Rapport de Projet : Application Mobile TMDB

Table des matières

---

1. Introduction

1.1 Contexte du projet

Ce projet consiste à développer une application mobile utilisant l'API The Movie Database (TMDB) pour afficher des informations sur les films et séries TV.

1.2 Objectifs

- Créer une application React Native fonctionnelle
- Intégrer l'API TMDB pour récupérer les données
- Implémenter des fonctionnalités de favoris et watchlist
- Créer une interface utilisateur moderne et intuitive

1.3 Technologies utilisées

- **React Native** : Framework pour le développement mobile
- **Expo** : Plateforme de développement
- **React Navigation** : Navigation entre les écrans
- **Context API** : Gestion d'état globale
- **TMDB API** : Source de données

---

2. Architecture de l'application

2.1 Structure des dossiers

TMDB-APP/

_____src/

_____api/

| | ├— MovieApi.js

| | ├— TvApi.js

| | └— SearchApi.js

| ├— components/

| | ├— MovieCard.js

| | └— TvCard.js

| ├— config/

| | └— api.js

| ├— context

| | ├— FavouritesContext.js

| | └— WatchlistContext.js

| ├— navigation/

| | └— BottomTabs.js

| └— screens/

| ├— MovieScreen.js

| ├— TvScreen.js

| ├— FavouritesScreen.js

| ├— WatchlistScreen.js

| └— SearchScreen.js

├— App.js

└— package.json

2.2 Flux de données

API TMDB → API Modules → Screens → Components → Context (State)

---

3. API TMDB - Documentation

3.1 Présentation de l'API

The Movie Database (TMDB) est une API REST qui fournit des informations complètes sur les films et séries TV.

URL de base : https://api.themoviedb.org/3

Authentification : API Key

const API_KEY = 'fd63c8e30a4b643d327e533e0aa10f61';

3.2 Endpoints utilisés

3.2.1 Films

Endpoint	Description	Exemple
/movie/now_playing	Films actuellement au cinéma	GET /movie/now_playing?api_key=XXX
/movie/popular	Films populaires	GET /movie/popular?api_key=XXX
/movie/top_rated	Films les mieux notés	GET /movie/top_rated?api_key=XXX
/movie/upcoming	Films à venir	GET /movie/upcoming?api_key=XXX

3.2.2 Séries TV

Endpoint	Description	Exemple
/tv/airing_today	Séries diffusées aujourd'hui	GET /tv/airing_today?api_key=XXX
/tv/on_the_air	Séries en cours	GET /tv/on_the_air?api_key=XXX
/tv/popular	Séries populaires	GET /tv/popular?api_key=XXX
/tv/top_rated	Séries les mieux notées	GET /tv/top_rated?api_key=XXX

3.2.3 Recherche

Endpoint	Description	Paramètres
/search/movie	Rechercher des films	query (requis)
/search/tv	Rechercher des séries	query (requis)

3.3 Structure des données retournées

Exemple de réponse pour un film :

```
{
  "results": [
    {
      "id": 823464,
      "title": "Godzilla x Kong",
      "poster_path": "/z1p34vh7dEOnLDmyCrlUVLuoDzd.jpg",
      "vote_average": 7.2,
      "release_date": "2024-03-27",
      "popularity": 2847.95,
      "overview": "Description du film..."
    }
  ]
}
```

3.4 Gestion des images

Les images sont récupérées via une URL construite :

```
const IMAGE_BASE_URL = 'https://image.tmdb.org/t/p/w500';
const imageUrl = IMAGE_BASE_URL + poster_path;
```

Tailles disponibles : w92, w154, w185, w342, w500, w780, original

3.5 Gestion des erreurs

```
try {
  const response = await fetch(url);
  const data = await response.json();
  return data.results;
} catch (error) {
  console.error('Error:', error);
  return [];
}
```

