**Ontwikkeling van een Android-bioscoopapplicatie**

**Technisch Ontwerp**

**VERTROUWELIJK**

**Versie 1.4**

Auteurs: Tobias Willemsen (2124631) en Szonja Hollós (2128202)

Laatst gewijzigd op: 3-4-2018

**Document**

|  |  |  |
| --- | --- | --- |
| **Versie** | **Datum** | **Beschrijving** |
| 1.0 | 24-3-2018 | Begin gemaakt aan het technisch ontwerp. Inleiding gemaakt op de samenvatting na. |
| 1.1 | 27-3-2018 | Logisch gegevensmodel gemaakt en toegevoegd met een beschrijving. Package diagram gemaakt en toegevoegd met een beschrijving. |
| 1.2 | 30-3-2018 | Technisch procesflow gemaakt en toegevoegd met een beschrijving. |
| 1.3 | 2-4-2018 | Klassendiagram gemaakt en toegevoegd met een beschrijving. |
| 1.4 | 3-4-2018 | Samenvatting toegevoegd. |

**Inhoud**

[1. Inleiding 4](#_Toc510621533)

[1.1 Samenvatting 4](#_Toc510621534)

[1.2 Achtergrond 4](#_Toc510621535)

[1.3 Doelstellingen 5](#_Toc510621536)

[1.4 Over dit document 5](#_Toc510621537)

[2. Data 6](#_Toc510621538)

[2.1 Logisch gegevensmodel 6](#_Toc510621539)

[3. Structuur 7](#_Toc510621540)

[3.1 Klassendiagram 7](#_Toc510621541)

[3.2 Package diagram 8](#_Toc510621542)

[4. Proces 9](#_Toc510621543)

# **Inleiding**

Voor dit technisch ontwerp worden een aantal diagrammen gemaakt die helpen bij het realiseren van de bioscoopapplicatie. Deze bioscoopapplicatie zal in de Java programmeertaal geschreven worden. Verder wordt er gebruikt gemaakt van een API waarmee een aantal films kunnen worden opgehaald. Dit wordt gedaan met een back end waarbij de API gekoppeld wordt aan de applicatie. De gebruikte API voor het ophalen van de films komt van <https://www.themoviedb.org/>.  
In dit document wordt duidelijk wat de architectuur van de applicatie is. Dit wordt duidelijk gemaakt door middel van een klassendiagram en een package diagram. Ook wordt de structuur van de database helder met behulp van een logisch gegevensmodel.

## ***Samenvatting***

In dit technisch ontwerp is duidelijk geworden hoe de applicatie gemaakt moet worden. Dit is gedaan door middel van een klassendiagram en een package diagram. Deze diagrammen maken namelijk duidelijk hoe de applicatie precies in elkaar moet zitten. Verder staat in technisch ontwerp hoe de database van de applicatie eruit komt te zien. Dit wordt afgebeeld met een logisch gegevensmodel. In dit logisch gegevensmodel staan alle entiteiten met hun attributen en de relaties tussen deze entiteiten. Hierdoor is het makkelijker om de database te ontwerpen en te koppelen aan de applicatie. Ook is er gebruik gemaakt van een technisch proces flow diagram. Hierin wordt het duidelijk wat de volgorde van de schermen is wanneer de gebruiker een ticket koopt. In tegenstelling tot het proces flow diagram van het functioneel ontwerp wordt er in het technische diagram duidelijk gemaakt dat de gebruiker altijd de mogelijkheid heeft om de stoppen. De gebruiker kan dus bij elk scherm de applicatie afsluiten. Ook wordt er aangegeven vanuit welke schermen de gebruiker terug kan navigeren naar het gedetailleerde scherm. Door dit technisch proces flow diagram wordt het duidelijk wat de volgorde van de schermen in de applicatie precies is.

## ***Achtergrond***

De baas van de bioscoop wilt graag voor de bioscoopbezoekers een android-bioscoop applicatie laten maken. Hiervoor heeft hij een team van vijf personen ingehuurd. Dit team bestaat uit: Szonja Hollós, Thomas de Lange, Julian van Noordennen, Tobias Willemsen en Stephan de Wit. Er zijn een aantal eisen en wensen aan de applicatie gesteld. Deze eisen en wensen zijn voornamelijk vanuit de visie van een bioscoopbezoeker gesteld. Er zijn echter ook een aantal eisen en wensen vanuit de visie van de manager van de bioscoop gesteld. Verder is niet alles gedetailleerd vastgelegd, waardoor de opdrachtnemers zelf voorstellen voor de applicatie moeten doen. Naast deze informatie zijn er niet veel meer achtergrondgegevens voor het project gegeven.

