuRate.java' zit.

Package *services* bevat de klassen die instaan voor verschillende diensten die moeten gebeuren in de applicatie. Zo is er de *HttpConnect* klasse, die de oproepen doet en antwoorden download van de Last.fm API en afbeeldingen per http binnenhaalt. De *InternalDatabaseManager* klasse is ook een dienst die verleend wordt aan de interne database.

4.2 Java server

De server code bestaat uit drie bestanden in een package. De 'Server.java' is het bestand dat eenmalig opgestart moet worden, om de server te doen draaien. De server draait standaard op poort 2005. Een ander poortnummer kan meegegeven worden door de server in commandline op te starten en een nieuw poortnummer mee te geven.

Wanneer een client probeert te verbinden met de server, zal de server een aparte task opstarten hiervoor. Op die manier kunnen er meerdere clients tegelijk hun informatie van de server krijgen zonder dat ze op elkaar moeten wachten. Zo een aparte task is een instantie van 'ServerConnectionTask.java'. Die task kan dan aan de 'ExternalDatabaseConnection.java' vragen om dingen in- of uit de database te halen. Alle SQL is dus behouden in één klasse.

com.dsverdlo.AMuRate.server

> ExternalDatabaseConnection.java.java

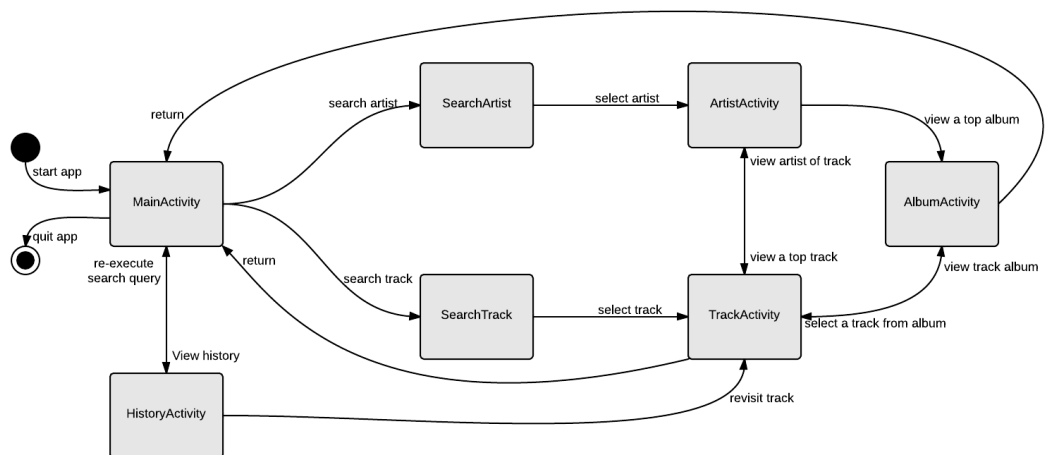
> Server.java.java

> ServerConnectionTask.java.java

4.3 Data flow

Om de overgangen tussen alle schermen overzichtelijk te illustreren, werd er een Data Flow Diagram gemaakt. Zie onderstaande afbeelding:

²Dit is de naam van een scherm in Android



Afbeelding 1: De data flow tussen alle activies.

Op de bovenstaande diagram zien we twee pijlen vertrekken vanuit MainActivity naar de SearchActivities. Wanneer enkel een artiest ingevuld wordt, zal de applicatie naar de ArtistSearchActivity gaan. Vanaf dat er ook een titel ingevuld wordt, gaat de applicatie naar SearchResultsActivity. Dit is visueel te zien in hoofdstuk GUI 6 en besproken in sectie ??.

4.4 Databases

Er zijn twee databases in dit project. Eén lokaal, op elk toestel met een AMuRate applicatie, en één grote centrale database op een server. De lokale databases houden enkel data bij van hun gebruiken, en de externe verzamelt de scores van iedereen. We zullen nu beide databases bespreken.

4.4.1 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 applicatie vooral beroep doet op een actieve internetverbinding, zal er niet meer dan dat opgeslagen worden.

