

Movie App

Martijn Reenaers

3ICT

Inhoud

[Inleiding 3](#_Toc156238194)

[Oogpunt Gebruiker 4](#_Toc156238195)

[Login Page en Register Pagina 4](#_Toc156238196)

[Movie Page 5](#_Toc156238197)

[Map Page 6](#_Toc156238198)

[Profile Page 6](#_Toc156238199)

[Structuur 8](#_Toc156238200)

[Model 8](#_Toc156238201)

[Fragment 8](#_Toc156238202)

[ViewModel 9](#_Toc156238203)

[Gradle 9](#_Toc156238204)

[Firebase Database 11](#_Toc156238205)

[Authenticatie Firebase 13](#_Toc156238206)

[Technische Werking 13](#_Toc156238207)

[Model 13](#_Toc156238208)

[Movie 13](#_Toc156238209)

[Liked 14](#_Toc156238210)

[User 14](#_Toc156238211)

[SignUpFragment 15](#_Toc156238212)

[LogInFragment 17](#_Toc156238213)

[MovieFragment 18](#_Toc156238214)

[MovieAdapter 19](#_Toc156238215)

[MovieViewModel 20](#_Toc156238216)

[DetailsViewModel 21](#_Toc156238217)

[MovieDetailFragment 23](#_Toc156238218)

[MapsFragment 24](#_Toc156238219)

[MapsViewModel 27](#_Toc156238220)

[ProfileViewModel 29](#_Toc156238221)

[ProfileFragment 30](#_Toc156238222)

[Nav.xml 31](#_Toc156238223)

## Inleiding

Voor het vak Mobile Apps 2 heb ik gekozen voor mijn eigen Movie App te maken in Android Studio Het doel van mijn app is om de gebruiker de meest recente films te laten zien. Ook kan de gebruiker een film leuk vinden, omdat we werken met een inlogsysteem kan elke gebruiker andere films leuk vinden. We kunnen ook een film op naam zoeken zodat je naar de detailspage van de film kunt gaan en een film liken. Ook kan je eigen locatie en de locaties van cinemas zien op de map page. Op de Profile page kan je je email en eigen ID zien, ook kan je films liken. De app maakt gebruik van Firebase als de backend voor het opslaan en ophalen van film gegevens, map gegevens en inloggegevens, en volgt de MVVM (Model-View-ViewModel) structuur.

Het ontwikkelen van mijn Movie App maakte mij meer vertrouwd met de Android Studio omgeving en de mogelijkheden voor het maken van Android applicaties. Daarnaast was het integreren van de Firebase een belangrijk aspect van de ontwikkeling, ik heb gekozen voor Firebase in plaats van een API omdat ik kon kiezen welke data ik er inzette waardoor ik meer vrij was voor het creëren van mijn app. En het heeft me ook veel bijgeleerd, zodat ik dit later ook kan gebruiken in andere projecten.

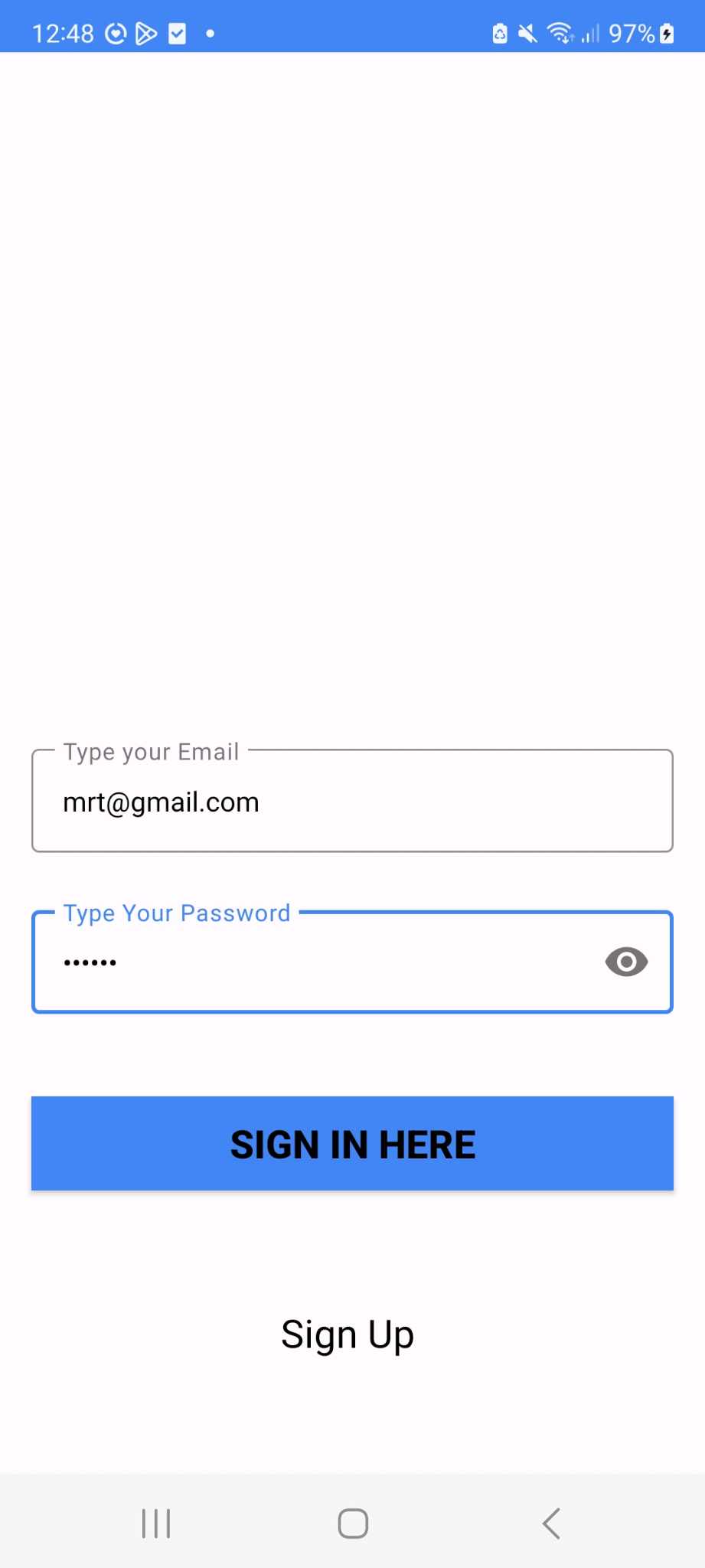
In dit verslag zal ik de belangrijkste aspecten bespreken, waaronder de structuur, de functionaliteiten en de gebruikersinterface.

