Ensimag ISI/MMIS 2A
2023/2024

Projet de spécialité

Objectif du projet :

Ce projet vise à créer une version en ligne du jeu MONOPOLY, mettant en vedette les rues emblématiques de Grenoble. Les joueurs pourront profiter de cette adaptation web pour jouer en ligne ou localement sur leur ordinateur.

Julian COUX
Ryan ELFATIHI
Loan GATIMEL
Yanis JOULOU
Ilyan MONESTIER-MEZE

Documentation du projet :

Sommaire :

I. Résumé du projet :

Le projet GRENOPOLY vise à créer une version web du jeu MONOPOLY, spécifiquement adaptée à la ville de Grenoble. Cette adaptation cherche à offrir une expérience de jeu captivante tout en mettant en avant les caractéristiques uniques de Grenoble. La création de ce jeu permet de pouvoir jouer à une version alternative du jeu de société classique. Aussi, c'est une version en ligne qui est développée afin de pouvoir être accessible plus largement et plus aisément.
Ce projet a pour but principal de reproduire fidèlement le jeu Monopoly tout en s'imprégnant au maximum des éléments propres à Grenoble tels que les lieux emblématiques de la ville qui remplaceront les propriétés du jeu ou encore la création de carte chance et caisse de communauté qui sont adaptées à l'environnement grenoblois. Le jeu pourra être à 6 maximum sur un même ordinateur (local) ou sur plusieurs machines (online).
Le jeu a été réalisé en utilisant les langages de programmation Javascript, HTML, CSS et les webSockets. De nombreux tests Cypress sont utilisés pour valider le bon fonctionnement du jeu.
Les résultats de ce projet seront quantifiés par le bon développement de la version web ainsi que par la bonne intégration des éléments grenoblois.

II. Introduction :

Ce rapport explore la conception d'une adaptation en ligne du jeu MONOPOLY, mettant en scène les rues emblématiques de Grenoble, que nous avons appelé, le GRENOPOLY. Inspirés par notre affection pour ce jeu intemporel, nous avons choisi de le moderniser afin de le rendre accessible à un public plus large, où que ce dernier se trouve.

Notre motivation découle de notre volonté de préserver le plaisir des jeux de société tout en surmontant les obstacles géographiques qui pourraient entraver nos moments de divertissement partagé. En effet, la possibilité de jouer ensemble, même à distance, représente un aspect fondamental de notre projet, offrant ainsi la promesse de perpétuer les liens sociaux forgés autour des jeux de notre enfance.

III. Description conceptuelle :

1. Besoins utilisateurs

Nous allons ici recenser tous les besoins utilisateurs relatifs à notre application.

Nous avons identifié la simplicité d'utilisation comme un besoin clé pour nos futurs joueurs. De ce fait, chaque personne pourra avoir une prise en main rapide et efficace.

Dans les besoins qui s'appliquent spécifiquement au développement d'un jeu, nous avons pensé à la sauvegarde d'une partie. Ainsi, le créateur d'une partie pourra la sauvegarder afin de la relancer plus tard, que ce soit une partie locale ou en ligne.

Ensuite, afin d'apporter une touche de nouveauté à ce jeu ancestral, nous avons décidé d'intégrer des statistiques pour chaque joueur, regroupant ses performances globales. Ceci peut amener à la création d'un classement en ligne entre les joueurs. On peut également apporter un récapitulatif à la fin de chaque partie avec quelques informations clé sur son déroulement.

2. Analyse des cas d'utilisation

Afin de pouvoir étudier les cas d'utilisations, nous avons réalisé des diagrammes d'utilisation.

