

AWESOMO 4000

Een revolutie in de muziekindustrie



Groep 1:

Liesbeth van Alebeek – 4166094

Bjorn van der Laan – 4151348

Rogier Slag – 1507761

INHOUDSOPGAVE

Inleiding	3
Doel van systeem	4
Doelgroep	Error! Bookmark not defined.
Schets van de oplossing en functionaliteit	5
Registreren en inloggen	5
Muziek luisteren en informatie	5
Databronnen	5
Recommendationsysteem	5
Userinterface	5
Implementatie keuzes	7
PHP	7
Bootstrap	7
Youtube	7
Facebook	7
EchoNest	8
Spotify	8
Database	8
Uitvoering	10
Database	12
Planning	12
Eerste sprint	13
Tweede sprint	14
Derde sprint	15
Evaluatie	16
Testplan	16
Testresultaten	16
Conclusie/discussie	18
Literatuurlijst	19
Appendixes	20
Sparql vs EchoNest	20

INLEIDING

De mens luistert al duizenden jaren naar muziek. Muziek is een onderdeel geworden van culturen en beschavingen. En recent heeft muziek weer een nieuwe sprong gemaakt.

Door de technologische vooruitgang als radio's en LP's werd muziek iets waar men naar kon luisteren op momenten dat men dat zelf wilde, onafhankelijk van de beschikbaarheid van zangers en orkesten. Inmiddels worden radio's en LP's minder gebruikt als primaire bron van muziek, maar hebben internetsdiensten en MP3-bestanden voor veel mensen deze rol overgenomen.

Steeds meer mensen hebben dan ook imposante collecties in hun kast of op hun harde schijf van hun PC staan. Maar met zo'n grote mate van aanbod kan het lastig zijn om specifiek te ontdekken wat je tegenwoordig nieuwe leuke muziek kan vinden. Tot in zekere mate kunnen toplijsten hierbij helpen, of hoor je wel eens suggesties van vrienden.

Echter kan men niet verwachten dat deze hulpmiddelen daadwerkelijk een significante basis hebben om zoveel muziek te kennen. Met de huidige stand van de techniek kan men dit proces automatiseren. Waar het van mensen niet verwacht kan worden om 17 miljoen nummers te kunnen analyseren, is dit voor computers geen enkel probleem. Op basis hiervan is het mogelijk om een systeem te bouwen wat dit zonder problemen uitvoert.

Exact dat was het doel van dit project. De AWESOMO 4000 is er op gericht om, gebaseerd op de luistervoorkeuren van mensen, nieuwe muziek aan te raden die hiermee overeenkomt. In dit verslag leest u hoe dit systeem tot stand is gekomen.

DOEL EN DOELGROEP VAN SYSTEEM

Een website met een zoekbalkje voor Youtube-filmpjes, een muziekspeler met extra informatie en luistersuggesties. Hier kan een link naar een muziekfilmpje ingevoerd worden en vervolgens begint deze af te spelen. Ondertussen wordt er vanuit allerlei bronnen informatie opgehaald en gecombineerd. Het scherm vult zich met informatie over de artiest, concert-data, mogelijkheden om het nummer te kopen, luistersuggesties en nog meer.

Onder het nummer verschijnt er een lijst met suggesties voor andere muziek. Deze suggesties zijn gebaseerd op het laatst geluisterde nummer, de verdere luistergeschiedenis en hoeveel nummers op elkaar lijken. Door de recommendations deels te baseren op het laatst geluisterde nummer wordt er muziek gesuggereerd die past bij de huidige voorkeur van de gebruiker. Wanneer de gebruiker ervoor kiest om een suggestie te beluisteren, verschijnt ook de mogelijkheid om de kwaliteit van de suggestie te beoordelen. Deze informatie wordt vervolgens weer gebruikt om toekomstige suggesties te verbeteren.

In ons project ligt de focus vooral op jongere mensen. Bij deze doelgroep is het belangrijk dat zij zelf niet al te veel moeite hoeven te doen. Jongeren zijn snel geneigd een website te verlaten als zij teveel stappen moeten maken om de informatie te krijgen die zij zoeken. Daarom hebben we gezorgd voor een simpel interface waarbij ze ons alleen de youtube link hoeven te geven. Zo kunnen ze elk liedje dat hun vrienden hun linken in een keer luisteren, terwijl er ook achtergrond informatie en luistersuggesties weergegeven worden. Een filmpje dat iemand linkt kan direct ingevoerd worden, en de gebruiker hoeft niet eerst op te zoeken welk nummer het is, of wie de artiest is.

SCHETS VAN DE OPLOSSING EN FUNCTIONALITEIT

Plaatje workflow

REGISTREREN EN INLOGGEN

Wanneer een gebruiker de site bezoekt komt deze als eerste terecht op een inlog-pagina. Nieuwe gebruikers kunnen hier naar de registratie-pagina gaan. Wanneer de gebruiker is ingelogd op facebook worden een aantal basisgegevens automatisch ingevuld. Anders moet dit handmatig gedaan worden. Na het registeren kan er voortaan ingelogd worden.

