Movies Application

Να γίνει ανάπτυξη ενός Android Application σε Kotlin, με χρήση του **TMDB API**. Η ανάπτυξη του User Interface θα πρέπει να γίνει με Jetpack Compose. Η εφαρμογή θα πρέπει να να είναι λειτουργική και δοκιμασμένη σε **mobile συσκευές** (όχι tablet) και να υποστηρίζει από Android 8 (minSDK 26), ενώ θα πρέπει να είναι διαθέσιμη και για Android 14 (targetSDK 34).

Για τη σύνδεση με το **TMDB API** θα δημιουργήσετε ένα λογαριασμό και θα εκδώσετε ένα API Key το οποίο θα χρησιμοποιήσετε ως **Bearer Token** για τις κλήσεις προς το API (με τη βοήθεια του **Retrofit**). Αναλυτικές πληροφορίες θα βρείτε εδώ.

TMDB: https://www.themoviedb.org/documentation/api?language=en-GB

Οθόνες και λειτουργικότητα

Το App πρέπει να έχει τις παρακάτω βασικές οθόνες, με τα απαραίτητα features:

A. Οθόνη Dashboard

Πρόκειται για τη βασική οθόνη την εφαρμογής. Θα περιέχει 3 sections με κατηγορίες ταινιών οι οποίες θα προβάλλονται σε οριζόντιες λίστες.

Κατηγορίες:

- Top 10 movies
- Popular movies
- Κατηγορία της επιλογής σας (π.χ Most recent comedies)

Κάθε section θα αποτελείται από:

1. Ένα section header **(Row)** το οποίο θα έχει στα αριστερά ένα **Text** με τον **τίτλο της κατηγορίας** και στα δεξιά ένα **TextButton "See All"**
2. Κάτω από το Section Header θα υπάρχει μια **οριζόντια λίστα με τις 10 πρώτες ταινίες (Lazy Row)**
3. Το Item κάθε ταινίας της οριζόντιας λίστας, θα αποτελείται από το banner (εικόνα) της ταινίας και τον τίτλο της. Το design θα είναι δικής σας επιλογής.
4. Όταν ο χρήστης κάνει click στο κουμπί **"See All"** θα πρέπει να οδηγηθεί στην οθόνη Movies Category (οθόνη B) ενώ όταν κάνει click σε κάποια από τις ταινίες (item της λίστας) να οδηγηθεί στην οθόνη Movie Details (οθόνη C).

Tip! Θα πρέπει να κρατήσετε το **id** της κατηγορίας που επιλέγει ο χρήστης όταν επιλέξει **"See All"** για να το περάσετε (ως navigation argument) στην οθόνη Movies Category με σκοπό να φορτώσετε όλες τις ταινίας της κατηγορίας με το συγκεκριμένο **id**. Αντίστοιχα είναι απαραίτητο να κρατήσετε το **id** της κάθε ταινίας που επιλέγει ο χρήστης για να το περάσετε (ως navigation argument) στην οθόνη Movie Details ώστε να πάρετε τα details της ταινίας που επέλεξε ο χρήστης και να τα απεικονίσετε.

B. Οθόνη Movies Category

Η οθόνη αυτή θα εμφανίζεται όταν ο χρήστης επιλέγει το **"See All"** κουμπί και θα πρέπει να έχει ένα υποτυπώδες navigation ώστε ο χρήστης να μπορεί να επιστρέψει στην οθόνη Dashboard.

Περιεχόμενα οθόνης:

1. Ένα **Toolbar** το οποίο θα περιέχει ένα **Image/Icon** με ένα back arrow το οποίο θα σε οδηγεί στην προηγούμενη οθόνη και ένα **Text** με τον **τίτλο της επιλεγμένης κατηγορίας** ταινιών.
2. Μια κάθετη λίστα **(Lazy Column)** με τις ταινίες που ανήκουν σε αυτή την κατηγορία.
3. Ο χρήστης θα μπορεί να πατήσει πάνω στο item της λίστας και να οδηγηθεί στην οθόνη Movie Details (οθόνη C).

Tip! Το UI item της κάθε ταινίας σε μια κάθετη λίστα, μπορεί να περιέχει το banner (πχ οριζόντιο) και τον τίτλο της ταινίας.

Bonus 1! Να μπει load more (pagination) μηχανισμός, για να φορτώνεται η επόμενη σελίδα ταινιών κάθε 20 ταινίες.

