Rapport de projet L1 CMI

Nom du projet : ProjectDrinX (non définitif)

Université de Montpellier - Faculté de Sciences
Cursus Master Ingénierie (CMI)
Année 2021 - 2022

Mathieu COLMON
Antoine SEGUIN

Sommaire

Développement du projet

1. Back-end
    - *BeamIO*
    - Structure de l’API
    - Gestionnaire de parties
    - Contrôleur de jeu
    - Les mini-jeux
2. Front-end
    - API Front
    - UX / UI
    - Les mini-jeux
3. Mise en place des tests unitaires

Bonus

1. Classement général
2. Scalabilité horizontale

Conclusion

Introduction

1. Choix du jeu

Dans un premier temps, nous avons essayé de trouver un concept qui sorte de l’ordinaire, et nous voulions que notre jeu soit accessible au plus grand nombre. Nous avons donc décidé que notre jeu devait être porté sur mobile, pour permettre à un plus grand nombre d'être susceptible de pouvoir y jouer. Une application Android est envisageable, mais concernant IOS, les conditions pour poster une nouvelle application sur le store ou en téléchargement sont relativement compliquées. Ainsi, un format mobile sur le WEB nous semble être un bon compromis (déjà utilisé par Ornikar par exemple).

Concernant l'idée du jeu, nous avons essayé de déterminer un besoin dans une population que nous voulions la plus large possible. C'est pourquoi nous avons décidé de cibler une population “jeune”, susceptible d'essayer le jeu plus facilement et de devenir des utilisateurs actifs. Maintenant que nous avons donc la population, nous avons essayer de trouver un concept qui puisse parler au plus grand nombre. Pour ça, nous avons essayer de trouver des jeux au format similaire qui ont pu avoir un succès. Les exemples ne manquent pas (Gartic phone, Le Loup Garou en ligne, Skribbl.io...), nous avons donc décidé de faire un jeu massivement multijoueur, étant donné que les jeux les plus appréciés sont tous de ce type. Nous avons également pu en déduire que les jeux qui ont rencontrés un grand succès sont également dans l’idée “jeu de société”, mais adapté à un format en ligne. En continuant nos recherches, nous avons mis l’accent sur l’identification d’une idée de jeu de société qui puisse être en lien avec les besoins liés à l’actualité et aux nouveaux enjeux. Suite à la pandémie qui frappe le monde depuis maintenant deux ans, l’une des conséquences à été les nombreux confinements et restrictions de déplacements. Suite à ça, la société à dû s’adapter pour garder contact et continuer à vivre malgré les conditions extrêmes. De très nombreuses activités sont donc devenus impossible, comme celles qui impliquent des regroupements ou des déplacements. Les jeunes ont été particulièrement touchés par l’impossibilité de pouvoir se regrouper pour partager des soirées autour d’un verre, ainsi que simplement pour partager des moments simples. Suite à ça, l’une des alternative qui s’est imposé comme une solution a été de l’émergence des “apéro-visio”. Nous avons donc décidé de cibler ces nouveaux moyens de partager des moments légers à

distance pour concevoir un jeu pour accompagner ces instants. Dans un soucis de rendre le jeu utilisable dans des conditions plus traditionnelles, nous avons tout de même pensé la possibilité que ce dernier puisse être utilisé avec plusieurs joueurs dans une même pièce.
C’est ainsi que nous avons déterminé que nous voulions développer un jeu pour accompagner les soirées alcoolisés à distance ou en physique, sur un gameplay massivement multijoueur (interactions entre les joueurs).

Le but du jeu repose sur la proposition de plusieurs mini-jeux à un groupe de joueurs qui forment une partie. A chaques tours, le mini-jeux change. A l’issue de chaques tours, les points sont compatiblisés, et des gages peuvent être donnés aux joueurs.