De lokale database bevat dus twee tabellen en de operaties op deze tabellen gebeuren d.m.v. adapters. Die adapters maken de brug tussen procedures als 'read-X-from-database' en SQLite statements (zoals "SELECT X FROM tablename"). Er bestaat 1 adapter per tabel.

- Ratings table

Naam	Type	Sleutel
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.

De laatste kolom in de tabel is een 'sync bit'. Dat betekent dat hij enkel de waarde 0 of 1 kan hebben. Wanneer hij waarde 0 nul heeft, betekent het dat de score nog niet verzonden is naar de externe database. Dit kan voorvallen als er slechte verbinding is, of de server even uit zou staan bijvoorbeeld. De DatabaseSyncer zal de score opsturen eens er terug verbinding is en de sync bit op waarde 1 zetten, zodat we weten dat hij opgestuurd is.

- Geschiedenis table

Naam	Type	Sleutel
id	integer	primary key
key	integer	
date	integer	
artist	text	
title	text	
mbid	text	

Tabel 3: De Geschiedenis tabel in de lokale database.

De 'key' in kolom 2 is een lokale identificeerder die onderscheid gaat maken tussen een zoek-geschiedenis-record en een track-viewed-geschiedenis. Deze twee soorten records staan samen in één tabel omdat ze, op 1 na, alle velden gemeenschappelijk hebben. Bij track-viewed-geschiedenis gaan we de MBID mee opslagen om sneller de informatie terug te kunnen downloaden. Voor de zoek-geschiedenis-records is dit veld leeg in de database.

4.4.2 Server Database

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

Naam	Type	Sleutel
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.

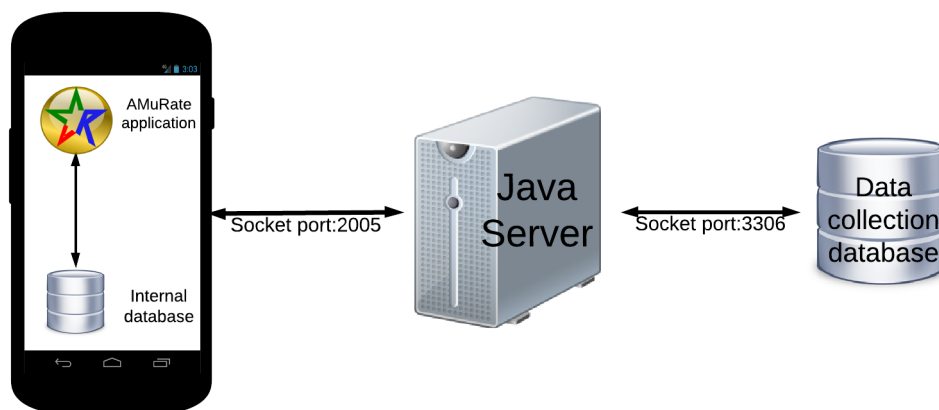
De synchronisatie van deze tabel met de lokale ratings database, gebeurt via een DatabaseSyncer in de applicatie. Die klasse zal, wanneer er ongesyncte scores lokaal staan, scores naar de server sturen zodat ze gelijk lopen. Dit wordt verder in detail besproken in sectie 5.6.

5 Technische beschrijving

5.1 Netwerkverbinding

De AMuRate data collection server is een combinatie van een Java server en een MySQL database, die momenteel lokaal op mijn laptop draait. Die externe database bevat één grote tabel, zoals in deze tabel:4.4.2 te zien is. Wanneer een score gegeven zou worden op de applicatie, probeert de applicatie een socket³ te openen met de server. Als de server draait, is er een serversocket die constant luistert naar nieuwe verbindingen. Als die serversocket een oproep van de applicatie opmerkt, zal die een nieuwe Thread aanmaken voor die verbinding vast te maken. De applicatie zet de score op de out-stream van zijn socket, waarna de serversocket de score van zijn in-stream afhaalt. De score is nu op de server.

De server geeft de score dan mee aan zijn ExternalDatabaseConnection klasse, die de nodige SQL instructies zal opstellen om de score in de database te kunnen voegen. Om de SQL op te sturen, maken we gebruik van een JDBC. Dit is een Java API die het makkelijk maakt om Java programma's te verbinden met databases, data te query-en en resultaten van queries te behandelen.