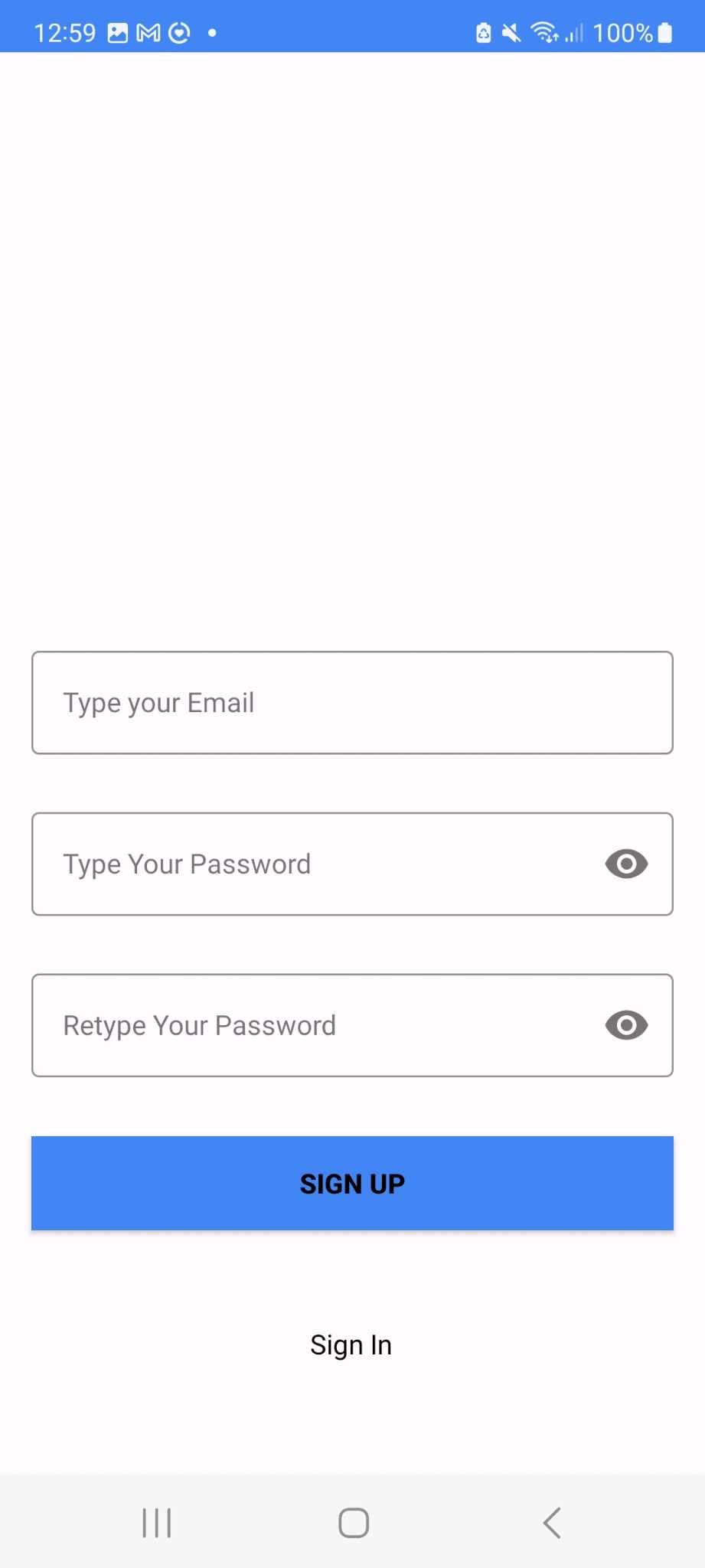
## Oogpunt Gebruiker

Hierin gaan we het visueel aspect van de movie app bespreken, vanuit het oogpunt van de gebruiker.