2. Cahier des charges

- Pour mobile / tablette : interface joueur
- Pour PC / TV : tableau des scores en temps réel
- Possibilité pour le créateur d’une partie de gérer les joueurs et les paramètres de la partie
- Possibilité d’activer un mode "sans alcool" pour tous les joueurs de la partie ou un joueur
- Possibilité de rentrer la liste des alcools disponibles
- Messages de préventions autour de l’alcool et de ses dangers (notamment au volant)
- Possibilité de choisir un mode “équipe” ou “individuel” (comptage des points)
- Système de sélection des jeux intelligent (variation entre des duels, des duels entre équipes ou tous les joueurs contre le jeu) en fonction du mode de jeu désiré
- Classement général de tous les joueurs (voir **Bonus**)

Organisation du projet

1. Organisation du travail

Avant tout, voici un échéancier des dates que nous nous sommes fixés pour porter le projet à son terme dans les délais demandés.

20 février 2022	Développement de BeamIO terminé. Charte graphique et maquette WEB du jeu terminé. Liste des mini-jeux à implémenter définitive. Début du développement du jeu (détaillé en bas).
27 mars 2022	Développement terminé.
17 avril 2022 (date imposée)	Vidéo de présentation du projet. Rapport complet.

Le développement de la librairie BeamIO (détaillé plus bas) a déjà commencé. Cette librairie est conçue dans le but d'être utilisée par le jeu, mais sera considérée comme une librairie "externe", utilisable dans d'autres projets à terme (voire disponible sur NPM).
Le développement du jeu commencera donc le 20 février, comme déterminé dans le calendrier prévisionnel ci-dessus. Dans cette première phase du développement, la structure (backend et frontend) du projet va être développée. Le but est donc de lancer de manière simultanée le développement du front et du back. Mathieu s'occupera du backend, avec notamment dans un premier temps, la mise en place du gestionnaire des joueurs et des parties, ainsi que le système chargé de contrôler les mini-jeux. Dans ce même temps, Antoine réalisera le frontend, ce qui inclut les différents menus, éléments graphiques, ainsi que le système de liaison à l'API.
La deuxième phase de développement commencera une fois que ces éléments auront été finalisés.
Dans cette deuxième phase, l'objectif sera d'implémenter chaque mini-jeu indépendamment (avec le frontend et le backend). Étant donné que la charte graphique aura déjà été déterminée, le travail nécessaire en frontend sera plus de l'assemblage d'éléments visuels, ou l'ajout de contenus simples. Le backend, quant à lui, devrait être relativement court pour chaque mini-jeu, étant donné que le but principal est d'avoir une expérience amusante mais simple (ce qui a un impact direct sur la logique derrière

chaque jeu).
Au 27 mars 2022, le but est d'avoir terminé l'entièreté du développement. Ainsi, à compter de cette date, nous pourrons réaliser des tests de stabilités (bien que des tests sont déjà prévus pendant les différentes phases de développement, pour minimiser les potentielles erreurs à corriger à l'issue du développement). Nous pourrons également réaliser la vidéo de présentation du projet, et la rédaction du rapport final.

2. Choix de la stack

Le principe de l'application web est idéal pour ce genre de jeu. Performant, très facilement accessible, multiplateforme, évolutif, etc... Nous avons donc décider d'utiliser les technologies suivantes :

- Les langages *JavaScript* / *TypeScript* ainsi que *HTML* / *CSS*
- Utilisation de NodeJS pour le back-end (API)
- Le framework front-end *VueJS*
- La librairie *BeamIO* (Module de networking réalisé dans le cadre du projet)
- Les services *Google Cloud* pour l'hébergement web

Analyse du projet

1. Un module de networking

Nous avons la volonté d'optimiser le plus possible la communication entre client et serveur afin de privilégier la stabilité ainsi qu'obtenir des latences et une utilisation du processeur les plus faibles possibles.

Pour cela nous développerons un module de networking appelé *BeamIO* qui gèrera l'encodage, le formatage, la compression et la transmission des données. Celles-ci transiteront par un tunnel WebSocket. La librairie *BeamIO* permettra aussi de simplifier le développement en gérant le typage et la structure de chaque paquet échangeable.

2. Un gestionnaire de parties

Ce module permet la création d'une partie que les autres joueurs pourront rejoindre par le biais d'un lien ou d'un code.
Le créateur d'une partie sera en mesure de gérer différents paramètres, ainsi que les joueurs qui ont rejoint sa partie.