C. Οθόνη Movie Details

Περιεχόμενα οθόνης:

1. Ένα **Toolbar** το οποίο θα περιέχει ένα **Image/Icon** με ένα back arrow το οποίο θα σε στην προηγούμενη οθόνη, είτε αυτή είναι η οθόνη Dashboard είτε η οθόνη Movie Details καθώς και ένα **Text** με τον **τίτλο της επιλεγμένης ταινίας**.
2. Στην οθόνη θα πρέπει να εμφανίζονται κατα και κάθετα η εικόνα της ταινίας (μπορεί να υλοποιηθεί είτε με **Colum** είτε με **LazyColumn)**

Bonus 2! Να μπει ένα section (τίτλος με οριζόντια λίστα) που θα εμφανίζει το Cast της ταινίας

Tip! Καθώς υπάρχει πληθώρα σε μεγέθη οθονών, προσπαθήστε να σχεδιάσετε τα UI σας με τη χρήση δυναμικών sizes (στους modifiers) ως προς το width της οθόνης (πχ fillMaxWidth, aspectRatio) και αποφύγετε τα στατικά dp. Αυτό θα έχει ως αποτέλεσμα, το UI σας να φαίνεται ομοιόμορφα στους χρήστες ανεξάρτητα με τις οθόνες που έχουν.

UI State με Sealed Class

Για την καλύτερη διαχείριση του UI αλλά και για να βελτιώσουμε το User Experience δημιουργούμε τα παρακάτω states με τη χρήση των sealed classes που αναφέραμε στην παρουσίαση:

- **Loading**, ο χρήστης θα πρέπει να βλέπει έναν Loader όταν γίνεται το βασικό request της οθόνης.
- **Data**, ο χρήστης θα βλέπει τα προβλεπόμενα δεδομένα στην οθόνη.
- **Error**, ο χρήστης θα βλέπει στην οθόνη ένα γενικό μήνυμα λάθους όταν το βασικό HTTP request της οθόνης έχει αποτύχει (HTTP Code != 200)
- **Empty**, σε περίπτωση που η οθόνη αποτελείται από μια λίστα (πχ Movies Category) η οποία είναι άδεια, ο χρήστης θα πρέπει να δει κάτω από τον τίτλο ένα μήνυμα οτι δεν βρέθηκε περιεχόμενο.

Tip! Σε περίπτωση που η οθόνη αποτελείται από παραπάνω από μια λίστες (όπως η Οθόνη Dashboard) μπορείτε να **μην** κάνετε χρήση του **Sealed Class** αλλά να εμφανίσετε τα sections με ένα ενημερωτικό μήνυμα “Δεν βρέθηκε περιεχόμενο”.

Tip! Σε περίπτωση που η οθόνη αποτελείται από πολλαπλά requests και κάποια αποτύχουν, αν είναι εφικτό μπορούν να εμφανιστούν τα δεδομένα από τα επιτυχημένα request. Σε περίπτωση όμως που το request το οποιο απέτυχε ήταν απαραίτητο κλειδί για τα επόμενα, τότε θα ο χρήστης θα πρέπει να βλέπει το Error State.

Σύνδεση Coroutines, Room & Retrofit

Ένας από τους βασικότερους συντελεστές για το Android Development όπως είπαμε στην πρώτη παρουσίαση είναι τα Coroutines τα οποία θα πρέπει να συνδυάσετε με τη χρήση του Room DB και του Retrofit για τα API Calls.

Προτείνεται η χρήση των παρακάτω βημάτων κατά σειρά. Αρχικά αφού συνδεθείτε με το TMDB και καταφέρετε να πάρετε τα πρώτα δεδομένα από το API θα πρέπει να μελετήσετε την JSON απάντηση και να σχεδιάσετε τη δομή του ROOM Database. Θα χρειαστεί μόνο ένας πίνακας για να αποθηκεύσετε τις ταινίες.

Στη συνέχεια θα πρέπει να δημιουργήσετε ένα Use Case (**FetchMoviesByCategoryUseCase**) το οποίο δέχεται ως παράμετρο την ζητούμενη κατηγορία, θα επικοινωνεί με το API και θα αποθηκεύει τις ταινίες στο Room DB. Το συγκεκριμένο Use Case δεν θα πρέπει να επιστρέφει αποτέλεσμα.

Έπειτα θα πρέπει να δημιουργήσετε ένα Use Case (**GetLocalMoviesByCategoryUseCase)** το οποίο δέχεται ως παράμετρο την ζητούμενη κατηγορία, θα επικοινωνεί με το Room DB και θα επιστρέφει ένα **flow** με ταινίες της κατηγορίας που ζητήθηκε.

