

Dufeu Rémi

Lemoine Alexandre

19/03/2021

# Rapport Final Projet :

Analyse Besoins et Appl Web

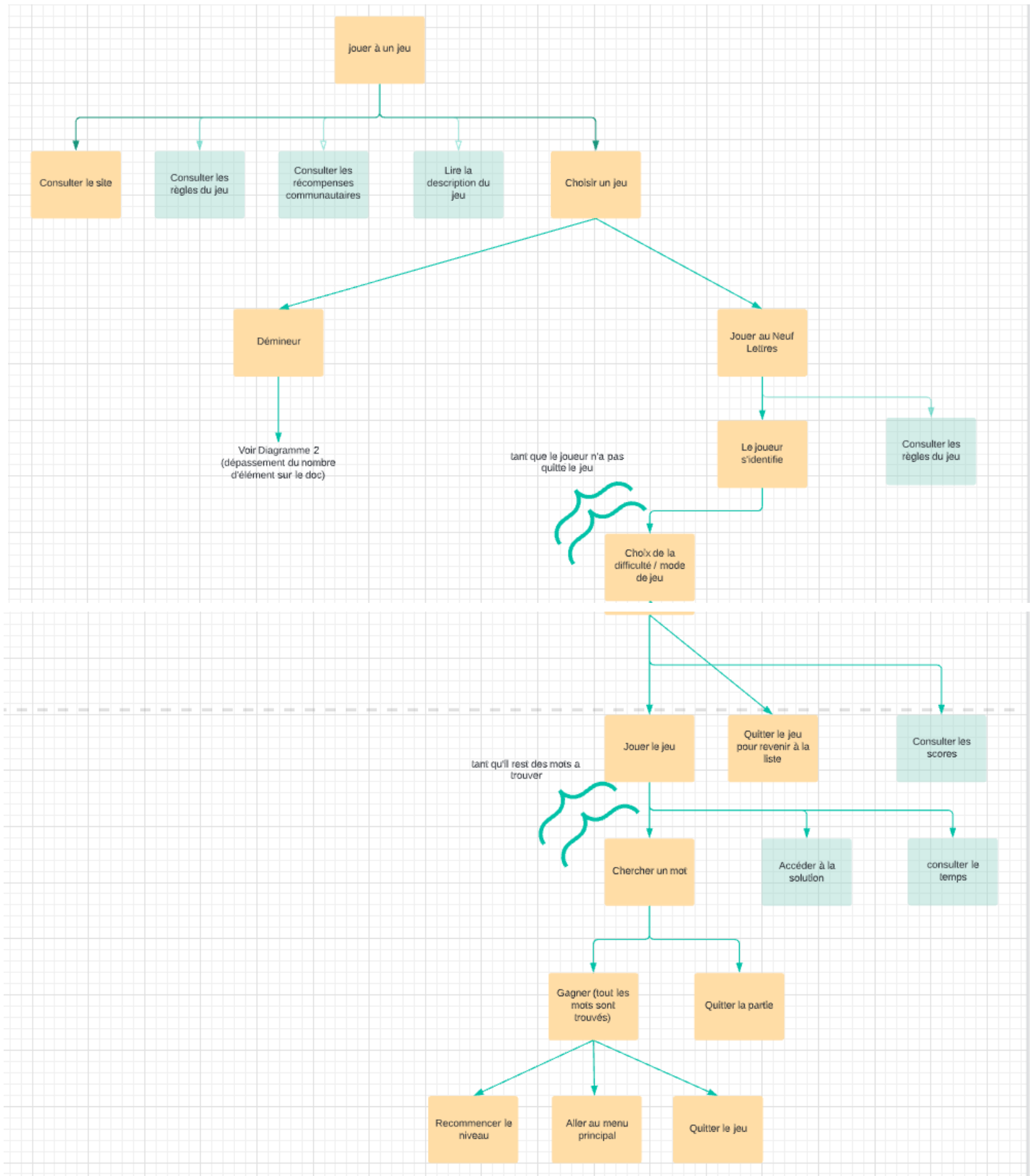


## **Table des matières**

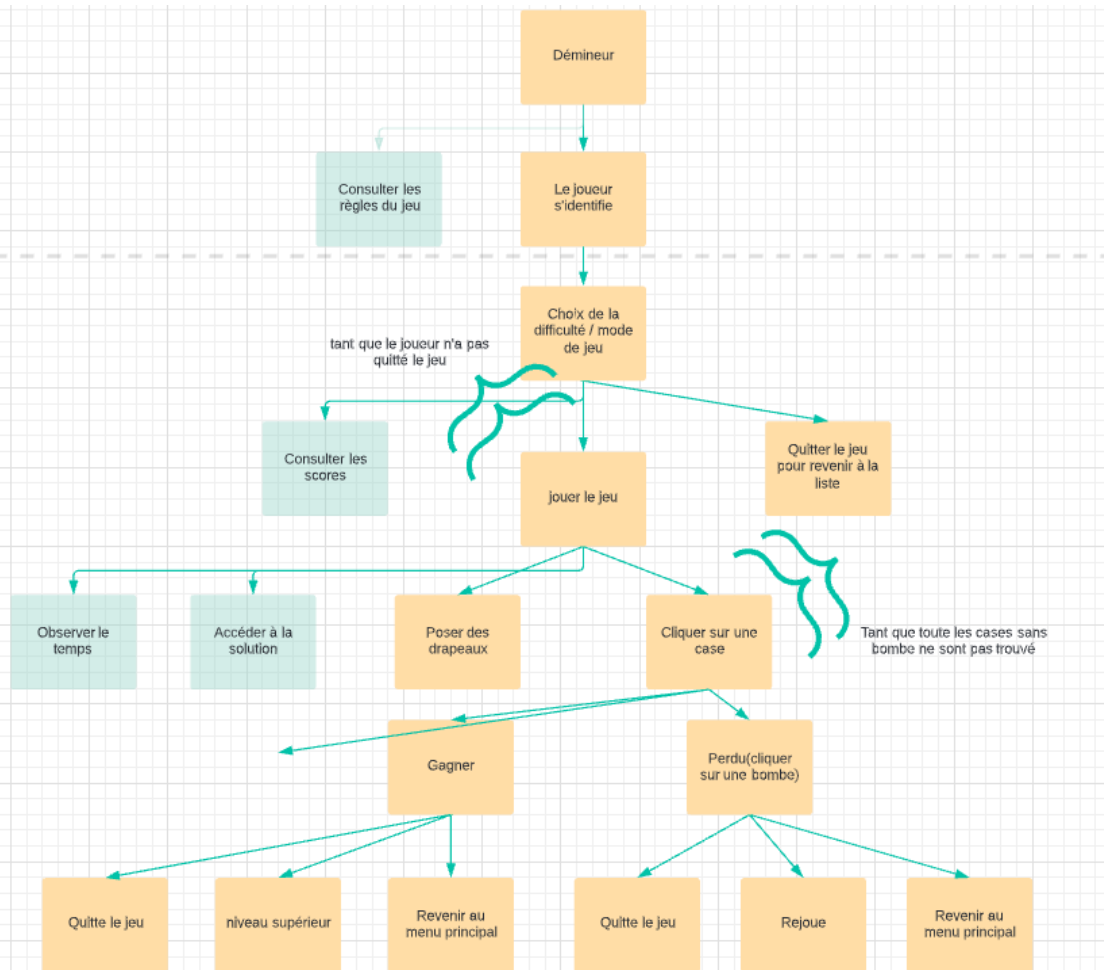
<b>Modèle des tâches</b>	<b>2</b>
Modèle des tâches de l'accueil et du jeu neuf lettres (version web)	2
Modèles des tâches Démineur (version web)	3
<b>Maquettes</b>	<b>4</b>
<b>Architecture logicielle</b>	<b>7</b>
Structure du code	7
Architecture du jeu neuf lettres (Rémi DUFEU)	8
Architecture du jeu démineur (Alexandre LEMOINE)	10
Partie commune aux deux jeux	13
<b>Planning des scénarios</b>	<b>14</b>
Planning des scénario pour le jeu neufs lettres (Rémi Dufeu)	14
Planning des scénario pour le jeu démineur (Alexandre Lemoine)	15
<b>Technique utilisées</b>	<b>16</b>
<b>Comparaison et différences</b>	<b>17</b>

