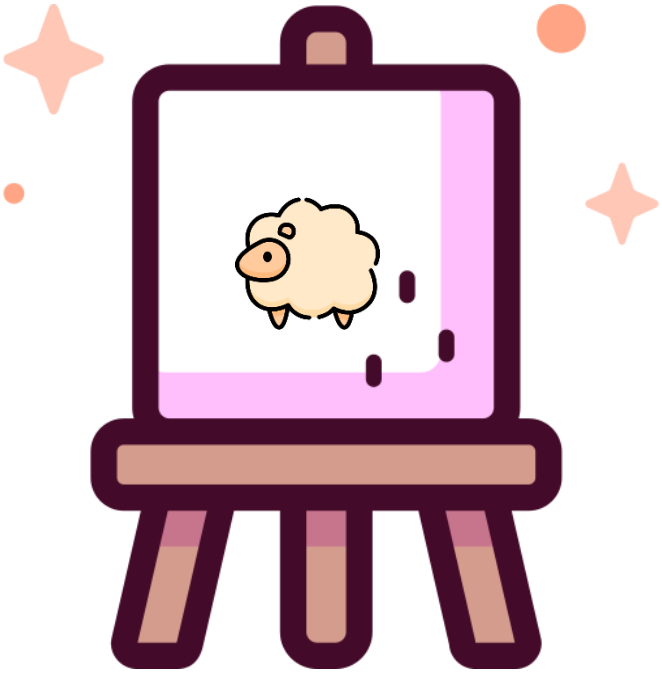


« DESSINE-MOI UN MOUTON »

Rapport d'un projet haut en couleurs

https://github.com/Ekhynox/Dessine-moi-un-mouton

RESUME
Mise en place d'un Pictionnary en ligne au travers de l'apprentissage de React.

Claire COUTURIER-PETRASSON
Kevin DOMINGUES
Ibrahim UCAR
LifProjet, printemps 2021

Gabriel RADANNE
Chargé de Projet

A - Présentation du projet

1) Introduction

Le but de l'UE est de développer une application en trinôme sur toute la durée du semestre, et de voir jusqu'où nous pouvons aller en nous auto-formant. Nous avons choisi comme sujet *Jeu de plateau multijoueur en ligne* car nous souhaitons tou.te.s les trois poursuivre nos études en master TIW (web) à Lyon 1. Ce sujet nous semblait donc tout indiqué. Plus spécifiquement, nous nous sommes tournés vers le Pictionnary car c'était le jeu de plateau qui semblait représenter le défi de conception le plus intéressant pour nous (par rapport aux autres jeux de plateau que nous connaissons).

Rappel :

> *« Le Pictionnary est originellement un jeu de société créé en 1985 dont le but est de faire deviner un mot, une expression ou une idée à son partenaire dans un temps limité, à l'aide d'un dessin. Le nom est une contraction des mots anglais picture et dictionary. »*[1]

Nous avons repris cette définition et l'avons adaptée à un jeu en ligne multijoueur. La partie « utilisateur » de notre projet se compose donc d'une page de connexion, une page *salle d'attente,* une page pour le dessinateur et une page *stream* pour les autres joueurs qui doivent deviner quel mot le dessinateur essaie de représenter.

Le projet est codé en Javascript et JSX (une extension syntaxique du Javascript). Nous avons décidé d'utiliser React comme Framework.