Στο ViewModel της Dashboard οθόνης θα πρέπει να δημιουργήσετε 3 διαφορετικές **coroutines.** Σε κάθε coroutine θα καλέσετε από μια φορά για κάθε section το **GetLocalMoviesByCategoryUseCase, θα κάνετε collect to flow** και θα απεικονίσετε τα δεδομένα. Στη συνέχεια θα δημιουργήσετε 3 νέα coroutines και θα καλέσετε από μία φορά για κάθε section το **FetchMoviesByCategoryUseCase** ώστε να αποθηκεύσετε τις νέες ταινίες που σας έφερε το API. **Προσοχή στα διπλότυπα!!**

Με την λογική αυτή μετά από το πρώτο άνοιγμα της εφαρμογής (αφού δηλαδή έχουν αποθηκευτεί κάποιες ταινίες ανα κατηγορία) ο χρήστης να μπορεί να δει το συγκεκριμένο περιεχόμενο ακόμα και χωρίς σύνδεση στο internet (wifi, data). Τέλος, η χρήση του **flow** στο Room DB μας επιτρέπει να έχουμε πάντα up to date data καθώς η αποθήκευση των νέων ταινιών γίνεται αυτόματα collect από το View Model μας.

Για την οθόνη MoviesCategory και στην περίπτωση που θέλετε να υλοποιήσετε το Bonus Task με το pagination θα πρέπει να κάνετε κλήσεις απευθείας στο TMDB API. Το ίδιο και για την οθόνη MoviesDetails.

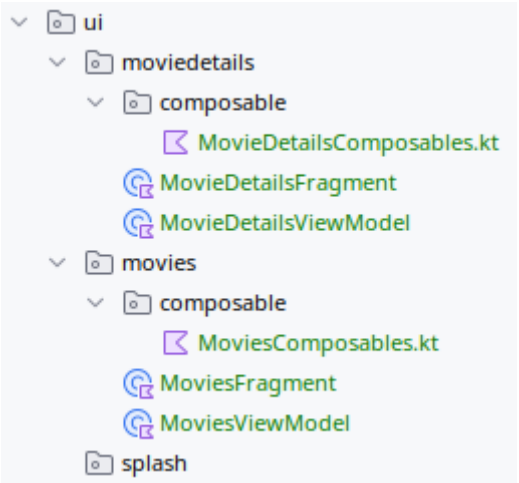
Βασικά components αρχιτεκτονικής

1. Ένα Activity που θα περιέχει ένα Fragment Navigation
2. Ένα Fragment για κάθε ζητούμενη οθόνη
3. Ένα View Model για κάθε Fragment
4. Jetpack Compose για το UI του κάθε Fragment
5. Hilt για το Dependency Injection (υπάρχει έτοιμο configuration στο base project)
6. Χρήση του Clean Architecture όσο είναι εφικτό (use cases, repository, datasource, api) με κατανοητά ονόματα και σωστή τοποθέτησή τους στους αντίστοιχους φακέλους.

Mandatory Third Party Libraries

1. Retrofit - για request στο TMDB API
2. Moshi - για μετατροπή του JSON Response σε Kotlin Object
3. Coil - για φόρτωση εικόνων από url

Tip! Η οργάνωση των αρχείων και των φακέλων θα πρέπει να ακολουθεί την ίδια δομή με το base project. Τα πακέτα και οι κλάσεις πρέπει να βρίσκονται στα σωστά σημεία, με λογικά ονόματα. Προσπαθήστε να δημιουργήσετε πακέτα για να διαχωρίζονται οι κλάσεις με βάση τις οθόνες, έτσι ώστε να είναι εύκολο να διαβαστούν και να κατανοηθούν απο όλα τα μέλη της ομάδας.



