Manuel d’utilisation

Cinéphoria est une plateforme dédiée à la gestion de cinéma, conçue pour offrir aux utilisateurs une interaction simple et complète avec les cinémas de leurs villes. L’application se compose de trois volets :

- une application web permettant de consulter les séances, réserver des billets, découvrir les films disponibles, gérer toute l’administration liée aux cinémas, se créer un compte, contacter les cinéma par mail;
- une application mobile offrant aux clients un accès à toutes leurs réservations ainsi qu’aux QR codes correspondants ;
- une application bureautique destinée aux employés pour la gestion des incidents techniques en salle.

Grâce à Cinéphoria, les utilisateurs peuvent créer un compte, sélectionner leurs séances selon le cinéma, la date, et la qualité d’image, puis réserver leurs places en ligne. Les employés disposent quant à eux d’outils complets pour administrer les films, les séances, les salles, et les avis assurant ainsi un service optimal et une expérience client améliorée.

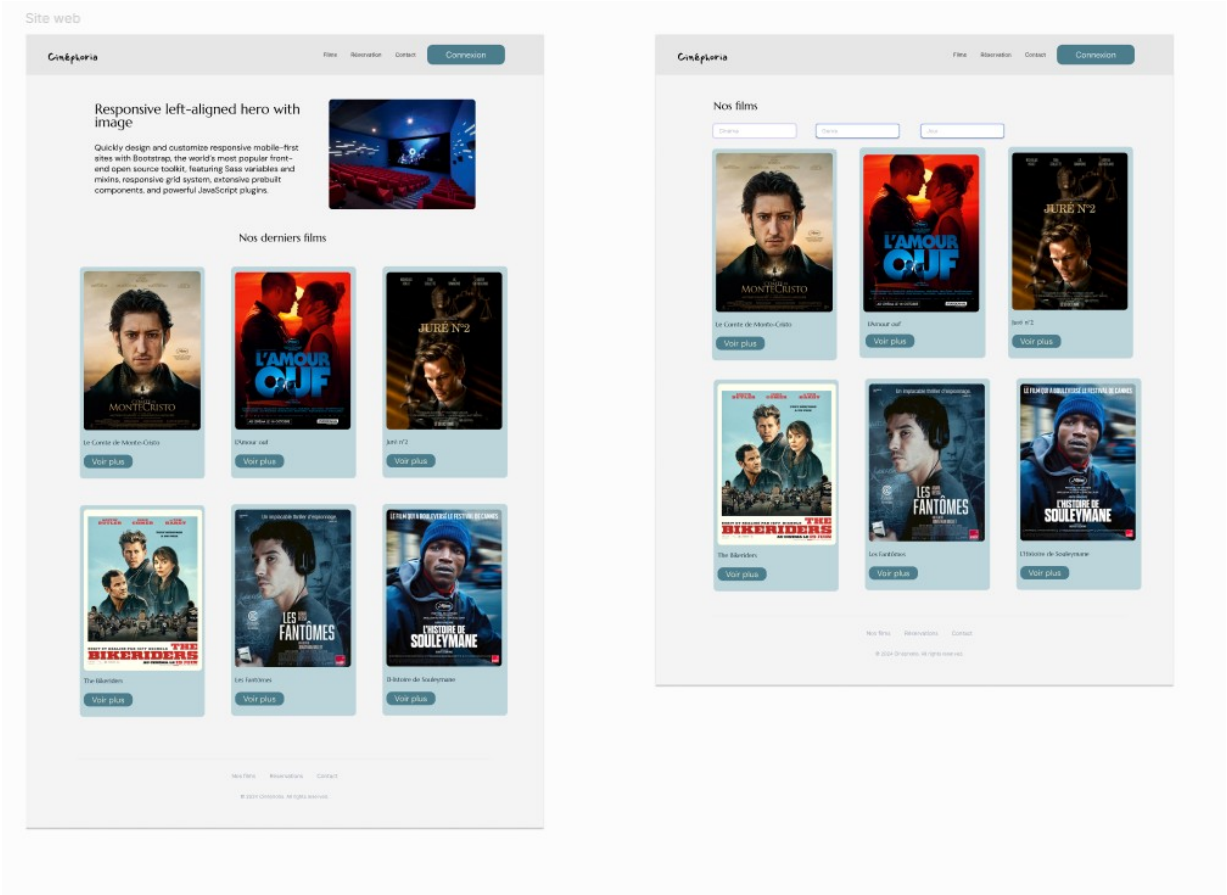
Gestion de projet

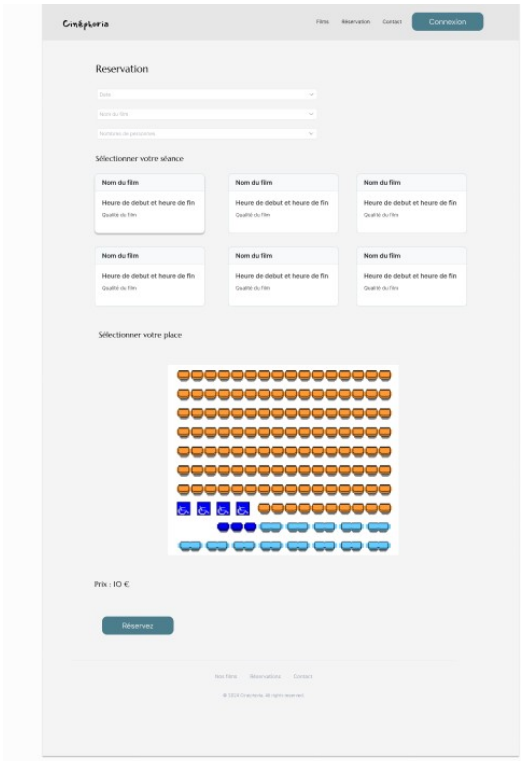
Pour organiser et suivre le développement de l'application Cinéphoria, j'ai utilisé Trello comme outil de gestion de projet. Trello m'a permis de structurer les tâches sous forme de cartes, organisées dans différentes colonnes correspondant aux étapes du projet :

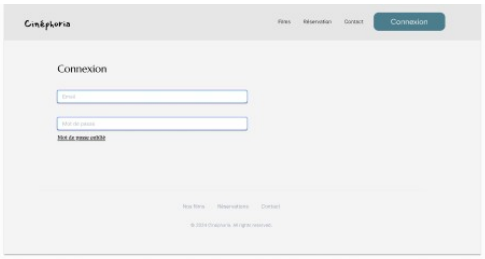
- **Fonctionnalités prévues** : où je liste toutes les fonctionnalités planifiées, classées par priorité.
- **Fonctionnalités à faire** : les tâches que je prévois de réaliser lors du prochain sprint ou dans la période en cours.
- **Fonctionnalités en cours** : les fonctionnalités sur lesquelles je travaille actuellement.
- **Fonctionnalités terminées** : les fonctionnalités finalisées et validées.

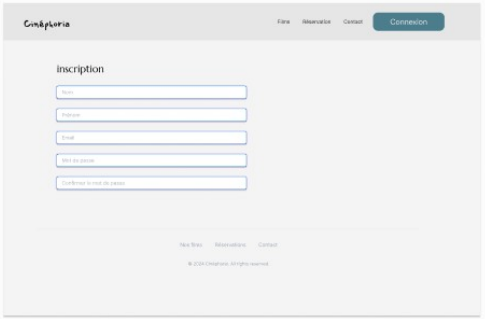
Cette organisation m'a aidé à garder une vision claire de l'avancement du projet, à gérer les priorités et à travailler de manière agile.