# Modèle des tâches

## Modèle des tâches de l'accueil et du jeu neuf lettres ([version web](#))

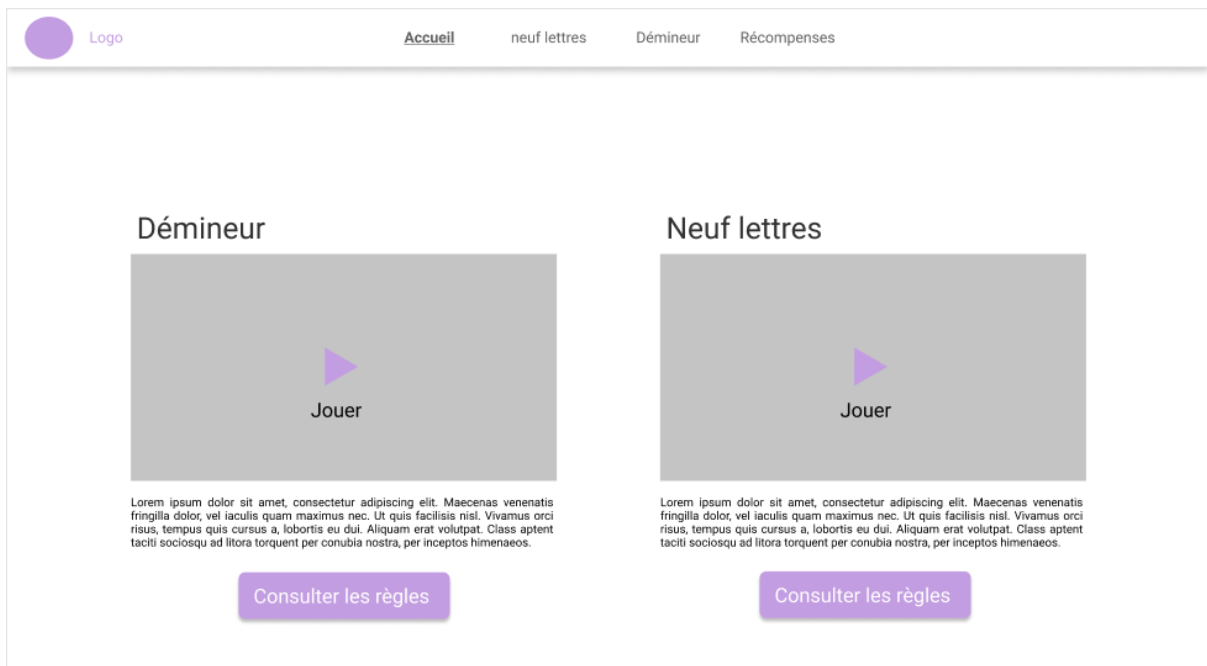


## Modèles des tâches Démineur ([version web](#))

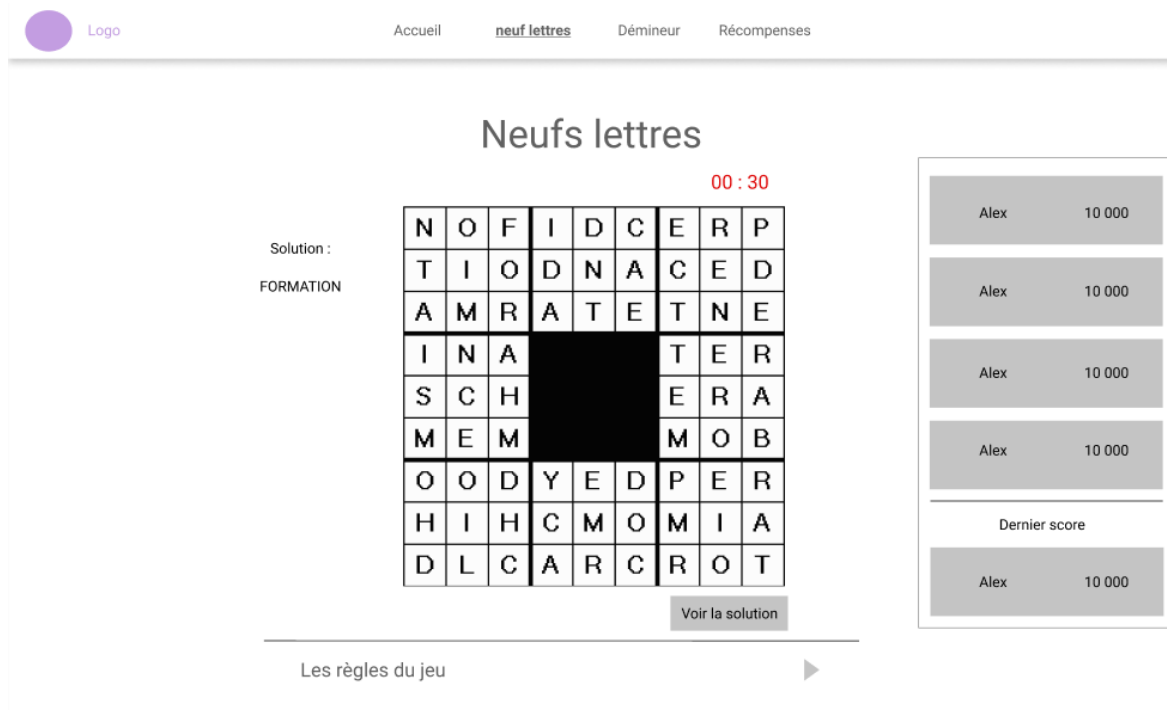


# Maquettes


## Page d'accueil



## Page Neuf lettre



## Page démineur

 Logo

Accueilneuf lettresDémineurRécompenses

### Démineur

00 : 30

	1		2	2	2		1		1
	3					2		1	
	2			3	3				2
									1
	2		3	2			2		
		4			1	2		2	
1									
2				3	4			2	
	3		1					4	
2		1		2	3	4		4	

Voir la solution (drapeau auto )

Les règles du jeu

Alex10 000

Alex10 000


Alex10 000

Alex10 000

Dernier score


Alex10 000


## Page récompense


 Logo


Accueilneuf lettresDémineurRécompenses


### Démineur


  
Trophée XX


  
Trophée XX


  
Trophée XX


  
Trophée XX

  
Trophée XX


  
Trophée XX


  
Trophée XX


  
Gagner en 30 s


  
????


### Neuf lettres


  
Trophée XX


  
Trophée XX


  
Trophée XX


  
Trophée XX

  
Trophée XX

  
Trophée XX

  
Trophée XX

  
Ne pas utiliser de solutions

  
????

Nos maquettes basse fidélité sont identiques à la version finale du projet. Les maquettes ont été pour nous un réel point d'appui pour la réalisation du projet et le fait qu'elles soient détaillées a été un plus pour harmoniser le design du site web.

