COMPTE RENDU DE PROJET

Application Mobile Flutter

« Netflim »

Technologie	Flutter (Dart)
API utilisée	TMDB (The Movie Database)
Architecture	MVC (Model – View – Controller)
Type de projet	Application mobile de streaming de films

1. Objectif du projet

Ce TP avait pour objectif de concevoir une application mobile complète avec Flutter, permettant d'afficher et d'explorer des films à partir de l'API externe TMDB. Le projet visait quatre grands axes :

- Mettre en place une architecture propre basée sur le modèle MVC
- Exploiter une API REST pour récupérer et afficher des données dynamiques
- Développer une interface utilisateur moderne et interactive
- Implémenter des fonctionnalités avancées telles que la recherche, les favoris et une liste personnalisée

2. Architecture et organisation du projet

Le projet repose sur le modèle MVC, qui assure une séparation claire des responsabilités entre les couches de l'application.

2.1 Modèle (Model)

- movie.dart : structure de données représentant un film
- user_preferences.dart : gestion des préférences utilisateur (thème, couleurs)

2.2 Vue (View)

Les écrans de l'application sont :

- home_screen.dart – Page d'accueil
- movie_screen.dart – Détail d'un film
- popular_movies_screen.dart – Liste des films populaires
- youtube_video_screen.dart – Lecteur de bande-annonce
- movie_actions_menu.dart – Composant réutilisable pour les actions sur un film

2.3 Contrôleur (Controller)

- api.dart – Définition des endpoints API
- api_service.dart – Gestion des appels HTTP via la bibliothèque Dio

3. Fonctionnalités développées

3.1 Fiche film enrichie

La page de détail d'un film a été considérablement améliorée. Elle affiche désormais les informations suivantes :

- Slogan (tagline)
- Date de sortie
- Genres
- Durée du film
- Note (score)
- Synopsis complet

Sur le plan de l'interface, les couleurs sont gérées de manière centralisée via UserPreferences, les images bénéficient d'un système de fallback en cas d'erreur, et les informations sont organisées en sections lisibles pour une meilleure clarté visuelle.

3.2 Lecture de bande-annonce

Une fonctionnalité de lecture de trailer a été intégrée :

- Appel à l'endpoint /movie/{id}/videos de l'API TMDB
- Extraction de l'identifiant YouTube de la vidéo
- Ajout d'un bouton « Regarder la bande-annonce »
- Navigation vers l'écran YoutubeVideoScreen dédié

Cette fonctionnalité améliore significativement l'expérience utilisateur en permettant de visionner les bandes-annonces directement dans l'application.

3.3 Recherche dynamique

Une SearchBar personnalisée a été intégrée directement dans l'écran principal, remplaçant l'ancien onglet dédié à la recherche. Le comportement est le suivant :

- Si le champ de recherche est vide : affichage des films populaires
- Si une saisie est détectée : affichage des résultats correspondants en temps réel

La gestion asynchrone des requêtes repose sur les concepts Flutter Future et FutureBuilder, permettant de gérer proprement les états de chargement, de succès et d'erreur.

3.4 Gestion des favoris

Un système de favoris a été mis en place via le design pattern Singleton (classe Favourites) :

- Ajouter un film aux favoris
- Supprimer un favori
- Vérifier si un film est déjà en favori
- Mise à jour dynamique de l'icône ❤️

- Persistance des données via SharedPreferences

Un problème de rafraîchissement de l'interface lors du retour à l'écran précédent a été résolu grâce à l'ajout d'un callback onGoBack combiné à un appel setState().

3.5 Liste « À regarder »

Une liste personnalisée a été ajoutée en complément des favoris. La navigation utilise un TabBar avec 4 onglets : Accueil, Recherche, Favoris, et À regarder, gérés via un TabController.

La classe Singleton ListToWatch fonctionne de la même manière que les favoris, avec en plus des améliorations UX :

- Message de confirmation lors de l'ajout d'un film
- Alerte si le film est déjà présent dans la liste

4. Fonctionnalités avancées

4.1 Accessibilité – Synthèse vocale (Bonus)

Un système de lecture du synopsis a été intégré grâce à la bibliothèque FlutterTts. Les événements gérés sont : start, complete et error. Cette fonctionnalité rend l'application accessible aux utilisateurs ayant des difficultés de lecture.