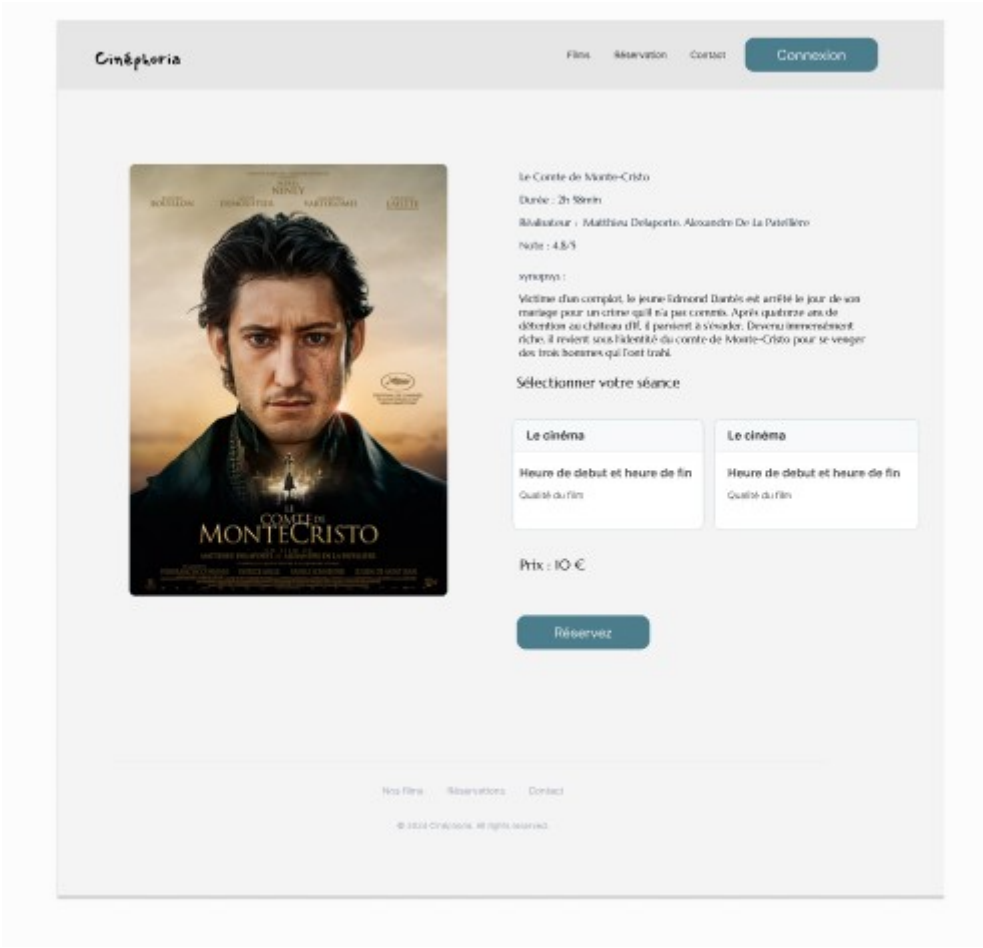
Maquette web



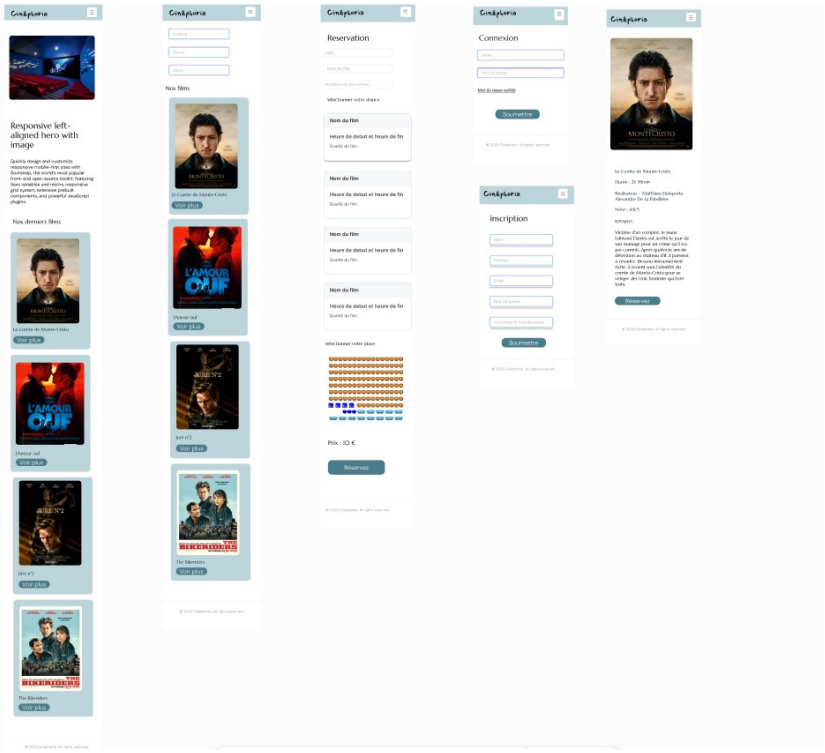




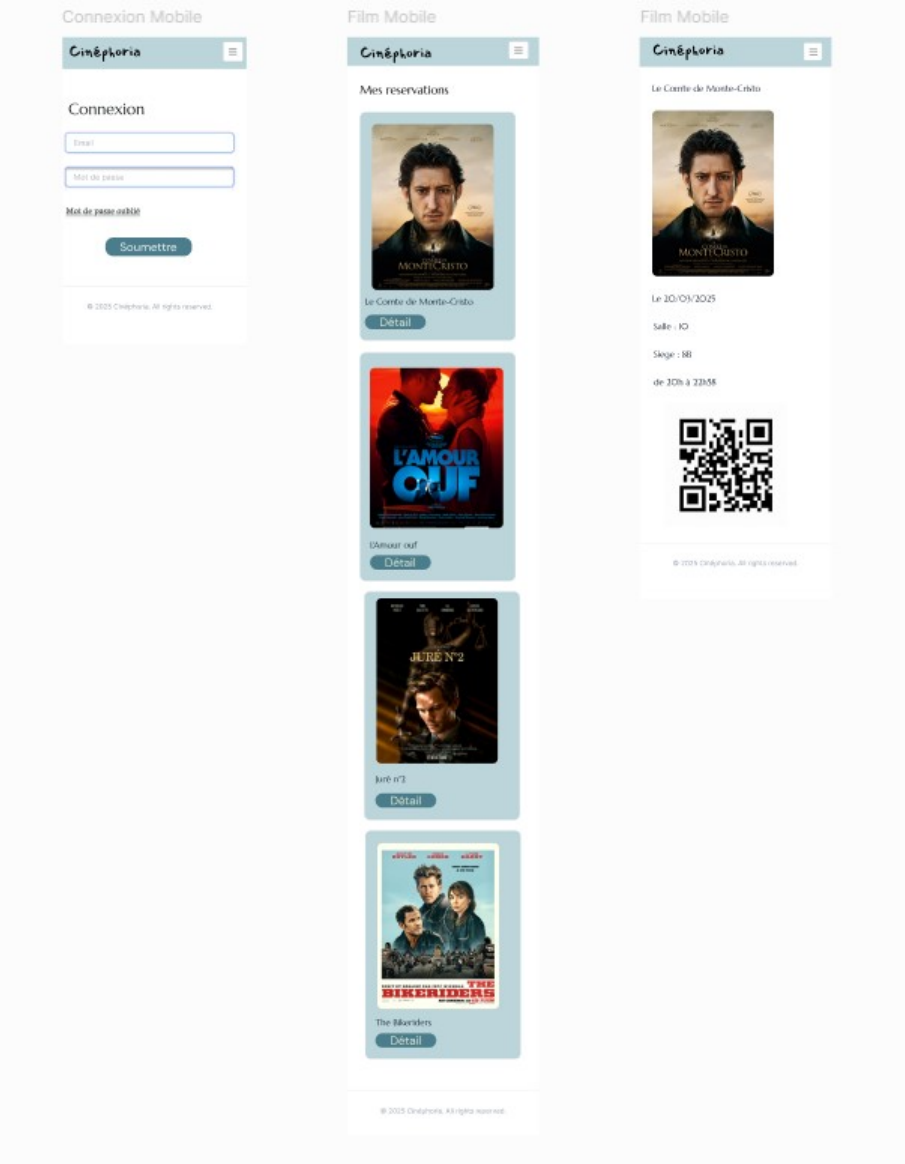




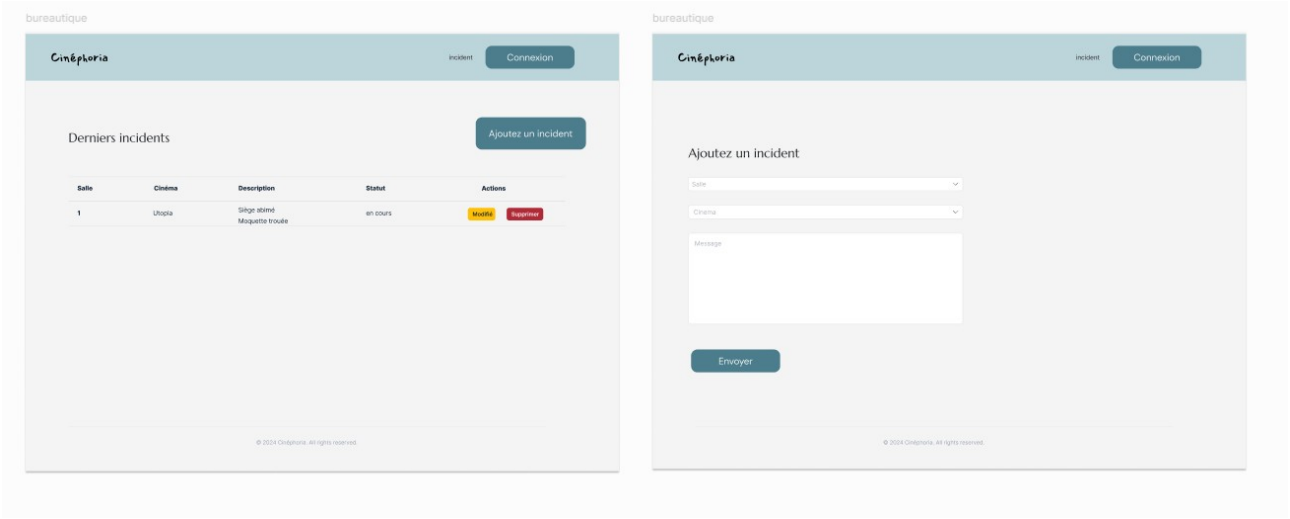
Maquette mobile



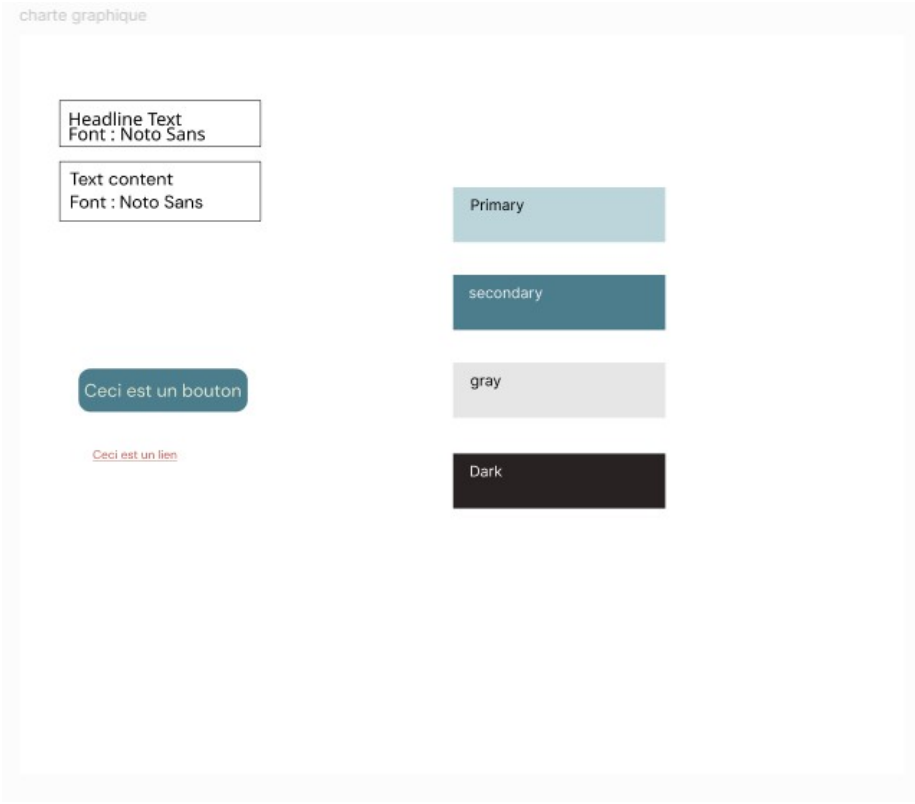
Maquette application Mobile