Βασικά βήματα υλοποίηση της εργασίας

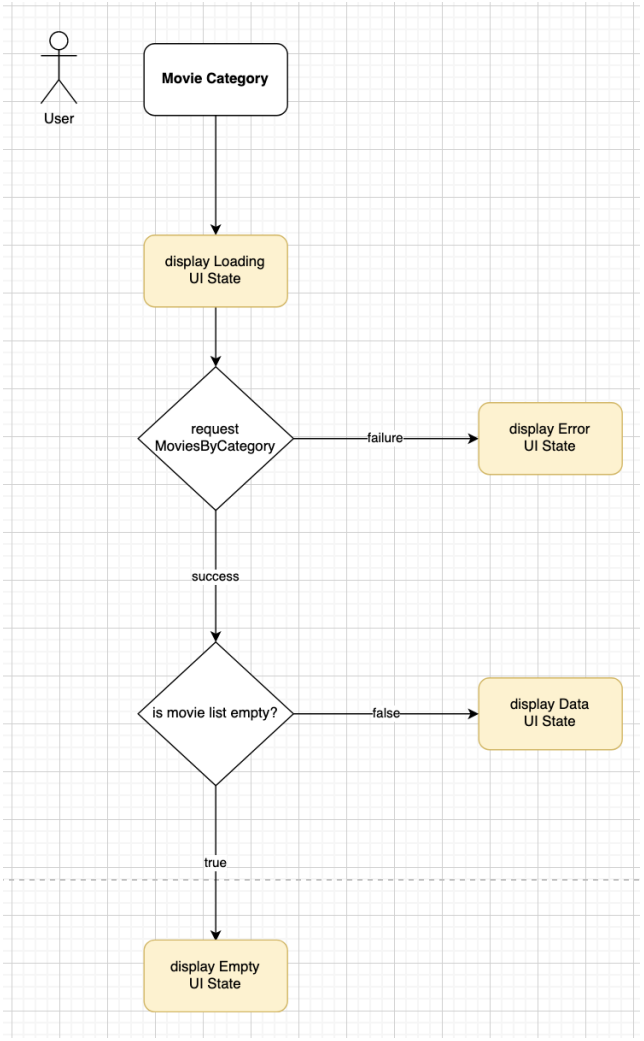
A. Analysis

1. Ανάλυση και καταγραφή των requirements της εφαρμογής (επιγραμματικά μια λίστα)
2. Έρευνα και εντοπισμός παρόμοιων εφαρμογών για κινητά (π.χ. IMDB, netflix κτλ) για να πάρετε ιδέες για το UI και την εμπειρία χρήστη (UX)

B. Design

1. Σχεδιασμός mockups/wireframes σε κάποιο free πρόγραμμα (π.χ. Draw.io, Figma trial, χαρτί και φωτογραφίες). Δεν χρειάζεται να δώσετε έμφαση στη λεπτομέρεια. Θα πρέπει να είναι απλά και να απεικονίζουν τη βασική δομή του UI.
2. Σχεδιασμός απλού UML διαγράμματος ροής (Draw.io) για την κάθε οθόνη που θα υλοποιηθεί.

   πχ θα περιγράφει την είσοδο του χρήστη στην εφαρμογή και τα requests προς το TMDB API (λίστες από τις ταινίες ανα κατηγορία).



C. Development

Ανάπτυξη των features και οθονών που έχουν περιγραφεί πιο πάνω. Τα **Bonus features** είναι προαιρετικά.

Παραδοτέο

Στο τελικό παραδοτέο θα πρέπει να υπάρχουν:

1. το Android Project σε ένα zip
2. την λίστα με τα requirements σε ένα doc file (A.)
3. το σχεδιασμό (B.) μια εικόνα του διαγράμματος (UML) για κάθε οθόνη στο ίδιο doc file.
4. Το APK της εφαρμογής
5. Ένα περιεκτικό video (screen record) της εφαρμογής, που να φαίνονται οι βασικές οθόνες και τα ζητούμενα features (όσα από αυτά μπορούν να φανούν)

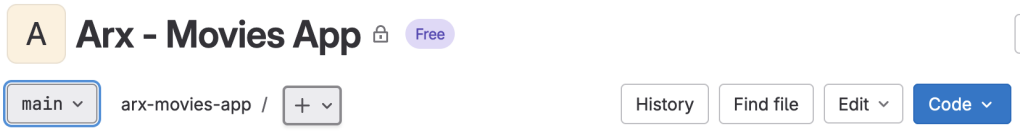
όπως αυτά περιγράφονται πιο πάνω στο document.

Base Project

Για την υλοποίηση της εφαρμογής θα **πρέπει** να κατεβάσετε το Base Project και να ξεκινήσετε την υλοποίηση σας πάνω σε αυτό. Το project θα σας βοηθήσει να ακολουθήσετε αρχιτεκτονική και μεθοδολογίες καθώς και ονοματολογίες κλάσεων.

