

Rapport projet AWS - Pictionary

Hugo Chanas
William Magalhaes Monteiro
Asmaa Ouguenoune
Christian Knayzeh

[Lien vers le répertoire GitHub](#)

1 Présentation du projet

1.1 L'idée de base

L'idée de base du projet est de proposer un jeu de type Pictionary. Un joueur doit faire deviner un mot en le dessinant et les autres doivent le trouver le plus vite possible. Le projet est très inspiré de Skribbl.io.

Il nous fallait donc au moins la possibilité de dessiner, avec le partage du dessin en temps réel, et un chat où les utilisateurs peuvent tenter de deviner le mot.

1.2 Les technologies que l'on a utilisées

Pour la partie dessin, nous aurions pu utiliser l'API de base pour dessiner sur des canvas HTML, mais nous avons préféré utiliser le framework Konva.js car il est plus facile à prendre en main et à utiliser, et apporte plus de fonctionnalités.

Pour gérer la communication, nous avons choisi Socket.io, qui permet de gérer la communication en temps réel, aussi bien pour le dessin que pour le chat. Pour le reste, nous avons utilisé Express.

Nous utilisons aussi Express-session pour gérer l'authentification et les sessions. Les comptes sont gérés avec SQLite. Nous avons commencé avec une base de données MySQL mais avons décidé de changer pour simplifier le projet et (surtout) l'hébergement (finalement sur Glitch).

2 Ce que l'on a fait - Description du projet

2.1 Authentification et base de données

Notre application ne permet qu'aux joueurs ayant un compte d'accéder au jeu. Il existe donc trois pages autres que celle du jeu :

- Une page d'accueil,
- Une page d'authentification pour se connecter,
- Une page de création de compte.

Pour gérer les réponses du serveur sur ces deux dernières, on utilise le moteur pug. On peut ainsi afficher des messages d'erreurs sans avoir besoin de script. On affiche donc un message rouge en bas de la page lorsqu'un nom d'utilisateur ou mot de passe est invalide, ou encore si le nom choisi est déjà utilisé, etc.

Dès qu'une requête de connexion ou de création est reçue, le serveur interroge l'instance SQLite et ajoute le compte ou autorise la connexion en conséquence. La base de données ne contient qu'une table pour les comptes, avec un nom d'utilisateur, un mot de passe haché, et une valeur aléatoire utilisée comme padding sur le mot de passe avant le hachage.

Pour la liste des mots à dessiner qui peuvent être proposés aux joueurs, elle est enregistrée dans un simple fichier. En effet, comme la liste est fixée, et non modifiable par un utilisateur, il n'a pas été nécessaire de l'ajouter à la base.

2.2 Partie jeu

2.2.1 Dessin

Pour la partie dessin, on utilise des objets Konva. Pour chaque outil disponible, une série de 'binds' est définie, liant un événement à une fonction. Dès que l'utilisateur change d'outil avec les boutons en bas, les binds sont modifiés.

Chaque nouvel élément du dessin est un nouvel objet Konva qui est ajouté à la liste de ceux qui sont affichés dans la zone de dessin. Certains attributs de ces objets, comme notamment la couleur, sont définis selon ce que l'utilisateur a sélectionné.

Pour le crayon ou la gomme, à chaque nouveau clic, un objet `Konva.Line` est créé. Tant que le clic est enfoncé, et à chaque mouvement de la souris, un point est ajouté à la ligne. Un tel objet représente en effet une suite de segments reliés entre eux bout à bout.

Pour l'outil 'rond', un disque est ajouté à chaque nouveau clic. Cet outil très rudimentaire a été ajouté en prévision d'une modification ou d'une amélioration, mais qui n'a finalement pas été faite.

L'implémentation de l'outil remplissage était de loin la plus difficile. En effet, il a fallu accéder au contexte 2D du canvas pour obtenir la matrice de pixel, et appliquer un algorithme de remplissage par diffusion. Après avoir obtenu une nouvelle matrice de pixels correspondant à la zone à colorier, une image bitmap est créée, et est utilisée pour initialiser un objet `Konva.Image`.

L'ajout de cet outil a demandé beaucoup de temps, mais il est assez important dans une application de dessin, et aurait manqué si il n'avait pas été implémenté.

Pour les boutons 'undo' et 'redo' (les flèches), un simple nombre entier est incrémenté ou décrémenté. Ce nombre représente le nombre d'objet de la liste qui sont affichés à l'écran parmi ceux qui ont été dessinés. Si l'utilisateur dessine un nouvel élément alors que certains ont été effacés, ceux-ci sont définitivement supprimés.