Maquette application Mobile

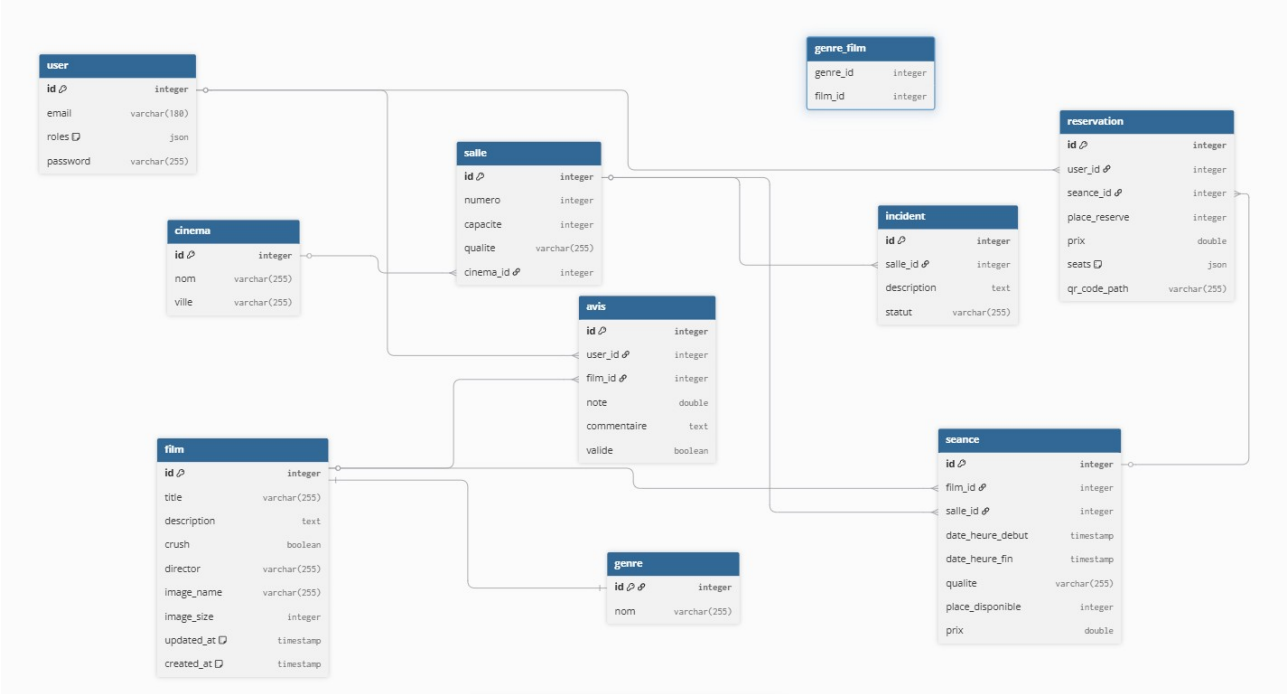


Charte graphique



Documentation technique

Diagramme MCD



Cinema -> Salle:

Un cinéma peut avoir plusieurs salles;

une salle appartient à un cinéma.

Film -> Seance:

Un film est projeté dans plusieurs séances;

une séance concerne un film.

Salle -> Seance:

Une salle accueille plusieurs séances;

une séance se tient dans une salle.

User -> Avis:

Un utilisateur rédige plusieurs avis;

un avis est rédigé par un utilisateur.

Film -> Avis:

Un film reçoit plusieurs avis;

un avis concerne un film.

User -> Reservation:

Un utilisateur effectue plusieurs réservations;
une réservation est liée à un utilisateur.

Seance -> Reservation:

Une séance a plusieurs réservations;
une réservation est pour une séance.