## ***Doelstellingen***

Het doel van dit project is om een android-bioscoop applicatie te maken voor de baas van de bioscoop. De baas van de bioscoop is namelijk de opdrachtgever van het project. Verder wordt de applicatie uiteindelijk ook vooral voor de bezoekers van de bioscoop gemaakt, aangezien zei de doelgroep zijn die de applicatie zullen gaan gebruiken. Een van de belangrijkste doelen van het maken van de applicatie is dus ook dat de bioscoopbezoekers gemakkelijk informatie over films kunnen vinden en makkelijk en snel kaartjes kunnen bestellen. Daarnaast is het ook een doel van de opdrachtnemers om hun vaardigheden bij het uitvoeren van een project te verbeteren. Dit houdt in dat ze hun samenwerkingsvaardigheden willen verbeteren en tegelijk ook hun software engineering vaardigheden willen verbeteren.

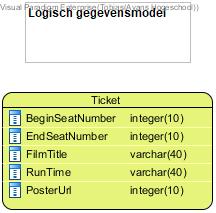
## ***Over dit document***

Dit document is gemaakt voor de baas van de bioscoop. Ook is het technisch ontwerp gemaakt voor de opdrachtnemers, zodat ze een beter beeld krijgen voor het realiseren van de applicatie. Dit document dient dan ook geactualiseerd te worden gedurende het project.

# **Data**

## ***Logisch gegevensmodel***

Hieronder staat het logisch gegevensmodel van de applicatie (zie figuur 1). In dit logisch gegevensmodel wordt duidelijk welke entiteit er bij de database hoort. Ook staan alle attributen met hun types in de ticket entiteit.