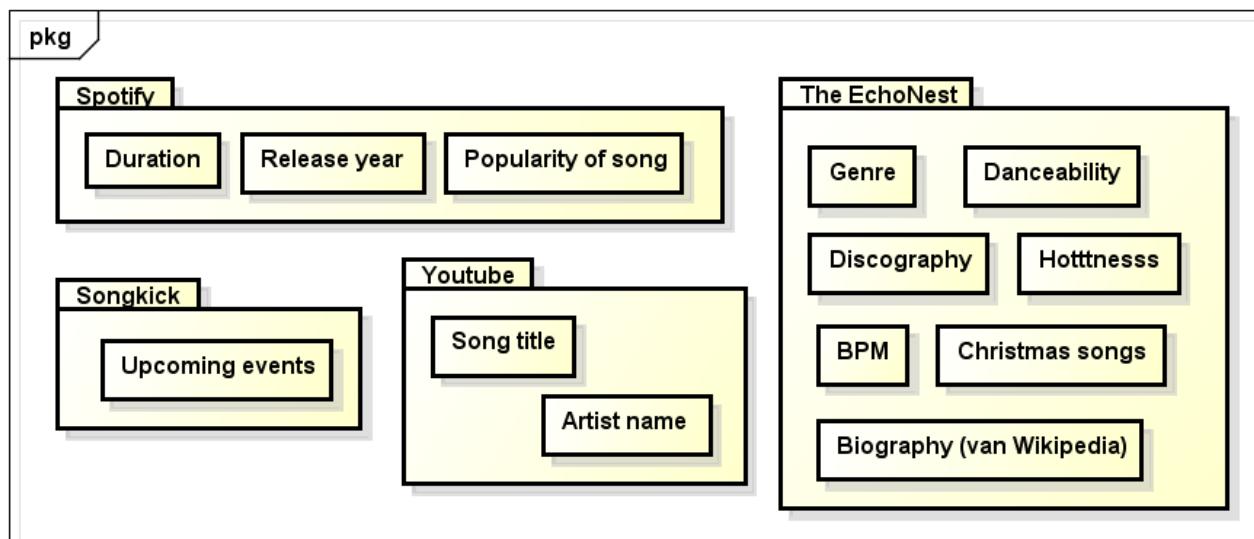
MUZIEK LUISTEREN EN INFORMATIE

Wanneer er ingelogd is, zijn er twee mogelijkheden voor het beluisteren van muziek. Er kan een linkje van een Youtubenummer ingevoerd worden, deze zal dan afgespeeld worden. Op de rest van de pagina verschijnt informatie over de artiest, het nummer, en het album waar het vanaf komt.

Als alternatief wordt er op het scherm een lijst van suggesties weergegeven. Hieruit kan er een nummer gekozen worden om te luisteren. Ook hiervan wordt de informatie weergegeven. Ook kan de gebruiker aangeven of het een goede suggestie was door middel van een thumbs up of down.

DATABRONNEN

Wanneer een nummer wordt ingevoerd in de AWESOMO 4000 wordt er uit verscheidende bronnen informatie opgehaald en samengevoegd. Vervolgens wordt deze data opgeslagen in de database en een selectie hiervan wordt ook weergegeven op de webpagina.



RECOMMENDATIONSYSTEEM

Luistersuggesties worden gebaseerd op hoeveel bepaalde nummers op elkaar lijken. Dit wordt bepaald aan de hand van onder andere artiest, BPM, genre en populariteit. Per nummer worden deze eigenschappen vergeleken, en wordt een waarde gegeven aan hoeveel twee nummers op elkaar lijken. Als een gebruiker een bepaald nummer luistert, wordt er in de database opgezocht welke nummers hier het meest op lijken, oftewel, de hoogste score hebben in relatie tot het oorspronkelijke nummer. In de toekomst zullen deze suggesties ook vergeleken worden met eerder geluisterde

muziek van de gebruiker. De nummers die ook hiermee vergeleken hoog scoren, zullen gesuggereerd worden.

Van elk nummer worden artiest, BPM, populariteit, genres, lengte, danceability en het jaar waarin het uitgebracht is opgehaald en opgeslagen. Elk van de eigenschappen wordt via een eigen methode vergeleken met de corresponderende eigenschap van de andere nummers. Op basis van deze vergelijking wordt er een score bepaald, die bij de totaalscore wordt opgeteld. Hoe hoger de score, hoe beter de nummers bij elkaar passen.

$$v_i = \sum_{p \in P, v=v_i} p_v$$

Op deze manier worden alle nummers onderling vergeleken en worden de totaalscores opgeslagen in een similarity-matrix. Wanneer een nummer geluisterd wordt, worden uit deze matrix de nummers met de hoogste score geselecteerd als luistersuggesties.

Het personaliseren van deze suggesties gebeurd door middel van vergelijken met de luistergeschiedenis. Er wordt een eerste selectie gemaakt van de 50 nummers die het beste passen bij het zojuist geluisterde nummer. Deze nummers worden vervolgens één voor één met de luistergeschiedenis vergeleken, om zo per nummer een persoonlijke score te berekenen. Na deze vergelijking met de luistergeschiedenis worden tenslotte de hoogst scorende vijf nummers getoond aan de gebruiker.

USERINTERFACE

Ons gebruikersinterface is qua indeling een verticaal drieluik. In het eerste luik staat het filmpje waar de gebruiker naar heeft gezocht, aangevuld met de naam van de artiest het de titel van nummer.

Daaronder vindt de gebruiker een voor hem op maat gemaakte recommendatie; 5 nummers die op basis van zowel objectieve als persoonsspecifieke gegevens gekozen zijn.

