ABSOLUTE CINEMA

Application web de réservation de places de cinéma

Michel MUSTAFOV, Edwin WEHBE

1. Introduction

Absolute Cinema est une application web complète de réservation de places de cinéma développée en PHP pur, suivant une architecture stricte Modèle–Vue–Contrôleur (MVC) sans aucun framework externe. L'application permet aux utilisateurs de consulter les films à l'affiche, de visualiser les séances disponibles et de réserver des places en ligne.

Le système implémente un modèle d'accès à double rôle : les utilisateurs standards gèrent leurs comptes et réservations, tandis que les administrateurs disposent d'un contrôle total sur les films, les salles, les séances, les réservations et les comptes utilisateurs.

Ce document présente les choix techniques, l'architecture, les fonctionnalités implémentées, les mesures de sécurité et le schéma de base de données du projet.

2. Architecture

2.1 Patron MVC

Le projet sépare strictement les responsabilités selon le patron MVC. Aucune requête SQL n'apparaît dans les vues et aucun HTML n'est généré dans les modèles. Le contrôleur frontal (index.php) sert de point d'entrée unique et route toutes les requêtes vers la méthode du contrôleur adéquate.

Couche	Responsabilité	Fichiers
Modèle	Accès BDD via PDO, logique métier, validation des données	Database.php, UserModel.php, MovieModel.php, RoomModel.php, ScreeningModel.php, ReservationModel.php
Vue	Rendu HTML avec échappement, templates de mise en page (header/footer)	views/layouts/*, views/auth/*, views/movies/*, views/reservations/*, views/admin/*
Contrôleur	Gestion des requêtes, validation des entrées, gestion de session, routage	AuthController.php, MovieController.php, ReservationController.php, AdminController.php

2.2 Structure du projet

```
cinema/
├── index.php                # Contrôleur frontal & routeur
├── config/database.php       # Configuration BDD avec .env
├── sql/schema.sql            # Script de création de la BDD
├── app/
│   ├── models/               # Couche d'accès aux données PDO
│   ├── controllers/          # Gestionnaires de requêtes
│   └── views/                # Templates PHP
└── public/
    ├── css/                  # Feuilles de styles modulaires
    ├── js/                   # Interactivité côté client
    └── img/                  # Ressources statiques
```

2.3 Mécanisme de routage

Toutes les requêtes HTTP sont traitées par index.php, qui lit le paramètre action et dirige vers la méthode du contrôleur correspondant via un switch. Cela fournit un routage centralisé et propre sans réécriture d'URL.

3. Conception de la base de données

3.1 Modèle entité-relation

La base de données utilise cinq tables avec des clés étrangères et des suppressions en cascade pour maintenir l'intégrité référentielle. Toutes les tables utilisent utf8mb4.

Table	Rôle	Champs clés
users	Comptes utilisateurs avec gestion des rôles	id, username, email, password (bcrypt), role (enum: user/admin)
movies	Catalogue des films	id, title, description, duration (minutes)
rooms	Salles de projection avec capacité configurable	id, name (unique), capacity
screenings	Séances planifiées liées aux salles	id, movie_id (FK), room_id (FK), screening_date, screening_time, available_seats
reservations	Réservations utilisateurs liées aux séances	id, user_id (FK), screening_id (FK), seats, created_at

3.2 Relations

- Un film peut avoir plusieurs séances (un-à-plusieurs)
- Une salle peut accueillir plusieurs séances (un-à-plusieurs)
- Un utilisateur peut avoir plusieurs réservations (un-à-plusieurs)
- Toutes les clés étrangères utilisent ON DELETE CASCADE

3.3 Singleton PDO

La classe Database implémente le patron Singleton, garantissant une connexion PDO unique réutilisée dans toute l'application. Elle est configurée avec gestion des erreurs par exceptions, mode fetch associatif, et les requêtes préparées réelles (emulate_prepares désactivé).

4. Fonctionnalités implémentées

4.1 Authentification et gestion des comptes

- **Inscription :** Validation de l'unicité du nom d'utilisateur, format email, longueur du mot de passe (min. 6 caractères) et confirmation. Hachage bcrypt via password_hash().
- **Connexion :** Vérification des identifiants avec password_verify(). En cas de succès, la session est alimentée avec user_id, username, role et last_activity.
- **Se souvenir de moi :** Définit un cookie HttpOnly (remember_email) avec TTL de 30 jours. Le cookie pré-remplit le champ email.
- **Expiration de session :** Le contrôleur frontal vérifie last_activity à chaque requête. Les sessions inactives depuis plus de 30 minutes sont automatiquement détruites.
- **Gestion du profil :** L'utilisateur peut modifier son nom, email et mot de passe. La suppression du compte entraîne la suppression en cascade de toutes ses réservations.
- **Déconnexion :** Détruit la session et efface le cookie.

4.2 Consultation des films

- **Liste des films :** Affiche tous les films sous forme de cartes responsives avec titre, synopsis tronqué et durée.
- **Détail d'un film :** Description complète, durée et toutes les séances à venir avec une barre visuelle de disponibilité.
- **Filtrage par date :** Seules les séances futures sont affichées (WHERE screening_date >= CURDATE()).

4.3 Système de réservation

- **Sélection de places :** L'utilisateur choisit de 1 à 10 places. Validation côté client (feedback instantané) et côté serveur (prévention du sur-booking).
- **Vérification de disponibilité :** Avant confirmation, le système vérifie available_seats >= places demandées via un UPDATE atomique avec clause WHERE.
- **Annulation :** L'utilisateur peut annuler ses réservations. Les places sont restituées de manière atomique avec vérification de la capacité maximale.
- **Contrôle d'accès :** La fonction requireLogin() redirige les utilisateurs non connectés vers la page de connexion.