> *React est une bibliothèque JavaScript libre développée par Facebook depuis 2013. Le but principal de cette bibliothèque est de faciliter la création d'application web mono-page, via la création de composants dépendant d'un état et générant une page HTML à chaque changement d'état.*

Nous avons choisi React, car c'est un Framework particulièrement bien documenté. Par ailleurs, il existe de nombreux tutos pour apprendre à s'en servir, il est facile à implémenter grâce aux composants React (*React components*[2]) et facile à déployer via GitHub Pages[3]. Nous avons également utilisé des composants React tels que React-color, React-icon, Hex-rgb, etc…

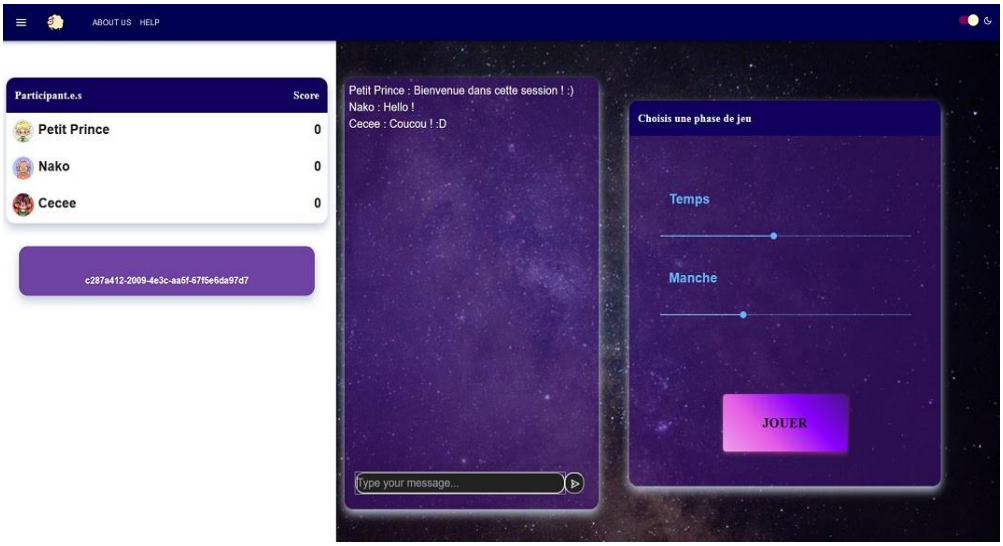
[1] Définition tirée de la page Wikipédia officielle du jeu : https://fr.wikipedia.org/wiki/Pictionary
[2] https://fr.reactjs.org/docs/react-component.html
[3] https://pages.github.com/

2) Comment se déroule une partie ?

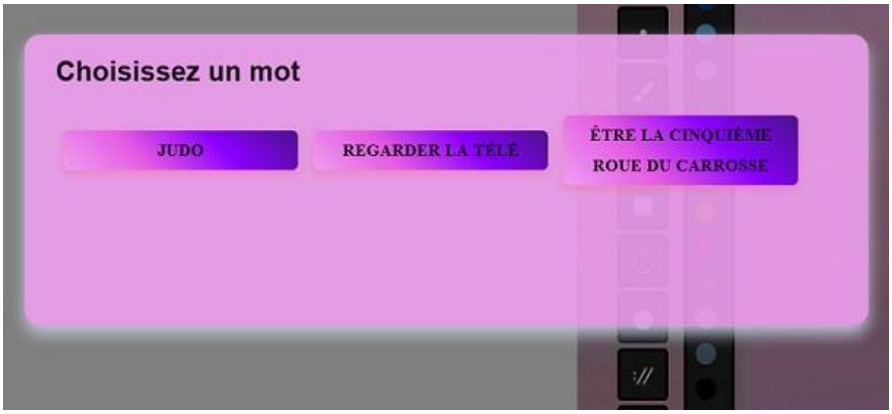
Le participant numéro 1 va se connecter via la page de connexion en choisissant un avatar et un pseudonyme. Si son avatar et son pseudonyme sont valides, cliquer sur le bouton « connexion » l'enverra dans la salle d'attente. Il y générera un identifiant de connexion (id) qu'il devra envoyer aux autres participant.e.s pour qu'ils et elles puissent le rejoindre dans la même instance de jeu (=salle d'attente).