Het derde deel staat in het teken van objectieve informatie. Dit luik is onderverdeeld in twee kolommen. In de linker wordt uitgebreide informatie gegeven over de artiest in kwestie. De kolom rechts bevat de overige informatie. Deze informatie omvat onder andere de discografie van de artiest, eventuele kerstliederen en aankomende evenementen, aangevuld met links naar de iTunes en Amazon webwinkels om de single te kunnen kopen.

Verder bevat het interface een sticky header waar een nieuwe link ingevuld kan worden. Deze header blijft altijd binnen het scherm, waardoor een gebruiker eenvoudig een nieuwe link kan invoeren om naar een nieuwe pagina te gaan.

[Screenshot user interface](#)

IMPLEMENTATIE KEUZES

PHP

Na het afwegen van de voor- en nadelen is er gekozen PHP als primaire ontwikkeltaal te gebruiken. Deze keuze is gebaseerd op de eigenschappen van PHP en het feit dat de voorkeur uitging naar het gebruiken van één taal als basis. PHP kent enkele grote voordelen op het gebied van webdevelopment, maar tegelijkertijd ook aan aantal nadelen.

Allereest de voordelen. PHP is een taal die breed wordt ondersteund op webplatformen. Het is daardoor erg eenvoudig om hosting te regelen. Door de brede ondersteuning is de gemeenschap, die ontwikkelt en bijdraagt aan PHP, erg groot. Hierdoor is het weer eenvoudig om bestaande libraries te kunnen gebruiken, waardoor er geen code opnieuw hoeft te worden geschreven. Tijd die vrijkomt kan zo worden besteed aan het schrijven van code die daadwerkelijk functionaliteit toevoegt. Tenslotte is PHP een taal waarmee de hele groep enigszins bekend is en snel valt aan te leren. Ook dit bevordert de ontwikkelsnelheid.

Helaas heeft iedere keuze ook een aantal nadelen. PHP's voornaamste nadeel is de inconsistentie in de taal zelf en het gebrek aan afdwingen van een correcte manier van coden. De inconsistentie is eenvoudig op te lossen door gewoon regelmatig documentatie te gebruiken en zelf, waar nodig, een interface gebruiken welke wel consistent is. De correcte manier van coden is lastiger (try-catch, exceptions, classes), PHP heeft ondersteuning voor deze functionaliteiten pas sinds versie 5 in de taal opgenomen. Hierdoor is veel informatie op Internet niet geschreven voor deze features. Wat nog sterker meespeelt is dat ook binnen de TU Delft het onderwijs in PHP nog wordt gegeven op de "oude" manier. Hierdoor is het voor het team noodzakelijk te schakelen ten opzichte van wat men gewend is.

Over het totaal genomen is de keuze voor PHP sterk verdedigbaar. Het moeten aanleren van een correcte codingstijl kan ook juist als pluspunt worden gezien.

BOOTSTRAP

Voor de opmaak en indeling van de website word gebruik gemaakt van Twitters Bootstrap-framework. Hiermee is het eenvoudig een overzichtelijke en flexibele indeling te maken. Elk onderdeel en stukje informatie heeft een eigen plek op de pagina. Ook maakt Bootstrap het makkelijk om verschillende media in een pagina weer te geven. Zo kunnen er thumbnails van de recommended videos in een net blok gezet worden, terwijl de tekstblokken breder gemaakt zijn, om het leesgemak te bevorderen.

YOUTUBE

De keuze voor YouTube is redelijk eenvoudig. Omdat er gebruik wordt gemaakt van de YouTube links voor user-input is het nodig om op enige wijze met YouTube te communiceren. Bovendien is de gratis API van YouTube erg uitgebreid wat betreft data die opgevraagd kan worden.

Door de grote hoeveelheid data die de API terug geeft kan er op veel verschillende kenmerken worden gematcht, wat de match en recommendation naar de gebruiker toe erg fijnmazig geregeld kan worden.

FACEBOOK

De Facebook API is gebruikt om het registratie formulier te maken. Daardoor wordt, als men op Facebook is ingelogd, een deel van het formulier automatisch ingevuld. De API kan ook gebruikt

worden om muziek interesse en de likes van een gebruiker ophalen, om zo een beter beeld te krijgen van wat deze gebruiker leuk vindt.

ECHONEST

Wij hebben voor The EchoNest gekozen omdat deze goed samenwerkt met verschillende andere API's. Zo ondersteunt EchoNest momenteel 33 verschillende ID spaces. Door deze integratie fungeert deze bron als een verbindend element voor al onze API's. Zo kunnen wij bijvoorbeeld Spotify ID's gebruiken in onze calls naar deze API. Daarnaast heeft EchoNest zelf een zeer uitgebreide API met een gunstig rate-limit. Dit limiet ligt bij niet commercieel gebruik bij ongeveer 120 calls per minuut.

Daarnaast levert EchoNest alle informatie in de vorm van JSON objecten. Deze objecten vinden wij erg fijn om mee te werken omdat JSON objecten ten eerste kleiner zijn dan hun XML tegenhangers. Daarnaast hebben ze een simpelere syntax.

SPOTIFY

Wij hebben voor Spotify gekozen voor zijn uitgebreide catalogus en snelle API. Bijna iedere artiest en bijna ieder liedje of album is in Spotify's vertegenwoordigd. Om deze reden gebruiken wij in onze database Spotify id's als uniek id voor iedere artiest. Daarnaast halen we verschillende gegevens op, zoals bijvoorbeeld popularity, die we gebruiken in onze recommendatie.

