RAPPORT DE PROJET WEB

RÉDIGÉ PAR

ELMOTACIME Omar, LALIC Geoffrey

SOMMAIRE:

[**I. Modèle des tâches**](#_bwsvg14stozy) **3**

[**II. Maquettes prévues**](#_d1341g8pbh2c) **4**

[**III. Architecture logicielle**](#_30sl9crfa6k2) **5**

[III.1 Organisation des fichiers :](#_xsdam3sissp9) 5

[III.2 Description des relations entre les fichiers js et html et leurs fonctionnalités:](#_z6y7796lvgxz) 6

[**IV. Planning de réalisation des scénarios**](#_k9zmiggcab9v) **11**

[IV.1 Planning prévue](#_kh8vjxefjtol) 11

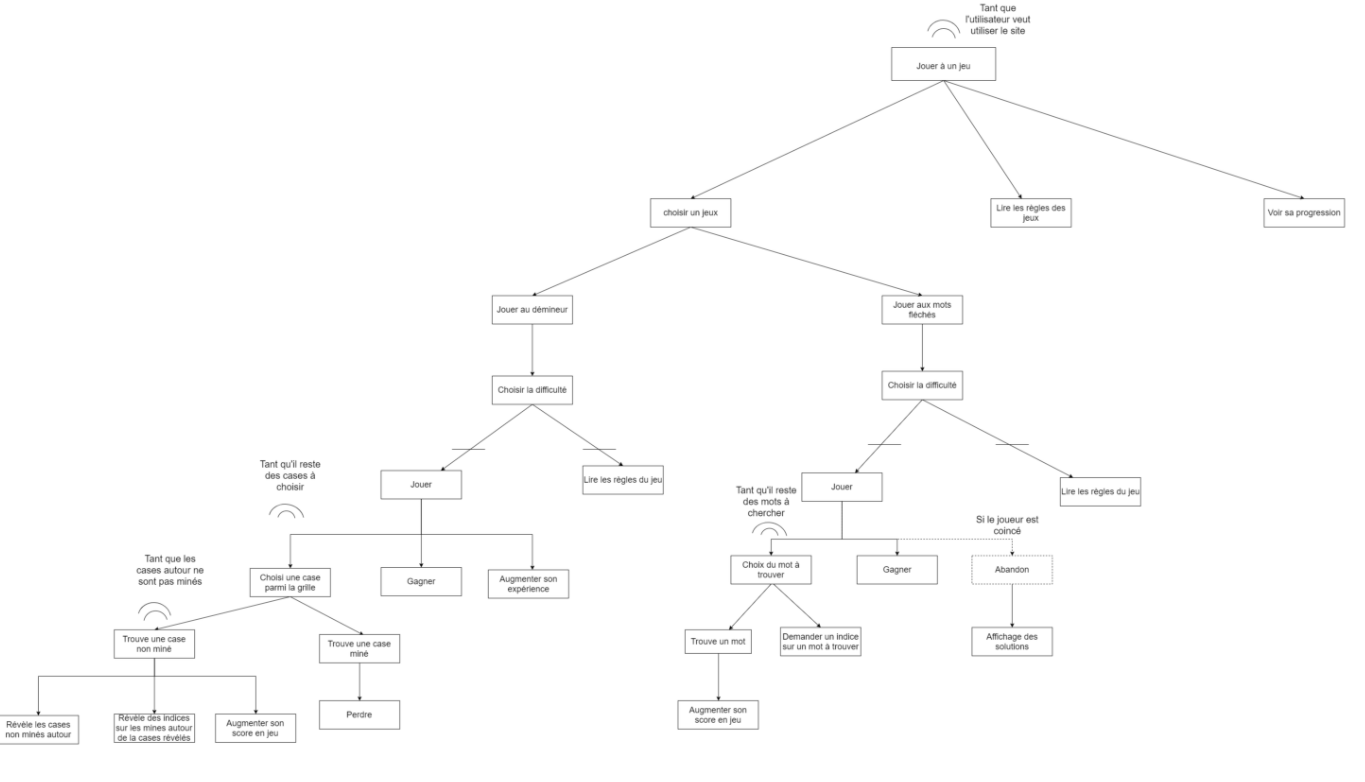
[IV.2 Planning réalisée](#_z5mpr3ll7fih) 11

