

Tu delft

Groep 5

Salim Hadri  
Kaj Oudshoorn  
Viresh Jagesser

IKE Project

Inhoud

Table of Contents

Inleiding 3

2 Ontwerp 4

Probleem beschrijving 4

Ontwerp beschrijving 4

Functionele beschrijving 4

Use case 7

Workflow 8

3 Implementatie 9

4 Evaluatie 10

Tussen product(wat is er eigenlijk bereikt) 10

Verloop van implementatie (Is het volgens plan gegaan?) 10

5 Test uitvoering 11

Test senario’s 12

Welk test-scenario is uitgevoerd? 12

Wat was de begintoestand van de database en/of de applicatie? 12

Welke testgegevens zijn ingevoerd? 12

Wat is het waargenomen gedrag van de software? 12

Wat is het verwachte gedrag? 12

Waarin wijkt het waargenomen gedrag precies af van het verwachte gedrag? 12

Hoe kan de fout gereproduceerd worden? 12

In welke versie van de software is de fout aangetroffen? 12

Wat heeft u gedaan om mogelijke oorzaken van de fout te achterhalen? 12

# Inleiding

De opbouw van dit rapport is als volgt. Hoofdstuk 2 beschijft globaal de oplossing voor de recommandatie systeem d.m.v. een probleem beschrijving ontwep beschrijving, functionele beschrijving en een ruwe oplossing met daarbij een workflow. In hoofdstuk 3 komen de details van het ontwerp aan de orde. De evaluaties van dit product worden besproken in hoofdstuk 4. Tot slot staat er in hoofdstuk 5 de tests die zijn uitgevoerd.

# 2 Ontwerp

## Probleem beschrijving

De gebruiker moet worden geadviseert met andere muziek, nadat hij zelf een nummer heeft heeft ingevoerd (hierbij ook de naam v/d artiest). Voorlopig werkt dit met een enkel nummer. Hierbij moet naar overeenkomsten tussen de nummers worden gezocht. Daarnaast moet het aantal geadviseerde nummers niet te groot zijn (dan wordt het onoverzichtelijk) of te klein (niet genoeg keuze).

## Ontwerp beschrijving

Het programma wordt in java geschreven, omdat de programeurs ervaring hebben met Sesame (een library voor het werken met SPARQL endpoints). Later bleek wel dat veel databases voor php gemaakt waren en dat een juiste database moeilijk te vindenwas.