# Architecture logicielle

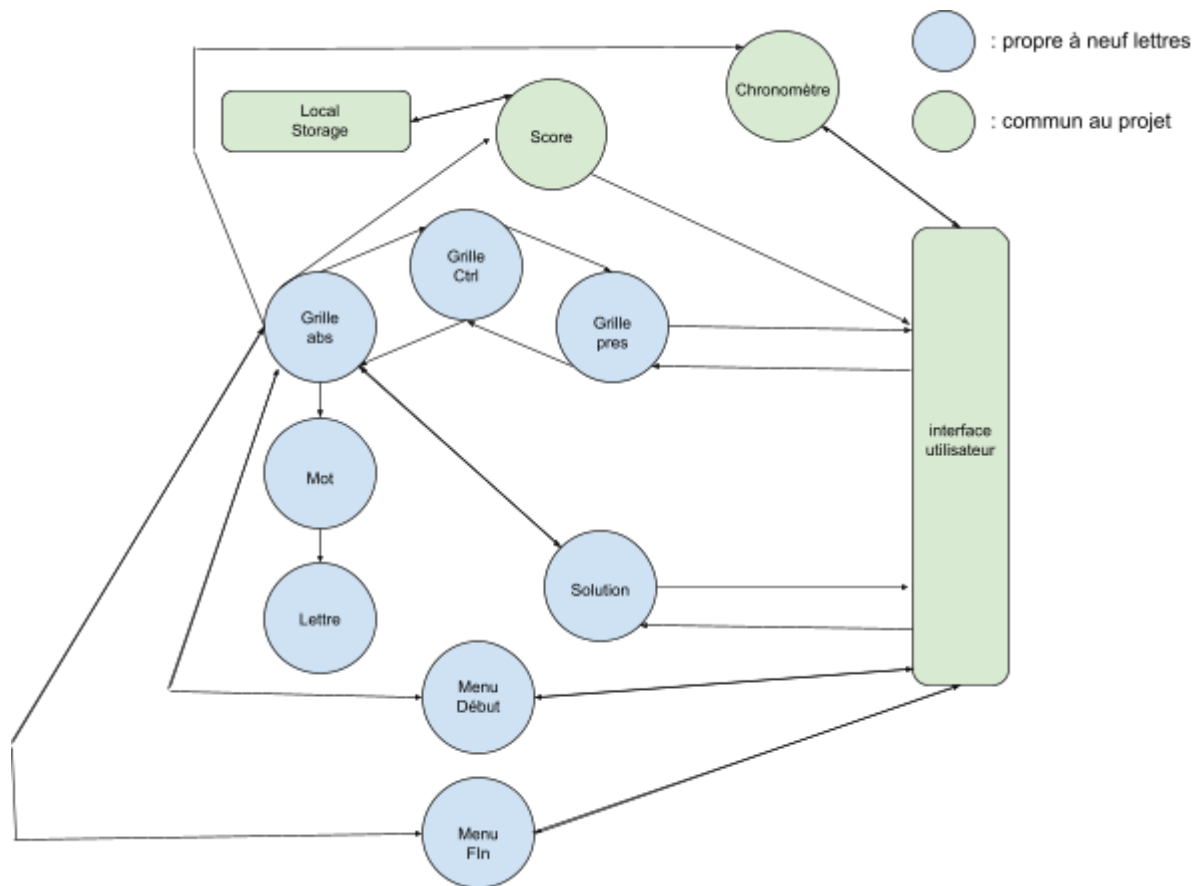
## Structure du code



Au niveau du code nous pouvons distinguer ici 3 groupes de fichiers : les fichiers propres au fonctionnement du jeu démineur dans le dossier `demineurGames`, les fichiers propres au fonctionnement du jeu neuf lettres dans le dossier `neufLettresGames` et enfin le reste qui est commun : le code du site web, les fichiers javascript utilisé par les deux jeux les assets etc...



## Architecture du jeu neuf lettres (Rémi DUFEU)



## Les classes relatives au Canvas

### GrilleAbs

GrilleAbs joue le rôle de classe centrale au jeu neuf lettres. Il centralise les données du jeu, gère les états de la partie et fait le lien avec les autres classes du jeu (solution, chronomètre, menu...). Le nom de la classe provient du fait que j'ai essayé d'implémenter la structure PAC au sein de mon projet. En théorie la classe devait uniquement gérer les données du jeu mais elle gère aussi une partie de la présentation avec l'inclinaison des lettres mais aussi la boucle d'animation qui devrait être implémentée dans la présentation.

### **GrilleCtrl**

Cette classe est responsable de la communication entre l'abstraction et la présentation en transmettant des messages avec une étiquette et une pièce jointe (un objet avec des données).

### **GrillePres**

GrillePres est la classe qui ajoute les écouteurs sur le canvas et qui envoie les informations des clics à GrilleAbs au travers de GrilleCtrl. GrillePres est aussi responsable d'afficher sur le canvas les données du jeu comme les mots et la grille de jeu.

### **Mot**

Cette classe représente un mot du jeu 9 lettres. C'est une classe utilitaire utilisée par GrilleAbs pour simplifier le code. Elle a un rôle important puisqu'elle est responsable de l'ordre aléatoire des lettres dans la grille.

### **Lettre**

Lettre est une classe utilitaire de Mot et gère les états des lettres et leur inclinaison. Étant une classe en rapport avec l'abstraction, cette donnée ne devrait pas se trouver dans cette partie du code. Cette erreur provient du fait que pour moi la présentation ne devait pas posséder de copie de données du jeu et devait seulement présenter les données présentes dans l'abstraction. A refaire je ferais une copie des informations dans la présentation et donc l'inclinaison serait présente dans la partie présentation.