[**V. Techniques utilisées ( js , css, ...)**](#_btcq4wcq3ux4) **12**

[VI. Comparaison et explication des différences entre ce qui était prévu et ce qui a été fait](#_mya20rjz5znx) **12**

# I. Modèle des tâches

Le joueur choisit un jeu parmi ceux proposés. Il peut choisir aussi de lire les règles ou de voir sa progression. Il peut choisir le démineur ou les mots fléchés. Sur chaque jeu, il choisit la difficulté. Si le joueur sait déjà jouer, il jouera une partie sinon il va au préalable lire les règles des jeux. Le joueur peut gagner, perdre ou abandonner (pour les mots fléchés). Pour le démineur, le joueur choisit une case qui va lui révéler des cases vides ou le faire perdre s'il tombe sur une mine. Il va répéter l'expérience jusqu'à qu'il révèle toutes les cases non minées. Il sera aidé par les cases contenant des indices sur la position des mines et augmente son score et son expérience à chaque partie. Pour les mots fléchés, le joueur choisit un mot à trouver dans la grille. S'il en trouve un mot, cela va augmenter son score en jeu sinon il peut demander des indices jusqu'à trouver les mots. Il va répéter l’expérience également jusqu'à trouver tous les mots de la grille ce qui le fera gagner la parti et de l’expérience.



# 

# 

# II. Maquettes prévues

Tout d’abord, nous avons une page d’accueil page, on y retrouve une barre de menu placé en haut de page, dont on retrouve différents boutons comme :

* Titre qui permet de rediriger vers la page d'accueil
* Démineur qui redirigera vers la page de sélection de niveaux du démineur
* Mots fléchés qui redirigera vers la page de sélection de niveaux des mots fléchés
* Règle des jeux qui redirigera vers la page des règles des jeux
* Progression qui redirigera vers la page de progression