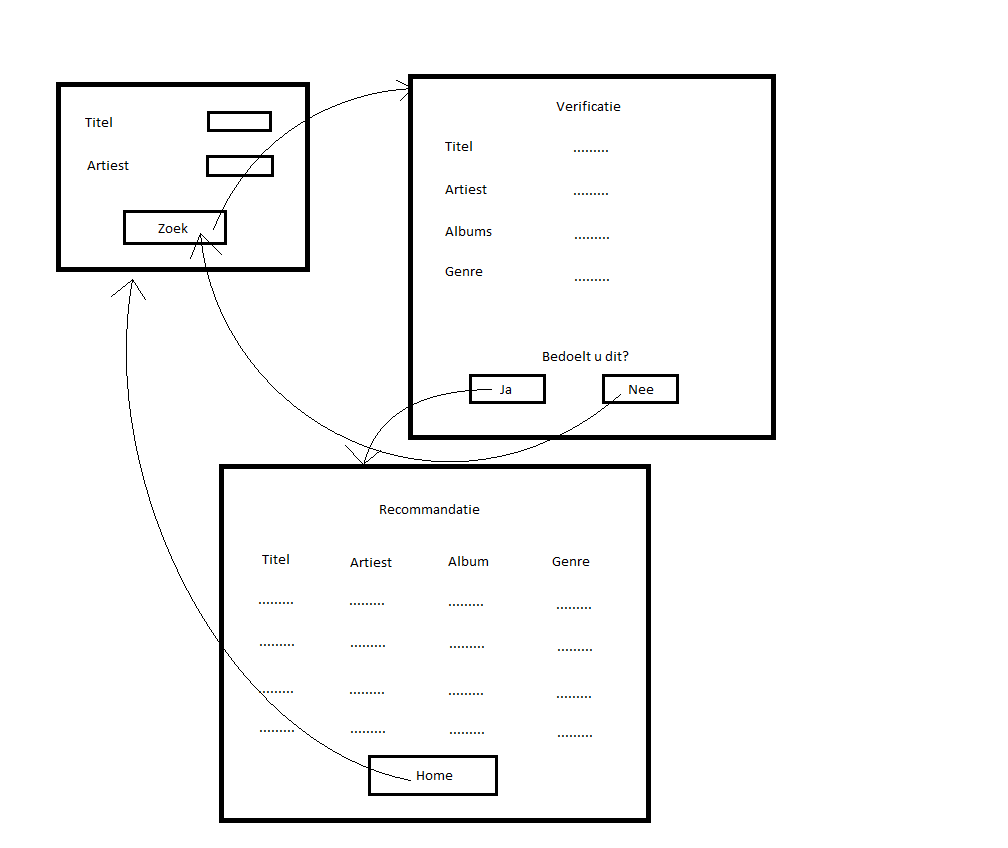
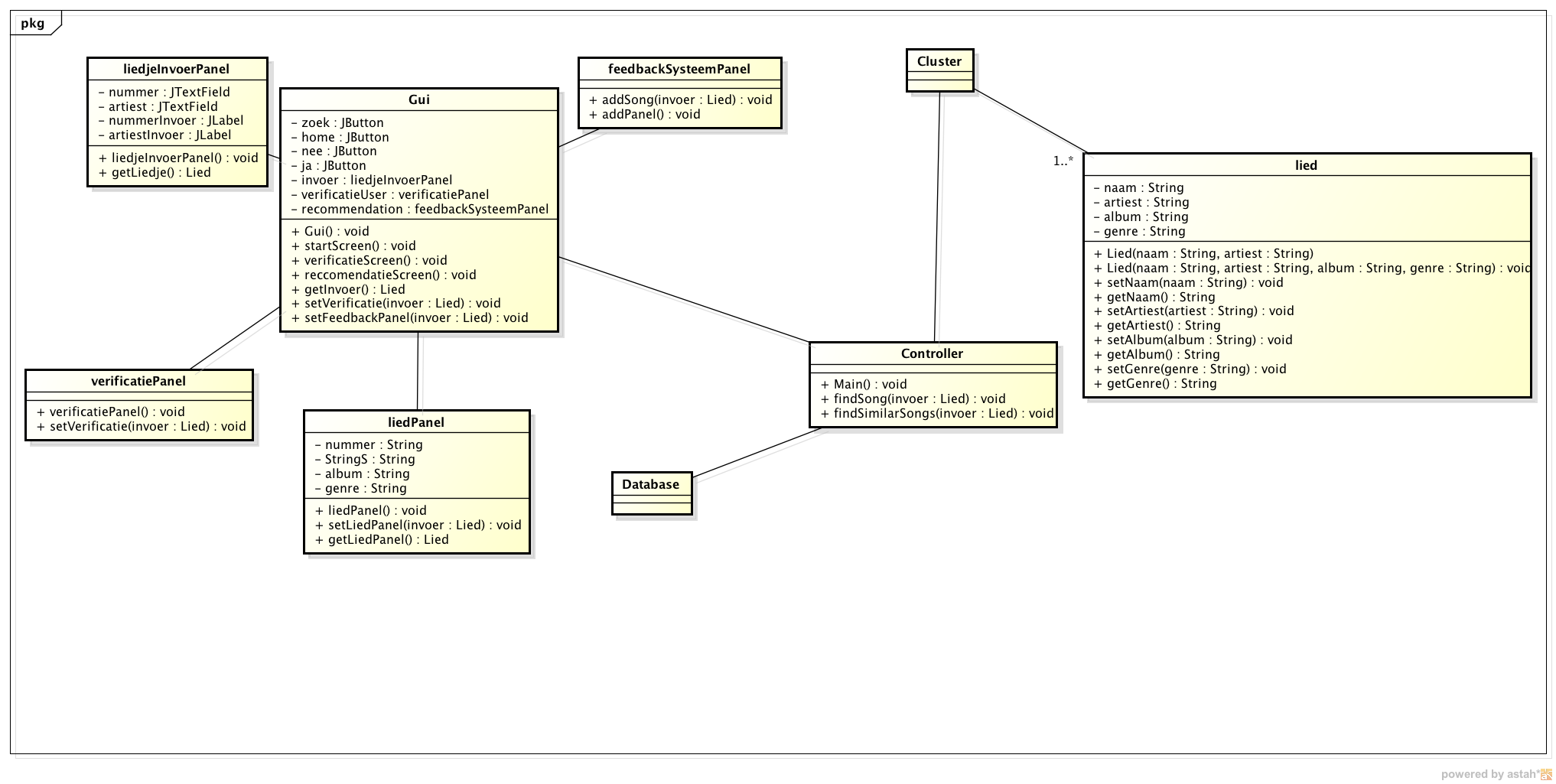
## Functionele beschrijving