## Login Page en Register Pagina

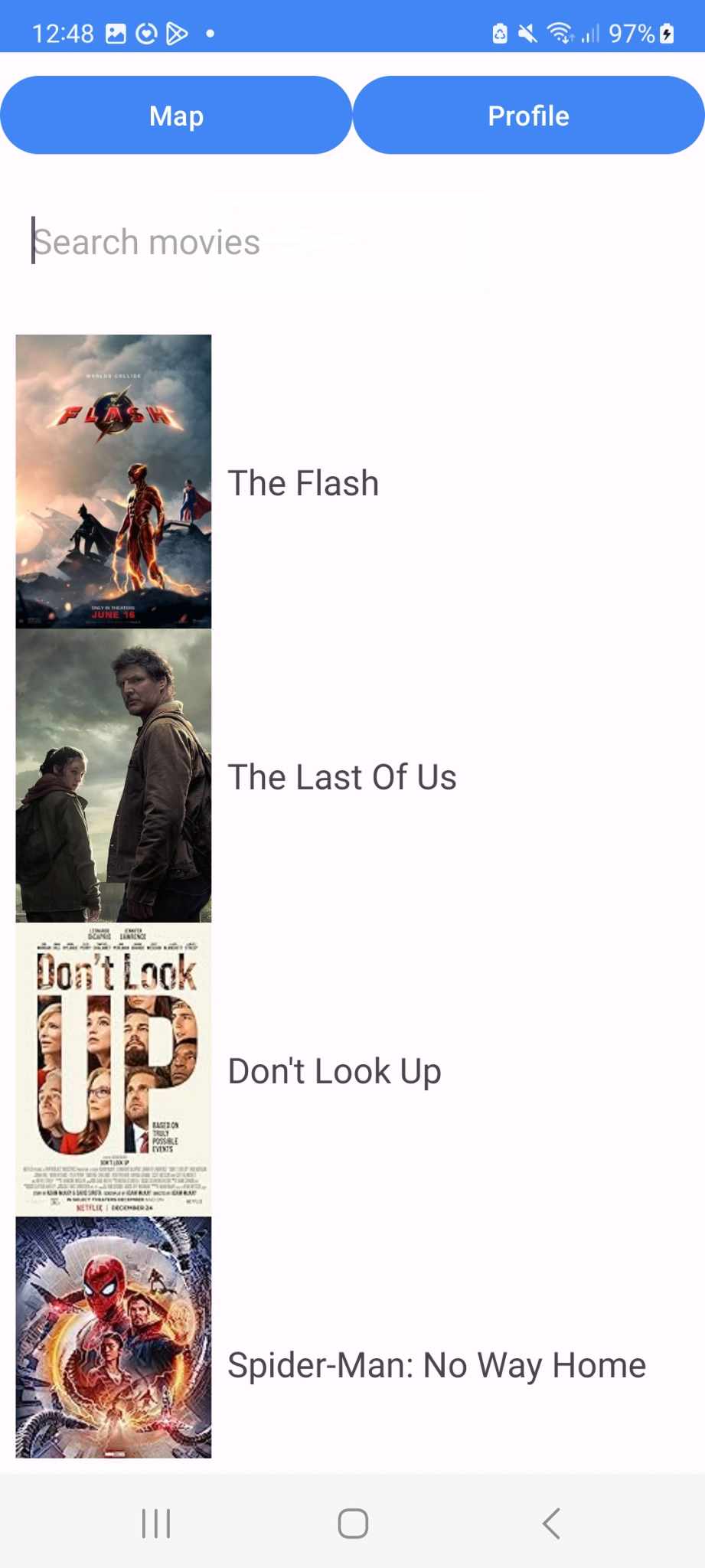
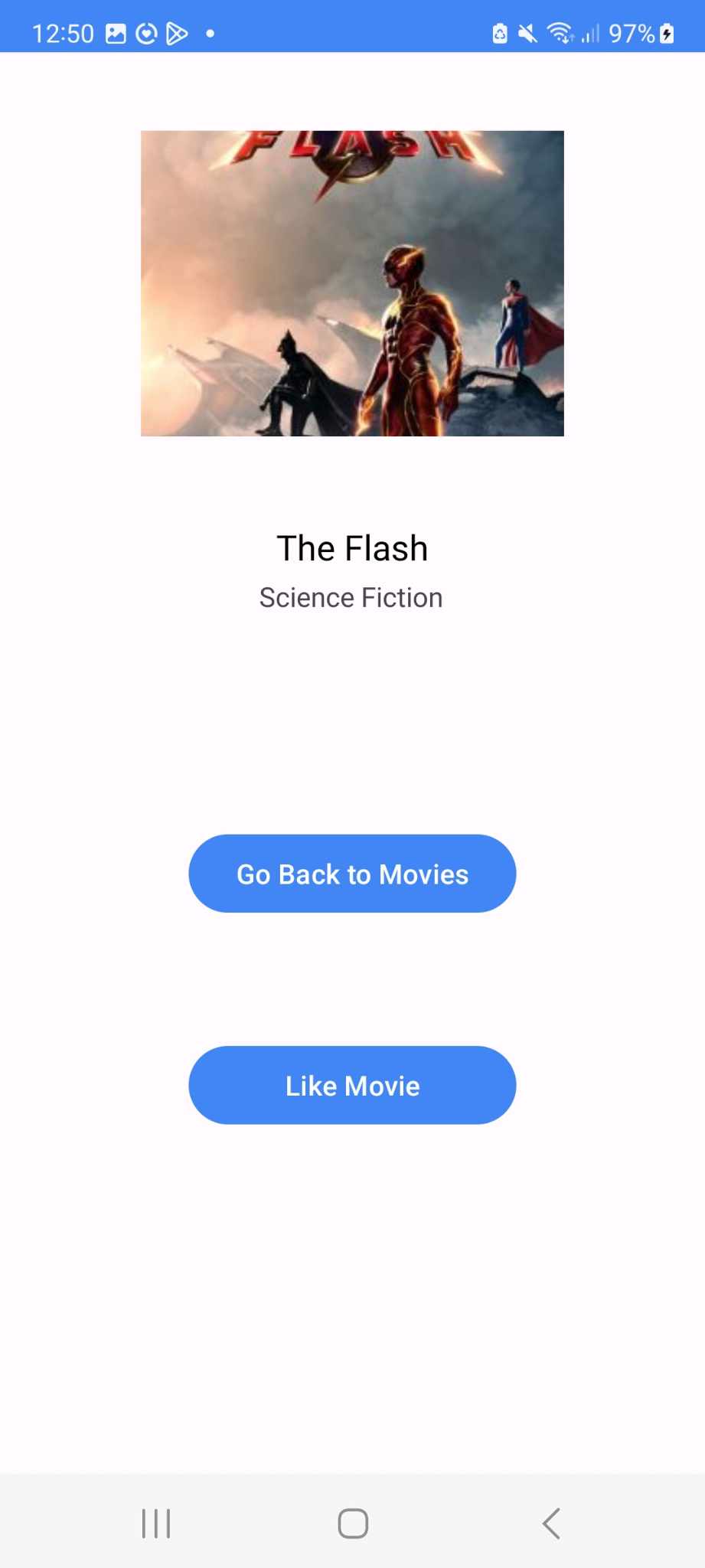
Op onze zoekpagina kan je een uitgebreide lijst aan films zien, je kan ook opzoeken via tekst welke film je wilt opzoeken. Als je op de film klikt ga je automatisch naar de detailpagina van die film.





