Réflexions initiales technologiques sur le sujet

Pour le projet Cinephoria, j'ai choisi d'utiliser plusieurs technologies qui, selon moi, répondent parfaitement aux besoins variés de notre application.

Bases de données : PostgreSQL et MongoDB

Pour la gestion des données, j'ai opté pour PostgreSQL et MongoDB. PostgreSQL est un système de gestion de base de données relationnelle open-source, connu pour sa robustesse et ses performances. Il est particulièrement utile pour les données structurées qui nécessitent des relations complexes, comme les comptes utilisateurs et les transactions. Ce qui me plaît avec PostgreSQL, c'est sa capacité à gérer des requêtes complexes et à maintenir l'intégrité des données grâce à sa conformité ACID. En parallèle, MongoDB sera utilisé pour les données non structurées ou semi-structurées. MongoDB, en tant que base de données NoSQL, est extrêmement flexible et scalable. Il est parfait pour stocker des données sous forme de documents JSON, ce qui facilite la gestion des données variées et évolutives que nous pourrions rencontrer dans notre projet.

Langage de requêtes : SQL

Pour interagir avec PostgreSQL, j'utiliserai SQL. C'est un langage de requête bien établi et puissant, idéal pour tirer parti des fonctionnalités avancées de PostgreSQL.

Serveur : Express.js

Pour la partie serveur, mon choix s'est porté sur Express.js. Express.js est un framework pour Node.js qui est à la fois minimaliste et flexible. Il facilite la création de notre serveur web en

gérant les routes, les middlewares et les requêtes HTTP de manière efficace. Cela rend notre application plus performante et plus facile à maintenir.

Frontend : EJS, TailwindCSS et JavaScript

Côté frontend, j'utiliserai EJS pour générer du HTML dynamique. EJS est pratique pour créer des templates HTML avec des données dynamiques, ce qui simplifie le rendu côté serveur. Bases de données : PostgreSQL et MongoDB Langages de requêtes : SQL Serveur : Express.js Frontend : EJS, TailwindCSS et JavaScript Pour le style, TailwindCSS sera utilisé. C'est un framework CSS basé sur des classes utilitaires qui permet de concevoir rapidement des interfaces utilisateur élégantes et cohérentes. TailwindCSS rend le processus de stylisation plus flexible et maintenable. Enfin, pour rendre le site interactif, j'utiliserai JavaScript côté client. JavaScript est indispensable pour manipuler le DOM, gérer les événements utilisateur et communiquer avec le serveur de manière asynchrone via AJAX. Cela permettra d'améliorer l'expérience utilisateur en rendant l'application plus réactive et dynamique.

Backend (Node.js, Express.js)

Pour le backend de notre application, j'ai choisi d'utiliser Node.js avec Express.js. Node.js est une plateforme JavaScript côté serveur qui permet de construire des applications web performantes et évolutives grâce à son modèle asynchrone et basé sur les événements. Sa capacité à gérer un grand nombre de connexions simultanées avec une faible latence en fait un choix idéal pour notre application.

Express.js, étant un framework léger et minimaliste pour Node.js, simplifie grandement le développement de notre serveur web. Il offre une multitude de fonctionnalités robustes pour gérer les routes, les middlewares et les requêtes HTTP, ce qui nous permet de structurer notre application de manière modulaire et maintenable. Grâce à Express.js, nous pouvons facilement implémenter des API RESTful pour notre application, facilitant ainsi la communication entre le frontend et le backend.

Sécurité

Pour garantir la sécurité des applications Cinéphoria, plusieurs mesures seront mises en place pour se protéger contre les vulnérabilités courantes telles que les injections SQL et les failles XSS. Les entrées utilisateur seront validées côté client et serveur pour assurer leur

conformité. Les données reçues seront échappées pour prévenir les injections SQL, et des requêtes préparées seront utilisées pour les interactions avec la base de données. Les sorties affichées seront échappées pour éviter les scripts malveillants. La gestion des autorisations restreindra l'accès aux ressources spécifiques. L'authentification sécurisée sera assurée par des tokens JWT, et les mots de passe seront chiffrés avec Bcrypt. Enfin, des contrôles stricts sur les sessions utilisateurs seront en place pour prévenir les attaques.

Mobile (React Native, Expo, TailwindCSS via NativeWind)

Pour la partie mobile de notre application, j'utiliserai React Native, un framework open-source développé par Facebook qui permet de créer des applications mobiles natives en utilisant JavaScript et React. React Native nous offre la possibilité de partager une grande partie de notre code entre les plateformes iOS et Android, réduisant ainsi le temps de développement et les coûts.

Expo est un outil précieux dans notre stack technologique mobile. C'est un framework et une plate-forme pour les applications universelles React qui facilite grandement le développement, le test et le déploiement de nos applications React Native. Expo fournit un ensemble d'outils et de services intégrés qui simplifient la gestion des dépendances, la compilation du code natif et l'accès aux fonctionnalités spécifiques des appareils, comme la caméra et les notifications push.

Pour le style, j'intégrerai TailwindCSS dans l'application mobile grâce à NativeWind. NativeWind permet d'utiliser les classes utilitaires de TailwindCSS dans les applications React Native, ce qui rend le processus de stylisation rapide, flexible et maintenable. Grâce à cette approche, nous pouvons appliquer des styles cohérents et élégants à nos composants mobiles, tout en bénéficiant de la réutilisabilité et de la simplicité offertes par TailwindCSS.

Desktop (Electron)

Pour la version desktop de notre application, j'ai choisi d'utiliser Electron. Electron est un framework open-source développé par GitHub qui permet de créer des applications de bureau multiplateformes en utilisant des technologies web telles que JavaScript, HTML et CSS. Avec Electron, nous pouvons empaqueter notre application web comme une application de bureau native pour Windows, macOS et Linux, en réutilisant le même code base.

Electron offre également des capacités d'intégration profonde avec le système d'exploitation, telles que l'accès au système de fichiers, les notifications natives, et la gestion des fenêtres. Cela nous permet de fournir une expérience utilisateur riche et native, tout en conservant la flexibilité et la rapidité du développement web.