Le bouton 'suppression' est quant à lui plus simple, il supprime simplement tous les objets.

2.2.2 Communications

Une fois sur la page du jeu, toutes les communications entre les clients et le serveur sont gérées avec `Socket.io`.

Lorsque le serveur désigne un ou plusieurs dessinateurs, ceux-ci sont placés dans une 'room', ce qui permet de leur envoyer des messages ciblés. Par exemple, le mot qu'ils doivent faire deviner n'est envoyé qu'à eux. Les autres reçoivent en effet une chaîne de caractères avec les lettres remplacées par des tirets bas (comme au pendu).

Une autre room existe où sont placés les utilisateurs qui doivent deviner le mot et qui ne l'ont pas encore trouvé. Les messages du chat envoyés par les dessinateurs et ceux qui ont devinés ne sont ainsi pas visibles par ceux qui cherchent.

Dès qu'un dessinateur modifie le dessin, l'information est transmise au serveur, qui la transmet à son tour aux autres utilisateurs. Pour les traits

et les disques, des coordonnées et une couleur sont transmises ce qui permet aux autres clients de re-cr  er des objet parfaitement indentiques de leur c  t  .

Pour le remplissage, c'est un peu plus compliqu  . Transmettre la matrice de pixels enti  re serait en effet assez lourd ($\approx 1\text{Mo}$ par image). Il est assez simple de r  duire le volume de donn  es, et on a utilis   deux solutions.

Dans un premier temps, le remplissage n'a qu'une seule couleur. On peut donc transmettre la couleur en parall  le du tableau, une seule fois et r  duire l'information de chaque pixel    un bool  en (0 = transparent, 1 = couleur du remplissage).

Cela donne un tableau de 0 et de 1. Pour r  duire encore le volume de donn  es, chaque suite de 1 est remplac  e par deux nombres : la position du premier 1 dans le tableau, et la longueur de la suite horizontale. Il ne reste plus qu'   transmettre ces informations pour reconstituer l'image compl  te.

Avec ces techniques, le remplissage de toute le zone de dessin pour changer la couleur du fond, par exemple, tient en quelques octets.

L'ensemble des modifications apport  es au dessin sont trait  es et enregistr  es par le serveur, y compris undo et redo. Ainsi, lorsqu'un nouvel utilisateur rejoint le jeu en cours de route, la pile d'objets dessin  s lui est envoy  e pour qu'il puisse afficher le dessin tel qu'il est    cet instant.

Tout le d  roulement du jeu est planifi   par le serveur. L'encha  nement des manches, des tours de jeu, le choix des dessinateurs, du mot    deviner, la v  rification des messages dans le chat pour voir qui a devin  , etc, est g  r   par le serveur.

3 Ce que l'on a appris

Travailler en groupe (avec les 3 r  les    permuter par semaine)

Cela a s  rement   t   la partie la plus difficile. En effet, nous ne nous sommes pas rencontr  s souvent, et avons donc d   communiquer presque exclusivement sur un serveur discord.

Le fait d'interchanger les r  les chaque semaine   tait une experience assez int  ressante, nous obligeant    fournir un travail que l'on n'aurait pas fait naturellement. Cela nous a permis d'exp  rimer dans plusieurs postes, en divisant le travail diff  remment de nos habitudes.

Gestion des requ  tes des clients, r  ponses, redirections

Nous avons appris comment fonctionne une application web classique, avec un ensemble de pages dans lesquelles naviguent des utilisateurs, envoyant

régulièrement diverses requêtes au serveur.

Nous avons utilisé des middlewares pour gérer l'authentification, les sessions et l'interrogation de la base de données.

Écriture de script javascript complexe pour une application

Les scripts côté serveur et client qui gèrent la partie jeu sont assez longs et complexes. Nous avons donc lors de leur écriture beaucoup progressé en javascript.

Certains aspects notamment comme l'outil remplissage avec des matrices de pixels et des images bitmap ont été particulièrement techniques et instructifs.

Communication en temps réel entre un serveur et plusieurs clients

Pour obtenir un jeu fonctionnel en temps réel, nous avons dû apprendre à bien manipuler Socket.io. Cette partie est en effet une nouveauté, bien différente de la programmation séquentielle à laquelle nous sommes plutôt habitués.

Il nous a fallu écrire tout un tas de fonctions qui sont appelées lorsqu'un message du serveur est reçu. Toutes ces fonctions doivent interagir avec les mêmes variables, sans pour autant avoir un ordre d'exécution prévisible. Cela nous a donné du fil à retordre, surtout pour penser à toutes les possibilités et gérer tous les cas.