## Movie Page

Op de Movie Pagina kan de gebruiker scrollen tussen de films en zoeken naar de films die opgeroepen worden uit de database.



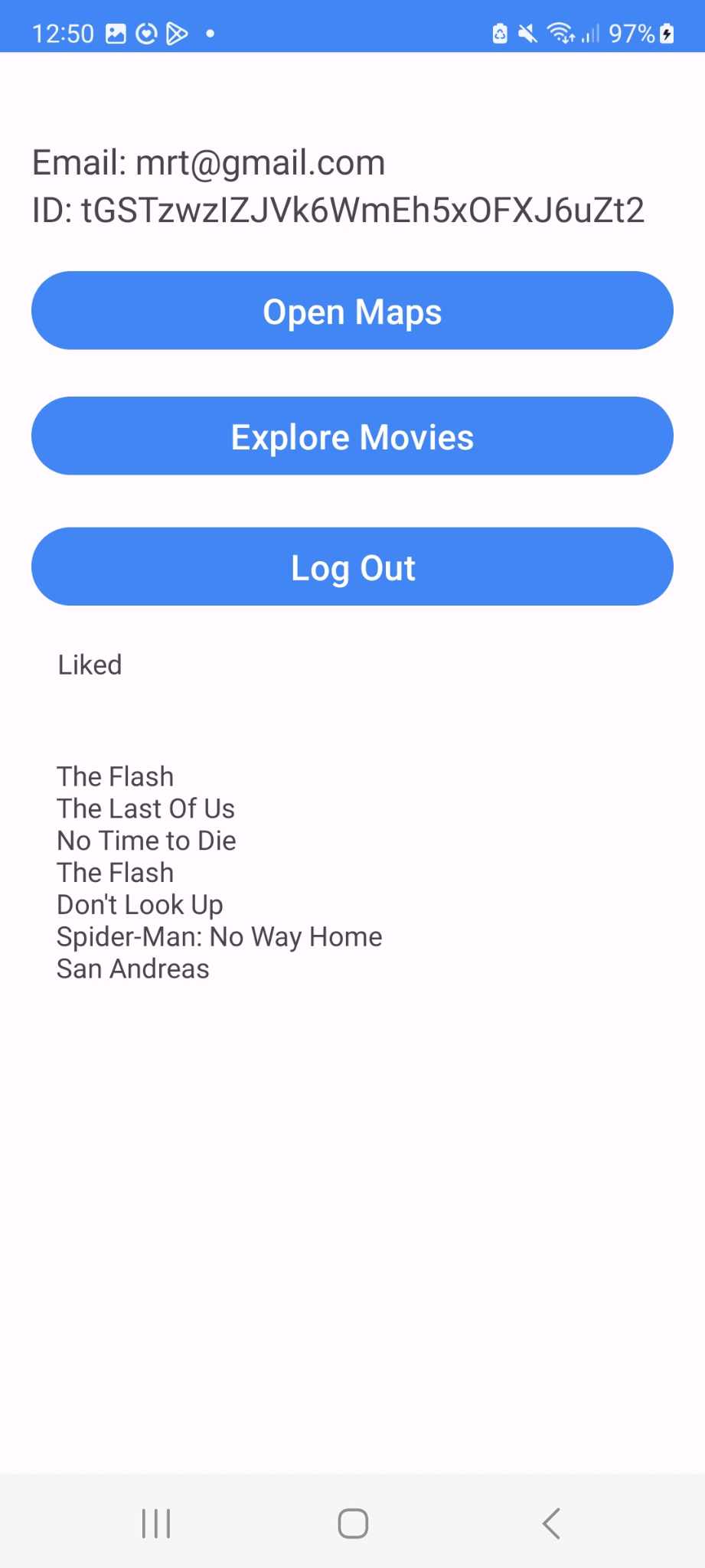
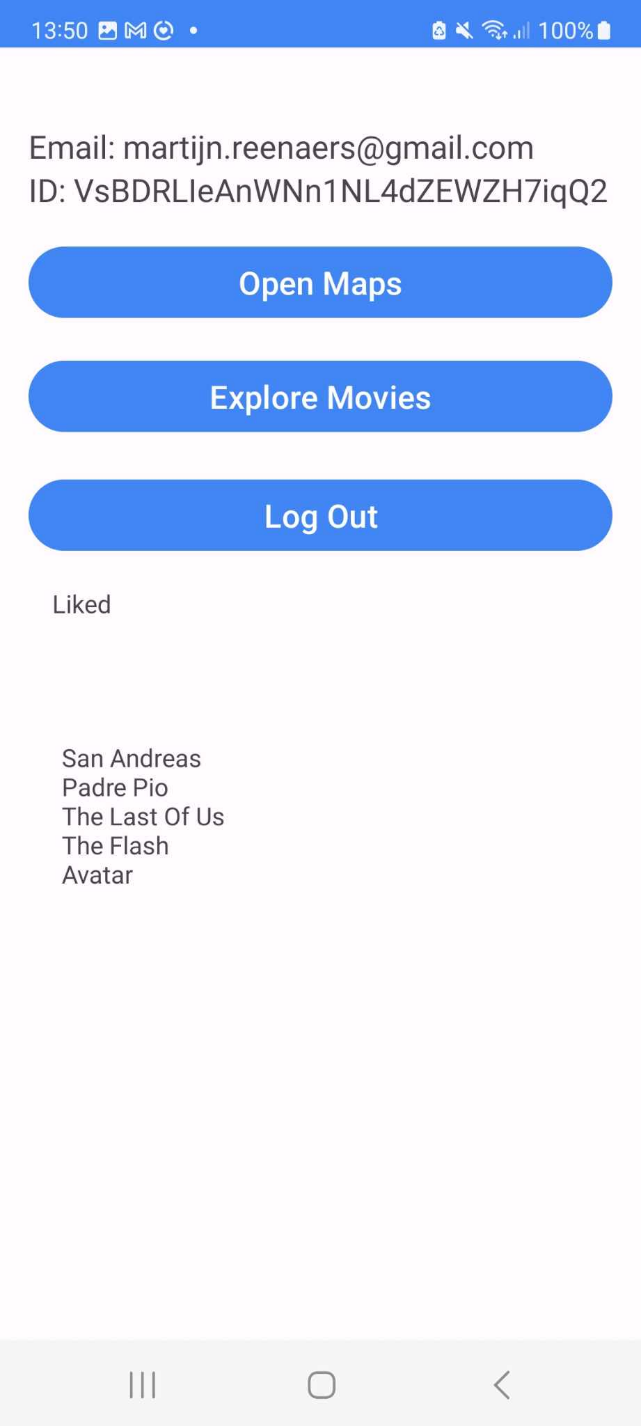
Als je op een film klikt dan ga je naar de detail pagina van die film. Hier kan je de categorie zien en kan je een film leuk vinden.