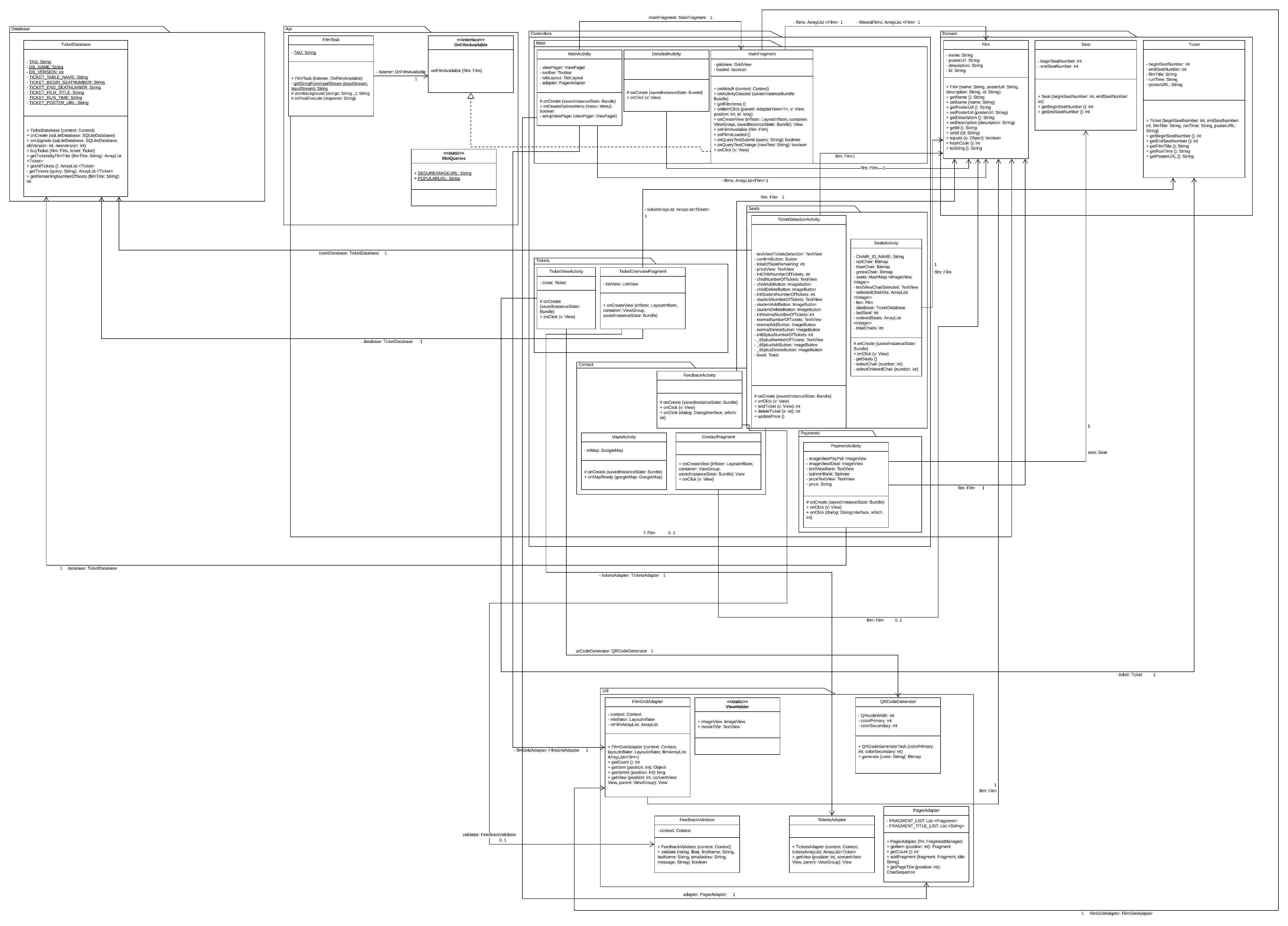


Figuur 1: logisch gegevensmodel

# **Structuur**

## ***Klassendiagram***

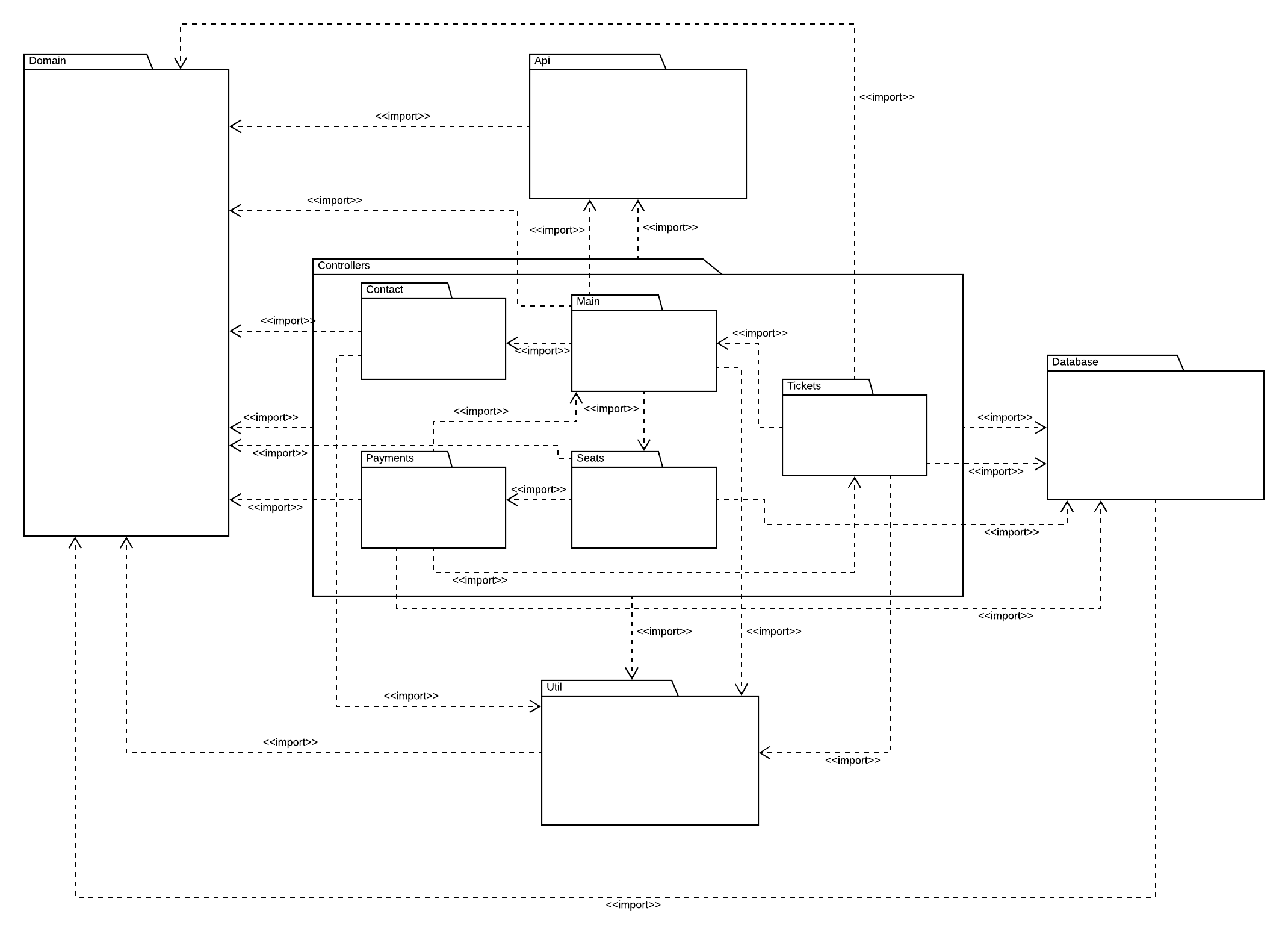
Hieronder staat in figuur 2 het klassendiagram van de applicatie. Door dit klassendiagram wordt het duidelijk welke klassen er in de applicatie zitten. Ook wordt er aangetoond welke attributen en methodes er in de klassen zit. Verder laat het klassendiagram ook zien wat de relaties tussen alle verschillende klassen zijn. Deze relaties worden aangeven met verschillende pijlen.