Au fur et à mesure que les participant.e.s se connectent à la salle d'attente, leurs pseudonymes et leurs avatars apparaissent dans le tableau « participants ». Ce faisant, le tableau « choisir une phase de jeu » disparait pour ces nouveaux participant.e.s, afin que seul.e l'hôte de la session ai la main sur les paramètres de la manche, et que lui / elle seule puisse lancer la partie.

On peut choisir deux critères pour une partie : le temps de dessin en secondes (entre 10 et 120 secondes) et le nombre de manches (entre 1 et 10). Le nombre de manches correspond au nombre de fois où chaque joueur / joueuse va dessiner.



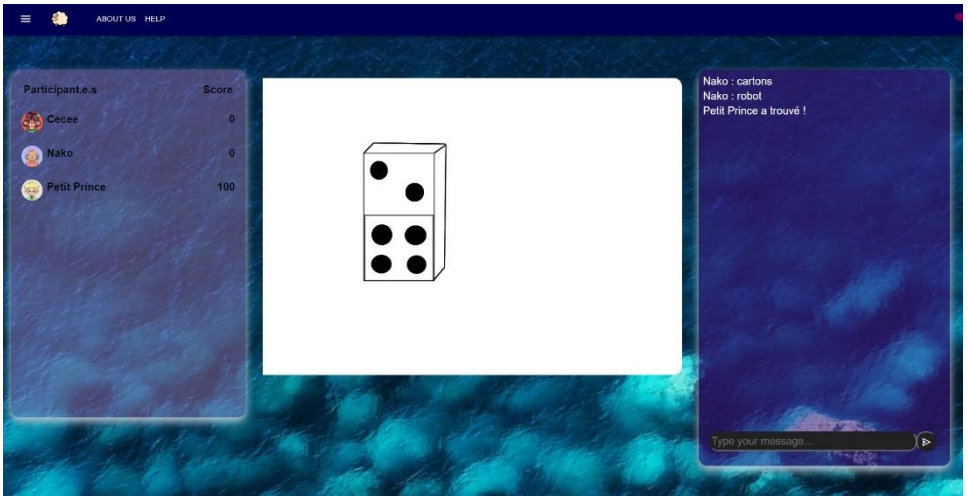
Une fois que tous les participant.e.s sont là, le joueur qui a créé la session (participant 1) clique sur le bouton « jouer » pour commencer une partie et tous les participant.e.s sont redirigé.e.s sur une page de jeu (la page de dessin pour le premier joueur, la page *viewers* pour les autres).



Celui ou celle qui dessine doit choisir un mot ou une expression parmi trois propositions. Une fois le mot choisi, le minuteur se lance et les participant.e.s ont jusqu'à la fin du temps imparti - l'un.e pour dessiner, les autres (les *viewers*) pour deviner de quel mot il s'agit. Pour deviner, ils et elles doivent taper le mot ou l'expression dans le chat.

Lorsque l'un.e des participant.e.s tape le « bon » mot dans le chat, au lieu du mot, un message apparait indiquant quel.le participant.e a deviné le mot, et le score de ce/ cette participant.e s'incrémente.

A la fin du temps imparti, une fenêtre modale apparaît sur la page du dessinateur ou de la dessinatrice, indiquant que sa phase de jeu est terminée. En cliquant sur le bouton « prochaine manche », il ou elle passe le rôle de dessinateur à un.e autre participant.e et devient *viewer*.



Si tou.te.s les participant.e.s ont été dessinateur ou dessinatrice au moins une fois, le rôle de dessinateur /dessinatrice revient au premier ou à la première participant.e et ainsi de suite.

A la fin, un tableau de score apparaît avec le score final de tous les participant.e.s

B – Organisation

La conception du projet s'est déroulée en plusieurs étapes. Les deux premières semaines ont été des phases d'apprentissage et d'observation : nous nous sommes auto-formés à React js grâce à différentes ressources en ligne (principalement **Openclassroom**[4] et le site officiel de React) et en parallèle, nous avons testé des applications similaires à notre projet (telles que Skribl.io[5] ou encore Gartic Phone[6]) afin d'analyser leurs points forts et leurs points faibles, et d'avoir une idée plus précise de ce à quoi nous voulions faire ressembler notre application.