---

4. Fonctionnalités implémentées

4.1 Affichage des films (MovieScreen)

- **Now Playing** : Films actuellement au cinéma
- **Popular** : Films populaires du moment
- **Top Rated** : Films les mieux notés de tous les temps
- **Upcoming** : Films à venir prochainement

Code exemple :

```
useEffect(() => {
  fetchMovies('now_playing').then(setNowPlaying);
  fetchMovies('popular').then(setPopular);
  fetchMovies('top_rated').then(setTopRated);
  fetchMovies('upcoming').then(setUpcoming);
}, []);
```

4.2 Affichage des séries TV (TvScreen)

- **Airing Today** : Séries diffusées aujourd'hui
- **On The Air** : Séries actuellement en cours
- **Popular** : Séries populaires
- **Top Rated** : Séries les mieux notées
-

4.3 Système de favoris (FavouritesContext)

Permet aux utilisateurs de marquer des films/séries comme favoris.

Implémentation :

```
const toggleFavourite = (item) => {
  setFavourites((prev) => {
    const exists = prev.find((fav) => fav.id === item.id);
    if (exists) {
      return prev.filter((fav) => fav.id !== item.id);
    }
    return [...prev, item];
```

```
  });
};
```

4.4 Watchlist (WatchlistContext)

Liste de films/séries à regarder plus tard.

Différence avec les favoris :

- Favoris = Contenu aimé
- Watchlist = Contenu à voir
-

4.5 Recherche (SearchScreen)

Recherche en temps réel de films et séries.

Fonctionnalités :

- Recherche instantanée
- Résultats séparés (Films / Séries)
- Affichage horizontal scrollable

```
const handleSearch = async (text) => {
  if (text.length > 2) {
    const movies = await searchMovies(text);
    const tv = await searchTv(text);
    setMovieResults(movies);
    setTvResults(tv);
  }
};
```

---

5. Design et Interface utilisateur

5.1 Choix du thème

Thème sombre (Netflix-style)

- Fond noir : #000000
- Cartes : #1a1a1a
- Texte principal : #ffffff

- Couleur d'accent : #e50914 (rouge)
- Texte secondaire : #999999
-

5.2 Navigation (Bottom Tabs)

Navigation par onglets avec icônes :

- Movies
- TV Shows
- Favorites
- Watchlist
- Search

Avantages :

- Navigation intuitive
- Accès rapide à toutes les sections
- Icônes visuellement claires
-

5.3 Composants réutilisables

MovieCard / TvCard

Carte affichant :

- Poster (180x120px)
- Titre (1 ligne max)
- Note ⭐ + Année
- Badge "Popular" (si applicable)
- Boutons Favoris / Watchlist

5.4 Responsive design

- Largeur adaptative
- ScrollView pour le contenu long
- FlatList horizontal pour les listes

---

6. Collaboration et Gestion du projet

6.1 Outils de collaboration

- **GitHub** : Gestion de version
- **Git** : Branches et commits
- **Communication** : Messages réguliers

6.2Workflow Git

- Récupérer les modifications

git pull

- Ajouter ses changements

git add .

git commit -m "Description"

-Envoyer sur GitHub

git push

6.3 Commits principaux

- Initial project setup
- Add API integration
- Implement favorites and watchlist
- Add dark theme design
- Final improvements

---

7. Conclusion

7.1 Objectifs atteints

Application fonctionnelle

Toutes les fonctionnalités requises

Design moderne et intuitif

Code bien structuré

Collaboration réussie

7.2 Apprentissages

- Utilisation d'une API REST
- React Native et Expo
- Gestion d'état avec Context
- Collaboration avec Git/GitHub
- Design d'interface mobile

7.3 Perspectives

Ce projet nous a permis de développer des compétences essentielles en développement mobile et en travail d'équipe. L'application peut être étendue avec de nombreuses fonctionnalités supplémentaires.

Annexes

Références

- The Movie Database API Documentation
- React Native Documentation
- Expo Documentation
- React Navigation Documentation