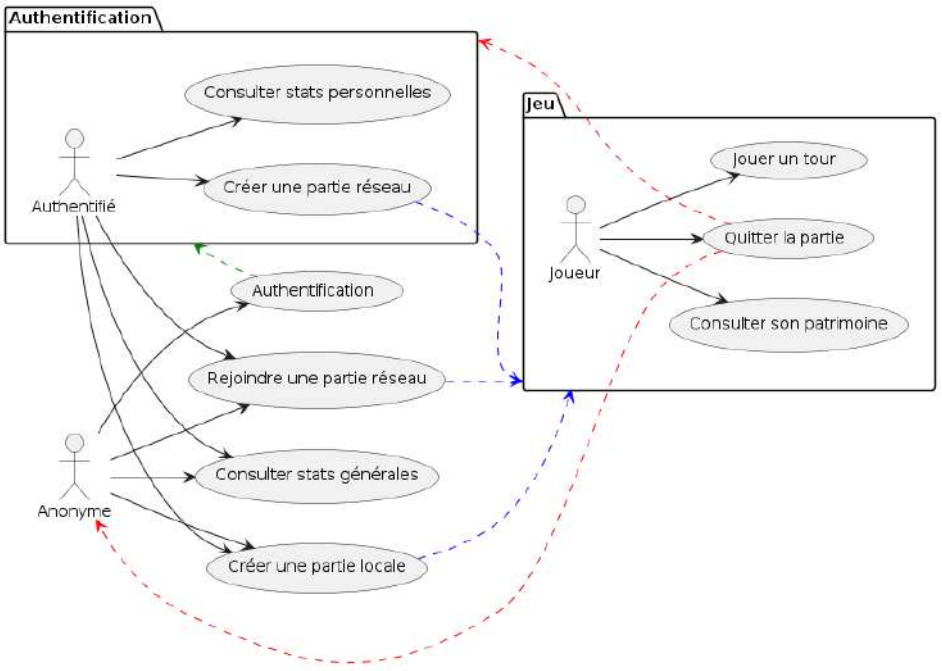

Figure 1: Diagramme général d'utilisation du jeu *Grenopoly*

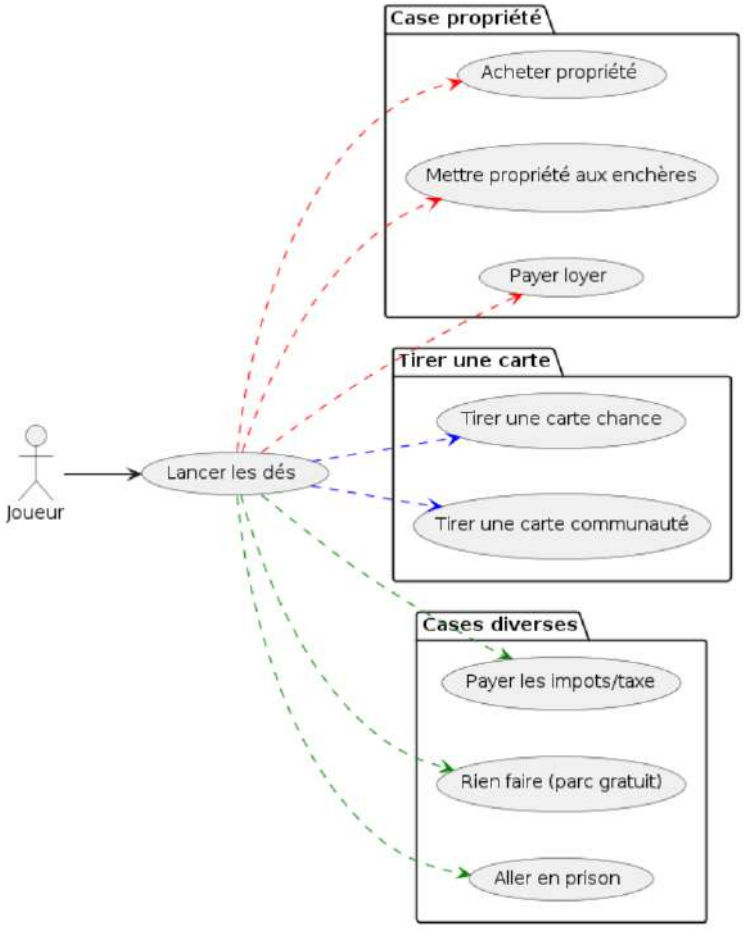

Figure 2: Diagramme d'actions après un lancer de dés

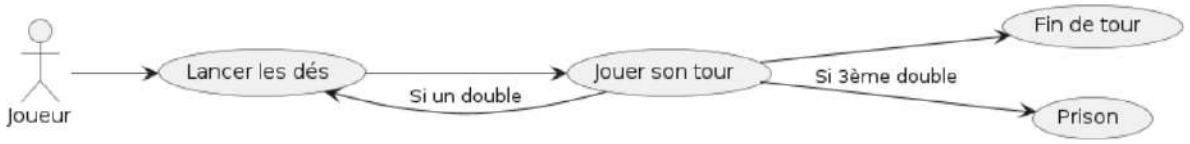

Figure 3: Diagramme de déroulement d'un tour d'un joueur

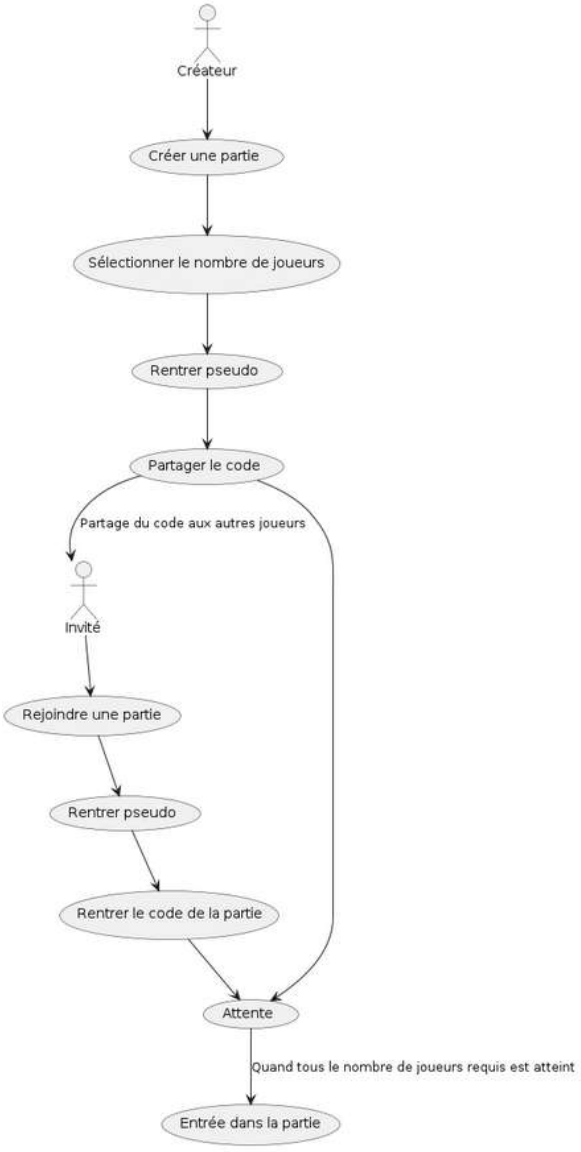

Figure 4: Diagramme général d'utilisation du jeu *Grenopoly*

IV. Présentation de la solution :

Pour mener à bien notre projet, il était impératif de définir un cahier des charges précis et d'adopter une méthode de travail rigoureuse. Avant même d'aborder les aspects techniques de la programmation, nous avons pris le temps de jouer ensemble au Monopoly afin de saisir les besoins et attentes des utilisateurs lorsqu'ils jouent sur leur ordinateur. Cette immersion nous a permis d'appréhender les éléments essentiels pour offrir une expérience de jeu satisfaisante.
Ensuite seulement, nous avons réfléchi au visuel de notre site WEB.

1. Imagination du visuel

La première étape de notre démarche a été d'imaginer le visuel de notre site web. Nous avions pour défi de créer un plateau de jeu original, fidèle à l'esprit du Monopoly tout en reflétant l'identité unique de la ville de Grenoble. Nous nous sommes donc largement inspirés du jeu classique, tout en ajoutant notre touche personnelle pour le personnaliser à notre image.

En parallèle, nous avons élaboré des maquettes détaillées pour chaque page du site, notamment les pages de connexion et de jeu. Cette approche nous a permis de définir clairement notre direction esthétique et de garantir un environnement de jeu convivial et attractif pour les utilisateurs.