Cela nous a permis de faire un cahier des charges comprenant une liste des fonctionnalités que nous souhaitions réaliser : des fonctionnalités « obligatoires » (celles sans lesquelles l'application n'est pas réellement « jouable ») et des fonctionnalités « optionnelles » (celles que nous avions envie de faire, mais dont la réalisation était conditionnée au temps de développement restant). Le cahier des charges a bien sûr évolué au cours du projet, à mesure que nous avancions dans le développement et que nos idées se précisaient.

Nous avons créé un GitHub pour pouvoir travailler en parallèle, « chacun de notre côté », mais nous avons également fait beaucoup de *peer-programming*[7] : l'un.e d'entre nous partage son écran en codant, tandis que les autres donnent leur avis, proposent de nouvelles solution, etc… L'avantage principal est bien sûr de rendre la conception plus uniforme. A l'échelle individuelle, cela donne à tous les membre de l'équipe une vision globale du code, et permet à chacun.e de bien connaitre toutes les parties du projet, et pas seulement celles que l'on a codées soi-même.

Grâce au travail réalisé en amont, nous avons pu répartir le travail relativement équitablement. Nous nous sommes tou.te.s impliqués dans toutes les parties du projet, mais nous avons également attribué à chacun.e un domaine « de référence » en fonction de ses compétences : Ibrahim est responsable de toutes les fonctionnalités concernant Peer-js, Claire s'est spécialisée dans les composants react et tout

[4] https://openclassrooms.com/fr/courses/7008001-debutez-avec-react
[5] https://skribbl.io/
[6] https://garticphone.com/fr
[7] Pour en savoir plus sur cette méthode de travail : https://www.dnd.fr/2018/12/le-peer-programming-lunion-fait-la-force/

ce qui touche à l'aspect visuel de l'application, et Kevin a la charge de toutes les fonctionnalités tierces (notamment tout ce qui est mathématique / algorithmique).

La fin du semestre approchant, nous avons organisé un « playtest » avec trois personnes extérieures à la promo (dont deux non-informaticien.ne.s). Le but : tester l'application en conditions réelles. Cela nous a permis de déceler de petits bugs que nous n'aurions pas vu autrement et de les corriger.

C – Fonctionnalités

L'application comporte de nombreuses fonctionnalités, que l'on peut répartir comme suit : tout ce qui touche au dessin et les options qui y sont liées, les fonctionnalités qui permettent de jouer, les fonctionnalités qui rendent le site viable, et la connexion entre différent.e.s joueurs et joueuses.

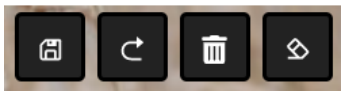
Toutes les fonctionnalités que nous avions indiquées comme « obligatoires » dans notre cahier des charges ont été réalisées et sont fonctionnelles (ci-après en gras). Les fonctionnalités « optionnelles » sont soulignées. Certaines fonctions « optionnelles » inscrites au cahier des charges n'ont pas été réalisées, mais nous les avons néanmoins indiquées, car elles font partie de notre processus de conception du projet.

Les difficultés rencontrées les plus notables au cours du développement concerne peer-js : depuis la disparition du serveur OVH, nous subissons régulièrement des pertes de connexion plus ou moins longues, nous empêchant de travailler (et notamment de débugger) comme nous le souhaiterions. Le point positif étant que nous sommes maintenant tous trois convaincus de l'importance d'avoir, pour toute chose, plusieurs sauvegardes à différents endroits.

