**Diagramme de Cas d'Utilisation**

Le diagramme de cas d’utilisation permet de représenter les interactions entre les utilisateurs (acteurs) et le système (fonctionnalités). Voici les principaux acteurs et leurs interactions :

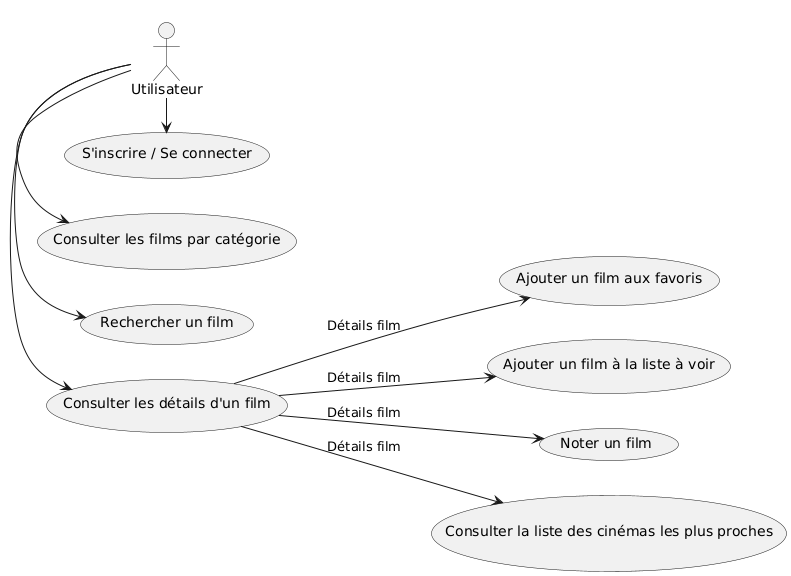
**Acteurs :**

* **Utilisateur** : Un utilisateur peut consulter les films et les salles de cinéma les plus proche.
* **Administrateur** : Gère les utilisateurs (gestion de statut, activation, suppression, …).

**Cas d'utilisation principaux :**

1. S'inscrire / Se connecter (pour les utilisateurs).
2. Consulter les films classés par catégorie.
3. Consulter les détails d’un film.
4. Recherche d’un film.
5. Interaction avec les films pour les utilisateurs inscrits :
   * Ajouter un film aux favoris.
   * Ajouter un film à la liste à voir.
   * Noter un film.
6. Consulter la liste des salles de cinémas les plus proches.
7. Gestion des salles de cinéma (administrateur).
8. Une image contenant texte, croquis, Police, cercle

   Le contenu généré par l’IA peut être incorrect.Gérer les utilisateurs (administrateur).



Modèle Conceptuel de Données (MCD)

Le MCD représente les entités et leurs relations.

Entités :

1. **User**
   * uid (PK)
   * email
   * password
   * displayName
   * photoURL
   * createdAt
   * lastLoginAt
   * role
2. **Theatre**
   * id (PK)
   * dep
   * nom
   * adresse
   * code\_insee
   * commune
   * ecrans
   * fauteuils
   * geo (lon, lat)
3. **Movie**
   * id (PK)
   * title
   * overview
   * genre\_ids (relation avec Genre)
   * release\_date
   * vote\_average
   * vote\_count
4. **Genre**
   * id (PK)
   * name

Relations :

* Un utilisateur peut gérer plusieurs films et théâtres.
* Un film peut appartenir à plusieurs genres.
* Un théâtre peut projeter plusieurs films.

Modèle Logique de Données (MLD)

Le MLD affine le MCD en ajoutant des clés étrangères pour représenter les relations.

Tables principales :

1. users
2. theatres
3. movies
4. genres
5. movie\_genres (table associative pour gérer la relation plusieurs-à-plusieurs entre movies et genres).

Modèle Physique de Données (MPD)

Le MPD traduit le MLD en tables SQL avec types de données.

sql

**CREATE** **TABLE** users (

uid **VARCHAR**(255) **PRIMARY** **KEY**,

email **VARCHAR**(255) NOT NULL,

password **VARCHAR**(255) NOT NULL,

displayName **VARCHAR**(255),

photoURL **TEXT**,

createdAt **BIGINT**,

lastLoginAt **BIGINT**,

role **ENUM**('admin', 'user') NOT NULL **DEFAULT** 'user'

);

**CREATE** **TABLE** theatres (

id **INT** **AUTO\_INCREMENT** **PRIMARY** **KEY**,

dep **INT** NOT NULL,

nom **VARCHAR**(255),

adresse **TEXT**,

code\_insee **INT**,

commune **VARCHAR**(255),

ecrans **INT**,

fauteuils **INT**,

geo\_lon **FLOAT**,

geo\_lat **FLOAT**

);

**CREATE** **TABLE** movies (

id **INT** **PRIMARY** **KEY**,

title **VARCHAR**(255),

overview **TEXT**,

release\_date **DATE**,

vote\_average **FLOAT**,

vote\_count **INT**

);

**CREATE** **TABLE** genres (

id **INT** **PRIMARY** **KEY**,

name **VARCHAR**(255)

);

**CREATE** **TABLE** movie\_genres (

movie\_id **INT**,

genre\_id **INT**,

**PRIMARY** **KEY**(movie\_id, genre\_id),

**FOREIGN** **KEY**(movie\_id) **REFERENCES** movies(id),

**FOREIGN** **KEY**(genre\_id) **REFERENCES** genres(id)

);

Diagramme UML : Diagramme de Classe

Le diagramme de classe modélise la structure statique du système.**Exemple de classes principales :**

* Classe User avec attributs comme uid, email, etc.
* Classe Theatre avec attributs comme nom, adresse, etc.
* Classe Movie avec attributs comme title, release\_date, etc.
* Classe Genre.