## **Les classes responsable du menu du jeu**

### **MenuDebut**

MenuDebut est une classe qui gère naturellement le menu de début du jeu. Elle affiche et détruit le menu du jeu à l'aide de l'API du dom dans une div. Une partie ne peut être lancée uniquement si un pseudo est rentré grâce à la propriété "disabled" des boutons HTML.

### **MenuFin**

MenuFin est une classe qui gère le menu de fin du jeu. Elle affiche et détruit le menu du jeu à l'aide de l'API du dom dans une div. Elle donne la possibilité de retourner dans une partie ou alors de retourner au menu principal pour changer de pseudo par exemple.

## **La classe solution**

Cette classe donne un coup de pouce au joueur en affichant les solutions du jeu à l'aide d'un bouton HTML. Elle génère une liste de mots sur la gauche de l'écran qui est au départ vide puis qui se remplit quand des mots sont trouvés ou quand le joueur demande une solution.

## **Les fichiers utilitaires**

La classe coordonnée permet de faciliter la gestion des coordonnées de la souris à l'écran.

Le fichier asset gère le chargement des sons du jeu à l'aide de la librairie HowlerJs.

Le fichier "dictionnaire mot" est une simple liste de mots à 9 lettres qui seront choisis aléatoirement lors du lancement du jeu.

## **Architecture du jeu démineur (Alexandre LEMOINE)**

### **Fonction de lancement de partie**

Lorsque l'utilisateur choisit le jeu démineur sur la page d'accueil ou le menu, la page se charge.

Lors de la fin du chargement de fenêtre, le gestionnaire d'événements window.onload appelle la fonction init().

La fonction init() est le commencement pour le chargement du jeu. On récupère le canvas html pour dire ou dessiner mais aussi on indique un jeu en 2D et sa taille. Dans un premier temps on crée l'objet menuDebut qui ensuite appelle la méthode construireMenu() pour pouvoir créer le menu où l'utilisateur va entrer son pseudonyme et cliquer sur le bouton jouer.

Quand le bouton jouer est actionné on stocke dans une variable le pseudonyme et on détruit le menu.

Ensuite init() crée l'objet new score avec comme paramètre le jeu auquel l'utilisateur joue, après la fonction init() appelle la méthode construireTableau() qui permet la création du tableau des scores.

Ensuite on génère les assets avant de lancer la fonction `startGame()` qui actualise le tableau des scores par rapport à ce qui est stocké dans `localStorage`.

Du coup on revient sur la fonction `init()` qui appelle la dernière méthode `requestAnimationFrame(animationLoop)`; qui permet de faire tourner le jeu à 60i/s.

Une fois sur la fonction, le mode de jeu prédéfinis et "MenuPrincipal" donc le code va passer par la méthode `afficherMenuPrincipal()`; pour afficher le menu principal

Tant que l'utilisateur ne clique pas sur un mode de jeu sur le menu principal, la fonction `animationLoop()` s'appelle elle-même tout en affichant le menu principal.

En premier lieu nous allons parler de l'écouteur de clic qui a 2 fonctions dans le projet, faire apparaître les cases du démineur mais aussi de sélectionner le mode de jeu auquel jouer.

Dans notre cas, l'utilisateur étant bloqué dans le menu principal, la méthode `canvas.onmousedown = traiteMouseDown` est en attente d'un clic. Lorsque l'utilisateur va cliquer sur un choix, grâce à la fonction `traiteMouseDown(event)`, on va récupérer la position de la souris pour connaître la difficulté sélectionnée et changer le mode en "débutant" ou autre mode de jeu et rappeler la fonction `startGame()`.

## **Fonction qui génère la grille**

La fonction `startGame()` va entrer dans le switch case qui attendait que l'utilisateur clique sur un mode de jeu. Une fois rentré dans le case du mode de jeu, on vérifie dans un premier temps si l'objet Grille est nul ou non. Si il est vide, on crée l'objet Grille avec en paramètre le nombre de ligne et colonne, la largeur et hauteur du canvas, les assets chargés avant et le nombre de bombes qui remplit le constructeur de la classe Grille. Si l'objet grille existe déjà, on appelle la méthode `resetGrille()` avec les paramètres de bombes, lignes et colonnes et rappelle les mêmes méthodes que le constructeur de la classe Grille ensuite on lance la méthode `compeur.init()` pour démarrer le chronomètre.

