MovieRadar  
  
 **Technisch Ontwerp**

Gemaakt door:

Naam studenten: Sietse ‘t Hooft, Jan Roelofs, Jochem ter Braak, Gijs van Vugt

Studentnummers: 2213722, 2218997, 2214876, 2216616

# Inhoudsopgave

**H.**[**1 Inleiding 3**](#_1vz1dbjv2kiq)

[**H.2 Database ontwerp 4**](#_9n5w9ca6r7bm)

[**H.3 De Unified Modeling Language 5**](#_xa0ykj13v5ly)

[H.3.1 Klassediagram 5](#_3bbdjgf0mmy9)

[H.3.2 Package Diagram 6](#_rqg27ptrsmcu)

[**H.4 Activity diagrammen 7**](#_uqt1f817of9i)

# 

# H.1 Inleiding

Dit technische ontwerp beschrijft gedetailleerd de plannen, specificaties en benaderingen voor het ontwikkelen van de bioscoop-applicatie MovieRadar. Deze app is speciaal ontworpen om gebruikers te voorzien van uitgebreide informatie over films, tijden en andere relevante gegevens die verband houden met bioscoopbezoeken.

Het document gaat diep in op de keuzes met betrekking tot technologieën, gegevensbeheer, verwerkingsmethoden en andere cruciale aspecten die essentieel zijn voor de ontwikkeling van MovieRadar.

Om een helder beeld te krijgen van hoe de app er op technisch vlak uit zal zien, zijn er verschillende diagrammen opgenomen. Het databaseontwerp biedt een nauwkeurige representatie van de database die zal worden gebruikt in de app. Het klassendiagram dient als een voorbeeld van hoe de code zal worden gestructureerd, met georganiseerde packages om de functionaliteit te categoriseren. Het activity-diagram visualiseert de flow van de applicatie, waardoor een beter begrip ontstaat van hoe gebruikers door de app zullen navigeren.

# H.2 Database ontwerp

In dit hoofdstuk laten wij ons database ontwerp zien. Dit ontwerp laat zien hoe wij de entiteiten in de database weergeven. We laten ook zien wat de Primary Keys (PK), Foreign Keys (FK) en de relaties die daarmee gelegd worden. Daarnaast laten we ook zien wat er gebeurt als een entiteit geüpdatet wordt of gedelete wordt.

# 

*Figuur 2.1 Database ontwerp*

# H.3 De Unified Modeling Language

In dit hoofdstuk behandelen wij de klassediagram en package diagram, om een overzicht te maken hoe wij de applicatie willen gaan schrijven.