## Map Page

Op onze map pagina kan je de dichtstbijzijnde cinema vinden. Alle cinema’s staan in de database.

## Geen beschrijving beschikbaar.Geen beschrijving beschikbaar. Profile Page

Op de Profile page kan je je email en Uniek ID wat je krijgt. Je ziet hier ook welke films je leuk vind. Ook kan je hier uitloggen. En ga je terug naar de inlog.



## Structuur

Ik maak gebruik van het MVVM (Model-View-ViewModel) structuur.

De MVVM-Structuur is een ontwerppatroon dat wordt gebruikt om de scheiding in onze applicatie te kunnen ondervinden. Dit bestaat uit onze 3 componenten:

## Model

Het Model vertegenwoordigt de applicatie. Dit bevat klassen of structuren die benodigde gegevens ophalen en verwerken. In dit geval hebben we 3 Klassen in ons Model, Liked, Movie en User.

Afbeelding met tekst, schermopname, Lettertype, Graphics

Automatisch gegenereerde beschrijving

## Fragment

Fragment is verantwoordelijk voor de presentatie van de gebruikersinterface. Het bevat de visuele elementen die de gebruikers te zien krijgen, en waarmee ze kunnen communiceren.

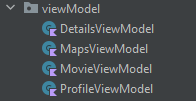
Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

## ViewModel

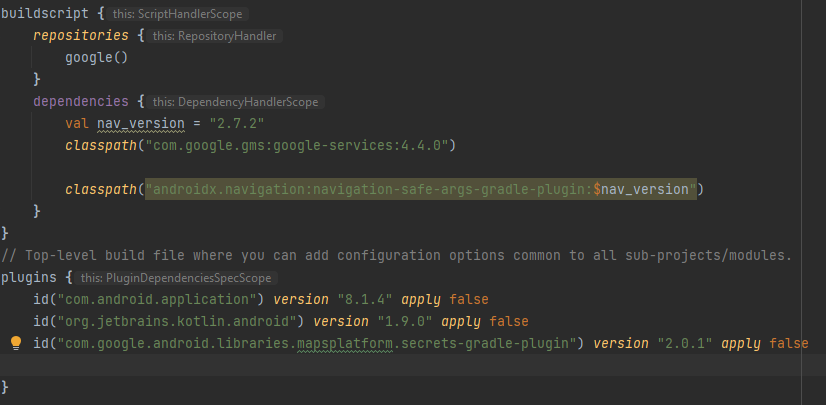
Het ViewModel fungeert als een tussenlaag tussen het Model en het Fragment. Het bevat de logica die nodig is voor het presenteren en verwerken van gegevens in de Fragment.