DATABASE

Het gebruik van de database is tweeledig. Enerzijds wordt de informatie over de geregistreerde gebruikers opgeslagen; anderzijds verdient ook de informatie over de muzieknummers er een plekje.

Voor de gebruikersdata is een tabel opgezet. Deze bevat de inloggegevens van de gebruiker, de gebruikerskenmerken (geboortedatum, geslacht ed) en eventueel een Facebook authenticatietoken. Via dit token kan er later extra data van een gebruiker worden opgehaald. Door een aantal gebruikerskenmerken op te slaan bestaat de mogelijkheid om later deze data in te zetten voor het personaliseren van zoekresultaten en suggesties.

Voorts wordt er van ieder beluisterd nummer een informatieset opgeslagen in de database. Volgende keren hoeft deze data dan niet opgehaald te worden. Bovendien vormt deze dataset de basis van het algoritme welke gebruikers gerelateerde muzieknummers aanbiedt.

Van ieder nummer wordt vanuit verschillende bronnen data gecombineerd; voorbeelden zijn Spotify, Youtube en Echonest. Deze data wordt vervolgens weer opgeslagen in de database. Voorbeelden van de data zijn genres, ratings, populariteitstatistieken, tempo. Door het combineren van de verschillende kenmerken wordt de data daadwerkelijk interessant.

De data wordt in matrixvorm gerepresenteerd, maar in tabelvorm opgeslagen. Voor ieder item in database bestaat een x en een y veld, beide zijn identifiers voor muzieknummers. Het (x,y)-veld represeneert dan ook in hoeverre y aan te raden is als x wordt beluisterd.

Tenslotte wordt de data van beluisterde nummers ook gelinkt naar de individuele gebruikers. Door hen te volgen en zo direct informatie te kunnen krijgen over hun muzieksmaak wordt er een voordeel gegenereerd bij het personaliseren van resultaten. Deze data is vrij eenvoudig en bevat simpelweg een link tussen een nummer en een gebruiker, plus het aantal keer dat het betreffende nummer is beluisterd.

Voor een compleet overzicht van de tabellen en hun inhoud verwijzen we de lezer door naar appendix B.

UITVOERING

RECOMMENDATIONSYSTEEM

Omdat nummers van dezelfde artiest vaak relatief veel op elkaar lijken, is er een risico dat er enkel nummers van dezelfde artiest gesuggereerd worden. Om dit te voorkomen wordt er bij het vergelijken van artiest juist één punt bij de totaalscore opgeteld wanneer deze ongelijk is.

Van elke artiest wordt een lijst met de genres van zijn/haar muziek opgehaald. Deze lijsten worden doorlopen, en voor elk overeenkomend genre worden twee punten toegekend. Dit gebeurd op basis van de genres van de artiest, omdat het niet mogelijk bleek om dit per nummer te bepalen.

$$v_{genre} = \begin{cases} 2, & \text{Genres komen in beide nummers voor} \\ 0, & \text{Anders} \end{cases}$$

Lengte, BPM, danceability en jaar van uitkomen worden vergeleken door het verschil te berekenen. Er is voor gekozen om deze verschillen stapsgewijs te beoordelen, in plaats van hiervoor een continue functie te schrijven. Dit maakt het eenvoudiger en sneller in gebruik.

Als er meer dan 25 seconden verschil zit tussen de lengte van twee nummers, wordt er voor de lengte geen punten bijgeteld. Als het verschil tussen 15 en 25 seconden zit wordt er één punt bijgeteld, als het binnen 15 seconden valt twee.

$$v_{lengte} = \begin{cases} 0, & \Delta t > 25 \\ 1, & 15 < \Delta t \leq 25 \\ 2, & \Delta t \leq 15 \end{cases}$$

Verschil in BPM is opgedeeld in gelijk, minder dan vijf, minder dan tien en minder dan vijftien BPM verschil. Hiervoor worden respectievelijk drie, twee, één of een half punt toegekend. Wanneer het verschil buiten deze grenzen valt, passen de nummers hierin dus niet bij elkaar.

$$v_{BPM} = \begin{cases} 0, & \Delta bpm > 15 \\ 0.5, & 10 < \Delta bpm \leq 15 \\ 1, & 5 < \Delta bpm \leq 10 \\ 2, & 0 < \Delta bpm \leq 5 \\ 3, & \Delta bpm = 0 \end{cases}$$

Danceability wordt weergegeven door een score tussen nul en één. Als het verschil minder dan 0.05 is, zijn de nummers ongeveer even dansbaar, en worden er drie punten gegeven. Wanneer het verschil minder dan 0.1 is, wordt er twee punten bijgeteld, anders geen.

$$v_{danceability} = \begin{cases} 0, & \Delta danceability > 0.1 \\ 2, & 0.05 < \Delta danceability \leq 0.1 \\ 3, & \Delta t \leq 0.05 \end{cases}$$