Afbeelding 2: Overzicht van de score verbindingen.

5.2 Uniek datamodel

Gebruikers mogen maar één 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

³Standaard communicatiemiddel tussen programma's en computerprocessen

hij/zij geïnformeerd 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 5.6 Database Syncing.

5.3 Verschillende zoekopdrachten

Op het hoofdscherm zijn er twee inputvelden; 'artiest' en 'titel'. Gebruikers kunnen kiezen welke velden ze wel en niet willen invullen om te vinden wat ze zoeken. Dat levert ons 4 opties;

- **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 één veld ingevuld moet zijn.
- **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 query. 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.
- **Geen artiest en wel titel**
Er wordt gezocht naar alle liedjes met een gelijkaardige titel als die van de query. De artiesten kunnen variëren 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.
- **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 query 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.

5.4 Last.fm en hun data

Om data uit de Last.fm database te halen moet er een call (oproep) gestuurd worden via HTTP. Deze gebeurt door een GET oproep naar de Last.fm server te sturen. Die server gaat op zijn beurt antwoorden met een XML object (of JSON op aanvraag). De architectuur van client < – > server die hier gebruikt wordt heet 'REST(ful)'.

De url van de aanvraag bestaat uit 3 delen; de basis van de API, de methode met zijn parameters en de api-key. De basis is altijd 'ws.audioscrobbler.com/2.0/' en de methode is van de vorm '?method=category.function'. De parameters in de vorm van '& param1=value1& param2=value2'. Een api-key is een string van 32 karakters en gratis te verkrijgen door te registreren bij Last.fm. De Last.fm API biedt tientallen methodes aan, zoals bijvoorbeeld artist.search, artist.getTopAlbums, artist.getTopTracks, track.getInfo, track.search, track.getTags, album.search, album.getBuyLinks, Voor een overzicht van alle methoden verwijs ik door naar hun website: <http://www.last.fm/api>.

De methoden die in dit project gebruikt worden zijn

Activity	Methoden
MainActivity	track.search
"	artist.search
SearchResultsActivity	track.getInfo
SearchArtistActivity	artist.getInfo
TrackActivity	artist.getInfo
"	album.getInfo
ArtistActivity	track.getTopTracks
"	album.getTopAlbum
AlbumActivity	track.getInfo
HistoryActivity	track.getInfo

Tabel 5: Een overzicht van alle oproepen in de applicatie.

Een tegengekomen probleem in dit project is het feit 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 met verschillende 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'n object⁴ 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 func-

⁴In dit document gebruiken XML notatie omdat deze makkelijker leesbaar is dan JSON, maar in de broncode wordt er enkel met JSON gewerkt.

tie te maken die weet hoe X eruit ziet en waar alle informatie staat. Als we net een oproep hebben gedaan met methode 'track.search', zullen we de Track.loadFromSearch oproepen. Voor een oproep met 'track.getInfo', roepen we Track.loadFromInfo op.

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 3: Resultaat van track.search GET oproep voor Bohemian Rhapsody - Queen.