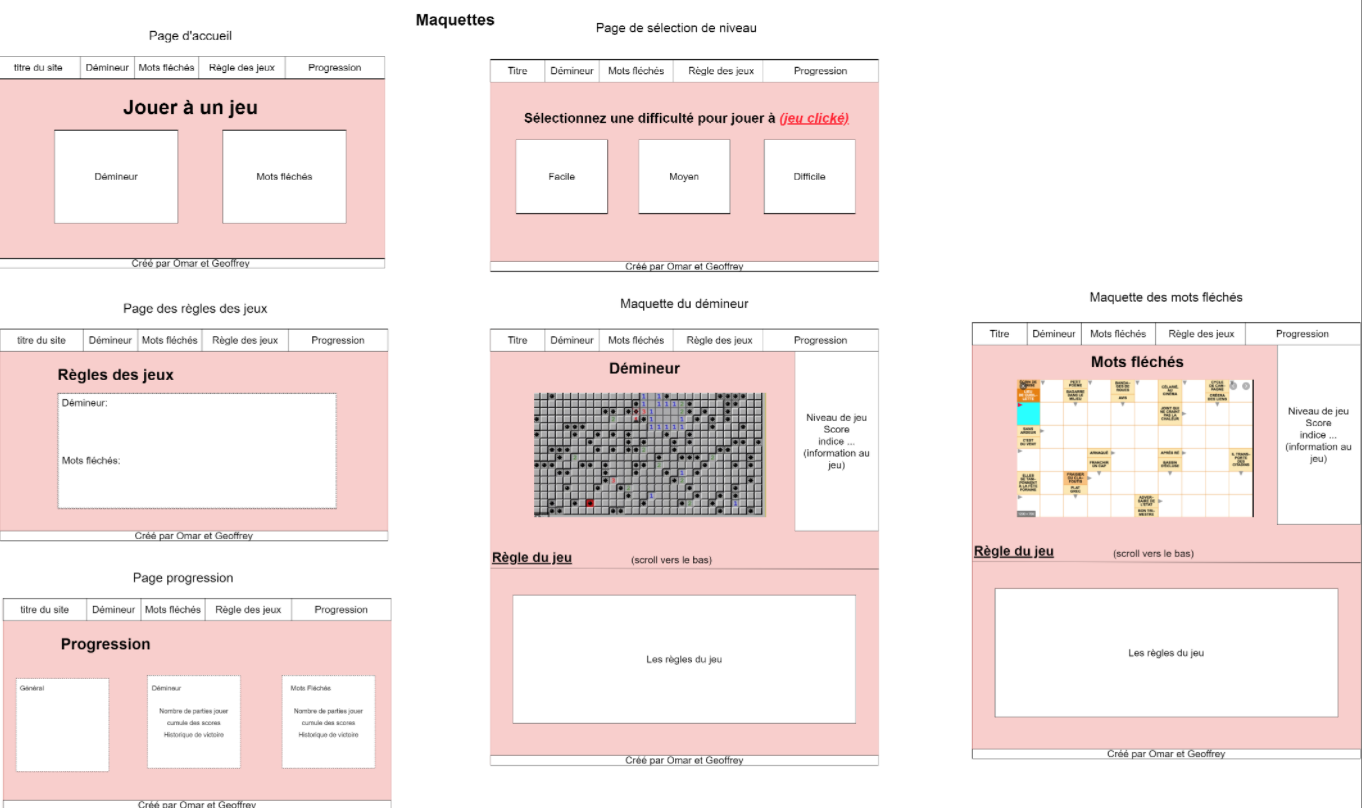
Pour ce qui est du contenu principal de cette page, qui est en dessous, on a 2 boutons qui redirigent vers une sélection de niveaux de démineur ou mots fléchés.

Pour la page de sélections niveaux, nous avons toujours la barre de menu. Pour ce qui est du contenu principal de cette page, qui est en dessous, on retrouve 3 boutons qui permettent à l'utilisateur de choisir son niveau de jeu par rapport au jeu sélectionné auparavant, qui sera mentionné et facilement différentiables par rapport à la typographie de la phrase "Sélectionnez une difficulté pour jouer à ... " (ici en rouge, italique, gras et souligné). Ceci permettra également de donner un rappel sur le jeu sélectionné dans une page précédente. Ainsi, on retrouve des niveaux facile, moyen et difficile.

Dans la page des règles, on a toujours la barre de menu qui nous permet de naviguer entre les différentes pages. On aura aussi différents paragraphes expliquant de manière concise et la plus explicite possible les règles des jeux démineurs et des mots fléchés.

Sur la page de jeu, nous retrouvons la même barre de menu sur le haut de la page, puis en descendant, il y a le nom du jeu sélectionné ainsi que la grille de jeu en dessous. Sur la droite, nous retrouvons des informations sur le jeu en cours comme le niveau de jeu sélectionné, le score, les indices, etc. Dans cette section, nous verrons apparaître toutes les informations liées au jeu. Puis sur une partie en dessous encore sur la partie visible, il y a un titre "Règle du jeu" ce titre permet d'indiquer à l'utilisateur de scroller vers le bas afin de voir les règles du jeu sélectionnées.

Enfin, pour la page progression, on aura encore la barre de menu. Pour le contenu principal de la page, on aura l’ensemble des progressions des différents jeux avec un système de points d’expérience. On aura aussi un système pour évaluer le niveau du joueur sur l’ensemble des jeux présent avec différents niveaux qui seront accordés au joueur en fonction du nombre de points d’expérience qu’il a.