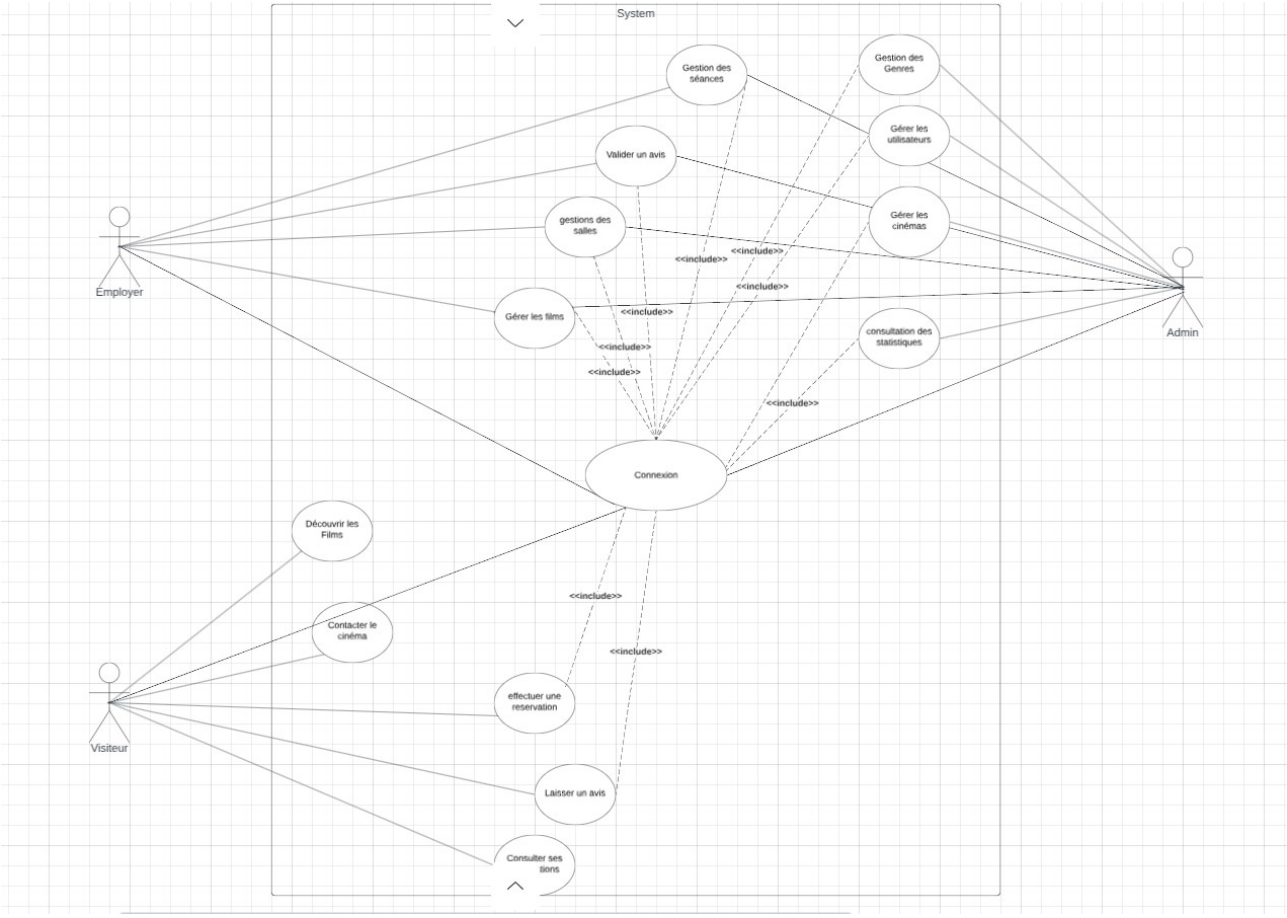
Salle -> Incident:

Une salle peut avoir plusieurs incidents;
un incident est lié à une salle.

Film -> Genre:

Un film peut avoir plusieurs genres;
un genre peut être associé à plusieurs films

Diagramme d’utilisation



Architecture logicielle Cinéphoria :

L'application Cinéphoria repose sur une architecture logicielle modulaire, pensée pour répondre à la fois aux besoins du personnel et du public, tout en assurant de bonnes performances, une sécurité renforcée et une séparation claire des responsabilités techniques.

Le cœur de l'application web a été développé avec Symfony 7, un framework PHP robuste, structuré autour du modèle MVC (Modèle-Vue-Contrôleur). Il permet de construire des interfaces dynamiques et sécurisées côté serveur grâce à Twig, son moteur de templates intégré. J'ai choisi Symfony pour sa fiabilité, sa forte communauté, ses outils intégrés (comme API Platform) et sa compatibilité naturelle avec Doctrine ORM pour la gestion de la base MySQL.

Concernant la base de données, j'ai utilisé deux technologies complémentaires :

- MySQL pour stocker les données métiers classiques de manière structurée (films, utilisateurs, incidents, réservations,...).

- MongoDB, une base NoSQL orientée documents, pour gérer spécifiquement les statistiques de fréquentation des 7 derniers jours. MongoDB a été choisie car elle permet une ingestion rapide et flexible des données temporelles, sans contrainte de schéma rigide. C'est donc une solution parfaitement adaptée pour suivre des tendances en temps réel sans ralentir ou surcharger la base principale.

L'application expose également une API REST, construite avec API Platform et protégée par un système d'authentification JWT. Cette API permet de centraliser les données et de les rendre accessibles de manière sécurisée aux interfaces web, mobile et desktop.