Par ailleurs, nous avons eu quelques soucis avec Material-Ui après le dernier déploiement de l'application. En effet, le navigateur ne tenait plus compte de certains des styles que nous avions et appliquait par défaut ceux de Material-Ui. Tout fonctionnait en local, et dans un même fichier, certains styles étaient pris en compte et d'autres non. Nous avons dans un premier temps contourné le problème (par exemple en ajoutant un élément supplémentaire et en lui donnant un autre style pour que le style s'applique à l'élément que nous voulions réellement modifier). Finalement, modifier l'ordre de certains *imports* a réglé le problème.

1) Dessin et options

- **Le canevas :** toile pour dessiner.
- **Le pinceau :** permet de dessiner « à main levée » sur le canevas. Quatre tailles de traits différentes sont disponibles.
- **Effacer :** On peut utiliser pour cela **la gomme** (dont fonctionnement s'apparente à celui d'un pinceau blanc) ou **la poubelle** (qui remet le canevas « à zéro »).
- **Les formes automatiques :** rectangle (plein ou vide), cercle (plein ou vide) et segments. La prévisualisation permet de voir la taille d'une forme en temps réel lorsqu'on la crée.
- **Le pot de peinture :** remplissage automatique d'une forme avec une couleur au choix.
- **Les couleurs :** palette de couleurs pour changer la couleur du trait, des formes automatiques ou du remplissage.
- **Le retour en arrière (ou ctrl+z) :** pour annuler une action (de la dernière à la première action réalisée). On peut annuler plusieurs actions à la suite (jusqu'à 20 actions), soit en cliquant sur le bouton prévu (en dessous du canevas), soit en utilisant la combinaison de touche « ctrl+z ».



2) Autres fonctionnalités du jeu

- Enregistrer le dessin : permet au joueur de sauvegarder le dessin qu'il ou elle vient de réaliser sur son ordinateur au format .jpg.
- **Le choix des mots :** le dessinateur doit choisir entre trois mots générés aléatoirement en cliquant sur l'un des mots. Le choix d'un mot déclenche le timer (=minuteur).
- **Le minuteur :** indique au dessinateur le temps qu'il lui reste avant la fin de son tour. Lorsqu'il est terminé, déclenche l'affichage d'une fenêtre modale contenant un message («Time's up ! ») et un bouton « Nouvelle Manche » pour passer la main à un autre joueur.
- **Deviner un mot :** les participant.e.s doivent deviner le mot choisis en le tapant dans le chat. La casse n'entre pas en compte. Si le mot est incorrecte, tous les participant.e.s le voient, sinon les autres voient seulement un message « @pseudo a trouvé ». Un.e participant.e ne peut pas deviner le même plus d'une fois par tour. Il ou elle peut cependant continuer à écrire dans le chat, une fois le mot trouvé.
- **Score :** pour chaque mot deviné, le score du joueur qui a deviné est incrémenté.

3) Fonctionnement général du site

- **Barre de navigation** : commune à toutes les pages et qui comprend plusieurs boutons.
  - **Bouton Logo** : permet de revenir sur la page de connexion.
  - « **A propos** » : redirige sur une page indiquant les crédits.
  - « **Aide** » : un formulaire de contact afin que les utilisateurs puissent signaler un problème sur le site.
    Cette fonctionnalité n'est malheureusement pas encore complète puisque le bouton « envoyer » redirige seulement l'utilisateur / utilisatrice sur une autre page, alors que sa version finale prévoit que les informations rentrées dans le formulaire soient envoyées par mail à une adresse email-type help.dessinemoiunmouton@gmail.com
  - Thème Sombre : permet aux utilisateurs et utilisatrices de changer le thème du site, soit lumineux, soit sombre. Les images de fond d'écran sont générées aléatoirement via deux dossiers (un pour les images « claires », un pour les images « sombres ») que nous avons créé grâce au site *Unsplash*[8].
- **Responsivité des pages** : Nous avons essayé de faire en sorte que toutes les pages et les éléments qu'elles contiennent soient « responsive ». La taille de chaque élément s'adapte à la taille de la fenêtre du navigateur.
  Petit bémol : la page « A Propos » (About Us) n'est pas du tout responsive et nous n'avons pas encore réussi à adapter certains outils de dessin pour faire en sorte que la page du canevas soit totalement responsive. Par exemple, la palette de couleurs a été créée avec React-Color[9]*. Pour avoir une palette de couleur responsive, une solution aurait été créer les formes une à une, avant de leur attribuer une couleur et de les rendre cliquables. Malheureusement, nous nous en sommes rendu compte tard dans le développement, c'est pourquoi cette fonctionnalité n'a pas été modifiée.*
- **Gestion des erreurs** : Lors de la connexion, si le joueur ou la joueuse ne remplit pas le champ « pseudo », celui-ci devient rouge et la connexion est impossible tant que le champ n'est pas rempli. De la même manière, l'envoie du formulaire de contact est impossible si l'un des champs (« pseudo », « adresse email » ou « explication ») est laissé vide - et le champ en question devient rouge pour indiquer d'où vient l'erreur.

