Documentation Technique – Cinéphoria

---

1. Architecture Logicielle de l'Application

1.1 Vue d'ensemble

- **Frontend :** React.js (Vite.js pour le bundling rapide et moderne)
- **Backend :** Node.js avec Express.js
- **Base de données :** MySQL
- **Authentification :** JWT (JSON Web Tokens)
- **Déploiement :** GitHub pour le versionnement et Fly.io pour la mise en production
- **Gestion de projet :** GitHub Projects (Kanban)

1.2 Pourquoi ces technologies ?

- **React.js :** composant réactif, maintenable, bonne gestion des routes avec React Router.
- **Vite.js :** Dev server rapide, meilleure expérience développeur.
- **Node.js + Express.js :** léger, performant pour une API REST.
- **MySQL :** relationnel, robuste et adapté à notre MCD.
- **JWT :** Sécurisation des endpoints via des tokens.

---

2. Réflexions Initiales Technologiques

- **Besoin d'un SPA** pour fluidité côté utilisateur ➔ **React.js**
- **Performances en développement ➔ Vite.js**
- **Sécurité API ➔ JWT + BCrypt**
- **Gestion relationnelle des données ➔ MySQL** (films, séances, utilisateurs)
- **Déploiement scalable ➔ Fly.io**

---

3. Configuration de l'Environnement de Travail

3.1 Prérequis

- **Node.js** (v20+)
- **npm**
- **MySQL** (via XAMPP ou local)
- **Git**

3.2 Installation

```
# Backend
cd Cinephoria
npm install
npm run dev

# Frontend
cd cinephoria-frontend
npm install
npm run dev
```

3.3 Variables d'environnement (.env)

```
MYSQLHOST=localhost
MYSQLUSER=root
MYSQLPASSWORD=p4ssw0rdSuperSecret
MYSQLDATABASE=Cinephoria
MYSQLPORT=3306
SECRET_KEY=supersecretkey
```

---

4. Modèle Conceptuel de Données (MCD)

Entités principales :

- **Utilisateur** (id, nom, email, mot_de_passe, role)
- **Film** (id, titre, description, durée, affiche)
- **Salle** (id, numéro, capacité)
- **Séance** (id, date, heure, film_id, salle_id)
- **Réservation** (id, utilisateur_id, seance_id, nb_places)
- **Avis** (id, utilisateur_id, film_id, note, commentaire)

Relations :

- Un utilisateur peut réserver plusieurs séances.
- Une séance appartient à un film et une salle.
- Un film peut avoir plusieurs avis.

---

5. Diagrammes

5.1 Diagramme d'utilisation

- Utilisateur ➔ Consulter films, réserver séances, laisser avis.
- Employé ➔ Signaler incidents.
- Administrateur ➔ Gérer films, séances et utilisateurs.

5.2 Diagramme de séquence *(Ex : Processus de réservation)*

1. L'utilisateur choisit une séance.
2. Le système vérifie la disponibilité.
3. Génération d'un QR Code pour la réservation.
4. Confirmation de la réservation.

---

6. Plan de Tests

6.1 Tests Backend

- **Jest** pour tests unitaires sur API.
- Tests d'authentification (login/register JWT).
- Tests CRUD films, séances, réservations.

6.2 Tests Frontend

- Vérification navigation React Router.
- Composants dynamiques (affichage films, QR Code).
- Gestion des erreurs d'authentification.

---

7. Déploiement & CI/CD

7.1 Déploiement sur Fly.io

- **fly.toml** configuré.
- Dockerfile pour image backend.
- Workflow GitHub Actions pour déploiement continu.

7.2 CI/CD avec GitHub Actions

- **Branche develop :** tests et développement.
- **Branche main :** production.
- Chaque **merge** vers main déclenche le déploiement automatique via Fly.io.

7.3 Commandes pour déploiement

```
# Build frontend
cd cinephoria-frontend
npm run build

# Déploiement avec Fly.io
fly deploy
```

---

8. Conclusion

Cette documentation fournit l’ensemble des aspects techniques, de la conception à la mise en production, garantissant une livraison continue et fiable de Cinéphoria.