Pour le front-end, j'ai utilisé :

- Bootstrap côté web, pour accélérer la création d'interfaces réactives, accessibles et compatibles avec tous les navigateurs, tout en respectant les standards de design modernes.

- React Native côté mobile, afin de développer une application native fluide, fonctionnant sur iOS et Android tout en partageant une seule base de code JavaScript. Ce choix offre un bon compromis entre performance et productivité.

- Electron pour l'environnement desktop. Cette solution permet d'emballer une application web dans une interface native, compatible Windows/macOS/Linux, ce qui est idéal pour les besoins spécifiques des employés (notamment la gestion autonome des incidents).

Enfin, pour le stockage des fichiers statiques (images de films, QR codes de billets...), j'ai utilisé AWS S3, intégré via l'addon Heroku Bucketeer. Ce choix s'explique par une contrainte spécifique de la plateforme Heroku : les fichiers stockés localement sur l'hébergement sont supprimés régulièrement (à chaque redéploiement ou mise en veille). Stocker les fichiers dans le cloud permet donc de garantir leur persistance, leur accessibilité depuis toutes les plateformes, et d'assurer une expérience utilisateur fluide.

L'application web est déployé sur Heroku, une plateforme cloud qui facilite la mise en production et l'intégration continue, ce qui correspond parfaitement à un projet agile et évolutif comme Cinéphoria.

Plan de test et déploiement

Le plan de test mis en place assure la fiabilité de ses fonctionnalités grâce à une combinaison de tests unitaires, fonctionnels et d'intégration, réalisés avec PHPUnit dans un environnement Symfony 7.

Les tests unitaires:

FilmTest, ReservationTest, et SeanceTest, valident les entités (Film, Reservation, Seance) en vérifiant leurs propriétés (ex. : titre, prix, dates, places disponibles) et leurs comportements (ex. : reserverPlaces dans Seance, qui réduit correctement les places disponibles ou échoue si insuffisant).

Les tests fonctionnels :

tels que ContactTest, contrôlent l'interface web Twig en testant la soumission du formulaire de contact, l'envoi d'emails via Symfony Mailer, et l'affichage des messages de succès.

Les tests d'intégration :

comme LoginTest, simulent l'accès sécurisé au tableau de bord admin pour un utilisateur avec ROLE_ADMIN et la redirection des anonymes vers /login, en recréant la base MySQL pour un environnement propre.

Le déploiement, effectué sur Heroku, utilise l'addon Bucketeer pour stocker images (image_name) et QR codes (qr_code_path) sur AWS S3.

Ce plan garantit une application robuste pour les interfaces web (Twig), mobile (React Native), et desktop (Electron), avec une API REST sécurisée (API Platform) et un déploiement évolutif.

Déploiement continu

J’ai mis en place une chaîne CI/CD complète pour l’application web Cinephoria en utilisant GitHub Actions, afin de garantir un développement sécurisé, fiable et déployable automatiquement.

1. Security Audit
À chaque *push* ou *pull request*, un workflow exécute composer audit avec PHP 8.2. Cela permet de détecter rapidement les vulnérabilités connues dans les dépendances PHP que j’utilise. Le rapport généré (security-audit.json) est archivé en tant qu'artefact, ce qui facilite la revue et l’intégration dans un suivi de sécurité continu.

2. Tests
Un second workflow exécute PHPUnit dans un environnement de test basé sur MariaDB. Il couvre :

•des **tests unitaires** (FilmTest, ReservationTest, SeanceTest) validant les entités métier,

•des **tests fonctionnels** (ContactTest) vérifiant le bon affichage et traitement des formulaires via Twig,

•des **tests d'intégration** (AdminAccessTest) testant les accès sécurisés ROLE_ADMIN via authentification.

Ce pipeline s’assure que toute modification du code n’introduit pas de régression fonctionnelle ou de bug critique.

3. Deploy
Enfin, un troisième workflow automatise le déploiement sur Heroku à chaque *push* sur la branche master. Il utilise l’interface Heroku CLI avec une clé API sécurisée (stockée via GitHub Secrets), garantissant une livraison rapide, fiable et sans action manuelle.

Cette architecture CI/CD me permet de livrer une application Symfony 7 stable, testée et sécurisée, avec une réactivité en cas de bug ou de vulnérabilité.