EpiAirConsole
Cahier des charges
Projet Professionnalisant

Description du projet

EpiAirConsole est un projet qui a pour but de reproduire au mieux les fonctionnalités du site AirConsole. Celui-ci consiste en la création d'une application web permettant de jouer à des jeux faits sur Godot directement sur son navigateur. La particularité principale du projet cependant, est la fonctionnalité d'utiliser des téléphones comme manettes pour un jeu en cours. EpiAirConsole a pour but de simuler un projet avec des attentes professionnelles, et sera donc accompagné d'un cadre professionnel (Outil de gestion de projets, tests automatisés, etc...).

Techo utilisé

Frontend

- Techno: **React.js**

- Pourquoi ? React est une techno très connu avec beaucoup de documentation ainsi qu'une grosse communauté. Il est également simple à connecter avec WebSockets pour les jeux, ainsi que pour les interfaces mobile. Une étude de cas à était réaliser ci-dessous.

CRITÈRE	REACT.JS	VUE.JS	ANGULAR
COURBE D'APPRENTISSAGE	MOYENNE	FAIBLE	ÉLEVÉE
ÉCOSYSTÈME	TRÈS RICHE	BON	TRÈS COMPLET
FLEXIBILITÉ	TRÈS ÉLEVÉE	ÉLEVÉE	MOYENNE
ADAPTIVITÉ À NOTRE SOLUTION	TRÈS BONNE (WEBSOCKET)	BONNE	MOYENNE (LOURD)
RAPIDITÉ DE DEV	RAPIDE	TRÈS RAPIDE	PLUS LENTE

Backend

- Techno: **Node.js**

- Pourquoi ? Node et React vont généralement de pair. Il gère très bien le temps réel pour le multi-joueur grâce à ça complicité avec Socket.io.

CRITÈRE	NODE.JS	FAST API	SPRING BOOT
COURBE D'APPRENTISSAGE	FAIBLE À MOYENNE	FAIBLE	ÉLEVÉE
ÉCOSYSTÈME	TRÈS RICHE	BON	TRÈS RICHE
TEMPS RÉEL (WEBSOCKET)	EXCELLENT (SOCKET.IO)	BON	BON
ADAPTIVITÉ À NOTRE SOLUTION	EXCELLENTE	BONNE (TEMPS RÉEL MOINS ADAPTÉ)	MOYENNE
RAPIDITÉ DE DEV	RAPIDE	TRÈS RAPIDE	LENTE

Base de donnée

- Techno: **MongoDB**

- Pourquoi ? Base de donnée légère et schéma flexible pour gérer facilement des données de jeux évolutives (Profils, scores, sessions...). Facilement scalable et réactif pour une meilleur fluidité pour l'expérience de jeu.

CRITÈRE	MONGODB	POSTGRE SQL	MYSQL
FLÉXIBILITÉ SCHÉMA	TRÈS ÉLEVÉE	MOYENNE	FAIBLE
SCALABILITÉ	EXCELLENTE (PENSÉ POUR)	BONNE	MOYEN
PERFORMANCE	TRÈS BONNE	EXCELLENTE	BONNE
DONNÉE EN TEMPS RÉEL	BONNE	MOYENNE	FAIBLE
ADAPTIVITÉ À NOTRE SOLUTION	EXCELLENTE	BONNE	MOYEN À FAIBLE

Jeux vidéo

- Techno: **Godot Engine**

- Pourquoi ? Godot permet de faire de petits jeux rapide à développer et légé. Il permet également de faire des export simple sur sur web, parfait pour notre solution.

CRITÈRE	GODOT	UNITY	UE5
2D	EXCELLENT	TRÈS BON	FAIBLE
POIDS / LÉGÈRETÉ	TRÈS LÉGER	MOYEN	LOURD
WEB EXPORT	TRÈS BON	MOYEN	MAUVAIS
COURBE D'APPRENTISSAGE	FAIBLE	MOYEN	ÉLEVÉE
ADAPTIVITÉ À NOTRE SOLUTION	EXCELLENTE	BONNE	FAIBLE

Stack suplémentaire

- Bcrypt - Algorithme de Hachage, donc pas de reverse possible. Lent, exprès pour contrer le *brut force*. Il ajoute également un Salt au MDP haché pour le complexifier et pour le rendre unique a chaque fois (contre les Rainbow tables / hash pré-calculé).
- Express - Utile pour gérer les requêtes HTTP entre le front et le back.
- Socket.io - Très utile pour l'envoie de donnée pour des jeux multi-joueurs.
- Cors - Autorisation unique de nos site pour des appels API (bloqué les sites non autorisé).
- Token JWT - Utile pour libérer de l'espace en DB (ne pas garder les sessions utilisateurs ouverte), ainsi que pour la sécurité des comptes client.

Déploiement

- Backend → Render / Pourquoi ? Node.js supporté, https automatique, gratuit
- Frontend → Vercel / Pourquoi ? Rapide, https automatique, CDN mondiale, gratuit
- DataBase → MongoDB Atlas / Pourquoi ? Directement par mongoDB, gratuit