Track.getinfo resultaat:

```

▼<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 4: Resultaat van `track.getInfo` GET oproep voor *Bohemian Rhapsody* - Queen.

5.5 Beveiliging

De beide databases zijn beschermd tegen SQL-injecties door het gebruik van preparedStatements, in plaats van String formatting. Wanneer er enkel validatie op de user input wordt gedaan, zou het nog mogelijk zijn dat de gebruiker een extra SQL-instructie meestuurt. Een preparedStatement compileert een SQL-instructie op voorhand en laat toe om erna nog parameters te binden. De preparedStatement heeft dan zijn executiepad volledig uitgewerkt, wat het onmogelijk maakt om een SQL-injectie in te voegen die, bijvoorbeeld, tabellen dropt.

5.6 Database Syncing

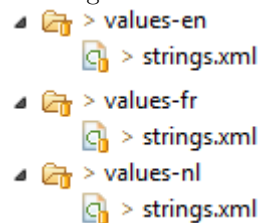
Het kan voorkomen dat de Java server of de MySQL database offline zijn voor een bepaalde onverwachte periode/reden. Als de gebruiker een score wilt geven en de applicatie vaststelt dat er geen verbinding tot stand kan komen, wordt de score tijdelijk opgeslagen in de lokale database. Hier komt de kolom '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, zal de DatabaseSyncer kijken of er unsynced scores in de lokale database zitten. Dit gebeurt volledig op de achtergrond en is onzichtbaar voor de gebruiker. Wanneer er unsynced scores zijn, zal er gekeken of er nu wel een verbinding gemaakt kan worden. Als er een verbinding gemaakt kan worden, gaan we ze een voor een opsturen naar de server. Wanneer ze allemaal zijn aangekomen zetten we hun sync bits op 1 (synced).

5.7 Uitbreidbaarheid

De uitbreidbaarheid van het systeem is zeer eenvoudig, in de veronderstelling dat de gebruiker enige voorkennis van Android development bezit. Om nieuwe functionaliteiten toe te voegen moeten er gewoon activiteiten bijgemaakt worden, en toegevoegd worden in het Android Manifest.