2. Programmation du visuel des pages

Une fois notre vision visuelle clairement définie, nous avons entamé la phase de programmation du site. En utilisant principalement HTML, CSS et Javascript, nous avons commencé à donner vie à nos maquettes en produisant les premières versions des pages.

Nous avons opté pour une approche sans intégration, en nous appuyant sur les fonctionnalités de base de HTML et CSS pour créer les différentes composantes visuelles du site. Pour la partie en ligne cependant, nous avons eu besoin du module Socket.io pour connecter différents clients ensemble.

a. Page d’accueil

Figure 5: Page d’accueil

b. Page de création de partie locale

En ce qui concerne la page de création d’une partie locale, les utilisateurs peuvent ajouter un joueur en l’identifiant avec un pseudo. Le nombre de joueurs doit être compris entre 2 et 6 joueurs, sinon une alerte s’affiche à l’écran.

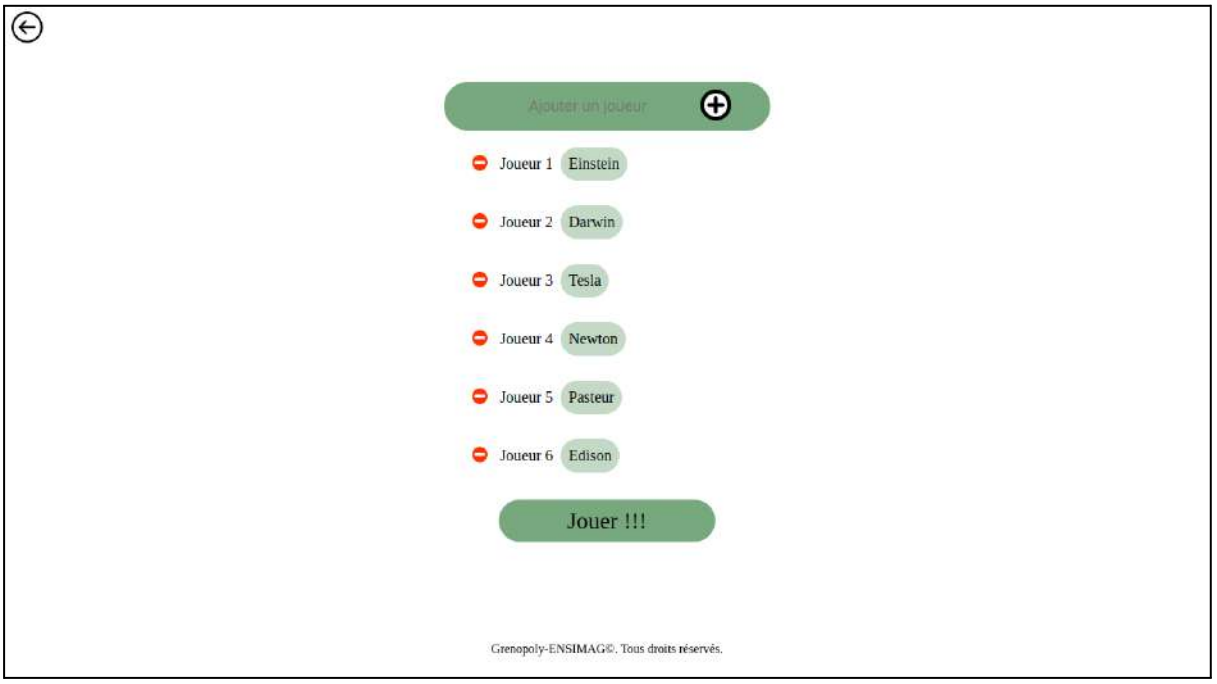

Figure 6: Page pour créer des joueurs

De plus, nous avons ajouté des contraintes sur le format du pseudo que l'utilisateur peut entrer. Le nom d'un joueur doit tout d'abord être unique, être composé de lettres avec ou sans chiffres (pas seulement des chiffres) et pas de caractères spéciaux. Si l'utilisateur met des espaces dans son pseudo, ceux-ci seront simplement supprimés.
Le format des noms des joueurs est strict car le nom sera utilisé dans le programme comme une manière d'identifier le joueur, donc les contraintes demandées sont les contraintes du javascript pour identifier correctement un string (notamment lorsqu'ils sont utilisés comme des clés dans un dictionnaire).

