Compte rendu : MateR, le site de rencontre pour Gamer !

LEGOUEIX Nicolas - ORNIACKI Thomas - COTREZ Leo

8 janvier 2020

Chapitre 1

MateR, c'est quoi ?

Nous devions à l'origine code un site de recontre, qui mettais en relation a l'immage de son homologue Tinder des utilisateurs ayant des profils communs à l'occasion de recherches. Les profils sont présentés à l'utilisateur à la suite (par ordre de similitude). Ce dernier à alors le choix de passer au suivant / précédant ou de "matcher" ce profile.
Nous avons voulu rendre ce projet plus original en en faisant un site de rencontre pour gamer. On ne met alors plus en relation des personnes habitant proches les unes des autres ou aimant le même film, mais des personnes jouant aux mêmes jeux sur les mêmes plateformes, ayant le même style de jeu... On ne cherche plus l'âme soeur, mais le coéquipier idéal, ce qui est tout aussi difficile à trouver, voir plus encore...

Chapitre 2

Choix de Tchnos et d'API

En plus de ce qui était déjà inclus dans la base de code fournie, nous avons ajouté :

- Busboy : Afin d'avoir plus de flexibilité dans nos envois au serveur, nous avons choisi d'inclure le packet npm BusBoy (voir ici). En effet, nous trouvions très limitant le fait de devoir utiliser les urls pour envoyer les données au serveur via la méthode get. Nous voulions par exemple envoyer des formulaires au serveur, ce qui s'est revelé impossible avec get. (la page de création renvoie plus d'une 10aine de champs) De plus, le passage par URL impose de ne pas utiliser de caractères spéciaux. Pour utiliser la méthode post, busboy s'est imposé.
- js-sha512 : Pour plus de sureté lors de la communication des mots de passe, nous encodons ces derniers en sha512. Cette API (voir ici)

Chapitre 3

Architecture du Programme et origanisation

Notre programme est divisé en 7 menus :

- *Menu d'authentification*, le premier vu.
- De ce menu, un utilisateur peut *créer un compte*, il est alors redirigé sur le menu d'enregistrement.
- Une fois connecté, l'utilisateur arrive sur son *menu Profile* où il peut voir et modifier ses informations.
- L'utilisateur peut ensuite *chercher* des équipiers en renseignant ses critères.
- Le programme lui présente alors le *résultat* des recherches.
- L'utilisateur peut également discutter avec ses match via le *menu Tchat.*
- Enfin, il peut consulter la liste de toutes ces conversations dans le *menu principal.*

Nous avons donc un fichier par menu, contenant l'ensemble MVC de ce menu. Côté serveur, toutes les bases de données sont représentées dans des fichiers JSON. Nous avons 7 fichiers :

- games.json : la liste des jeux supportés par MateR, avec les plateformes sur lesquelles ils sont jouables, et un booléen qui permet de savoir si le jeu est crossplay ou non (et donc si on peut matcher des utilisateurs ne jouant pas sur la meme plateforme).
- languages.json : la liste de languages supprotés.
- levels.json : la liste des niveaux de jeux possibles (newbie, casual, experienced, pro player)
- locals.json : la liste régions du mondes et les pays sélectionables pour ces dernières.
- playstyles.json : la liste des styles de jeux (chill, serious, try hard).
- users.json : la liste des utilisateurs et leur données (id, mail, username, password, bio, genre, année de naissance, region, country, liste de

langues parlées, conversations, jeux (dont la plateforme, playstyle et niveau) et plateformes de communication utilisées).

— vocals.json : la liste des plateformes de communications utilisables (Discord...).

Enfin, chaque conversation entre utilisateur est stockée dans un json nommé : tag1_tag2_date, où tag1 est le plus petit des deux tags, tag2 le plus grand et date est la date au format JourMoisAnnee (année au format long).
Dans ces fichiers figurent des listes contenant l'id de l'envoyeur, le contenu du message, son état (vu/pas vu), l'heure d'envoi et la date d'envoi.

Chapitre 4

Problèmes rencontrés

Le principal problème que nous avons recontré a été le temps : à cause d'un autre projet très difficile, nous n'avons pû nous mettre à travailler sur celui-ci avant le 1er janvier ce qui ne nous a clairement pas donné assez de temps. Trois ou quatre jours supplémentaires auraient été bienvenus. Nous continueront d'ailleurs a commit des modifications sur le github dans les prochains jours. Une autre grosse difficulté (qui nous a couté du temps précieux) a été de comprendre le squelette de code fourni : le modèle MVC était une grosse inconnue pour nous malgré les informations trouvées en ligne, et la manière dont doit fonctionner la relation client/serveur n'était également pas très claire . Nous pensons que le squelette de code aurait mérité des explications en cours.