4.2 Personnalisation du thème

Un thème global NetflimTheme a été créé, permettant une cohérence visuelle dans toute l'application. Il inclut :

- Des couleurs dynamiques configurables via UserPreferences
- La personnalisation des styles de texte
- Le style des champs de saisie

5. Compétences développées

Domaine	Compétences acquises
Développement mobile	Flutter, widgets, navigation, StatefulWidget, gestion d'état
Backend / API	Consommation d'API REST, requêtes HTTP avec Dio
Architecture	Modèle MVC, organisation modulaire du code
Concepts avancés	Design Pattern Singleton, asynchronisme (Future, FutureBuilder), stockage local (SharedPreferences)

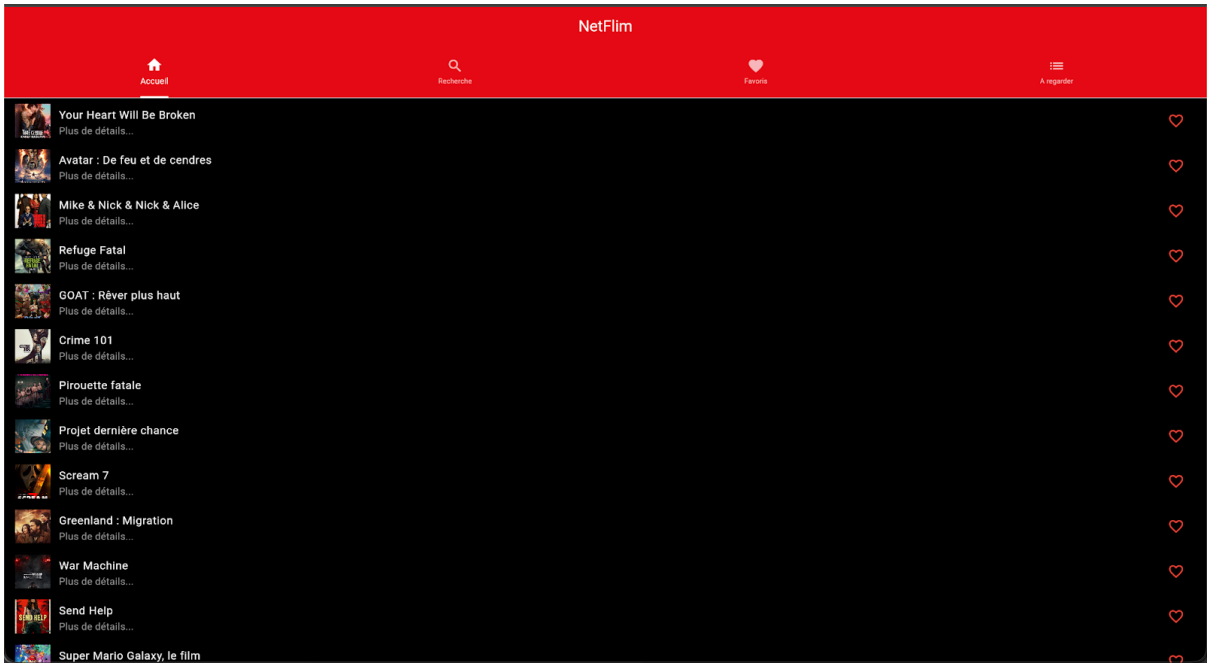
UX / UI	Interfaces dynamiques, gestion des erreurs, feedback utilisateur