c. Page de jeu avec le plateau

Une contrainte majeure était l'affichage du plateau de jeu de GRENOPOLY, qui devait être carré et entièrement visible, quel que soit la taille de l'écran. Cette décision, prise pour simplifier notamment le déplacement des pions, nous a amenés à positionner le plateau sur la partie gauche de l'écran, occupant ainsi toute la hauteur disponible, tandis que la partie droite était réservée à l'affichage des données de la partie.

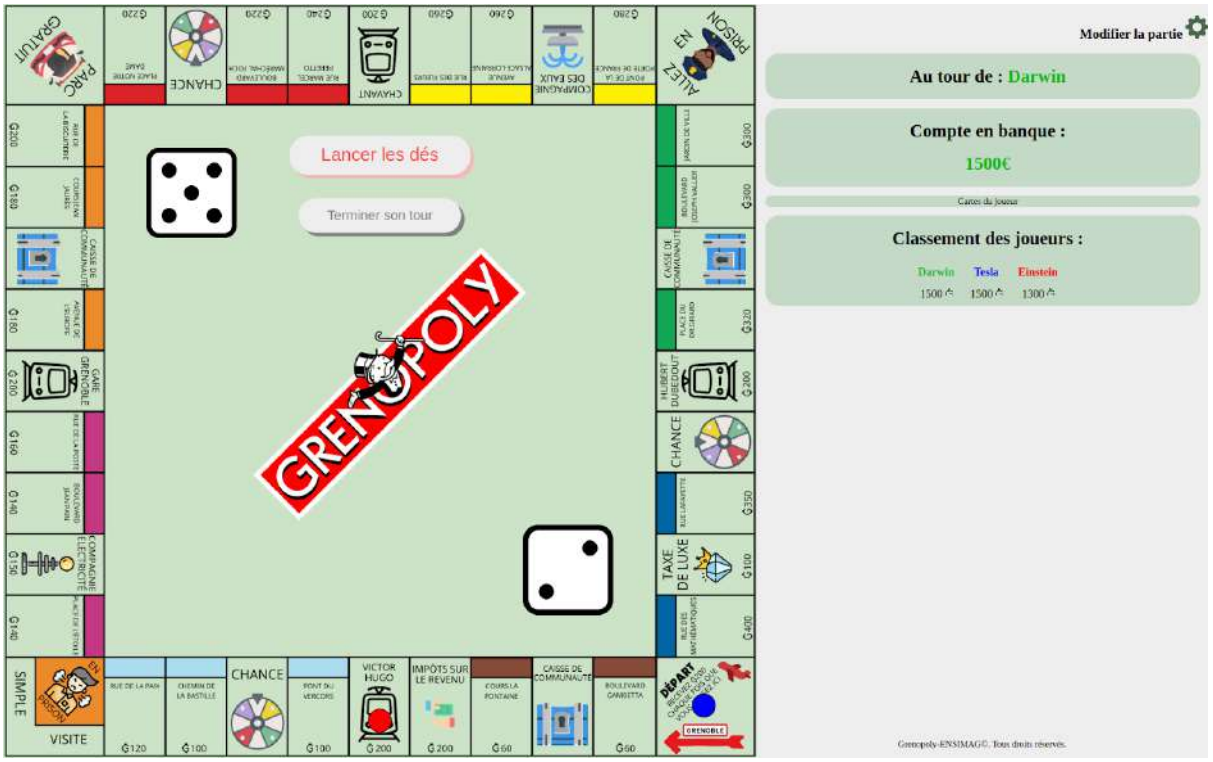

Figure 7: Page de jeux en local

Sur cette partie droite, nous avons identifié les informations essentielles à afficher :

- Le nom du joueur actif;
- Son solde bancaire;
- Les cartes qui sont en sa possession (pas encore implémenté à l'heure actuelle);

- Le classement des joueurs en fonction de leur solde bancaire;
- Un bouton paramètre qui permet d'entrer dans une page où l'utilisateur peut avoir des informations sur les règles du jeu, supprimer un joueur ou finir la partie.

Par ailleurs, afin d'optimiser l'espace utilisé sur le plateau, nous avons décidé d'utiliser la partie centrale, qui initialement est vide, pour afficher les informations relatives au tour actuel. Nous avons décidé d'y inclure :

- Un bouton qui permet de lancer les dés, lorsqu'il est activé;
- Un bouton qui permet de terminer son tour lorsqu'il n'y a plus aucune action à réaliser pour le joueur;
- Une animation des 2 dés qui affichent un nombre aléatoire;
- Les alertes utiles lors de la partie (aller en prison, pas assez d'argent, …);
- Les informations d'une carte lorsqu'on tombe dessus :
    - Si personne ne possède l'objet, on affiche un bouton "Acheter".
    - On affiche un bouton "Mettre aux enchères" afin de pouvoir mettre en enchère la carte si on ne souhaite pas l'acheter
    - Un bouton payer le loyer si elle appartient à quelqu'un d'autre.