3. Un contrôleur de jeu

Celui-ci aura pour but de gérer les différents tours. A chaque tour correspond un mini-jeu. Il doit donc en choisir un en fonction de l'avancée de la partie et des paramètres choisi par le créateur de la partie.

4. Les mini-jeux

Il faudra également coder chaque mini-jeu. Cela inclut la partie back-end et la partie front-end.
La liste définitive des mini-jeux qui seront ajoutés sera connue prochainement, comme détaillé plus haut.
Chaque mini-jeux appartiendront à une catégorie (comme jeu de réflexion, jeu de culture, jeu de stratégie ou de rapidité par exemple).
A l'issue de chaque tour, les points seront comptabilisés.

5. Une application PWA

L'application web sera développée avec le framework front-end *Vue 3*. Cela permet de faciliter le développement, de permettre une meilleure compatibilité grâce à *Babel* et de meilleures performances grâce à *Webpack*. Le principe de la PWA permettra de générer un paquet Android (APK). L'interface doit être réactive, moderne et intuitive.

Développement du projet

1. Back-end

BeamIO (Librarie de networking)

Cette librairie doit gérer la totalité de la communication entre serveur et client. Pour permettre une communication symétrique on décide de se baser sur des WebSockets car c'est le protocole compatible web le plus proche du TCP. Pour des raisons de performances et de stabilité, les paquets envoyés contiendrons seulement les données (sans schéma).

Une trame sera donc constituée d'un premier octet contenant le type de paquet (256 possibilités), puis des données brut. Les données brut seront obtenues en transformant l'objet que l'on souhaite envoyer (objet JSON) en une liste en retirant tous les indices (en gardant seulement les valeurs) en fonction d'un schéma qui contient le nom de chaque indice et le typage de chaque valeur.

Par exemple, l'objet :

```
{
  username: 'test',
  session: 'XXXXXX',
}
```

Transformé avec le schéma :

```
{
  username: String,
  session: String,
}
```

Sera d'abord organisé en une liste :

```
['test', 'XXXXXX']
```

Puis formatté dans un buffer que l'on peut représenter par la chaine de caractère :

```
'test^XXXXXX'
```

Le caractère `^` est utilisé ici comme séparateur.

On peut utiliser un second caractère pour éviter les problèmes si le caractère séparateur est présent dans un des membres de la liste ⇒

```
['te^st', 'XXXXXX'] -> 'te!^st^XXXXXX'
```

Ce caractère est équivalent au '\' en JSON.

On considère que les différents schémas sont connus par le serveur et le client. Ainsi, les deux peuvent communiquer de manière extrêmement fiable sans échanger de données inutiles.

Une autre optimisation serait de compresser les valeurs de type nombre et booléen. En effet, lors de l'envoi classique d'un nombre, 1 octet est utilisé pour chaque chiffre. Pour un booléen c'est pire : *true* pèse 4 octets et *false* en pèse 5. Alors qu'on pourrait stocker 1 booléen par bit soit 32 ou 40 fois moins.

Ainsi, on choisi de faire cette compression au moment du passage de l'objet à la liste. Les booléens sont eux tous placés à la fin du paquet et codés chacun sur un bit.

Afin de s'assurer de la stabilité de tout ce système, une procédure contenant de nombreux tests aléatoires est exécutée à chaque modification du module.

Lorsque différentes méthodes sont possibles, on détermine la meilleure en effectuant des benchmarks. Des différences de performances d'exécution du code JavaScript peuvent exister entre les navigateur et NodeJS. On effectue donc ces benchmarks sur différentes plateformes.

Structure de l'API

Cette étape consiste seulement à faire une liste de chaque requête nécessaire et de la typer. Cela consiste concrètement à déclarer des requêtes *BeamIO* ce qui va assigner un identifiant (le premier octet des trames) et générer le schéma correspondant.

Gestionnaire de partie

Cette couche est la base du back-end. Elle permettra de router les différentes requêtes au contrôleur de jeu concerné, ou d'en créer un si la demande en est faite. Elle est donc l'intermédiaire entre les contrôleurs de jeu et les clients. Le routage se fera au niveau de l'endpoint du socket.