De taal van de applicatie is standaard in het Engels, maar via de menu knop kan de gebruiker kiezen om de taal te veranderen. De applicatie ondersteunt momenteel 3 talen: Nederlands, Engels en Frans. Android ondersteunt een simpel systeem om makkelijk talen toe te voegen aan een applicatie. Alle tekst staat gedefinieerd in een XML bestand genaamd 'strings.xml'. Om een taal toe te voegen volstaat het om alle strings in dat XML bestand te vertalen in een folder met de naam 'values-' en dan de afkorting van de taal. Daarna moet de nieuwe optie aan het taal-menu toegevoegd worden. Die is gedefiniëerd in de BlankActivity die door alle activiteiten wordt ge-extend.



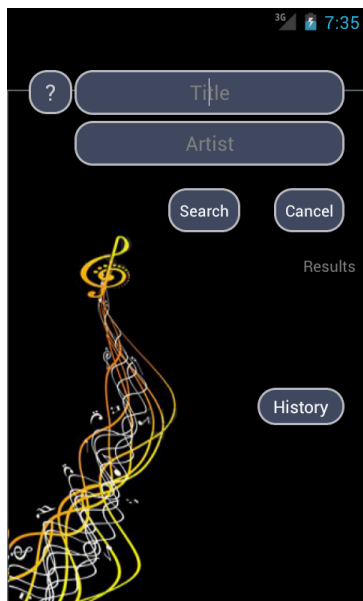
5.8 Installatie

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

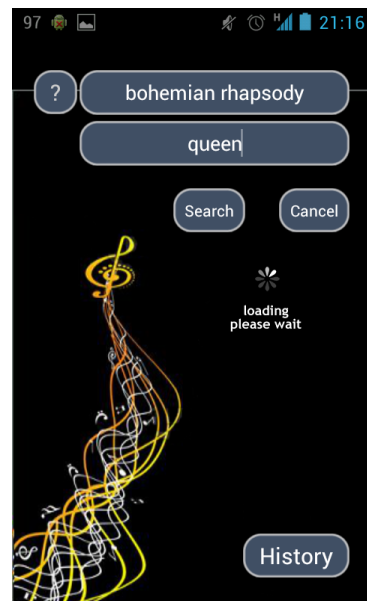
6 Graphical User Interface

Hieronder volgen enkele schermopnames van de uiteindelijke GUI met onderstaande uitleg.



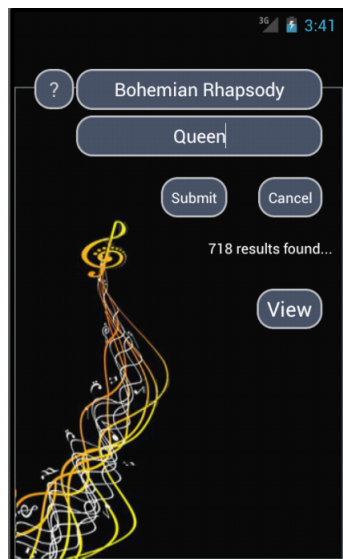
Afbeelding 5: Start scherm.

Dit scherm is het start scherm en het eerste wat de gebruiker ziet wanneer de applicatie gestart wordt. Er zijn twee velden om zoektermen in te geven (een voor artiest en een voor titel). De linkerboven knop met het vraagteken erop toont de instructies van de applicatie. De submit knop start een zoekopdracht, wanneer er zoektermen opgegeven zijn. De cancel knop wist alle ingevulde gegevens.



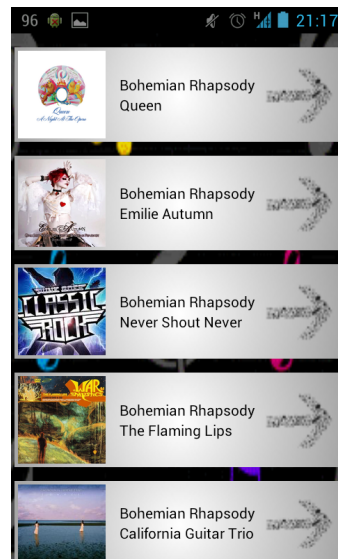
Afbeelding 6: Zoek- en downloadproces.

In het bovenstaande scherm werden de zoektermen "Bohemian Rhapsody" en "Queen" ingevuld, en op de zoekknop geklikt. De applicatie heeft de zoektermen verwerkt en start het informatie download proces. Een spinner met "Loading..." verschijnt op het scherm om de gebruiker te laten weten dat de applicatie aan het downloaden bezig is.



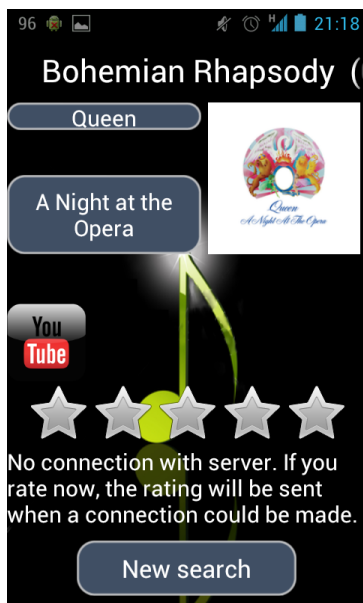
Afbeelding 7: Klaar met downloaden.

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.



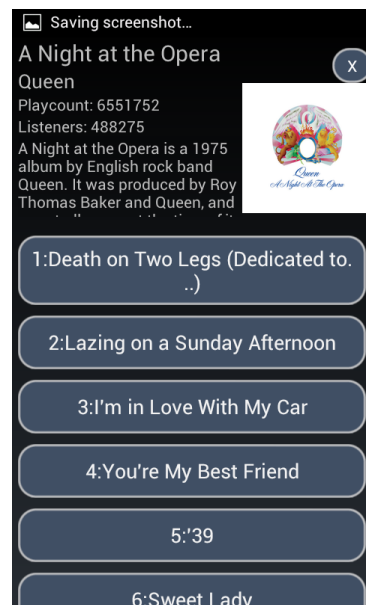