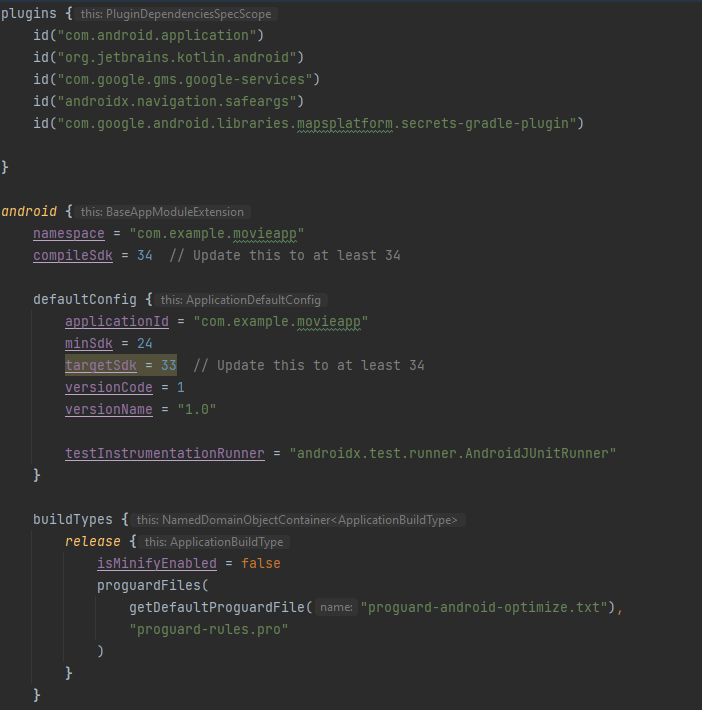


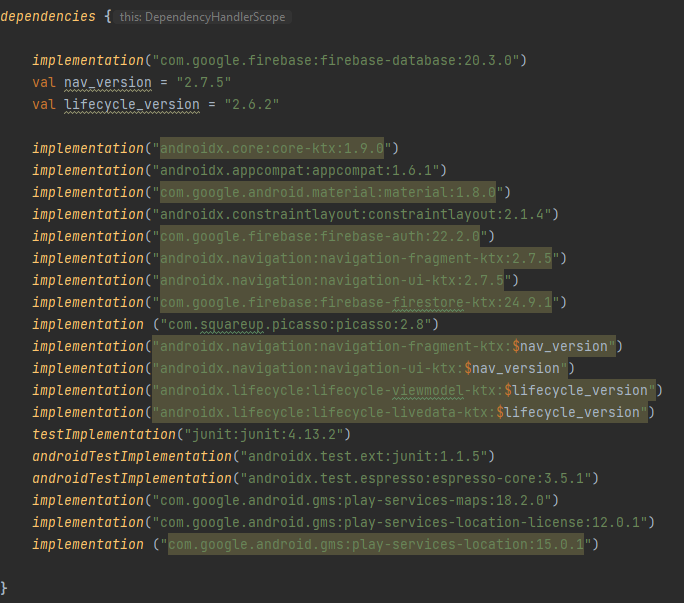
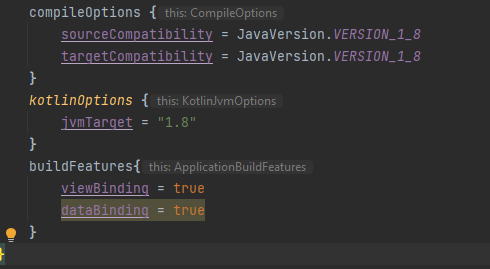
## Gradle

Gradle is een geautomatiseerd toolkit dat wordt gebruikt in softwareontwikkeling. Het fungeert als een krachtige bouwtool en projectbeheerder en wordt vaak gebruikt voor het bouwen, testen, publiceren en implementeren van softwareprojecten.

Je specificeert de afhankelijkheden van je project, zoals externe bibliotheken of andere projecten binnen hetzelfde multi-module project.

Je voegt Gradle-plugins toe om extra functionaliteit aan je build toe te voegen. Dit kunnen standaardplugins zijn die door Gradle worden geleverd of aangepaste plugins die door jou of anderen zijn gemaakt.





## Firebase Database

Voor het ophalen van de Movie Data en Map Data maken we gebruik van databases en externe API's. Voor de films en maps maak ik gebruik van de Firebase Database.

Firebase is een platform voor het ontwikkelen van mobiele webapplicaties. Firebase is van google en biedt ontwikkelaars hulpmiddelen en services die kunnen worden geïntegreerd in deze applicaties.

Firebase heeft verschillende functies, waaronder:

Realtime database: Een realtime, cloudgebaseerde database die synchronisatie in realtime mogelijk maakt tussen verschillende apparaten en gebruikers. Hiermee kunnen ontwikkelaars gegevens opslaan en bijwerken.

Firebase Firestore: is een schaalbare, cloudgebaseerde NoSQL-database die wordt aangeboden als onderdeel van het Firebase-platform van Google. Het biedt een flexibele, op documenten gebaseerde gegevensstructuur, wat betekent dat je gegevens kunt opslaan in de vorm van verzamelingen (analoog aan tabellen in relationele databases) en documenten binnen die verzamelingen (analoog aan rijen in relationele databases). Firestore is ontworpen om gemakkelijk schaalbaar te zijn en real-time synchronisatie te bieden tussen apparaten.

Afbeelding met tekst, Lettertype, Rechthoek, schermopname

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, diagram, lijn

Automatisch gegenereerde beschrijving

Als we naar de Realtime Database gaan zien we een Json bestand wat we gebruiken in onze movie app. De data is Movies en Cinemas

**Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving**

In de Firestore zetten we de gebruikers en kijken we op Id welke gebruiker een film leuk vind.

Afbeelding met tekst, Lettertype, software, schermopname

Automatisch gegenereerde beschrijving

**Afbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijving**

## Authenticatie Firebase

E-mail/Wachtwoord-authenticatie:

Gebruikers kunnen zich registreren met hun e-mailadres en een wachtwoord. Firebase zorgt voor de beveiliging van opgeslagen wachtwoorden en biedt functies zoals wachtwoordreset-e-mails.

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

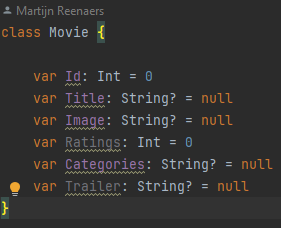
## Technische Werking

## Model

Ik gebruik 3 Model klassen, Liked, Movie, User

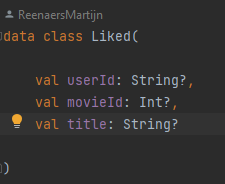
## Movie

In Movie klassen maak ik gebruik van een Id, elke film heeft een eigen uniek Id, dan hebben we de titel, foto rating, categorie en trailer van de films.



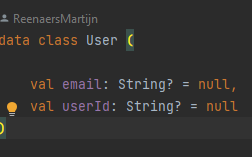
## Liked

Bij Liked gaan we een movieId koppelen aan een userId. Ook geven we een titel mee voor te zien welke film het is.

****

## User

Voor op de profielpagina te kunnen zien welke gebruiker ingelogd is. Geven we de userId mee en het emailadres.

****

## SignUpFragment

Bij SignUpFragment kan je je registreren en inloggen.

package com.example.movieapp  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.Toast  
import androidx.fragment.app.Fragment  
import androidx.navigation.Navigation  
import com.example.movieapp.databinding.FragmentSignUpBinding  
import com.google.firebase.auth.FirebaseAuth  
import com.google.firebase.firestore.FirebaseFirestore  
import com.google.firebase.firestore.FieldValue

De bovenstaande imports zijn nodig om de benodigde klassen en modules te gebruiken in dit bestand. Onder andere worden Firebase-authenticatie en Firestore-database geïmporteerd.

class SignUpFragment : Fragment() {  
 private var \_binding: FragmentSignUpBinding? = null  
 private val binding get() = \_binding!!  
 private lateinit var firebaseAuth: FirebaseAuth  
 private val firestore = FirebaseFirestore.getInstance()

SignUpFragment is een subklasse van Fragment.

binding wordt gebruikt om de layout van het fragment te binden.

firebaseAuth is een instantie van FirebaseAuth voor authenticatie.

firestore is een instantie van FirebaseFirestore voor de Firebase Firestore-database.

override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
): View? {  
 \_binding = FragmentSignUpBinding.inflate(inflater, container, false)  
 val view = binding.*root* firebaseAuth = FirebaseAuth.getInstance()  
  
 binding.SignUpTextS.setOnClickListener **{** val action = SignUpFragmentDirections.actionSignUpFragmentToLogInFragment()  
 Navigation.findNavController(requireView()).navigate(action)  
 **}**

onCreateView wordt aangeroepen wanneer de gebruikersinterface voor het fragment moet worden gemaakt.