Figuur 2: klassendiagram

## ***Package diagram***

Figuur 3 toont het package diagram van de applicatie. Dit package diagram laat zien welke packages de applicatie bevat. In dit diagram wordt ook aangegeven wat de relaties tussen deze packages zijn. Deze relaties bestaan allemaal uit dependency relaties. Hierbij is elke dependency relatie een import dependency relatie. Dit betekent dat de packages afhankelijk zijn van elkaar. Tussen de klassen binnen de packages lopen dus ook afhankelijkheden. Hierbij worden er bijvoorbeeld in allerlei klassen objecten van andere klassen aangemaakt die deze klassen nodig hebben voor bepaalde functies of implementaties. In het diagram wordt snel duidelijk dat elk package gebruik maakt van de domainklassen binnen het domain package. Hierbij spreekt het domain package dus nooit iets van de andere packages aan. Verder is zichtbaar dat het controllers package gebruik maakt van alle andere packages. Dit komt gedeeltelijk omdat in het controllers package de MainActivity klasse zit. Deze klasse maakt namelijk gebruik van minimaal één klasse uit alle andere packages.



Figuur 3: package diagram

# **Proces**

Op pagina 11 staat beschreven wat de volgorde van de schermen van de wireframes is (zie figuur 4). Dit wordt gedaan door middel van een technisch proces flow diagram. Dit diagram is dus niet hetzelfde als het proces flow diagram van het functionele ontwerp. Dit diagram is namelijk gedetailleerder. In dit technische proces flow diagram wordt duidelijk dat de gebruiker vaak de mogelijkheid heeft om te stoppen. Dit wordt aangegeven met het icoontje van de cirkel binnen nog een cirkel. Dit icoontje wordt ook wel het eindicoontje genoemd.  
Verder staat er ook een groene lijn in het technische proces flow diagram. Deze lijn geeft de volgorde aan die wordt uitgevoerd wanneer een gebruiker een ticket wilt kopen. Deze groene lijn gaat dus van boven naar beneden. Ook zijn er een rode lijnen in het diagram. Deze lijnen worden aangeroepen wanneer de gebruiker terug wilt navigeren naar een vorige pagina. Dit kan bijvoorbeeld gedaan worden als een gebruiker een ticket koopt, maar vervolgens annuleert om zo terug naar het gedetailleerde scherm van de desbetreffende film te gaan. Deze rode lijnen gaan dus van onder naar boven.  
Alle activiteiten die in het technisch proces flow diagram staan kunnen worden gekoppeld aan klassen binnen de applicatie. Per genummerde activiteit wordt er een koppeling gemaakt aan een of meerdere klassen. Deze koppelingen staan hieronder opgesomd.