# 

# 

# 

# 

# 

# 

# 

# 

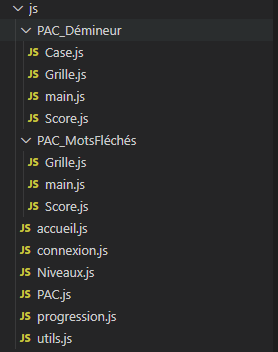
# 

# 

# III. Architecture logicielle

## III.1 Organisation des fichiers :

Pour ce qui est de l’organisation des fichiers, nous avons placé tous les fichiers js dans un dossier se nommant js placé à la racine du projet. Comme on peut le voir dans la l’image ci-dessus, le dossier PAC\_Démineur contient Case.js, Grille.js, main.js et Score.js. Le dossier PAC\_MotsFléchés contient également les fichiers Grille.js, main.js, Score.js. Ensuite, nous avons placé à l'extérieur des ses dossiers les fichiers accueil.js, connexion.js, Niveaux.js, PAC.js, progression.js et utils.js



## III.2 Description des relations entre les fichiers js et html et leurs fonctionnalités:

Avant tout, chaque page html contient une navbar qui permet de faire une navigation sur tout le site à n’importe quelle page. Ainsi, “Mini jeux” redirige vers la page d’accueil, “Mot Fléchés” redirige vers la page de sélection de niveaux de mots fléchés, “Démineur” redirige vers la page de sélection de niveaux de démineur, “règles” redirige vers la page de règles des jeux, “Progression” redirige vers la page de progression du joueur, “Connexion” redirige vers la page de connexion, déconnexion ou création de nouveau compte et le nom du joueur connecté si personne n’est connecté, il y aura écrit “aucun”. De plus, tous les fichiers html contiennent le fichier utils.js car il permet d’afficher le nom du pseudo en navbar grâce à la fonction localStorage.getItem(“listeJoueur”) qui permet de récupérer la variable stocké dans listeJoueur qui est une liste d’objet de type Joueur et vérifie s'il y a un joueur.actif = true qui se traduira par l’obtention du pseudo dans la navbar. Ainsi, utils.js est implémenté dans chaque page html. En outre, le fichier utils.js contient une fonction create2DArray qui permet de créer un tableau à deux dimensions selon le nombre de lignes et de colonnes choisies.

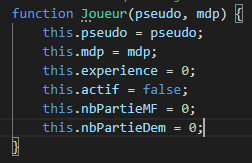
Les relations entre les fichiers html et les fichiers js sont les suivantes.

Le fichier index.html permet d’afficher la page d’accueil du site et est connecté au fichier accueil.js. Ce dernier, permet de stocker dans la variable listeJoueur une liste si la variable n’existe pas. Cette liste permettra par la suite de stocker des joueurs.

Les fichiers niveauxDémineur.html et niveauxMotFléchés.html contiennent uniquement le fichier utils.js car le contenue est du html, sur les balise nous avons ajouté un onclick=”localStorage.setItem(‘niveau’, uneValeur)” qui nous permettra par la suite de récupérer le niveau choisi par l’utilisateur.

Ensuite, le fichier règle ne contient aucun fichier js, car il affiche seulement les règles des jeux.

Le fichier connexion.html est connecté au fichiers connexion.js et utils.js. connexion.js contient un constructeur de Joueur et permet de créer des joueurs dont les propriétés sont: pseudo, mdp, expérience, actif qui permet de savoir si le joueur est connecté nbPartieMF qui calcule le nombre de parties effectué de mot fléchées nbPartieDem qui calcule pour le démineur.



Ensuite connexion.js permet également d’ajouter un profil, se connecter à un profil et se déconnecter d’un profil. Pour cela, nous avons au préalable ajouté des écouteurs sur les boutons de connexion déconnexion et d'enregistrement et nous avons récupéré les valeurs indiquées dans les champs texte grâce aux id dans le html et à la fonction document.getElementbyID(). Pour la création de profil, nous avons récupéré la liste de joueurs avec JSON.parse(), car la valeur était JSON.stringify auparavant. Nous vérifions si le pseudo entré existe déjà auquel cas nous demandons à l’utilisateur d’entrer un autre pseudo sinon, le profil est créé. Et nous renvoyons la liste de joueurs mis à jour en faisant JSON.stringify et localStorage.setItem(“listeJoueur”).

