Challenge Semestriel S2 - NodeJS, VueJS

Fernandes Amilcar
Talbi Marouane
Bourdel Théo
Cherigui Amine

Memo

28 juillet 2023

CONCEPT

On a décidé de prendre le sujet *“JV multijoueur tour par tour”*. On a réalisé un jeu de mémoire à deux joueurs, dans lequel les joueurs retournent des cartes pour tenter de trouver des paires correspondantes. Ce type de jeu est également connu sous le nom de "Memory" ou "Jeu de paires".

AUTEURS

Pour réaliser ce projet on était 4 : Théo, Marouane, Amine, Amilcar. Etant donné la contrainte qui était que tout le monde devait faire à la fois du NodeJS et du VueJS, on a décidé de se répartir les tâches par feature (fonctionnalités).

FONCTIONNALITÉS

Notre jeu de mémoire contenait plusieurs fonctionnalités, dont des statistiques, différents modes de jeux, un panel administrateur, un shop pour acheter des skins, un système d’elo (points), un système de ranking et une monnaie virtuelle.

On avait prévu également de faire un chat (in-game) avec possibilité de signaler un message (les admins auraient une ‘boîte de réception’ des signalements). On avait aussi prévu de faire des tournois / rejoindre une partie en spectateur.

CONTRAINTES

On avait comme contraintes le temps réel. Pour ce faire, on a utilisé la library socket.io. On a donc décidé de l’utiliser pour la partie “game” / “gamemode” du projet. Quand un utilisateur clique sur un gamemode, il cherche si une room est en attente (en fonction de ce gamemode), sinon il en crée une.

On avait aussi une seconde contrainte qui était d'utiliser du noSQL, et plus précisement MongoDB. Etant donné que mongoDB est rapide pour récupérer de la data, on a décidé de l'utiliser pour récupérer les utilisateurs côté panel d'administration. Du coup au moment de la registration, on enregistre un User en Postgres, et ensuite il y a un hook qui est appelé (grâce à sequelize) pour insérer le User en question en MongoDB.

On a aussi utilisé MongoDB pour afficher les statistiques d'un utilisateur. Pour ce faire, on a utilisé les aggregates de mongoDB. Etant donné que les aggregates sont très puissantes, on a récupéré toutes les stats (nombre total de game, game win, game lose, win rate, lose rate) en 1 aggregate.

On avait une 3e contrainte qui était d'utiliser Stripe. On a donc intégré Stripe à notre projet. On peut maintenant acheter un Skin avec notre argent virtuel (coins), et de l'argent réel.

La dernière contrainte était que notre projet devait être RESTFUL. On a donc fait attention à renvoyer les bons codes https et les bons objets. On fait aussi attention à bien nommer nos routes en fonction des verbes HTTP en question.

UX / UI

Pour l'UI on a utilisé figma. Si on regarde notre maquette figma, on peut se rendre compte qu'elle n'est pas totalement fidèle à notre projet, parce que l'on utilisé pour se donner une idée de la direction artistique du projet. On a décidé d'avoir un thème assez sombre, pour soulager nos yeux :D.

Concernant l'UX on essayé d'imaginer les choses de la manière la plus simple possible. On a utilisé des modals pour update / créer des choses. On a aussi utilisé des toast, pour avoir un retour sur nos actions.

lien figma :
https://www.figma.com/file/rV5JKCaVZQZgtlBeaF505x/Game-Project?type=design&mode=design&t=kElXm14K6Jy3T75A-1

GIT

lien github : https://github.com/marouaneTalbi/nodejs_vuejs

pseudo github : TheoBourdel ➜ Bourdel Théo

pseudo github : marouaneTalbi ➜ Talbi Marouane

pseudo github : acherigui1 ➜ Cherigui Amine

pseudo github : Amilcar-AFK ➜ Fernandes Amilcar