Cette approche d'affichage dynamique nous permet de présenter de manière fluide et concise les informations pertinentes à chaque étape du jeu, sans surcharger l'écran. Grâce à JavaScript, nous pouvons choisir d'afficher ou de masquer certains éléments en fonction de leur pertinence, assurant ainsi une expérience de jeu fluide et intuitive pour les utilisateurs.

De plus, nous avons prévu le cas où un utilisateur appuierait par accident sur la touche F5 ou perdrait la connexion subitement. Nous ne voulons pas que celui-ci perde les données de sa partie en cours. Pour cela, nous utilisons le localStorage du navigateur pour sauvegarder et restituer les données relatives à la partie lors du chargement du plateau.

d. Page création partie en ligne

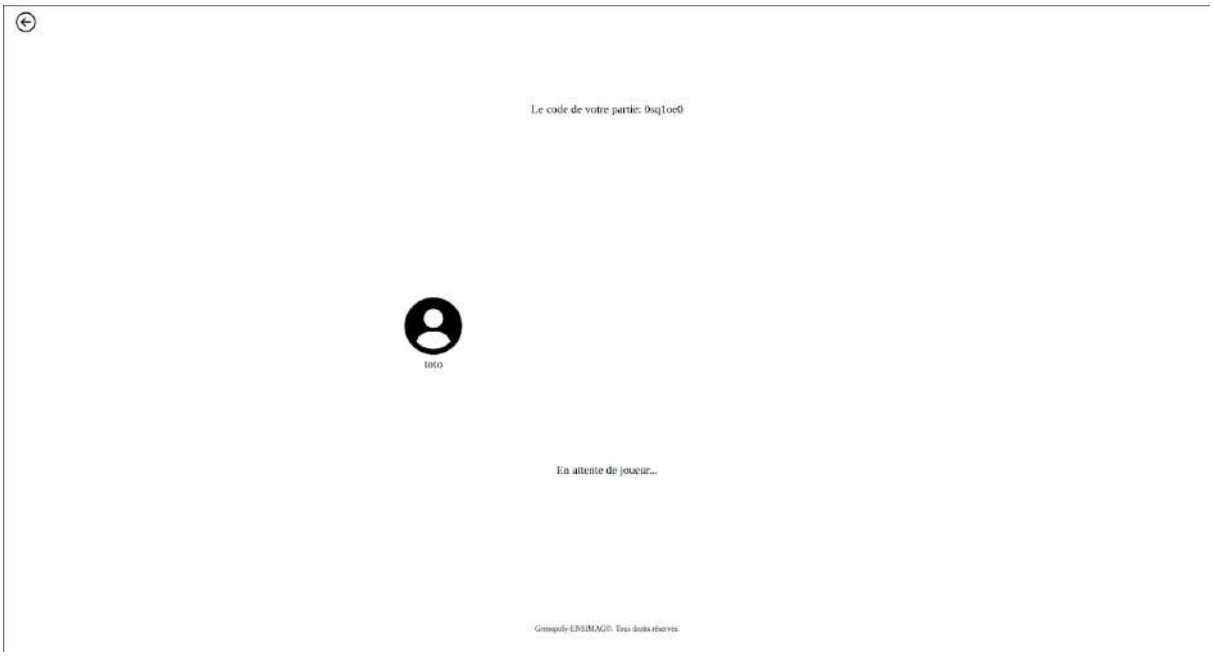

Figure 8: Page de création en ligne

e. Page pour entrer dans une partie en ligne

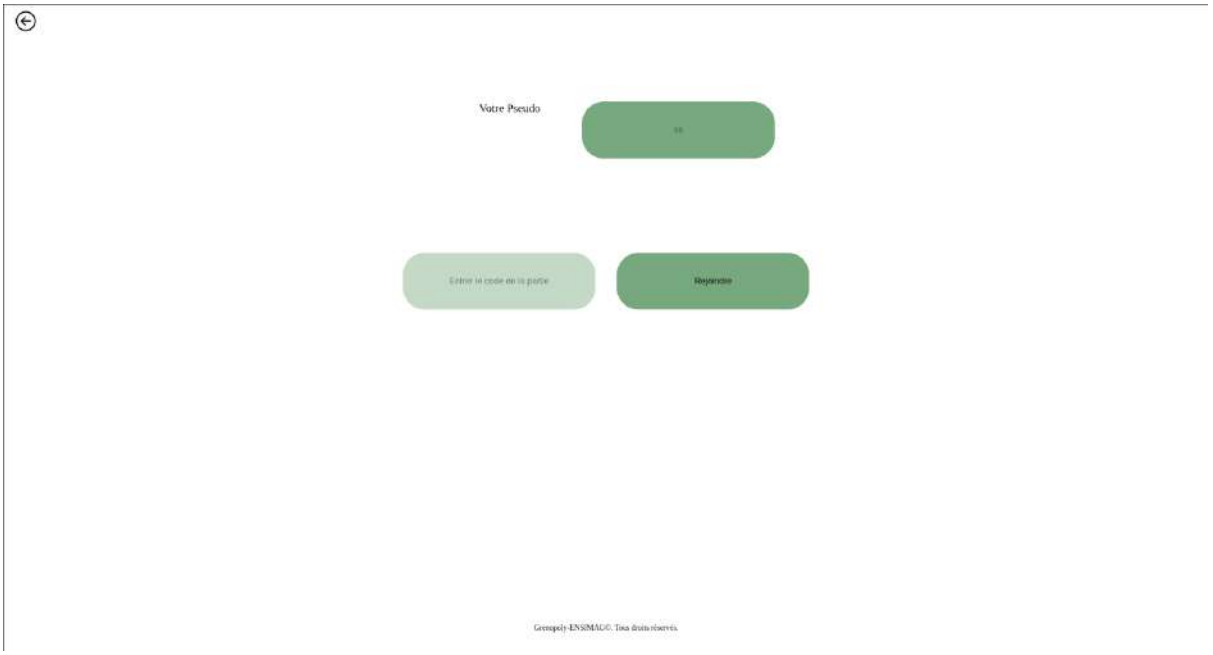

Figure 9: Page de connexion en ligne