Afbeelding 8: Lijst met alle zoekresultaten.

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



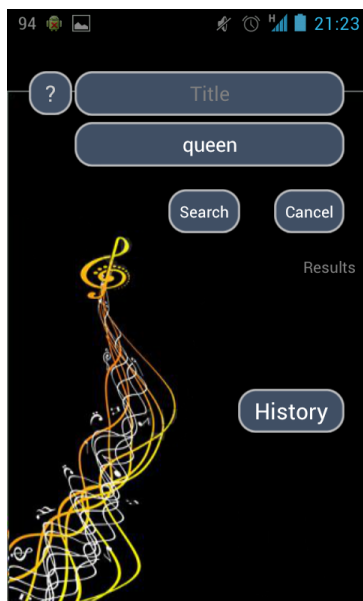
Afbeelding 9: Gedetailleerd scherm van een lied.

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 scoresysteem. 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 men op de youtube knop klikt zal de lokale youtube app starten. Wanneer de tekst en duratie te lang wordt voor het scherm, zal de tekst als marquee bewegen.



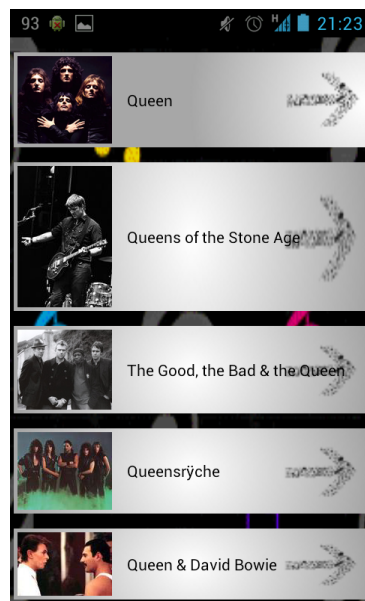
Afbeelding 10: Gedetailleerd scherm voor een album.

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.



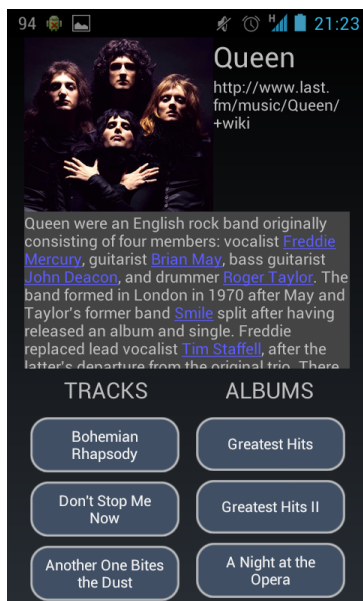
Afbeelding 11: We willen hier enkel op artiestnaam zoeken.

Als we enkel het artiest veld invullen, zal er een oproep gedaan worden naar de Last.fm API om artiesten te zoeken, niet tracks. Zoals normaal verschijnt de knop 'View' als er resultaten gevonden zijn, of verschijnt er een boodschap dat er geen matches waren voor de opgegeven zoekstring.



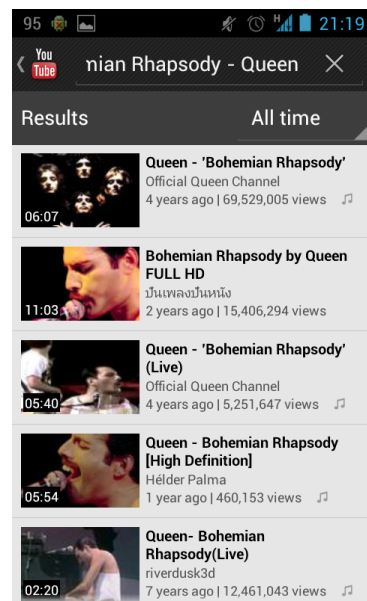
Afbeelding 12: Gevonden artiest resultaten voor query.

De ArtistSearchActivity is zeer gelijkaardig aan de zoek tracks pagina. We tonen een afbeelding om te helpen kiezen welke artiest de gebruiker wou bezoeken.



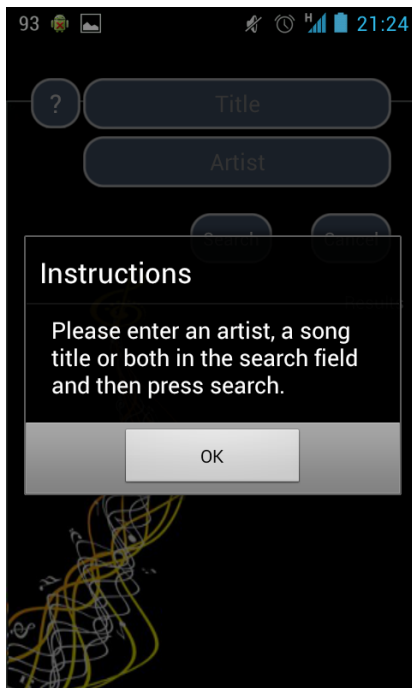
Afbeelding 13: Gedetailleerd scherm van een artiest.

Wanneer een gebruiker meer informatie wilt over een artiest of groep, komt hij/zij op dit scherm terecht. Er is een link naar de wiki (als die bestaat), een scrollbare biografie, en een lijst van de artiest's top tracks en top albums. Men kan in beide lijsten scrollen en op elke item klikken om naar de gedetailleerde track/album pagina te gaan.



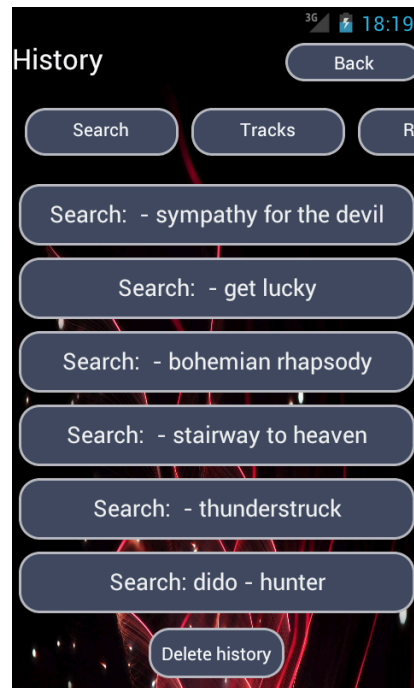
Afbeelding 14: Link naar de youtube app.

Wanneer iemand in de TrackActivity op de youtube knop klikt, zal de lokale youtube app opgeroepen worden met als zoekterm de artiest en titel van het lied, zodat de gebruiker het liedje nog eens kan afspelen voordat hij/zij een score wilt bedelen.



