Configuration de l’environnement de travail

Pour le projet Cinephoria, j'ai décidé d'utiliser plusieurs technologies pour répondre aux besoins variés de notre application. Voici comment j'ai configuré mon environnement de développement :

Choix de l'IDE

Pour commencer, j'ai choisi Visual Studio Code comme IDE. C'est un outil très populaire et puissant, avec plein de fonctionnalités et d'extensions qui facilitent la gestion du code, le débogage et l'intégration avec d'autres outils.

Structure des dossiers

- **Cinephoria/**: Dossier racine du projet contenant tous les sous-dossiers et fichiers nécessaires au projet.
  - **CinephoriaMobile/**: Contient le code source pour l'application mobile développée avec React Native.
    - **app/** : Écrans de l'application, chaque écran correspondant à une vue distincte de l'application mobile.
    - **assets/** : Ressources statiques telles que des images et des fichiers de style.
    - **components/** : Composants React Native réutilisables dans l'application mobile.
    - **constants/** : Configuration de la navigation dans l'application mobile.
  - **client/**: Contient le code source pour l'application web.
    - **public/**: Fichiers statiques accessibles côté client, tels que les images, les fichiers CSS, et JavaScript.
      - **css/** : Fichiers CSS pour le style de l'application web.
      - **images/** : Images utilisées dans l'application web.
      - **js/** : Fichiers JavaScript côté client.
      - **videos/ :** Fichiers vidéo utilisés dans l'application web.
    - **src/**: Code source principal de l'application web.
      - **auth/** : Gestion de l'authentification et des utilisateurs.
      - **components/** : Composants réutilisables de l'application web.
      - **layouts/** : Pages de l'application web, chaque page représentant une route.

- **desktop/**: Contient le code source pour l'application de bureau développée avec Electron.
  - **views/** : Processus principal d'Electron, gérant la logique principale de l'application de bureau.
    - **assets/** : Ressources statiques pour l'application de bureau, telles que des icônes et des images.
  - **renderer/** : Processus de rendu d'Electron, gérant l'interface utilisateur de l'application de bureau.
- **server/**: Contient le code source pour le serveur backend.
  - **config/** : Fichiers de configuration, tels que les paramètres de base de données.
  - **controllers/** : Contrôleurs pour gérer les requêtes HTTP et la logique de l'application.
  - **middlewares/** : Middlewares pour le traitement des requêtes HTTP.
  - **models/** : Modèles de données pour interagir avec les bases de données.
  - **routes/** : Définition des routes de l'application backend.
  - **services/ :** Services pour la logique métier et les interactions avec les ressources externes.
  - **uploads/ :** Gestion des fichiers téléchargés par les employés et l'administrateur.
  - **utils/** : Fonctions utilitaires et helpers pour le backend.
- **sql/**: Scripts SQL pour la gestion de la base de données.

Initialisation du projet

-Web

À la racine du dossier Cinephoria, j'ai initialisé un nouveau projet Node.js avec npm en utilisant la commande npm init. Cela a créé un fichier package.json pour gérer toutes les dépendances du projet. J'ai ensuite installé tous les modules nécessaires au projet, comme Express.js pour le serveur, EJS pour les templates, et TailwindCSS pour le style, avec des commandes comme npm install express ejs tailwindcss.

-Mobile

Pour commencer, j'ai installé react native et Expo go (npx create-expo-app@latest) , un outil précieux pour créer et gérer des projets React Native. Cette étape m'a permis de créer un nouveau projet Expo, où j'ai choisi un modèle de base approprié.

Après avoir initialisé mon projet, j'ai procédé à l'installation des dépendances nécessaires. J'ai installé React Navigation pour gérer la navigation au sein de l'application, et NativeWind (npm install nativewind) pour intégrer Tailwind CSS. J'ai ensuite configuré Tailwind CSS avec NativeWind en créant les fichiers de configuration nécessaires. Cela impliquait notamment

de configurer un fichier de configuration Tailwind et d'importer NativeWind dans le fichier principal de mon application.

Une fois ces configurations en place, j’ai pu installer Android studio dans mon bureau puis j'ai pu lancer l'application en utilisant Eas CLI (npm install -g eas-cli). Cela m'a permis de démarrer le serveur de développement et de tester l'application sur un émulateur Android .

-Desktop

J"ai installé Electron (mkdir my-electron-app && cd my-electron-app), un framework qui permet de créer des applications de bureau en utilisant des technologies web.

Pour configurer le projet Electron, j'ai créé les fichiers nécessaires, y compris le fichier principal et les fichiers de rendu (npm install --save-dev electron). Ces fichiers sont configurés pour charger un fichier HTML et exécuter le processus principal de l'application. J'ai ensuite ajouté des scripts npm dans le fichier de configuration pour faciliter le démarrage et le développement de l'application.

Après avoir configuré ces éléments, j'ai lancé l'application Electron en utilisant les scripts npm, ce qui m'a permis de vérifier son bon fonctionnement. Pour finaliser le processus, j'ai téléchargé et installé Inno Setup. Avec Inno Setup, j'ai créé un script d'installation pour compiler mon projet Electron en un installateur exécutable. Ce script d'installation était configuré pour inclure tous les fichiers nécessaires et définir les options de l'installateur.

Configuration des fichiers principaux

J'ai créé deux fichiers principaux dans mon dossier server : le premier, server.js : Ce fichier gère la configuration et le démarrage du serveur. Il écoute sur un port spécifique et initialise les routes et les middlewares nécessaires. Le second, app.js : Ce fichier configure l'application en elle-même, définissant les routes, les vues, les middlewares et d'autres configurations spécifiques à l'application.

Design Pattern

Pour structurer mon projet, j'ai suivi le design pattern MVC (Model-View-Controller). Ce pattern aide à séparer l'application en trois parties principales : Model pour gérer les données. View pour gérer l'affichage et la présentation. Controller pour gérer les interactions entre le Model et la View.

Gestion du versioning

Enfin, j'ai initialisé un dépôt Git à la racine du projet avec la commande git init. Cela me permet de suivre les modifications du code source. J'ai également connecté mon dépôt local à un dépôt distant sur GitHub en utilisant des commandes comme git remote add origin [URL du dépôt] et git push -u origin main.