Het systeem moet de volgende dingen kunnen:

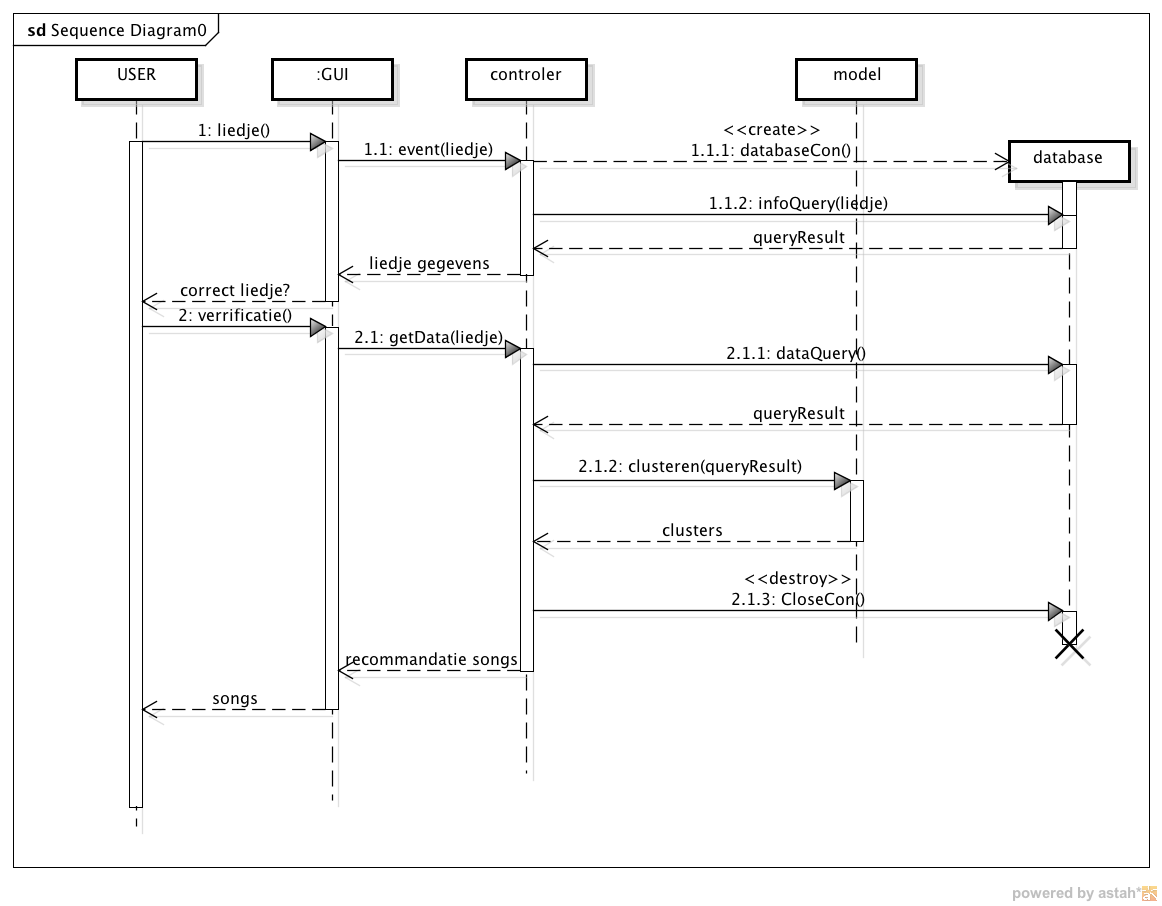
* Systeem moet gegeven nummer + artiest herkennen.
* Gevonden nummer verificeren: met nummer, artiest, genre, album.
* Recommandaties teruggeven van nummers die vergelijkbaar zijn met het gegeven nummer.
* Bij recommandaties nog extra info geven (zoals een review).

De volgende dingen zijn gewenst:

* Geen lange wachttijd (niet langer dan een paar seconden)
* De optie om opnieuw een recommendatie te vragen, zonder het programma opnieuw te hoeven opstarten.

Interface model Klassendiagram

## Use case Workflow

Ruwe oplossingen

Er wordt gebruik gemaakt van de lastFM database. Met deze database kunnen tracks, artiesten en albums worden opzoeken. Met de methode getSimilar (uit de lastFM API) worden er vergelijkbare tracks gevonden en deze worden geretourneerd.