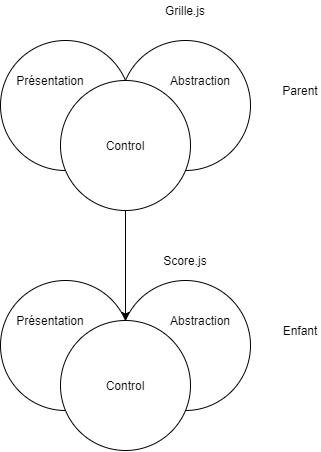
Pour ce qui est de la connexion, nous récupérons la liste de joueurs et nous vérifions si le pseudo et le mot de passe existent dans la liste de joueurs si il existe nous le passons actif = true. Et nous renvoyons la liste de joueurs mise à jour.

Et pour la déconnexion, nous parcourons la liste de joueurs et nous mettons tous les profil en actif = false et nous renvoyons la liste de joueurs mise à jour.

Le fichier progression.html contient les fichiers progression.js et utils.js. Progression.js permet de montrer les statistiques du joueur. En effet, il peut voir le nombre de parties jouées dans les deux jeux grâce à la récupération des id dans progression.html. Pour récupérer les données nous avons utilisé la fonction localStorage(“listeJoueur”) afin de récupérer le joueur connecté et nous avons récupéré les informations liées au nombre de parties jouées notamment joueur.nbPartieDem et joueur.nbPartieMF. Pour ce qui est des succès, nous avons comparé le nombre de parties jouées sur chaque jeu à 5 et nous lui octroyons un succès selon le jeu.

## III.2 Le modèle PAC:

Pour ce qui est des deux jeux, nous avons utilisé l’architecture PAC dont l’agent grille est le parent de l’agent score. Ces agents contiennent trois classe PresGrille, AbsGrille, CtrlGrille, PresScore, AbsScore, CtrlScore respectivement. Ces classes héritent des classes Pres, Abs et Ctrl respectivement qui se trouvent dans le fichier PAC.js. Les classes Pres permettent d’afficher sur le site, les classes Abs permettent de d’implémenter les fonctions ainsi que les variables et les classes Ctrl permettent de faire la jonction entre Pres et Abs. Ctrl permet aussi de communiquer à sa hiérarchie dans les deux sens (parent, enfant). Le fichier PAC.js contient également une classe MESSAGE qui contient les messages qui transitent à travers le modèle PAC. De plus, nous vérifions si un joueur est connecté grâce au localStorage.getItem(“listeJoueur”), en début de Grille.js.



Ainsi le fichier motFléchés.html contient tous les fichiers du dossier PAC\_MotsFléchés à savoir main.js, Grille.js, Score.js, Niveaux.js, PAC.js et utils.js. Le fichier main.js permet de lancer le modèle PAC des mots fléchés. En effet, on définit le parent et l’enfant de chaque PAC et grâce aux niveaux récupérés et à la fonction localStorage.getItem(“niveau”) dans la précédente page de sélection de niveaux, on charge le niveau avec les bonnes variables dans la grille. La variable niveau est un chiffre compris entre 0 et 5 qui retournera un élément à la position demandé dans la liste d’objet JEUX.

Le contenu du constructeur de PresGrille est composé des méthodes hérité de Pres, nbLignes; nbColonnes, tabCase qui permettra de construire la grille, tabCaseClick qui permet de stocker une case lorsqu'elle est cliquée et grille qui récupère l’id grilleMF nous l’avons pas créer la div grâce à un createElement(“div”) pour des raisons d’implémentation du BootStrap css.