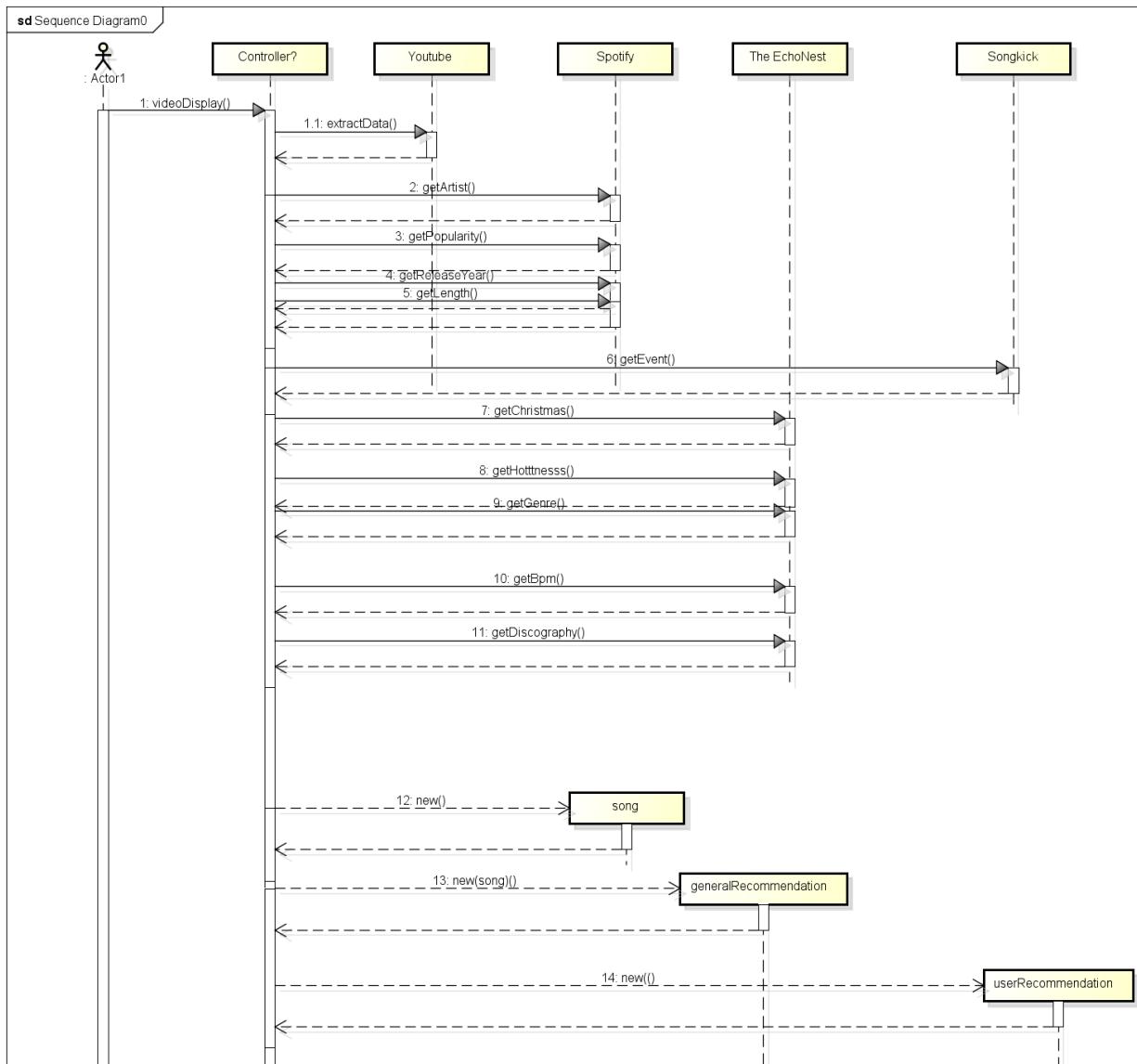
Het jaar van uitkomen van een nummer wordt ook vergeleken, als het verschil minder dan drie jaar is, wordt de score met drie punten opgehoogd. Als het verschil minder dan zes jaar is wordt er één punt toegevoegd aan de totaalscore en wanneer het verschil groter is geen.

$$v_{releaseYear} = \begin{cases} 0, & \Delta releaseYear > 6 \\ 1, & 3 < \Delta releaseYear \leq 6 \\ 3, & \Delta releaseYear \leq 3 \end{cases}$$