1. MainActivity, MainFragment, PagerAdapter, FilmTask, OnFilmAvailable, FilmGridAdapter, Film

2. MainActivity, MainFragment, Film

3. DetailedActivity, Film

4. TicketSelectionActivity, TicketDatabase, Seat, Film

5. TicketSelectionActivity, TicketDatabase

6. TicketSelectionActivity, Seat, Film

7. TicketSelectionActivity

8. SeatsActivity, Seat, Film

9. SeatsActivity, Seat

10. SeatsActivity, Seat

11. PaymentActivity, Seat, Film

12. PaymentActivity

13. PaymentActivity

14. PaymentActivity, TicketDatabase, Ticket, Seat, Film

15. PaymentActivity

16. MainActivity, ContactFragment, PagerAdapter

17. MapsActivity

18. MainActivity, TicketOverviewFragment, Ticket, TicketDatabase

19. MainActivity, TicketOverviewFragment, Ticket, TicketsAdapter

20. MainActivity, TicketOverviewFragment, Ticket

21. TicketViewActivity, QRCodeGenerator

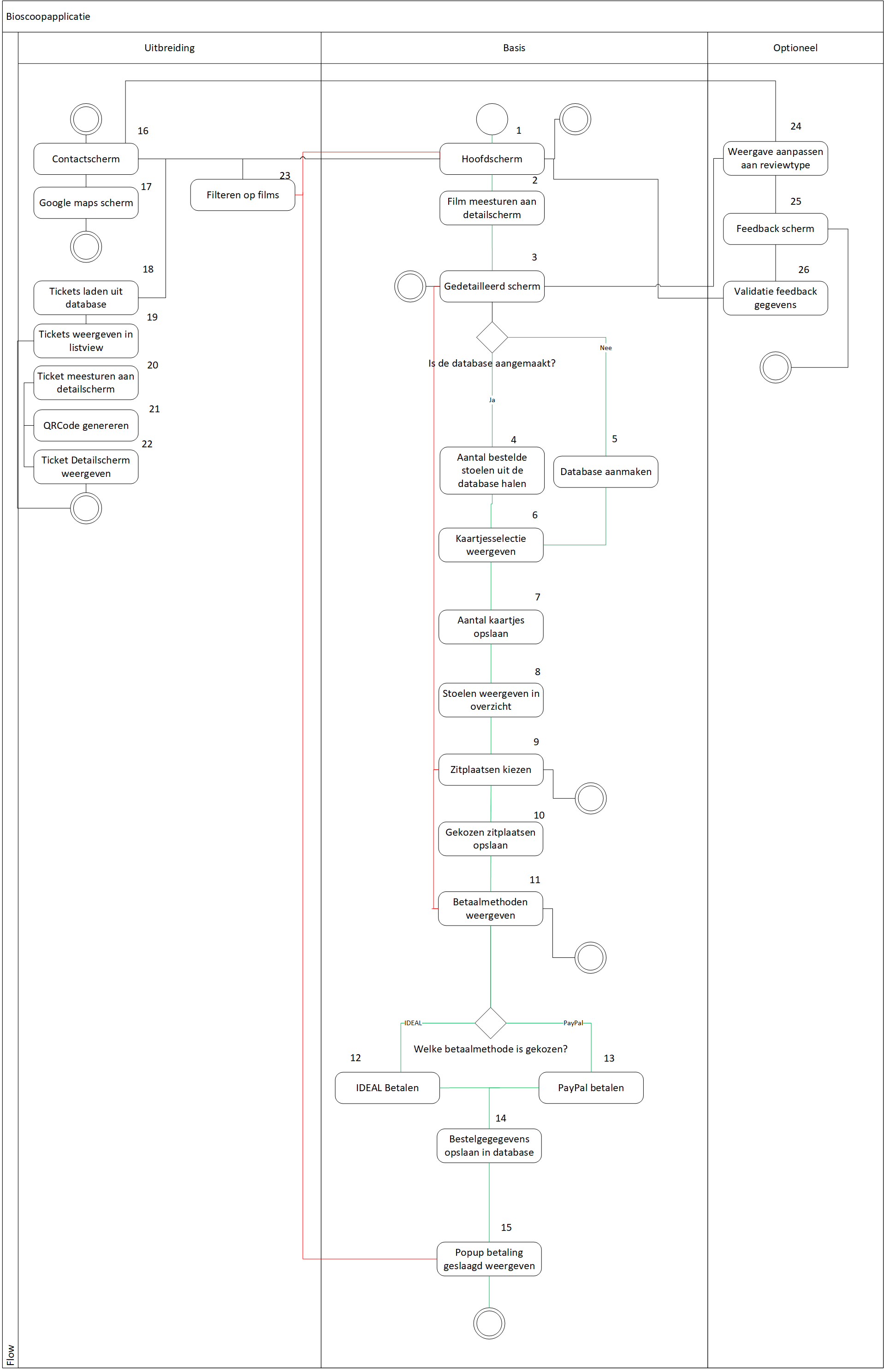
22. TicketViewActivity

23. MainActivity, MainFragment, FilmGridAdapter, Film

24. FeedbackActivity, Film

25. FeedbackActivity

26. FeedbackActivity, FeedbackValidator



Figuur 4: technisch proces flow diagram­