Pour cette raison, certains bugs, même basiques, sont restés inconnus longtemps. Nous avons eu besoin d'organiser des parties en groupe afin de bien pouvoir tester les différentes fonctionnalités du jeu. Faire jouer un grand nombre de personnes différentes permet aussi de vérifier plus de cas, chacun venant avec des innovations et une façon de jouer.

Par exemple, avoir un nom d'utilisateur avec des espaces provoquait des bugs, mais nous ne nous étions pas rendus compte pendant longtemps. Pour cause, aucun d'entre nous n'avait pensé à créer un compte avec un nom en plusieurs mots. Il a fallu qu'un joueur externe au projet teste l'application pour que l'on s'en rende compte.

Tout cela nous a bien instruit sur l'importance d'organiser des phases de tests avec plusieurs personnes différentes, et surtout des personnes qui ne font pas partie de l'équipe du projet. Cela devient de plus en plus important lorsque le projet gagne en complexité.

Sécuriser une base de données contenant des mots de passe.

Pour enregistrer les comptes, nous avons dû apprendre à manipuler une petite base de données SQLite dans le script du serveur.

Nous avons ensuite pu appliquer les méthodes que l'on a vues en cours pour sécuriser le contenu de la base de données, même en cas d'intrusion, avec le hachage. Cela nous a permis de nous initier à l'utilisation de la cryptographie dans les applications web en javascript.

Bonus.

Tout cela nous a aussi appris à utiliser les outils des navigateurs pour inspecter les pages, la console, etc.

4 Ce que l'on pourrait ajouter ou améliorer

Fonctionnalités diverses du jeu

- Ajouter et améliorer des outils

L'outil 'rond' est particulièrement discutable, et faisait partie du projet d'ajouter des formes prédéfinies diverses, pouvant être placées avec précisions, bougées, etc.

Konva permet de manipuler des objets graphiques facilement, et nous aurions pu en profiter pour aisément implémenter ces fonctionnalités.

- Ajouter un sélecteur de couleurs

Notre jeu ne dispose que d'une quantité limitée de couleurs parmi lesquelles choisir. Il serait pratique pour les dessinateurs de pouvoir choisir une couleur dans un sélecteur dédié.

Cependant, comme il s'agit aussi d'un jeu de vitesse, la majorité des joueurs se contentent des couleurs prédéfinies. Pour cette raison, nous nous sommes contentés de quelques couleurs.

- Le déroulement du jeu

Un assez grand nombre de petites améliorations peuvent encore être faites sur le jeu.

Par exemple, la façon de choisir les dessinateurs quand il y en a plusieurs repose en partie sur l'aléatoire. Le mieux serait de choisir de façon déterministe, en s'assurant que tout le monde dessine autant de fois, et en essayant de créer des groupes différents à chaque fois.

Le méthode pour compter les points n'est pas très avancées non plus, et pourrait largement être améliorée.

Page dédiée à la modification de la liste de mots par des admins.

Nous avons une liste de mots statique. Elle n'est donc pas incluse dans la base de données, et ne peut être modifiée que par l'administrateur du serveur.

Une bonne amélioration à apporter serait de l'y ajouter, et de créer une page accessible par des utilisateurs ayant un rôle admin qui leur permettrait d'ajouter ou d'enlever des mots. Nous aurions alors besoin d'ajouter des rôles et privilèges aux utilisateurs, ce qui aurait pu être un aspect intéressant à développer.

Autoriser aux utilisateurs à ne pas créer un compte

Les utilisateurs sont forcés de créer un compte pour jouer au jeu. Cependant, rien n'est associé au compte à part le nom d'utilisateur et le mot de passe. Nous pourrions donc simplement demander à des utilisateurs sans compte d'entrer un nom qui serait utilisé seulement pendant sa session de jeu.

Il serait aussi possible par la même occasion d'ajouter des données liées aux comptes, comme des statistiques, visibles sur une page dédiée. Cela justifierait vraiment la création des comptes pour les utilisateurs.

5 Conclusion

Finalement, ce projet nous a fait découvrir un domaine qui était assez nouveau pour nous. Nous n'avons pas choisi le sujet le plus simple à réaliser, et cela a mené à beaucoup d'investissement dessus (surtout dans la période suivant la dernière soutenance). Tout ceci nous a cependant menés à approfondir le sujet et à en apprendre encore plus.