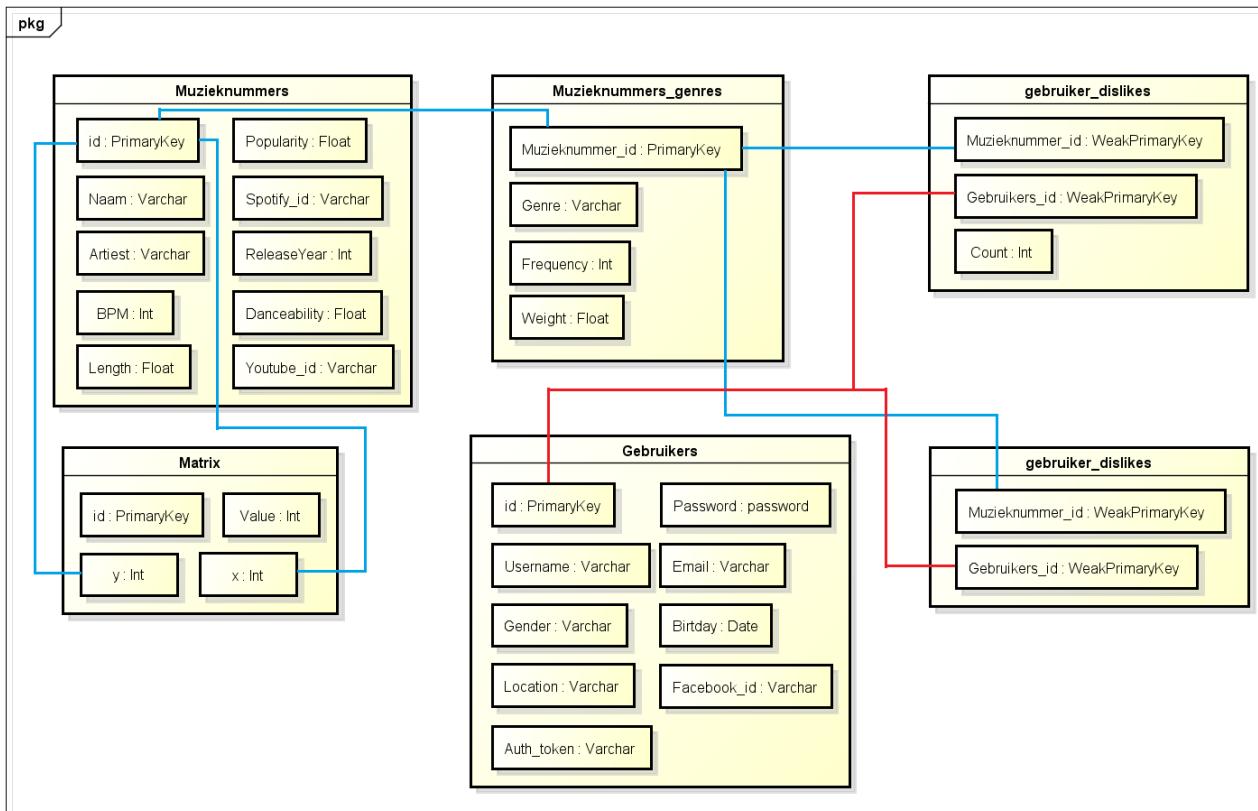
De populariteit van een nummer wordt opgehaald als een score tussen nul en één. Omdat dit aangeeft hoeveel dit nummer geluisterd wordt is dit een goede manier om slechtere nummers weg te filteren. Daarom wordt er simpelweg vier maal de populariteit toegevoegd aan de totaalscore.

$$v_{popularity} = 4 \cdot popularity$$

UML



DATABASE



PLANNING

Het project is opgedeeld in vier sprints, elke sprint bevat een aantal hoofdonderwerpen waar aan gewerkt zal worden. Op deze manier wordt er elke keer een volgend onderdeel van het eindproduct gerealiseerd.

1. De eerste sprint ligt de focus op het maken van een eenvoudig systeem, waarin de meeste basisfunctionaliteit gerealiseerd wordt. Er kan worden geregistreerd, ingelogd, video's worden afgespeeld en informatie over de artiest wordt weergegeven. Hiermee is er een goede basis om volgende sprints uit te breiden.
2. Tijdens de tweede sprint wordt een begin gemaakt aan het recommendationsysteem. Informatie wordt gecombineerd en afgestemd op de gebruiker. De informatie behorend bij een nummer wordt uitgebreid en de site wordt opgemaakt en gebruiksvriendelijk gemaakt.
3. In de derde sprint worden de eerste tests gedaan op de werking van het recommendationsysteem. De webpagina die gebruikers te zien krijgen wordt verder gepersonaliseerd en het informatiesysteem wordt afgemaakt.
4. De laatste sprint is bedoeld om de puntjes op de i te zetten. Het systeem wordt getest, de resultaten van de tests worden gebruikt om verbeteringen door te voeren. Aan het eind van deze sprint is het systeem voltooid en kan het gepresenteerd worden.

EERSTE SPRINT

De eerste sprint van het project stond in het teken van het ontwikkelen van basisfunctionaliteit die het project ondersteunen. Denk hierbij vooral aan het uitzoeken van informatie over koppelingen en het opstarten van verschillende frameworks welke de ontwikkeling ondersteunen. Voorbeelden zijn het opzetten van actiepunten systemen, documentatie, errorlogs, gebruikersaccount, GitHub en het framework zelf.

Allereerst is er begonnen met het kiezen van de databronnen die ingezet worden. De keuze hiervoor viel op Facebook, Youtube en DBpedia. Deze zijn vervolgens aan elkaar gelinkt in deze eerste sprint.

Voor het inloggen is er de mogelijkheid om dit via Facebook te doen. Indien een gebruiker ingelogd is binnen Facebook wordt het registratieformulier automatisch ingevuld met de al bekende parameters. Ook krijgen we in dat geval een authenticatietoken binnen waarmee we later extra informatie kunnen ophalen over de betreffende gebruiker.

Youtube wordt ingezet om de videokenmerken op te halen en om de video zelf af te spelen. Hiervoor biedt Youtube de gewenste functionaliteit binnen haar API's. Voor de verbindingen naar Youtube wordt gebruik gemaakt van de API van Zend Framework.

Tenslotte wordt de naam van de artiest die wordt ontvangen via de API van Youtube via een SparQL query doorgestuurd naar DBpedia, waar een korte samenvatting over die artiest ophaalt.