## **Fonction qui crée les cases**

Le constructeur de la classe Grille appelle dans un premier temps la méthode `remplirGrille()`.

Dans cette méthode on crée un tableau 2D via la méthode `create2DArray()` avec le nombre de lignes en paramètre. Ensuite on boucle sur toutes les cases de la Grille et on crée pour

chaque case une case avec en paramètre type prédéfinie en CaseVide pour le moment, la position sur la ligne et la colonne, l'image de la case et la valeur nulle pour le clic.

Ensuite on rajoute les images du drapeau dans constructeur de la case.

remplirGrilleDeBombe() est appelé après avec le nombre de bombes à placer sur la grille. De là, on place les bombes aléatoirement dans la grille et change le type et l'image cachée.

Et pour finir, la méthode detectionDesBombes() est appelée et vérifie pour chaque case qui n'est pas une bombe, combien de bombes sont positionnées autour d'elles et change le type de la case par rapport au nombre de bombes autour. Mais aussi on change l'image cachée pour y mettre l'image du chiffre en question.

On retourne dans la fonction animationLoop() ou l'on entre dans le switch case qui correspond au mode de jeu actuel. Dans la case, la fonction updateGame() est appelée.

UpdateGame() vérifie dans un premier temps si l'objet Grille est nul et ensuite elle appelle la méthode showCase() de grille qui elle boucle sur le nombre de cases en appelant la méthode draw() avec en paramètre le canvas, la position, la hauteur et la largeur. Cette méthode permet à chaque changement d'image du jeu de dessiner et changer l'image.

## **Fonction qui permet au joueur de jouer**

Pour la première partie du clic

On regarde dans un premier temps si on est dans une partie sinon on revient à ce qui est écrit plus haut.

Si l'utilisateur est dans une partie, on appelle la méthode getCase() avec en paramètre la position du clic en x,y.

Cette méthode permet de retourner à la case que l'utilisateur a cliquée.

A chaque clic gauche, on définit la case comme étant cliqué et tout dépend quel est le type de la case. Si il clique sur un type bombe, on joue un son avec la méthode play() des assets (une explosion de bombe et change l'état du jeu en perdu)

A chaque case cliquée, ça appelle la méthode PartieFinie(), la méthode vérifie si toutes les cases hormis les bombes ont été cliquées, si c' est le cas, ça return "fini" sinon rien.

On vérifie si aussi l'utilisateur fait un clic droit, si c'est de la cas alors on pose un drapeau sur la case en question et la case devient cliquable qu'avec le clic droit. Si un drapeau est déjà sur la case alors on peut re cliquer gauche sur la case.

Si le mode de jeu passe en fini ou perdu, alors on affiche un menu de fin de partie (finDePartiePerdu()) pour quand la partie est perdu et finDePartie() quand la partie est gagnée). L'utilisateur peut dans ce cas rejouer ou retourner au menu principal.

Quand une partie est gagnée en mode expert on appelle la méthode miseAJourScore() avec le mode de jeu en paramètre, cette méthode permet d'envoyer dans localStorage le score du joueur ainsi que son nom d'utilisateur.

## **Bouton de solution**

Si le joueur se retrouve bloqué il a la possibilité de cliquer sur le bouton solution, le bouton appelle la fonction revelerUneBombe() qui elle aussi appelle la méthode donnerUneSolution() de la classe grille. Ce bouton permet d'afficher un drapeau sur une bombe du jeu.

## **Partie commune aux deux jeux**

La gestion des scores utilise le même code pour les deux jeux. Cela est rendu possible car nos deux jeux fonctionnent avec un scoring basé sur le temps. Les scores sont localisés dans le localStorage à une clef qui dépend du nom du jeu.

### **Chronomètre**

La classe permet de gérer le chronomètre qui est le score du jeu. En début de partie elle lance le chronomètre et en fin de partie elle transmet le score et l'identité du joueur au localStorage.

### **Score**

Cette classe permet l'affichage des scores présent dans le localStorage à l'écran : le top 5 des meilleures scores et le dernier score du jeu courant.