Contrôleur de jeu

Son principal rôle est de gérer les tours en proposant un mini-jeu à chacun de ceux-ci. Il est crée avant le début de la partie et détruit à la fin de celle-ci.

Il a également pour rôle de gérer les utilisateurs ainsi que leur statut. En effet, on peut imaginer plusieurs scénarios (problème de connexion, pause toilettes, etc...) qui causerait une déconnexion du client en pleine partie (techniquement, un simple rechargement de la page suffit). Cependant, il ne faut pas exclure le joueur de la partie, mais plutôt le faire revenir automatiquement dans la partie en cours lors de la reconnexion. Le contrôleur conservera donc chaque instance de *Player* crée au moment du début de la partie mais supprimera l'endpoint du socket. Le gestionnaire de partie lui fournira le nouvel endpoint si le client se reconnecte. A chaque début de tour, un mini-jeu sera choisi en fonction du nombre de joueurs disponibles (actifs) et seulement ceux-ci participeront.

Les mini-jeux

Une instance d'un mini-jeu est construite par le contrôleur de jeu au début de chaque tour. Celle-ci aura accès aux instances *Player* et pourra donc communiquer directement au front-end en utilisant ses propres paquets. Lorsque le tour est terminé, elle annoncera grâce à un callback au contrôleur de jeu que le tour est terminé.

2. Front-end

API Front

La partie front-end de l'API sera développée en parallèle du backend. Les modules seront d'abord utilisés à des fins de tests, puis implémentés dans la partie web.

UX / UI

L'interface doit être accessible, moderne et réactive. Dans un soucis d'homogénéité globale, une charte graphique va permettre d'établir un thème pour l'ensemble du jeu. Cette charte graphique sera réalisée en même temps que la maquette WEB. Ces dernières seront terminées à une date ultérieure.

Les mini-jeux

Lors de l’ajout d’un mini-jeu, la majorité du travail se fera au niveau du frontend. Comme déjà détaillé plus haut, les mini-jeux reposeront sur une logique simple et intuitive, ce qui va forcément impliquer un travail moins important au niveau du backend.
En revanche, un travail plus important pourrait être nécessaire sur les mini-jeux qui demanderont des éléments graphiques spéciaux. Un ensemble d’éléments graphiques auront déjà été développé durant la première phase, pour permettre un travail d’assemblage des éléments sur les mini-jeux simples.

3. Mise en place de tests unitaires

La pratique du Test Driven Development sera au cœur de la réalisation du projet. En effet tout au long du développement, de nombreuses procédures de test seront mises en place pour assurer la stabilité après chaque modification majeure.

Bonus

> ✔️ Certaines fonctionnalités peuvent être ajoutées à la fin du développement dans le cas où on ajouterait un service intermédiaire tel Google Cloud Firestore ou équivalent. Cette base de données pourrait être lue par n’importe qui mais l’écriture nécessiterait une clé privée.

Classement général

On peut imaginer qu’à la fin d’une partie le controleur de jeu envoie à une instance Cloud Firestore le score de chaque joueur et son pseudo. Ainsi, on peut créer un classement général des joueurs en fonction de leur nombre de points.

Scalabilité horizontale

On peut imaginer qu'au démarrage en production d'une instance d'un back-end, celui-ci lit dans une variable d'environnement le domaine ('io1.example.com') et l'ajoute à la base de données Firestore en tant que serveur disponible. Ainsi, lors de la création d'une partie, le client peut choisir à quel serveur il va se connecter. Les liens/codes d'invitation contiendront (dans le premier caractère par exemple) l'information leur permettant de se connecter au bon serveur.

Conclusion

Ce projet n'étant encore qu'à ses prémices, nous ne pouvons que spéculer sur sa finalité.
Il n'en reste pas moins qu'il est l'opportunité pour nous de mettre en pratique certaines compétences dans un cadre plus stricte et plus proche du monde professionnel.
Nous sommes particulièrement pressés de pouvoir porter ce projet à son terme, et espérons pouvoir relever les différents épreuves techniques qui se poseront durant son développement.