Qua layout wordt er op dit moment nog gebruik gemaakt van simpel ogende pagina's. Het is de bedoeling dit in de toekomst echter via het Twitter Framework Bootstrap te laten lopen, waarbij de data asynchroon wordt opgehaald voor de gebruiker.

TAAKVERDELING

Taak:	Verantwoordelijk:
Facebook-registratie	Bjorn
Querier RDF-data	Bjorn
Youtube API	Liesbeth
Bootstrap voor lay-out website	Liesbeth
Framework systeem	Rogier
Integreren onderdelen	Rogier

TWEEDE SPRINT

In de tweede sprint hebben we het ophalen van informatie verbeterd. Er is overgestapt van sparql naar EchoNest. Er is voor EchoNest gekozen omdat dit veel sneller werkt. Daarnaast is niet alle benodigde data beschikbaar in RDF, maar wel op te halen via EchoNest. Op dit moment worden BPM, rating en genres opgehaald van Echonest. Verder slaan we ook het Spotify-ID op, omdat dit een unieke waarde is, die gebruikt kan worden voor het ophalen van verdere informatie.

Wat betreft de website stond de eerste sprint puur in het teken van functionaliteit. In deze sprint is er aandacht besteed aan de opmaak van de website. Hiervoor hebben wij gebruik gemaakt van Twitter's Bootstrap. Met dit framework kan je in korte tijd een mooie opmaak bouwen, waar makkelijk extra's aan toegevoegd kunnen worden.

Daarnaast is er begonnen met het recommendationsysteem. Op dit moment nemen we 4 variabelen mee in de berekening: bpm, artiest, genre en populariteit. In de volgende sprint willen we het systeem uitbreiden om de zoekresultaten te verfijnen. Ook zullen dan de resultaten gepersonaliseerd worden.

TAAKVERDELING

Taak:	Verantwoordelijk:
Informatie EchoNest	Bjorn
Verslag	Bjorn
Recommender algoritme	Liesbeth
Bootstrap voor lay-out website	Liesbeth
Opbouwen database	Rogier
Integreren onderdelen	Rogier

DERDE SPRINT

In de derde sprint is het recommendationsysteem voltooid. Er zijn extra vergelijkingskenmerken toegevoegd en methodes gemaakt om deze eigenschappen van nummers te vergelijken. Behalve BPM, genre, artiest en populariteit wordt er nu ook gekeken naar jaar van uitkomen, lengte van het nummer en danceability. Er is gekeken naar verschillende mogelijkheden om deze diverse kenmerken met elkaar te vergelijken. Ook is het belangrijk om te definieren hoever deze kenmerken uit elkaar mogen liggen om alsnog punten te scoren, en hoe deze scoreverdeling plaats vindt. Uiteindelijk zijn deze waarden grotendeels empirisch bepaald op basis van een testset.

De recommendations worden nu als thumbnails weergegeven op de website. Hierop kan geklikt worden om deze nummers te beluisteren. Tenslotte worden er nu ook de komende concerten van een artiest weergegeven. Deze informatie zorgt ervoor dat een gebruiker vrijwel alle gewenste informatie kan vinden binnen AWESOME 4000.

TAAKVERDELING

EVALUATIE

TESTPLAN

Om te controleren of het systeem werkt zoals oorspronkelijk voor ogen stond is er besloten verschillende soorten tests uit te voeren. In dit hoofdstuk zal erop worden ingegaan hoe deze tests plaatsvinden en hoe gekwantificeerd kan worden of het systeem voldoet aan de gestelde eisen.

Er zijn verschillende soorten tests mogelijk om de kwaliteit van het systeem te verifiëren. De bekendste zijn de unit tests, de integratietesten, de gebruikerstesten en validatietesten. Van ieder van deze tests zal worden uitgelegd hoe en waarom deze zijn ingezet.

Allereerst is binnen de AWESOMO 4000 gebruik gemaakt van unit tests. Verschillende klassen voor communicatie met API's van derde partijen lenen zich bijvoorbeeld goed om individueel te testen. Binnen deze klassen draait het voornamelijk om verschillende databronnen op te halen en op een correcte manier te decoderen en via een consistente interface aan te bieden aan de rest van de applicatie.

Om deze tests te draaien is er een losse testmethode per klasse aangemaakt waarin verschillende randvoorwaarden per API call worden getest. Hierbij kan men meteen denken aan een incorrecte API aanroep, een API aanroep welke geen resultaat geeft, een API aanroep welke één resultaat teruggeeft en een API aanroep welke meerdere resultaten teruggeeft. In geen van deze gevallen mag de klasse foutmeldingen teruggeven aan de client. De unit tests zijn zodanig ontworpen om deze onvolkomenheden in de implementatie aan het licht te brengen en worden dagelijks opnieuw gedraaid om te voorkomen dat ergens later fouten blijken op te treden. Voor gevonden nieuwe bugs worden er vervolgens nieuwe tests aangemaakt die de bug kunnen detecteren; op deze manier wordt er voorkomen dat er een regressiebug kan optreden in de code.

Ten tweede wordt er vaak gebruikt van integratietesten in ontwikkelprocessen. Echter is besloten voor dit project om deze testen te laten zitten. De software is hiervoor nog niet gecompliceerd genoeg. Bovendien wordt het gemis aan integratietesten in principe opgevangen door meer focus te leggen op gebruikerstesten en validatietesten.