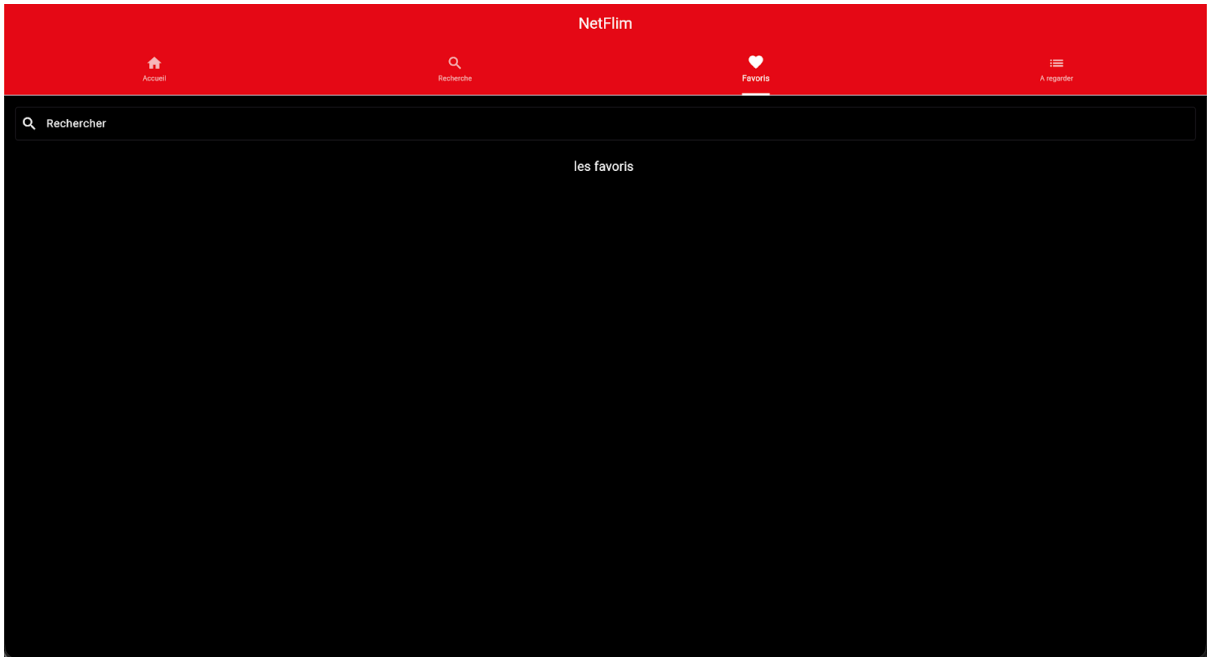
6. Conclusion

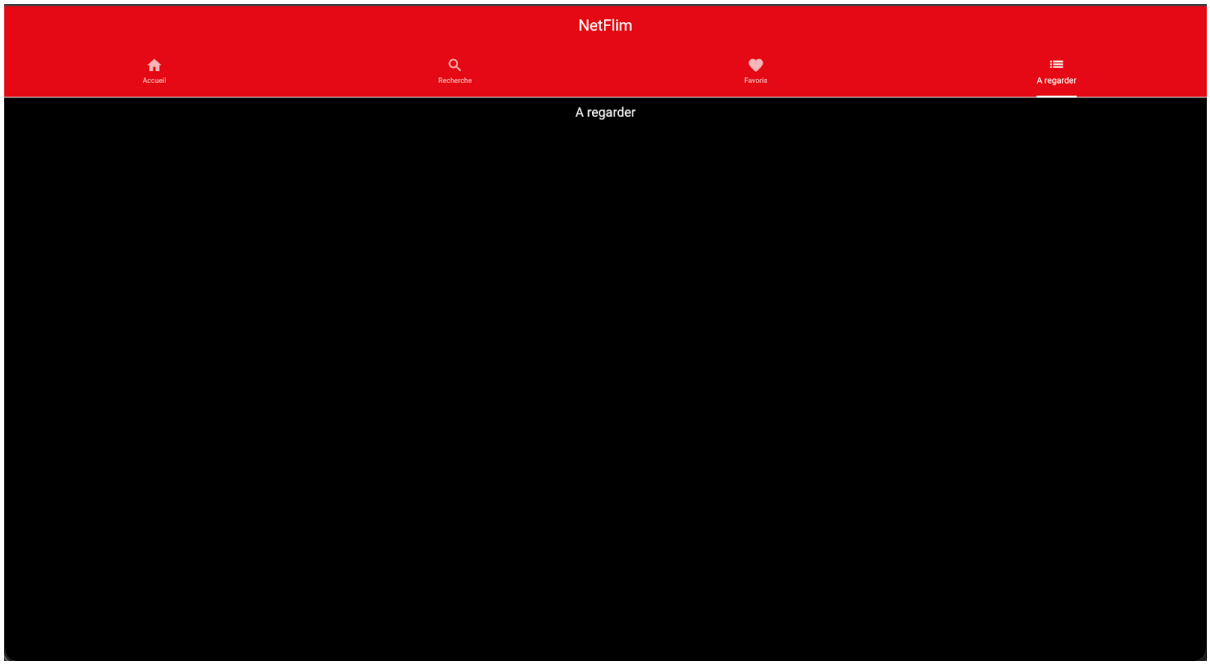
Ce projet m'a permis de concevoir une application mobile complète, structurée et proche d'une application réelle de streaming. À travers le développement de Netflim, j'ai pu manipuler des concepts clés du développement Flutter, améliorer progressivement l'application avec des fonctionnalités avancées, et renforcer mes compétences en architecture logicielle ainsi qu'en UX/UI.

Ce TP constitue une expérience concrète et valorisable dans mon parcours en développement informatique, illustrant ma capacité à mener un projet applicatif de bout en bout, de la conception à l'implémentation de fonctionnalités avancées.

7. Illustration




NetFlim
Accueil
Recherche
Favoris
A regarder
Rechercher
les favoris


NetFlim
Accueil
Recherche
Favoris
A regarder
A regarder