## H.3.1 Klassediagram

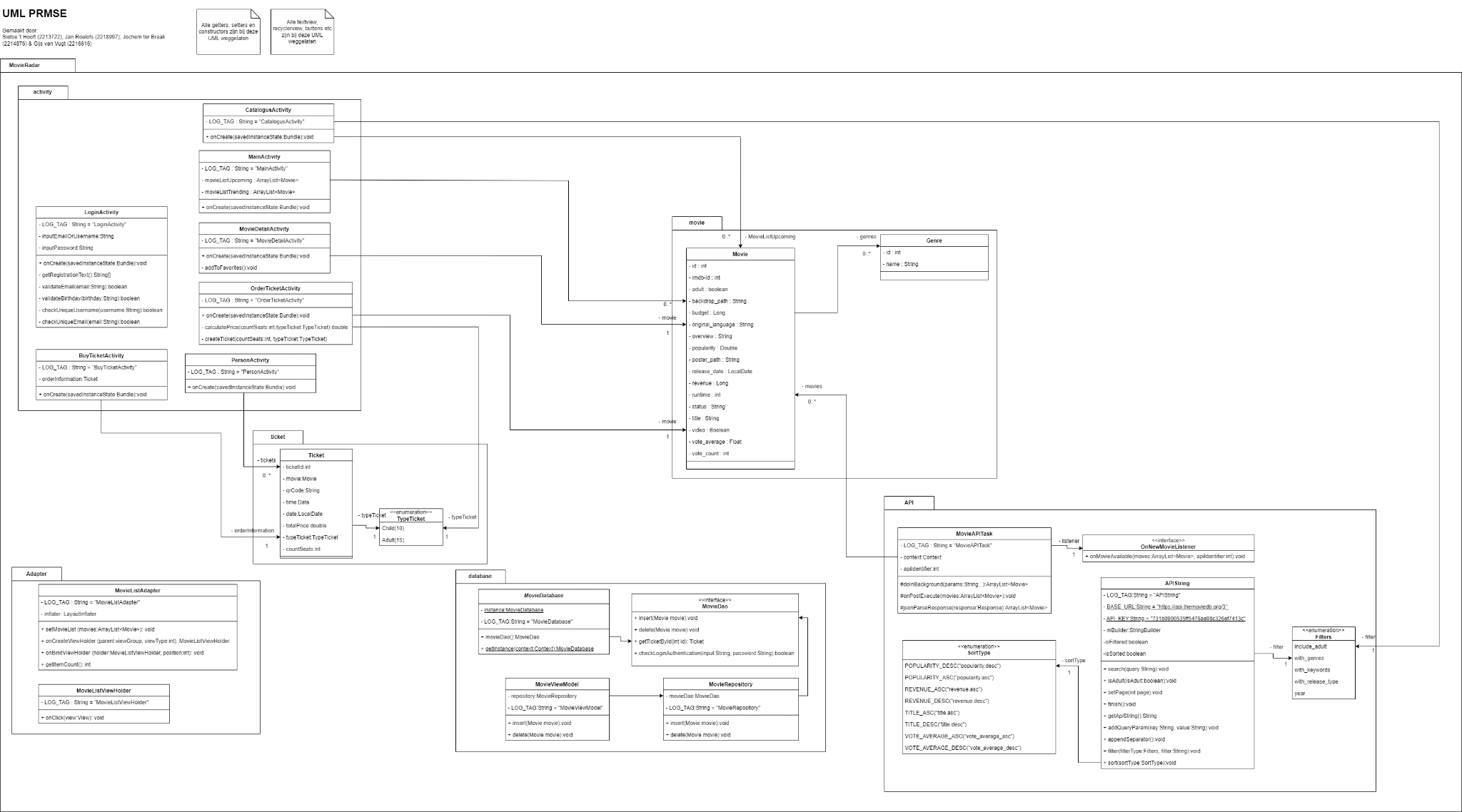
In dit klassediagram geven wij het ontwerp van onze code weer. De entiteiten staan voor onze klassen, met daarin attributen en methoden. Ook zie je de relaties tussen klassen en welke data ze delen.

# 

*Figuur 3.1 Klassediagram*

## H.3.2 Package Diagram

Alle klassen bij elkaar bevinden zich in de Movieradar package.  
Daarbinnen hebben we de klassen onderverdeeld in Activity, Ticket, Movie, API, Adapter en Database lagen.  
  
In de Activity laag vindt alle logica plaats, we sturen data vanuit de Movie en ticket laag naar de Database laag.  
De Ticket laag slaat data op van Tickets die je koopt  
De Movie laag haalt data van de API en vult de Movie objecten met deze data.  
De Adapter laag vult de UI elementen met de data die wordt gestuurd vanuit de Activity laag  
De database laag krijgt data binnen zoals inloggegevens en ticketdata en bewaart deze in de database.



*Figuur 3.2 Package Diagram*

# H.4 Activity diagrammen

# 

*Figuur 4.1 Ticket bestellen activity diagram*

In dit activity diagram laten wij zien hoe je vanuit het Homescherm een ticket kan betalen. Je leest het diagram van boven naar beneden, links naar rechts.

# 

*Figuur 4.1 Ticket bestellen activity diagram*

In dit activity diagram laten wij zien hoe je vanuit het Homescherm een film toevoegt aan een lijst. Je kan kiezen om een lijst toe te voegen aan je favorieten, of om het aan een andere lijst toe te voegen. Je leest het diagram van boven naar beneden, links naar rechts.

# 

*Figuur 4.1 Ticket bestellen activity diagram*

In dit activity diagram laten wij zien hoe je inlogt/registreert. Wanneer een gebruiker nog niet geregistreerd is, dan kan een gebruiker niet inloggen. Je leest het diagram van boven naar beneden.