Les relations incluent :

* Une association entre Movie et Genre.
* Une association entre User et ses actions administratives.

Diagramme UML : Diagramme de Séquence

Un diagramme de séquence montre comment les objets interagissent dans un scénario spécifique, par exemple :

**Scénario : Ajouter un film**

1. L'administrateur envoie une requête au système pour ajouter un film.
2. Le système vérifie les droits d'accès.
3. Le film est ajouté à la base de données.
4. Confirmation renvoyée à l'administrateur.

Vous pouvez utiliser des outils comme Microsoft Visio, Lucidchart, ou Altova UModel pour créer ces diagrammes[1](https://clickup.com/fr-FR/blog/69659/logiciel-de-diagramme-uml)[4](https://www.microsoft.com/fr-fr/microsoft-365/visio/uml)[9](https://www.guru99.com/fr/best-uml-tools.html). Pour le MCD/MLD, des logiciels comme Looping ou Draw.io sont également recommandés3[6](https://laconsole.dev/formations/sql/cas-pratique/).

**Diagramme de Cas d'Utilisation**

Les cas d'utilisation restent similaires, mais avec Firebase, la gestion des données est simplifiée grâce à sa synchronisation en temps réel et ses SDK multiplateformes.

Cas d'utilisation principaux :

1. **Utilisateur classique** :
   * Consulter la liste des films.
   * Rechercher un film par titre ou genre.
   * Consulter les détails d'un théâtre.
2. **Administrateur** :
   * Ajouter, modifier ou supprimer un film.
   * Ajouter, modifier ou supprimer un théâtre.
   * Gérer les utilisateurs (changer les rôles, désactiver un compte).

**Modèle Conceptuel de Données (MCD)**

Avec Firebase, les données sont stockées sous forme de collections et de documents JSON. Voici une structure conceptuelle adaptée :

Collections principales :

1. **Users**
   * uid : Identifiant unique.
   * email, password, displayName, etc.
2. **Theatres**
   * id : Identifiant unique.
   * nom, adresse, geo (coordonnées GPS), etc.
3. **Movies**
   * id : Identifiant unique.
   * title, overview, release\_date, etc.
4. **Genres**
   * id : Identifiant unique.
   * name : Nom du genre.

Relations :

* Les genres sont référencés dans les films via un tableau (genre\_ids).
* Les utilisateurs peuvent avoir des rôles spécifiques (admin ou utilisateur classique).

**Modèle Logique de Données (MLD)**

Firebase n'utilise pas de relations strictes comme dans une base relationnelle. Les relations sont souvent représentées via des références ou des sous-collections.

Exemple de structure JSON pour Firebase :

json

{

"users": {

"uid1": {

"email": "n.djamel@free.fr",

"displayName": "Mat95rix7",

"role": "admin"

}

},

"theatres": {

"theatre1": {

"nom": "UGC GEORGE V",

"adresse": "146 AVENUE DES CHAMPS ELYSEES",

"geo": { "lon": 2.298394, "lat": 48.873073 },

"ecrans": 11

}

},

"movies": {

"movie1": {

"title": "Sonic 3, le film",

"overview": "...",

"genre\_ids": [28, 878],

"release\_date": "2024-12-19"

}

},

"genres": {

"28": { "name": "Action" },

"878": { "name": "Science-Fiction" }

}

}

**Diagramme UML : Diagramme de Classe**

Avec Firebase, chaque collection peut être vue comme une classe contenant des attributs.

Classes principales :

1. **User**
   * Attributs : uid, email, role, etc.
2. **Theatre**
   * Attributs : nom, adresse, geo (longitude, latitude), etc.
3. **Movie**
   * Attributs : title, overview, genre\_ids (référence à Genre), etc.
4. **Genre**
   * Attributs : id, name.

Les relations entre les classes sont représentées par des références dans les documents JSON.

**Diagramme UML : Diagramme de Séquence**

Voici un exemple pour l'ajout d'un film par un administrateur :

1. L'administrateur envoie une requête via l'application pour ajouter un film.
2. L'application vérifie le rôle de l'utilisateur via Firebase Authentication.
3. Les données du film sont ajoutées à la collection **Movies** dans Firebase Realtime Database ou Firestore.
4. Une confirmation est renvoyée à l'administrateur.

**Avantages spécifiques avec Firebase**

1. **Synchronisation en temps réel** : Toute modification est instantanément visible sur tous les appareils connectés[2](https://appmaster.io/fr/blog/base-de-donnees-en-temps-reel-firebase-pour-la-synchronisation-des-donnees-en-temps-reel)[9](https://firebase.google.com/docs/database?hl=fr).
2. **Flexibilité** : Les schémas dynamiques permettent d'ajouter ou modifier facilement des champs[3](https://firebase.google.com/docs/firestore?hl=fr)[7](https://appmaster.io/fr/blog/base-de-donnees-firestore-nosql).
3. **Sécurité intégrée** : Grâce aux règles de sécurité Firebase, vous pouvez restreindre l'accès aux collections selon les rôles utilisateurs[4](https://blog.back4app.com/fr/les-10-principaux-avantages-de-firebase/)[9](https://firebase.google.com/docs/database?hl=fr).

Ces ajustements garantissent que votre application est optimisée pour tirer parti des fonctionnalités uniques de Firebase tout en respectant vos besoins fonctionnels et structurels.