De SignUpTextS voert een navigatie-actie uit naar het inlogfragment.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

De SignInButton voert de logica uit voor het aanmelden van gebruikers met Firebase-authenticatie.

Id1 is de userId dat in de database word gezet.

Als de aanmelding succesvol is, worden de gebruikersgegevens opgeslagen in de Firestore-database.

Als er een fout optreedt tijdens het aanmelden, wordt een foutmelding weergegeven.

Er wordt gecontroleerd of de ingevoerde wachtwoorden overeenkomen.

Er wordt gecontroleerd of de invoervelden leeg zijn.

Dit fragment behandelt het aanmeldingsproces van gebruikers, inclusief Firebase-authenticatie en het opslaan van gebruikersgegevens in Firebase Firestore. Eventuele fouten of waarschuwingen worden weergegeven met behulp van Toast-meldingen.

## LogInFragment

Bij LogInFragment kan je inloggen.

class LogInFragment : Fragment() {  
 private var \_binding: FragmentLogInBinding? = null  
 private val binding get() = \_binding!!  
 private lateinit var firebaseAuth: FirebaseAuth

\_binding wordt gebruikt om de layout van het fragment te binden.

binding is een eigenschap die \_binding teruggeeft en wordt gebruikt om toegang te krijgen tot de views in het fragment. Die we later returnen

firebaseAuth is een instantie van FirebaseAuth voor authenticatie.

override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
): View? {  
 \_binding = FragmentLogInBinding.inflate(inflater, container, false)  
 val view = binding.*root* firebaseAuth = FirebaseAuth.getInstance()

onCreateView wordt aangeroepen wanneer de gebruikersinterface voor het fragment moet worden gemaakt.

binding.SignUpText.setOnClickListener **{** val action = LogInFragmentDirections.actionLogInFragmentToSignUpFragment()  
 Navigation.findNavController(**it**).navigate(action)  
**}**

De SignUpText voert een navigatie-actie uit naar het aanmeldfragment.

binding.SignInButton.setOnClickListener **{** val email = binding.emailEt.*text*.*toString*()  
 val pass = binding.passET.*text*.*toString*()  
  
 if (email.*isNotEmpty*() && pass.*isNotEmpty*()) {  
 firebaseAuth.signInWithEmailAndPassword(email, pass).addOnCompleteListener **{** if (**it**.*isSuccessful*) {  
 var user = firebaseAuth.*currentUser* val action = LogInFragmentDirections.actionLogInFragmentToMovieFragment(user?.*uid*.*toString*())  
 Navigation.findNavController(requireView()).navigate(action)  
  
 } else {  
 Toast.makeText(*activity*, **it**.*exception*.*toString*(), Toast.*LENGTH\_SHORT*).show()  
 }  
 **}** } else {  
 Toast.makeText(*activity*, "Empty Fields Are not Allowed !!", Toast.*LENGTH\_SHORT*).show()  
 }  
**}**return view

Als het inloggen succesvol is, wordt de gebruiker doorgestuurd naar het filmscherm.

Als er een fout optreedt tijdens het inloggen, wordt een foutmelding weergegeven.

Dit fragment behandelt het inlogproces van gebruikers, inclusief Firebase-authenticatie. Het controleert op lege velden en navigeert de gebruiker naar het filmscherm na succesvol inloggen. Eventuele fouten of waarschuwingen worden weergegeven met behulp van Toast-meldingen.

## MovieFragment

private lateinit var movieViewModel: MovieViewModel  
private lateinit var movieAdapter: MovieAdapter

movieViewModel: Een instantie van MovieViewModel, verantwoordelijk voor het beheren van filmgegevens.

movieAdapter: Een adapter om filmgegevens weer te geven in een RecyclerView.

override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
): View? {  
  
 \_binding = FragmentMovieBinding.inflate(inflater, container, false)  
 val view = binding.*root* movieViewModel = ViewModelProvider(this).get(MovieViewModel::class.*java*)  
 movieAdapter = MovieAdapter()  
  
 val args = MovieFragmentArgs.fromBundle(requireArguments())  
 val userId = args.*id* val recyclerView: RecyclerView = view.findViewById(R.id.*recyclerViewMovies*)  
 recyclerView.*layoutManager* = LinearLayoutManager(requireContext())  
 recyclerView.*adapter* = movieAdapter  
  
 movieViewModel.getMovies().observe(  
 *viewLifecycleOwner*,  
 ) **{** movies **->** movieAdapter.submitList(movies)  
 **}** movieAdapter.setOnItemClickListener **{** movie **->** val action = MovieFragmentDirections.actionMovieFragmentToMovieDetailFragment(  
 movie.Id,  
 userId  
  
 )  
 *findNavController*().navigate(action)  
 **}**

Initialisatie van movieViewModel en movieAdapter.

Opzet van de RecyclerView om films weer te geven.

Observer om veranderingen in de filmgegevens te detecteren en de adapter bij te werken.

Een TextWatcher om de zoekfunctie te activeren wanneer de tekst in de zoekbalk verandert.

Navigatie naar het profielfragment en kaartfragment.

val searchBar: EditText = view.findViewById(R.id.*searchBar*)  
searchBar.addTextChangedListener(object : TextWatcher {  
 override fun afterTextChanged(s: Editable?) {  
 // No action needed  
 }  
  
 override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {  
 // No action needed  
 }  
  
 override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {  
 movieViewModel.searchMovies(s.*toString*())  
 }  
})

Voor het zoeken naar een movie movieViewModel.searchMovies(s.*toString*()), die we uit het viewmodel halen.

## MovieAdapter

Een aangepaste adapter voor het weergeven van films in de RecyclerView.