4.4 Panneau d'administration

Le panneau admin est protégé par requireAdmin() qui vérifie le rôle en session. Il offre des opérations CRUD complètes pour :

- **Films :** Créer, modifier (via modale) et supprimer des films. La suppression cascade aux séances et réservations.
- **Salles :** Gérer les salles avec noms et capacités personnalisés.
- **Séances :** Planifier des séances en sélectionnant film, salle, date et heure. Le système valide la disponibilité de la salle en vérifiant les chevauchements.
- **Réservations :** Visualiser toutes les réservations et pouvoir en annuler.
- **Utilisateurs :** Voir les inscrits avec badges de rôle. Un admin peut supprimer tout utilisateur sauf lui-même.

4.5 Validation de la disponibilité des salles

Lors de la création d'une séance, le système empêche les conflits d'horaire via la méthode isRoomAvailable(). Cette méthode calcule les intervalles de temps basés sur la durée du film et vérifie les chevauchements dans la même salle. Deux séances se chevauchent si :

```
débutNouveau < finExistant  ET  finNouveau > débutExistant
```

5. Sécurité

La sécurité a été une priorité de conception à chaque niveau de l'application.

Menace	Protection	Implémentation
Injection SQL	Toutes les requêtes utilisent des requêtes préparées PDO	$stmt->execute([$param]) dans tous les modèles
XSS	Toutes les sorties utilisateur sont échappées	h() encapsule htmlspecialchars(ENT_QUOTES, UTF-8)
Vol de mots de passe	Aucun mot de passe stocké en clair	password_hash(PASSWORD_BCRYPT) + password_verify()
Détournement de session	Expiration après inactivité	Timeout de 30 minutes vérifié à chaque requête
Élévation de privilèges	Contrôle d'accès basé sur les rôles	requireAdmin() sur toutes les routes admin
Vol de cookies	Cookie "se souvenir" en HttpOnly	setcookie() avec le flag httponly = true

5.1 Configuration PDO

- **ERRMODE_EXCEPTION :** Les erreurs BDD sont levées en exceptions plutôt que silencieusement ignorées.
- **FETCH_ASSOC :** Retourne des tableaux associatifs propres.
- **EMULATE_PREPARES = false :** Force le driver à utiliser de vraies requêtes préparées, assurant une protection réelle contre l'injection SQL au niveau du protocole.

6. Interface utilisateur

6.1 Architecture CSS

Les styles sont modulaires, répartis en 16 fichiers dédiés importés via main.css. Chaque fichier a une responsabilité unique : variables (vars.css), reset, navigation, boutons, cartes, formulaires, alertes, tableaux, tags, modales, films, authentification et administration.

6.2 JavaScript

- **main.js :** Auto-fermeture des messages de succès (4s), boîtes de dialogue de confirmation, système d'ouverture/fermeture de modales, mise en surbrillance du lien actif.
- **reservation.js :** Validation en temps réel de l'entrée du nombre de places avec feedback visuel.
- **admin.js :** Remplissage de la modale d'édition de films à partir des attributs data.

6.3 Thème sombre & responsive

L'interface utilise un thème sombre avec accent violet (#7c3aed) et fonds dégradés subtils. Les propriétés CSS personnalisées assurent la cohérence. Le design est responsive : grilles de cartes adaptatives, liens de navigation masqués en mobile (< 700px), tableaux à défilement horizontal.

7. Organisation du travail

7.1 Répartition des tâches

Domaine	Michel	Edwin
Backend	Tous les modèles, contrôleurs, routage, schéma BDD, sécurité, gestion de session	(entraide, débogage)
Frontend	(entraide, débogage)	Toutes les vues (templates PHP), architecture CSS, JavaScript
Intégration	Câblage des routes, validation de disponibilité, profil	Templates layout, vues admin, corrections de bugs

7.2 Méthodologie

Le projet a suivi une approche backend-first : les modèles et contrôleurs ont été construits en premier pour établir la couche de données et la logique métier, suivis des vues qui consomment les données des contrôleurs. Cela a garanti une séparation propre et évité le couplage entre les couches.

8. Installation

8.1 Prérequis

- PHP 8.0 ou supérieur
- MySQL 5.7+ ou MariaDB
- Serveur web (Apache, Nginx) ou le serveur intégré PHP

8.2 Base de données

Importer le fichier de schéma pour créer la base, les tables et les données d'exemple :

```
mysql -u root -p < sql/schema.sql
```

8.3 Configuration

Créer un fichier .env à la racine du projet :

```
DB_HOST=localhost
DB_NAME=cinema
DB_USER=root
DB_PASS=votre_mot_de_passe
DB_CHARSET=utf8mb4
```

8.4 Lancement

```
php -S localhost:8000
```

Puis ouvrir `http://localhost:8000` dans un navigateur.

9. Conclusion

Absolute Cinema implémente avec succès toutes les fonctionnalités demandées : authentification avec gestion de session, consultation des films, réservation de places avec vérification de disponibilité, et un panneau d'administration basé sur les rôles. L'application suit une architecture MVC stricte sans SQL dans les vues ni HTML dans les modèles.

Au-delà des exigences de base, le projet inclut des fonctionnalités supplémentaires : gestion des salles avec capacité par salle, détection des conflits de séances basée sur la durée des films, fonctionnalité « se souvenir de moi » avec cookies sécurisés, et une architecture CSS entièrement modulaire avec thème sombre.

L'implémentation de la sécurité couvre toutes les protections spécifiées : requêtes préparées contre l'injection SQL, htmlspecialchars contre le XSS, hachage bcrypt des mots de passe, et expiration de session après inactivité.