Comme on le voit dans la Figure 1, un utilisateur peut soit créer une partie ou rejoindre une partie déjà créée.

Lorsqu'il crée une partie, il choisit d'abord le nombre de joueurs qui va participer puis son nom de joueur, une fois que c'est fait, le code de sa partie s'affiche. Chaque fois qu'un utilisateur rejoindra la partie, il le verra s'afficher sur son écran.

Lorsqu'il rejoint une partie, l'utilisateur doit choisir son nom de joueur puis rentrer un code de partie valide. Si c'est le cas, il est alors en attente et peut voir qui se trouve dans la même partie que lui.

Lorsque la partie a fini de se remplir, le jeu se lance automatiquement. Le premier joueur est décidé aléatoirement.

V. Structure du code :

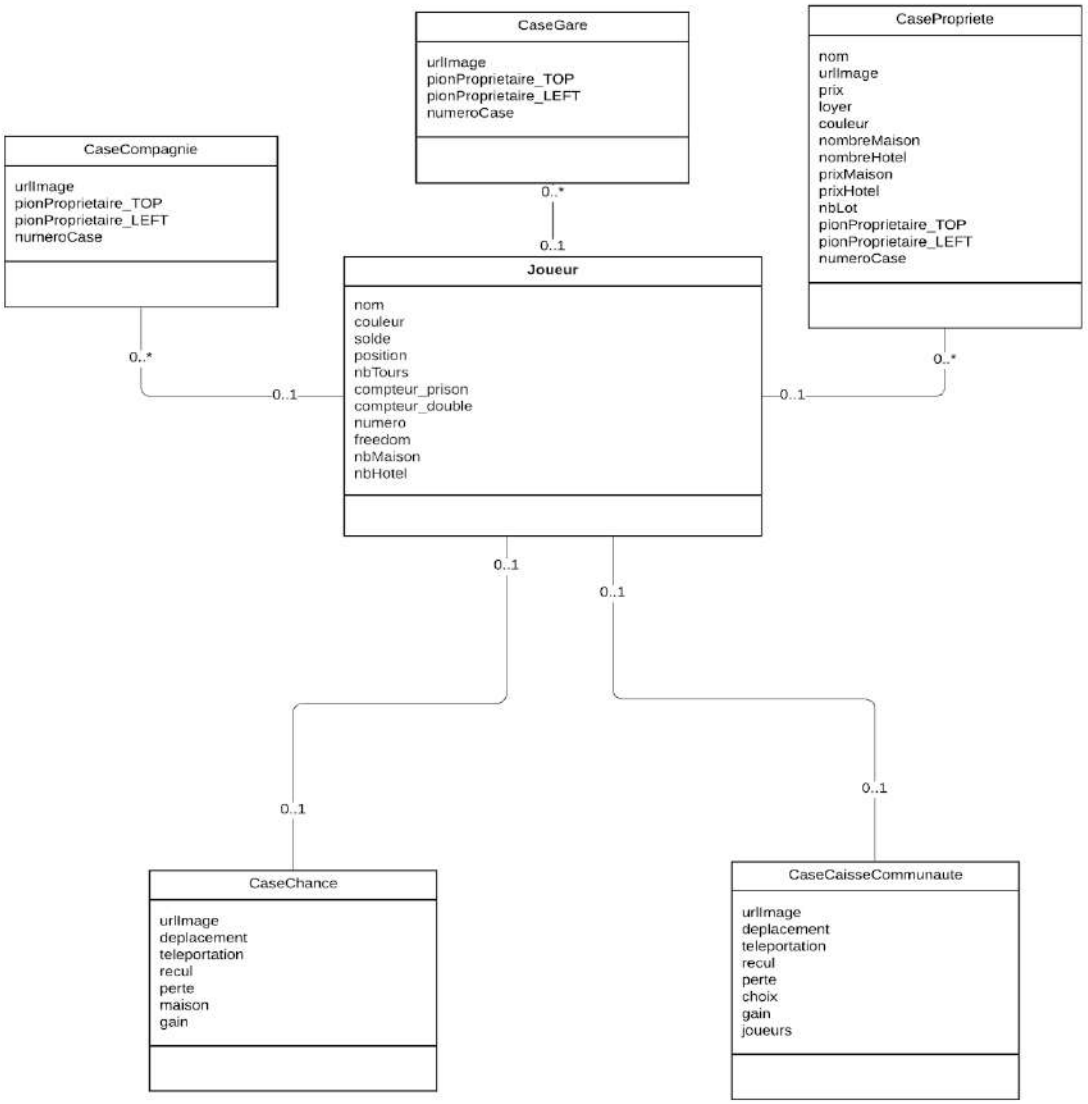

Figure 10: Diagramme de classe

Dans le code principal tous les joueurs sont stockés dans un dictionnaire avec pour clé leur pseudo ce qui permet des accès moins coûteux dans le reste du code.