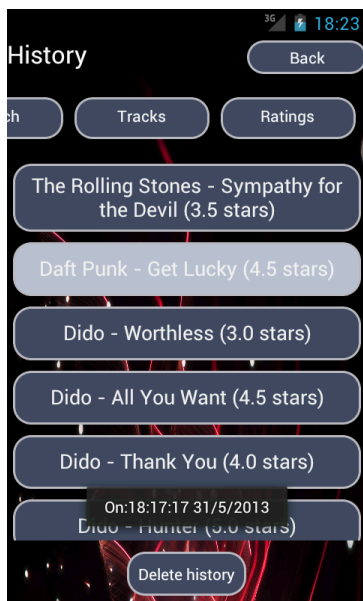
Afbeelding 15: Instructie pop up.

Als men op het vraagteken in de linksboven hoek klikt, komt deze pop-up tevoorschijn. Het geeft de instructies die nodig zijn om een eerste zoekopdracht uit te voeren.



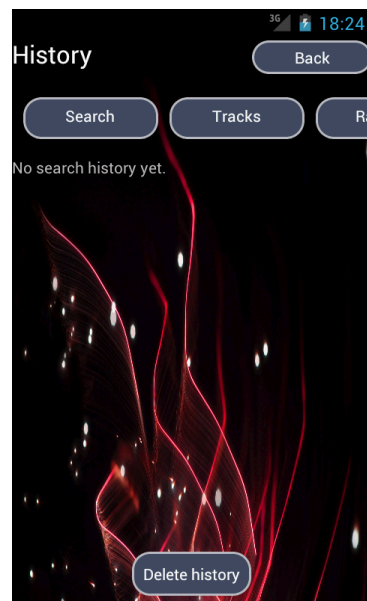
Afbeelding 16: Geschiedenis scherm.

Dit is het geschiedenis scherm. Vanboven is er een knop om terug te keren naar het hoofdscherm en er vlak onder enkele tabs. De tabs zitten in een horizontale scrollbar en kunnen dus verschoven worden. Wanneer de gebruiker op een van deze tabs klikt, worden de respectievelijke geschiedenisrecords in het midden van het scherm ingeladen. Met de onderste knop kan de gebruiker deze records wissen. Op bovenstaande afbeelding zien we de 'search' geschiedenis. Door op een item te klikken in de lijst, worden die zoektermen terug ingevuld in de zoekvelden van het hoofdscherm.



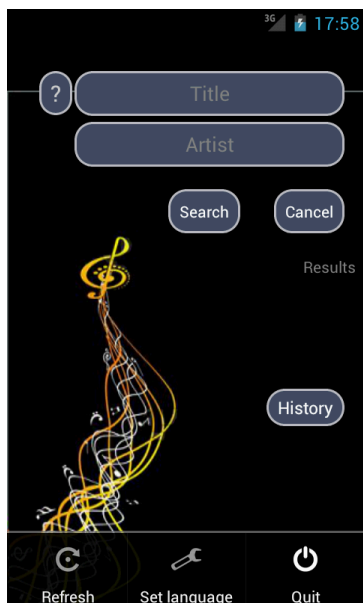
Afbeelding 17: Geschiedenis ratings.

Deze tab geeft een overzicht van alle gegeven scores. We tonen de titel, artiest en aantal gegeven sterren. Door op een score te klikken verschijnt er een Toast die zegt wanneer die rating precies gegeven werd.

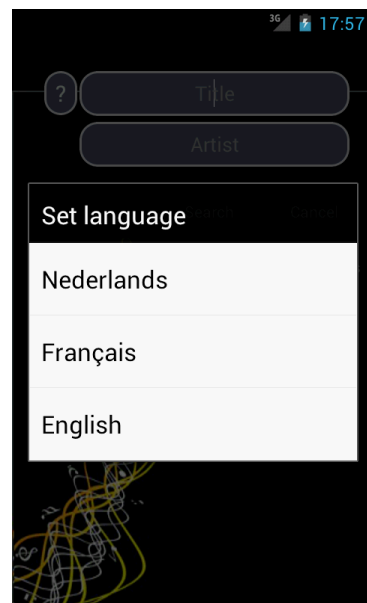


Afbeelding 18: Geschiedenis gewist.

Delete history verwijdt de geschiedenis records van de huidig geopende tab.



Afbeelding 19: AMuRate menu van elk scherm.



Afbeelding 20: Taal veranderen menu.

Op bovenstaande afbeelding zien we het menu dat tevoorschijn komt wanneer de gebruiker op de 'menu' knop van zijn telefoontoestel duwt. Er is een optie die het huidige scherm refresht, een optie om de taal te veranderen (zie volgende afbeelding) en een knop om de applicatie volledig af te sluiten.