[8] https://unsplash.com/
[9] https://casesandberg.github.io/react-color/

4) Peer-js et la connexion multijoueur

- **Connexion :** pour se connecter, un joueur doit choisir un avatar[10] et un pseudonyme. Une fois ces deux éléments choisis, l'utilisateur / utilisatrice se connecte soit en cliquant sur le bouton « connexion », soit en appuyant sur la touche « entrée » de son clavier. Pour que d'autres personnes se connectent à la session, l'hôte génère un ID qu'il ou elle leur transmet.
- **Le chat (envoie et réception) :** permet d'échanger des messages avec les autres participants une fois que l'on est connecté à la session créée par l'hôte. Le chat apparait dans « la salle d'attente » et sur les pages de jeu. Une fois la partie commencée, c'est dans le chat qu'on tapera les mots que l'on veut deviner. Si le mot est incorrecte, tous les participant.e.s le voient, sinon les autres voient seulement un message « @pseudo a trouvé ».
- **Le Stream (envoie et réception) :** permet aux autres participants de voir ce qui est dessiné en temps réel.
- **Manche suivante** : permet de passer le rôle de dessinateur / dessinatrice à un.e autre participant.e.

5) Fonctionnalités « optionnelles » non réalisées

Les fonctionnalités suivantes ont été inscrites en tant que fonctionnalités « optionnelles » sur le cahier des charges, mais n'ont pas été réalisées :

- Bouton de changement de langues : donner la possibilité aux utilisateurs et utilisatrices de choisir une langue (au moins anglais et français) pour le site.
- Système de vote « dessin » : permettre aux participants de donner une appréciation sur le dessin qui vient d'être réalisé.
- Système de vote « ban » : permettre aux participant.e.s de signaler un.e participant.e qui aurait un mauvais comportement. Si un.e même participant.e est signalé.e plusieurs fois, il ou elle est « banni.e » du jeu (automatiquement déconnecté, et ne peut pas se reconnecter pendant une durée déterminée).
- Curseur polymorphe : le curseur change de forme en fonction de ce qu'il fait (forme de pinceau lorsque l'on clique sur le pinceau, forme de gomme lorsque l'on se sert la gomme, forme de pot de peinture lorsque l'on utilise le pot de peinture, etc…).
- Plusieurs modes de jeu : en plus du mode jeu « classique » (celui qui est proposé par défaut), nous voulions offrir la possibilité de jouer de plusieurs manières. Par exemple, le mode « par équipe » répartirait les joueurs en deux équipes (A et B). L'équipe A choisi un mot, que le dessinateur de l'équipe B doit faire deviner aux autres membres de l'équipe B.
- « Toggle Bar » : faire en sorte que la barre de navigation de « rétracte » en menu déroulant lorsque la taille de la fenêtre diminue.