Afbeelding met tekst, schermopname, Lettertype, software

Automatisch gegenereerde beschrijvingEen ViewHolder met een methode om filmgegevens aan views toe te wijzen.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Een DiffUtil-implementatie om de items in de lijst te vergelijken en te bepalen of ze hetzelfde zijn.

## MovieViewModel

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Een verwijzing naar de Firebase Realtime Database voor het ophalen van filmgegevens.

Livedata om filmgegevens aan de fragmenten door te geven.

Een lijst voor het bewaren van de oorspronkelijke filmgegevens.

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving

Ophalen van films uit de Firebase Database.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Een methode om films te filteren op basis van zoekopdrachten.

## DetailsViewModel

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

databaseReference: Verwijzing naar het Firebase Realtime Database-knooppunt "Movies" voor het ophalen van filmgegevens.

moviesLiveData: Een MutableLiveData die een lijst van films bevat.

originalMoviesList: Een lijst voor het bewaren van de oorspronkelijke lijst met films.

Afbeelding met tekst, schermopname, software, Lettertype

Automatisch gegenereerde beschrijving

Deze methode controleert of een gebruiker een film al heeft geliked. Als dat niet het geval is, wordt de like toegevoegd aan Firestore.

Afbeelding met tekst, schermopname, software

Automatisch gegenereerde beschrijving

Deze methode haalt alle films op uit de Firebase Realtime Database en roept de meegegeven callback aan met de geselecteerde film op basis van het gegeven movieId.

## MovieDetailFragment

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

\_binding: Een optioneel gebonden klasse voor het efficiënt omgaan met ViewBinding.

movieDetailsViewModel: Instantie van DetailsViewModel.

Afbeelding met tekst, schermopname, software, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Deze methode initialiseert de UI en haalt de filmgegevens op met behulp van DetailsViewModel.

## MapsFragment

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

mapsViewModel: Een instantie van MapsViewModel om kaartgerelateerde gegevens te beheren.

fusedLocationClient: Een instantie van FusedLocationProviderClient om de locatie van de gebruiker op te halen.

googleMap: Een variabele om de Google Map te bevatten.

Afbeelding met tekst, schermopname, software, Lettertype

Automatisch gegenereerde beschrijving

OnMapReadyCallback om de kaart en locaties te initialiseren van de bioscopen.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Voor de locatie van de gebruiker.

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving

Initialisatie van mapsViewModel, ophalen van bioscooplocaties, en instellen van de kaart.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Knop om dichtstbijzijnde bioscoop te vinden en te markeren op de kaart.

Afbeelding met tekst, schermopname, Lettertype, software

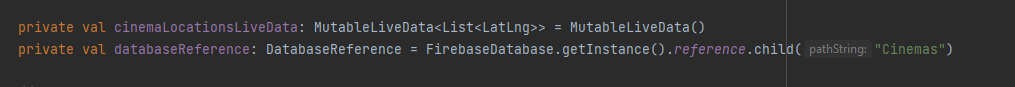
Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, software, Lettertype

Automatisch gegenereerde beschrijving

Methoden om de laatst bekende locatie van de gebruiker op te halen en de dichtstbijzijnde bioscoop te vinden.

## MapsViewModel



cinemaLocationsLiveData: Een MutableLiveData die een lijst van LatLng (coördinaten) bevat voor bioscooplocaties.

databaseReference: Een verwijzing naar de Firebase Realtime Database, specifiek naar het knooppunt "Cinemas".

Afbeelding met tekst, schermopname, Lettertype, software

Automatisch gegenereerde beschrijving

Deze methode haalt bioscooplocaties op uit de Firebase Realtime Database en actualiseert cinemaLocationsLiveData met de lijst van LatLng-coördinaten.

Afbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijving

Deze methode retourneert cinemaLocationsLiveData, waardoor andere delen van de app live-updates kunnen ontvangen van de lijst met bioscooplocaties.

Afbeelding met tekst, schermopname, software

Automatisch gegenereerde beschrijving

Deze methode berekent de dichtstbijzijnde bioscooplocatie ten opzichte van een opgegeven gebruikerslocatie (userLocation). Het resultaat wordt als LiveData verpakt en kan worden waargenomen voor live-updates.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Een private hulpmethode om de afstand tussen twee LatLng-punten te berekenen met behulp afstand.

## ProfileViewModel

Afbeelding met tekst, schermopname, Lettertype, software

Automatisch gegenereerde beschrijving

Deze methode haalt gebruikersgegevens op uit Firestore voor de opgegeven userId en roept de meegegeven callback aan met een User-object als de gebruiker bestaat.

Afbeelding met tekst, schermopname, software

Automatisch gegenereerde beschrijving

Deze methode haalt films op die zijn geliked door de gebruiker met de opgegeven userId en roept de meegegeven callback aan met een lijst van Liked-objecten.

## ProfileFragment

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

profileViewModel: Instantie van ProfileViewModel.

firebaseAuth: Instantie van FirebaseAuth voor authenticatie.

\_binding: voor ViewBinding.

Afbeelding met tekst, schermopname, software, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype, software

Automatisch gegenereerde beschrijving

De methoden haalt gebruikersgegevens en gelikede films op met behulp van ProfileViewModel en zet het in de UI

## Nav.xml

Afbeelding met tekst, schermopname, diagram, ontwerp

Automatisch gegenereerde beschrijving

Deze nav is een visuele representatie van hoe gebruikers kunnen navigeren door verschillende schermen in je app. Elke actie vertegenwoordigt een mogelijke gebruikersinteractie of systeemgebeurtenis die een overgang tussen bestemmingen triggert.

Argumenten:

Wanneer je van de ene bestemming naar de andere gaat, wil je mogelijk gegevens doorgeven, zoals een gebruikers-ID, een geselecteerde film-ID, of andere relevante informatie door wilt geven naar een ander fragment.

Afbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijvingAfbeelding met schermopname, tekst, lijn, diagram

Automatisch gegenereerde beschrijving