Θα χρειαστεί να μας στείλετε τα **email** από τα άτομα κάθε ομάδας για να σας κάνουμε invite στο gitlab project έτσι ώστε να μπορέσετε να το κατεβάσετε.
Gitlab base project: **https://gitlab.com/ihu-workspace/arx-movies-app**

Αφού κατεβάσετε το zip του main branch μπορείτε να το ανεβάσετε σε ένα δικό σας repository, στο οποίο θα δουλέψουν όλα τα μέλη της ομάδας.

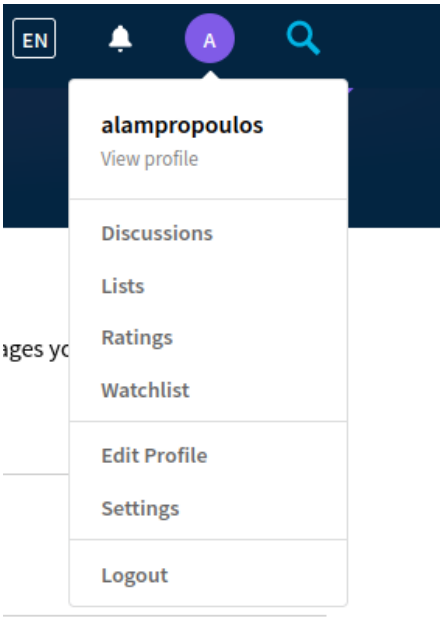


Tip! Σε μια ομάδα είναι απαραίτητη η χρήση του Git για την καλύτερη οργάνωση, συνέπεια της ομάδας, αλλά και την αποφυγεί λαθών και “χάσιμο” του κώδικα μας. Τα commit messages να είναι σύντομα και περιεκτικά. Τα commits θα πρέπει να γίνονται συχνά και να αφορούν συγκεκριμένο μέρος ενός feature. Πριν κάνουμε commit πρέπει πάντα να κάνουμε έλεγχο οτι το application κάνει compile και τρέχει.

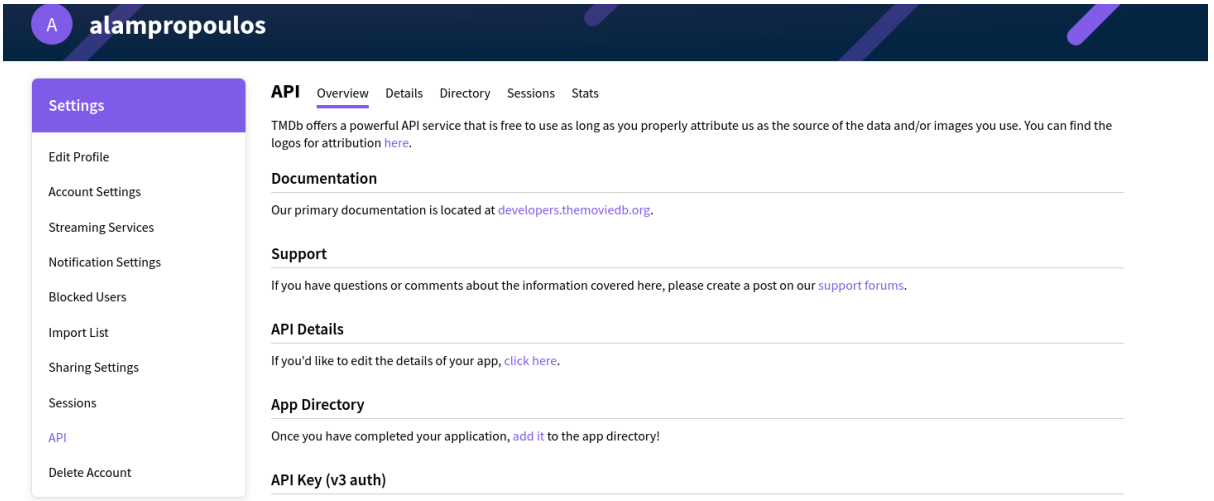
Οδηγίες για το TMDB

Για την υλοποίηση πρέπει να κάνετε λογαριασμό στο TMDB και έπειτα να ζητήσετε ένα API key.

1. Μετά την εγγραφή, κάνετε σύνδεση και πατήστε πάνω δεξιά στο icon με το account σας. Επιλέξτε “Settings”



2. Επιλέξτε απο το μενου αριστερά το “API” και εκεί θα κάνετε request API Key



3. Στις πληροφορίες για την εφαρμογή συμπληρώστε κάτι τέτοιο