6) Zoom sur certaines fonctionnalités

Parmi toutes ces fonctionnalités, certaines ont demandé plus de travail et de réflexion que d'autres. Deux aspects du projet en particulier ont nécessité du temps et de nombreux remaniements avant que nous ne parvenions à un résultat satisfaisant :

[10] Nous avons créé nous-mêmes les photos de profils / avatars via le site *PlaceIt*, un site spécialisé dans la génération de logo et avatars personnalisés. https://placeit.net/

- **Peer-JS** : Nous avons utilisé Peer-JS pour générer l'interconnexion entre les participant.e.s. Cette étape s'est révélée plus complexe que prévue, car notre première approche était difficile à implémenter. A un certain point, nous avons donc décidé de changer d'angle d'attaque. Au début, nous tentions de connecter tous les participant.e.s entre elleux. Cela créait des conflits, des problèmes de connexions ainsi que des duplications d'événements ou de messages. Pour corriger cela, nous avons décidé de modifier l'implémentation : chaque participant.e n'est connecté qu'a l'hôte et toutes les informations (participant.e.s, scores, messages) passent uniquement par l'hôte – à l'exception de l'envoi du Stream (envoie / réception en temps réel du dessin).

- **Stream** : C'est une des fonctionnalités les plus importantes de l'application. En effet, si les participant.e.s ne peuvent pas voir le dessin en cours de réalisation, ielles ne peuvent pas deviner le mot ! Contrairement aux autres fonctionnalités de Peer-JS, ici celui ou celle qui a le canevas est connecté directement aux autres participant.e.s. Le canevas est redistribué aux *viewers* au début de chaque nouvelle manche. Malgré les apparences, c'est une des fonctionnalités Peer-JS que nous avons eu le plus de facilité à implémenter.

- **Le pot de peinture :** pour réaliser cette fonctionnalité incontournable des jeux de dessin, nous avons décidé d'utiliser l'algorithme de Dijkstra. L'implémentation a été longue, quoique pas aussi complexe que nous le redoutions. Nous avons par ailleurs eu du mal à débugger (et même à simplement réaliser que notre code fonctionnait), car l'algorithme de Dijkstra étant assez « gourmand », il a tendance à faire planter Mozilla Firefox (soit le navigateur que Claire et Kevin utilisent). Nous avons hésité à le recoder en utilisant l'algorithme de Metropolis. Il s'agit d'un algorithme statistique qui permet de prendre en compte une marge d'erreur – moins de précision mais beaucoup moins de calculs que Dijkstra ! Cependant, nous n'étions pas certain.e.s de réussir, aussi avons-nous préféré le laisser de côté et avancer sur d'autres aspects du projet.

D – Bilan

Ce projet est l'un des projets les plus longs que sur lequel nous ayons eu à travailler depuis le début de notre licence. C'est également la première fois que nous démarrions un projet à partir de rien. C'était très agréable de pouvoir choisir notre sujet, les outils qui nous semblaient les plus adaptés et surtout de pouvoir contrôler tous les aspects du projet.

Nous avons tous beaucoup progressé en Javascript et avons atteint une bonne maitrise de React et de ses composants. La découverte de Peer-JS, bien que rude, a été également très intéressante. Les rendez-vous hebdomadaires avec notre chargé de projet ont en outre été très utiles, car cela nous a montré l'importance de faire le point régulièrement sur notre travail, individuellement et en équipe. Par ailleurs, le peer-programming a été très formateur : nous avons pu progresser plus rapidement et nous familiariser plus facilement avec les manières de coder des un.e.s et des autres. Cela a également permis de compenser le fait que nous avons dû travailler exclusivement à distance.

Nous sommes très fier.e.s de ce projet. Bien qu'il ne soit pas complètement terminé, nous sommes allés beaucoup plus loin que ce que nous espérions et nous avons tous réussi à nous organiser et à progresser, aussi bien en tant qu'individus qu'en tant qu'équipe. Nous en sortons tous grandis, et nous sommes de meilleur.e.s informaticien.ne.s et de meilleur.e.s personnes que lorsque nous avons commencé ce travail ensemble.