Un joueur possède un dictionnaire appelé propriétés qui a pour clé une couleur et pour valeur un tableau avec potentiellement des propriétés correspondant à cette couleur. Ce dictionnaire sur les couleurs est utile pour vérifier par exemple qu'un joueur possède toutes les propriétés d'une certaine couleur.

Un joueur peut posséder aussi deux tableaux correspondant à des gares ou des compagnies et pour finir potentiellement une carte libérer de prison.

Toutes les cases Propriétés, Gares et Compagnies sont stockées dans des dictionnaires avec comme clé la position sur le plateau et comme valeur l'objet correspondant.

Toutes les cartes Chances et Communauté sont stockées dans des tableaux, à l'initialisation d'une partie on mélange le tableau puis on traite ces tableaux comme une pile de cartes, on pioche la carte à l'indice 0 et on la remet en queue une fois traité.

En ce qui concerne la sauvegarde dans une partie locale, nous avons une méthode qui récupère toutes les informations de la partie stockées dans un dictionnaire. De ce fait, nous pouvons les récupérer après le rafraîchissement de la page.

Pour la partie en ligne du Grenopoly, nous nous sommes appuyés sur les websocket de Socket.io.

Le serveur stocke toutes les rooms (un objet ce qui correspond à un partie) dans un tableau. Chaque room a un ID, un tableau de joueurs, une longueur attendue, un booléen à vrai si la room est pleine et le numéro du joueur dont c'est le tour.

Plusieurs événements (ex. une connexion) sont attendus par le serveur et par les clients. Lorsqu'un événement est reçu, du code est exécuté. Voici quelques exemples :

Lorsqu'un utilisateur se connecte, ses données sont envoyées au serveur. Si il a déjà un ID de room alors le serveur ajoute l'utilisateur dans le tableau des joueurs de la room. Sinon il crée une room et ajoute l'utilisateur dedans.

Lorsqu'un utilisateur se déconnecte, il est supprimé de la liste des joueurs de la room. Si cet utilisateur est l'hôte, la partie est dissoute.

Le serveur gère la gestion des tours en incrémentant le numéro du joueur dont c'est le tour et en envoyant le nom de ce dernier. Chaque joueur stocke dans son url, son nom de joueur et peut donc se reconnaître si c'est son tour.

Lorsque la partie est remplie, les utilisateurs sont renvoyés vers la page du jeu. Un script se lance alors pour déterminer le mode de jeu (local ou online) et ainsi ajouter le script correspondant (*PlateauLocal.js* ou *PlateauEnLigne.js*).

Lors d'une partie, après que le joueur se soit déplacé, il envoie les informations de déplacement du pion au serveur. Ce dernier le renvoie aux autres joueurs pour qu'ils mettent à jour leur plateau. La même chose est mise en place lors d'un achat de propriété.

Le cas des enchères est un peu particulier et le système est différent du système en local. Quand un joueur décide de placer sa propriété aux enchères, le

serveur prévient tous les joueurs qui affiche la fenêtre aux enchères. Chacun leur tour, les joueurs enchérissent ou quittent les enchères. La somme enchérit est envoyée au serveur qui prévient les autres joueurs. Il envoie aussi le nom du prochain joueur à enchérir. Le dernier en lice est débité de la dernière somme donnée. Cependant si tous les joueurs quittent l'enchère sans enchérir alors la propriété est remise en jeu.

VI. Résultats actuels :

1. Développement du jeu

Dans l'optique de la création du jeu GRENOPOLY, nous avons d'abord conçu le plateau de jeu. Pour ce faire nous avons identifié les lieux emblématiques à intégrer et nous avons au préalable créé les différentes images à l'aide de canva.

Dans un deuxième temps nous avons implémenté les fonctionnalités dites "basiques" telles que le déplacement des pions sur le plateau, le paiement des loyers ou encore l'achat de propriété.

Peu après cela, nous nous sommes chargés de la gestion des cases très spécifiques comme la case prison, la case départ, les cases impôts et la case du parc gratuit permettant une meilleure gestion de l'argent au cours de la partie.

Nous avons ensuite traité le cas des cartes caisse de communauté et chance que nous avons conçu de manière à représenter au mieux la ville de Grenoble.

Parallèlement, nous avons implémenté les fonctionnalités pour un jeu en ligne en créant une page de création de partie et une pour rejoindre. Pour le jeu en lui-même nous avons essayé de réutiliser au maximum ce qui avait été fait pour le local (ex. fonction de lancé de dés, déplacement des pions), malgré tout, nous avons dû modifier et rajouter des fonctions. Par exemple, pour que ce soit le serveur qui gère les tours de joueurs et les enchères.