Wanneer men klikt op de menu knop om de taal te veranderen, zal hij een pop-up krijgen met de beschikbare taal-opties. Een gebruiker kan kiezen om de taal in te stellen die al actief is, dit zal niets veranderen en heeft dan dezelfde werking als een refresh.

7 Tegengekomen problemen

7.1 Synchronous 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 gedownload moet worden. Hierom werd een klasse "MyConnection" gemaakt, die `async`⁵ data begint binnen te laden terwijl de applicatie vloeiend kan blijven voortdraaien.

7.2 Veranderende MBID's

In de Last.fm API hebben alle liedjes, artiesten en albums een geassocieerde MBID. Tijdens het programmeren door, viel het op dat ik scores kon geven op liedjes die normaal gezien al een score hadden gekregen van mij. Nu blijkt dat de MBID's op willekeurige tijdstippen en intervallen veranderen. In onderstaande afbeelding is een query gezien, die alle MBID's weergeeft van één lied.

idRat	MBID	Artist	Title	Date
53	2424dc3a-b451-4415-a5bf-93abfab320a1	Dido	Hunter	2013-05-11 18:26:30
29	8b96d259-a4a2-42d9-bb9a-c009eeea1e70	Dido	Hunter	2013-05-05 15:43:13
123	e583857b-de4e-4745-9665-25e6772088e3	Dido	Hunter	2013-05-31 17:03:52

Afbeelding 21: Verschillende MBID's voor één track.

Over deze veranderingen is geen documentatie te vinden om de Last.fm website. Het probleem van deze veranderingen is pas in mei opgemerkt, en is dus nog niet opgelost. Een oplossing voor dit probleem zou zijn om liedjes te identificeren via hun track en artiest. (Eventueel ook duratie.)

⁵Asynchronous betekent dat een process tegelijkertijd naast een ander process kan werken.

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>
- **MySQL**
Op MySQL workbench werd de data collection aangemaakt en kon de luisterende socket aangemaakt worden.
<http://www.mysql.com/>
- **LucidChart**
Op deze website werden de diagrammen en andere afbeeldingen in dit document gegenereerd.
lucidchart.com
- **AMuRate homepage**
<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/>