Later implementeren wij de getSimilar functie zelf m.b.v. clusteren. Tracks met overeenkomsten komen hierbij bij elkaar te liggen en vervolgens wordt er rondom het gegeven nummer gezocht naar tracks.

Er is ook een werkende connectie met een andere database (kasabi), hiermee kan er meer informatie over de track worden opgezocht. Zoals: genre, componist, producer, review. Maar er is nog geen tijd geweest om dit verder te implementeren.

Verificatie wordt gedaan door info over het gegeven nummer terug te sturen, waarmee de gebruiker kan beslissen of hij dat nummer daadwerkelijk bedoeld.

De info bij recommandaties kan worden gegeven door info van de tracks op te vragen bij de kasabi database.

Beide databases zijn remote, de data hoeft dus niet worden ingeladen bij het opstarten van het programma en ligt vanaf het begin al klaar om opgehaald te worden. Hierdoor is er geen lange laadtijd.

# 3 Implementatie

Als eerste werd de GUI geimplementeerd, hiervoor werd gebruik gemaakt van de awt java library. Vervolgens werd de Last Fm library geimporteerd. In de LastFm klasse werden de functies findSong en FindSimilar geimplementeerd. Zoals de naam al doet vermoeden kan je met de findSong functie op basis van een naam van een nummer en artiest het liedje vinden in de database van last fm. Dit was heel makkelijk te implementeren aangezien last Fm een heel duidelijke API heeft. Bij het implementeren van findSimilar songs wordt er gebruik van een standaard functie die op basis van een lied al recommendaties kan doen. Dit is echter tijdelijk, aangezien er bij het implementeren van de { Ontology…..} geen werkende connectie met een database kon worden gemaakt. Daarom werd er een zijsprong naar de last Fm database gemaakt.

Als laatst is er ook nog een Controller klasse die geimplementeerd is, dit verliep ook soepel . De controller klasse zorgt voor het dataverkeer tussen de GUI en de last Fm database.

# **4** Evaluatie

## Tussen product(wat is er eigenlijk bereikt)

Op dit moment werkt dit tussen product als verwacht. Er kan een lied ingevoerd en geverifieerd worden. Vervolgens kan op basis daarvan een recommondatie gegeven worden. Naast de connectie met lastfm, is er nu ook een connectie met een ontology database. De connectie met deze muziek ontology wordt in de volgende iteratie geïmplementeerd.

## Verloop van implementatie (Is het volgens plan gegaan?)

De implementatie is niet volgens plan verlopen, aangezien het erg moeilijk was om een werkende endpoint te vinden naar een muziek ontology. Hierdoor is er een zij spring gemaakt naar de lastfm database. Verder is de implementatie van de gui en de klassen volgens plan verlopen.

# 5 Test uitvoering

Dit systeem heeft een aantal functionaliteiten.

• Het Systeem moet nummer + artiest herkennen.

• Verificatie: met nummer, artiest, genre, album.

• Recommandaties teruggeven van nummers die vergelijkbaar zijn.

• Bij recommandaties nog extra info geven (zoals een review).

Deze functionaliteiten zullen getest worden met behulp van de userinterface. Daarbij letten we op een aantal aspecten. Om te beginnen met het gedrag van het systeem en hoe zinvol dit systeem reageert op foute invoer. Verder wordt er gekeken of de output de verwachte data bevat. Tot slot moet dit systeem makkelijk te gebruiken zijn, dus wordt er getest op gebruiksvriendelijkheid.

Het testen van de recommondation wordt echter op een andere manier gedaan. Hier worden de resultaten van de lastfm getSimilar functie vergeleken met onze resultaten, die zijn gevonden met behulp van onze cluster algorithmen.

## Test senario’s

Op de ”<http://www.applinet.nl/artikelen/acceptatietest.html>” site is een vragenlijst gegeven. We zullen dit gebruiken om een aantal test senario’s uit te voeren.

## Welk test-scenario is uitgevoerd?

## Wat was de begintoestand van de database en/of de applicatie?

## Welke testgegevens zijn ingevoerd?

## Wat is het waargenomen gedrag van de software?

## Wat is het verwachte gedrag?

## Waarin wijkt het waargenomen gedrag precies af van het verwachte gedrag?

## Hoe kan de fout gereproduceerd worden?

## In welke versie van de software is de fout aangetroffen?

## Wat heeft u gedaan om mogelijke oorzaken van de fout te achterhalen?