Ten derde is er gekozen voor relatief eenvoudige gebruikerstesten. Er wordt gekozen voor een methode waarbij de gebruikersresultaten worden vergeleken met de algemene suggesties. Deze worden gemixt (5 om 5) en samen getoond. Vervolgens is het de taak van de gebruiker deze suggesties te waarderen tussen de 1-10. Voor een gebruiker zonder opgebouwde geschiedenis is het verwachte resultaat een evenwichtige verdeling van de verschillende nummers. Voor een gebruiker die het systeem enige tijd heeft gebruikt dienen de gebruikersresultaten consequent beter te zijn dan de algemene resultaten of suggesties die YouTube doet bij eenzelfde nummer.

Tenslotte wordt er gebruikt gemaakt van validatietesten. Hiervoor wordt er bijvoorbeeld een afspeellijst gemaakt door iemand van bijvoorbeeld 40 nummers. Van deze lijst worden er vervolgens 30 beluisterd op een specifiek account, waarna er wordt gekeken naar de aanbevelingen welke volgen uit deze luistergeschiedenis. Het systeem voldoet indien de resterende 10 nummers bijvoorbeeld worden aanbevolen door het systeem. Voor de volledigheid dient dit type test enkele malen herhaald te worden met verschillende muzieklijsten. Ook dienen de resterende nummers present te zijn in de database; anders kunnen deze immers niet aanbevolen worden.

TESTRESULTATEN

ALGEMENE BEVINDINGEN

Allereerst is er bij het gebruik van het systeem al een aantal dingen opgevallen. Wanneer een nummer geen spotify-ID heeft, kan er niet genoeg informatie opgehaald worden om een goede recommendatie te doen. Uiteindelijk is er besloten om deze nummers dan maar niet mee te nemen in het recommendatiesysteem.

AFSPEELLIJST

Voor deze test hebben we een willekeurige selectie gemaakt van ongeveer 50 nummers van drie verschillende gebruikers uit de last.fm dataset met luistergeschiedenissen. In de onderstaande tabel staat per afspeellijst het aantal nummers wat op deze lijst staat, de opnieuw beluisterde nummers, hoeveel van de 10 achtergehouden nummers van de playlist er in de recommandaties verschenen, en het totaal aantal keer dat er nummers van deze 10 in de recommandaties verschenen.

afspeellijst	Aantal nummers	Ingevoerd nummers	Nummers uit laatste 10	Totaal aantal overeenkomsten
User 1	40	15	5	11
User 2	52	20	6	10
User 3	58	25	4	18

CONCLUSIE/DISCUSSIE

LITERATUURLIJST

Voor het project wordt er gebruik gemaakt van verschillende API's. Deze documentatie is dan ook ingezet om de resultaten te verwerken:

<https://developer.spotify.com/technologies/web-api/>

<http://developer.echonest.com>

<https://developers.google.com/youtube/>

<https://github.com/twitter/bootstrap>

Verder is er natuurlijk ook gebruik gemaakt van research papers

paper: A Survey Paper on Recommender Systems, D. Almozra, G.Shahatah, L. Albdulkarim, M. Khrees, R. Martinez, W. Nzoukou

https://blackboard.tudelft.nl/webapps/portal/frameset.jsp?tab_tab_group_id=_10_1&url=%2Fwebapps%2Fblackboard%2Fexecute%2Flauncher%3Ftype%3DCourse%26id%3D_40547_1%26url%3D

APPENDIXES

SPARQL VS ECHONEST

In eerste instantie werd met sparql de informatie opgehaald over de artiest en het nummer. Dit gebeurde door middel van een library voor php genaamd SparqlLib. Dit is een lichte doch krachtige library is, die alle benodigde functies bevat. Helaas bleek dat DBpedia.org niet snel genoeg resultaten terug te geven. Daarom is er overgestapt op EchoNest, hier kunnen JSON-objecten opgevraagd worden met alle gewenste informatie.

APPENDIX B

DATABASESCHEMA

De database bestaat uit een zestal tabellen waarin de data voor het systeem is opgeslagen.

Tabel: Muzieknummers	Deze tabel bevat alle informatie specifiek voor één muzieknummer.
Data:	
- <u>id</u> - Naam - Artiest - Spotify_id - BPM - Popularity - Length - ReleaseYear - Danceability - Youtube_id	

Tabel: Muzieknummers_genres	Deze tabel bevat de informatie welke genres koppelt aan muzieknummers
Data:	
- <u>Muzieknummer_id</u> - Genre - Frequency - Weight	

Tabel: Matrix	Deze tabel geeft aan hoe goed nummer X met nummer Y.
Data:	
- <u>id</u> - <u>X</u> - <u>Y</u> - Value	

Tabel: Gebruikers	Deze tabel bevat de informatie over de gebruikers van het systeem
--------------------------	---

Data:

- Id
- Username
- Password
- Gender
- Email
- Location
- Birthday
- Facebook_id
- Auth_token

Tabel: gebruikers_muzieknummers	Deze tabel geeft aan hoevaak een gebruiker een bepaald nummer heeft geluisterd
--	--

Data:

- Muzieknummer_id
- Gebruiker_id
- Count

Tabel: gebruiker_dislikes	Deze tabel bevat welke muzieknummers een gebruiker niet leuk vindt. Deze worden niet meer geselecteerd
----------------------------------	--

Data:

- Muzieknummer_id
- Gebruiker_id