Nous avons par ailleurs aussi traité la gestion de la sauvegarde des données dans le localStorage afin de pouvoir retrouver nos données en cas de rafraîchissement de la page de jeu.

Enfin nous avons traité le cas des enchères lors de l'achat des propriétés.

2. Tests de fonctionnalités

Afin de tester le bon fonctionnement de notre jeu, nous avons utilisé deux approches en parallèle.

La première est l'utilisation de tests cypress, lancés automatiquement sur gitlab après chaque commit, qui testent les fonctionnalités de notre jeu et assurent que l'application reste globalement fonctionnelle après chaque commit. Nous pouvons également lancer ces tests depuis une commande dans notre terminal.
Nous avons un test principal qui consiste en une partie deux joueurs où tous les coups sont décidés à l'avance. Cela permet de vérifier l'intégrité globale de notre jeu même si certaines parties en sont exclues. On a par exemple fait le choix de ne jamais arriver sur une case où l'on doit tirer une carte chance ou une carte de communauté puisque cela amène de l'aléatoire dans un test déterministe.
Ainsi, une batterie de petits tests visant à tester chaque composant de notre jeu a été créée : les créations et lancements d'une partie, le test de chaque type de case, l'ajout d'une maison, sortir de prison… Prenons l'exemple de la case prison, nous allons vérifier qu'un double nous fasse bien sortir, qu'on ait la possibilité de payer, que le paiement se fasse automatiquement au bout d'un certain nombre de tours …
Enfin, dès que nous découvrons un bug manuellement, nous ajoutons un test cypress afin de vérifier qu'il ne réapparaisse pas dans une version future du code.

La seconde est la plus évidente. Nous testons, nous même, manuellement notre jeu afin d'en trouver des bugs. Tout d'abord, nous avons cherché des comportements qui pourraient mener à un crash de notre application. Ensuite, nous nous sommes attardés au bon fonctionnement du jeu, c'est-à-dire à vérifier que chaque action amène la bonne réponse. Par exemple, nous avons fait attention à ce que la gestion des comptes soit bonne, que les tours s'enchainent bien ou encore que l'arrivée sur une case ait l'effet désiré.

VII. Discussion :

1. Points à retenir et limites

On peut retenir la bonne implémentation des fonctionnalités principales permettant de pouvoir continuer la conception du jeu avec des bases solides. Cependant, on peut aussi noter qu'uniquement une partie locale peut être jouée pour l'instant. Ils restent alors à implémenter toute la partie nécessaire au bon fonctionnement d'une partie multijoueur. De plus, un manque de retours utilisateurs peut être compromettant notamment sur certains choix subjectifs comme le choix des différents lieux de Grenoble.

2. Pistes à explorer

Bien que le projet avance de manière satisfaisante, plusieurs pistes d'améliorations sont possibles. Il pourrait être pertinent de vectoriser les différentes images présentes dans le jeu afin d'assurer une adaptation selon les différentes tailles de l'onglet.

De plus, l'obtention de retours utilisateurs pourraient être bénéfiques pour mieux cibler les aspects du jeu à améliorer.
Aussi on pourrait envisager lors de la conception d'une partie multijoueur de rendre compatible les différents navigateurs entre eux.
Enfin, il serait intéressant de rendre notre site responsive, de manière à ce qu'il s'adapte correctement à la taille de l'écran. Par la suite, on voudrait rendre le site utilisable sur téléphone en trouvant une nouvelle manière d'afficher les informations sur un petit écran.

3. Gestion du projet

Le travail étant assez conséquent, nous avons décidé de diviser le projet en plusieurs étapes et plusieurs parties. Premièrement, nous avons séparé celui-ci en une partie hors ligne et une autre en ligne. Nous avons aussi dédié une troisième partie du projet aux différents tests qui pouvaient être fait. Le plus gros du travail a été la partie hors ligne car la majorité des fonctionnalités du monopoly ainsi que le plateau ont dû être réalisés à ce moment-là. Afin de modéliser nos besoins et nos points de vue, nous avons réalisé tout au long de notre projet des schémas de conception, de classe ainsi que des prototypes imagés (pour les interfaces par exemple). Ceci nous a permis de fixer des objectifs clairs. Par exemple, nous voulions au début implémenter un système d'authentification et de statistiques comme expliqué sommairement en Figure 1. Grâce à une base de données, stocker le nombre de victoires, de défaites, le temps de jeu… des joueurs ayant un compte. Même s' ils ont pu changer et être modifiés au cours du projet pour des raisons diverses comme le temps, cela nous a grandement servi. Ce projet s'est déroulé parallèlement au cours de web, ce qui nous a permis de s'exercer mais cela nous a aussi freiné. Nos connaissances étant limitées au début, c'était compliqué de savoir comment réaliser toutes les fonctionnalités imaginées.

VIII. Conclusion :

En conclusion, ce projet GRENOPOLY permet de créer une version en ligne fonctionnelle du monopoly adaptée à la ville de Grenoble. Il est adapté à la fois pour des parties locales et des parties multijoueurs. Par ailleurs, nous avons toujours conservé cette volonté de concevoir un jeu MONOPOLY propre à la ville de Grenoble et à son patrimoine tout au long de ce projet.