Configuration de l'environnement de travail

Pour le projet Arcadia, 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

J'ai organisé mes dossiers de manière logique dans le dossier racine nommé Arcadia :

Arcadia/src/ : Contient tous les fichiers sources du projet.

models/ : Pour les modèles de données.

views/ : Pour les vues de l'application.

controllers/ : Pour les contrôleurs qui gèrent la logique de l'application.

utils/ : Pour les utilitaires et fonctions d'assistance.

config/ : Pour les fichiers de configuration.

routes/ : Pour la définition des routes de l'application.

middleware/ : Pour les middlewares utilisés dans l'application.

Arcadia/public/ : Contient les fichiers statiques accessibles côté client.

css/ : Pour les fichiers CSS.

js/ : Pour les fichiers JavaScript côté client.

images/ : Pour les images utilisées dans l'application.

videos/ : Pour les fichiers vidéo.

Arcadia/sql/ : Contient les scripts SQL pour la création des bases de données, des tables et l'intégration des données.

Initialisation du projet

À la racine du dossier Arcadia, 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.

Configuration des fichiers principaux

J'ai créé deux fichiers principaux à la racine du projet :

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.

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.

Configuration des bases de données

J'ai configuré PostgreSQL pour les données relationnelles et structurées, et MongoDB pour les données non structurées ou semi-structurées. Cela me permet de tirer parti des avantages des deux types de bases de données selon les besoins des différentes parties de l'application.

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.