# Planning des scénarios

Planning des scénario pour le jeu neufs lettres (Rémi Dufeu)

30 Janvier - 17 Mars	30 Janvier - 7 Février	7 Février - 12 Février	12 Février - 17 Février	17 Février - 22 Février	22 Février - 28 Mars	28 Février - 5 Mars	5 Mars - 10 Mars	10 Mars - 15 Mars	15 Mars - 17 Mars
conception des pages et réalisation de la maquette									
Jouer au jeu 9 lettres développement du jeu									
Consulter le temps chronomètre 9 lettres									
rejouer ou menu principale mise en place des menus du jeu									
Identification le joueur doit noter un pseudo pour lancer une partie									
Choisir le mode de jeu mise en place du mode facile									
Consulter son score mise en place du localStorage et de l'affichage des scores									
Jouer au jeu 9 lettres Ajout des sons									

Planning des scénario pour le jeu démineur (Alexandre Lemoine)

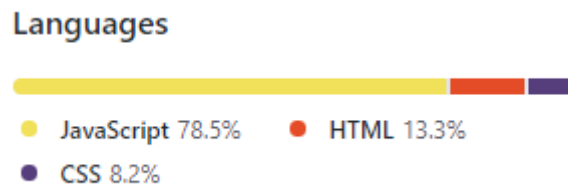
30 Janvier - 16 Mars	30 Janvier - 7 Février	7 Février - 12 Février	12 Février - 15 Février	15 Février - 19 Février	19 Février - 25 Février	25 Février - 10 Mars	10 Mars - 15 Mars	15 Mars - 16 Mars
conception des pages et réalisation de la maquette								
création de la grille, création des méthodes de placement de bombe et détection des valeur de case,								
ajout du clique droit et révélation des cases vides voisine, ajout de la fonctionnalité solution								
création du menu principal et du chronomètre								
Correction de bug qui faisait ralentir les fps du jeu								
Correction de bug sur rejouer, ajout de son dans le jeu								
Commentaire du code et résolution de problème lorsqu'on changer de niveau								
Modification de localStorage, mise en place des récompenses								



## Technique utilisées

Dans notre code nous avons utilisé du html5, CSS3 et JavaScript pour la mise en page et la navigation de notre site web. Le javascript nous a permis d'une part d'éviter la duplication de code HTML à travers des composants comme la topbar qui est réutilisé dans chaque page. D'une autre part, le javascript permet des fonctions dynamiques comme cacher du texte à l'aide d'un bouton ou bien de changer le style d'une image quand la souris passe dessus.

En ce qui concerne la partie technique des jeux, le langage utilisé est le javascript accompagné de l'API du canvas qui nous permet d'intégrer des effets visuels plus pertinents bien que le canvas présente une contrainte : la position de la souris doit être calculé.



Comme le démontre ce graphique JavaScript est le langage le plus utilisé dans le projet.

Nous avons fait appel à des librairies pour le son "howler JS".

Pour la persistance des données du site web et notamment des scores, nous utilisons le localStorage, une API permettant de disposer d'un espace de stockage dans la mémoire locale du navigateur web. Nous pouvons ainsi simplement profiter d'une sauvegarde des scores sur le même appareil.

## **Comparaison et différences entre ce qui était prévu et ce qui a été fait**

Durant la partie de développement nous avons changé quelques détails entre la maquette et le rendu final.

Pour commencer les boutons d'affichage pour consulter les règles du jeu on été changé nous sommes passés d'un gros bouton à un petit bouton qui déroule les règles.

Dans le jeu 9 lettres rien ne permet de quitter la partie contrairement à ce qui avait été annoncé. Le jeu peut bien évidemment recharger la page cependant cette option est moins pratique.

Nous avions aussi dans l'idée de créer des modes de jeu bonus comme un mode infini du jeu neuf lettres ou alors une course contre la montre de démineur mais nous avons abandonné ces projets.

Au niveau de l'architecture, Rémi qui a codé le jeu neuf lettre a voulu implémenter PAC au début puis par la suite le code fonctionne par composant (menu, solution...). Ce changement en milieu de projet est dommage car ducoup le projet ne respecte pas une méthode précise.