Le contenu du constructeur de AbsGrille est composé des méthodes hérité de Abs, niveau qui est récupérer dans le paramètre d’AbsGrille, difficulte, listeDeMots qui représente la toutes les solutions de la grille ainsi que leurs emplacement, nbLignes qui est calculer sur le nombre de ligne de listeDeMots, nbColonnes qui calcul également qui calcule le nombre de caractère qui sur une ligne de listeDeMots, grilleUser qui permettra de récupérer la grille que l’utilisateur remplit cette grille utilise create2DArray en prenant en paramètre nbLignes et nbColonnes, listeIndices qui la liste des indices de la grille, solutionLigne et solutionColonne qui sont les solutions en ligne et en colonne du niveaux

Enfin, le contenu du constructeur de CtrlGrille est composé des méthodes héritées de Ctrl.

Une fois le main.js lancé, et que toutes les variables des PACScore et PACGrille ayant été instancié, nous utilisons la fonction init() de CtrlGrille qui permet de mettre en place la grille grâce au fonction dessineGrille() et remplirIndices(). Dans remplirIndices() nous avons initialisé les écouteurs sur le click et nous avons ajouter un écouteur “keypress” si une case a été clické. En parallèle dans la section du score, le chronomètre est lancé et les boutons sont également initialisés et toutes les informations liées au jeu y sont inscrites.

Quand l'utilisateur clique sur une case, celle-ci est marquée ensuite lorsque le joueur trouve un mot, le mot est marqué grâce aux classList de la fonction lettreTrouve() qui appliquera un ajout de classe sur la lettre trouvée. Ensuite, une vérification de chaque ligne et colonne sera faite grâce au fonction verificationLigne() et verificationColonne() de la classe AbsGrille après chaque saisie de lettres de l'utilisateur. Une fois que la grille de solution est égale à la grille de l'utilisateur, le jeu prend fin grâce à la fonction finDePartie() et met à jour les progressions du joueur grâce à majJoueur().

En ce qui concerne le score, nous avons ajouté des écouteurs sur les boutons indices, triche, rejouer, changer de niveau, quitter. Lorsque le joueur clique sur une case et clique sur le bouton indice la lettre correspondant à la case s'affiche la grille de l’utilisateur se met à jour dans l’abstraction. Lorsque le joueur clique sur rejouer le chronomètre et la grille se réinitialise grâce à rejouerScore(). Lorsque le joueur clique sur tricher la partie et le chronomètre s'arrête grâce à la fonction finTemps().Ensuite, les autres boutons quitter et changer de niveau redirige l'utilisateur vers les pages d'accueil et sélection de niveaux du jeu respectivement.

Tout comme les classes de Grille.js et Score.js des mots fléchés, les classes Grilles.js et Score.js du démineur hérite des classes du fichiers PAC.js

Une fois le main.js lancé, et que toutes les variables de Score et de Grille du dossier PAC\_Démineur et ayant été instancié, nous utilisons la fonction init() de CtrlGrille qui permet de mettre en place la grille grâce au fonction construireGrille() ,remplirTableau() et ajouterIndices(). Nous avons initialisé les écouteurs sur les différents clicks: click droit et click gauche. Click gauche est lié à la propagation et à la révélation des cases miné et non miné quand au click droit il est lié à la pose de drapeaux sur les cases.Le click droit est utilisé dans la fonction ajoutDrapeau() qui va à son tour ajouter un drapeau si il n’y a pas d’image sur la div.Quant au clic gauche, il va être utilisé par diffusion() ,diffusionIndice() et afficherMine().

Pour ce qui est des fonctionnalités du jeu, si l’utilisateur clique sur une case il y a soit une case vide qui se découvre soit une mine et dans ce cas-là le joueur perd la partie et une alerte se déclenche notamment grâce à la fonction clickSurCase() . Le joueur ne peut plus modifier la grille. Si le joueur clique sur une case soit cela va lancer la propagation soit cela va découvrir un indice sur les mines autour de la case cliquée. La partie est chronométrée. Si, le joueur met trop de temps à finir sa partie, il aura perdu aussi. Le joueur peut s’aider de drapeau en faisant une clique droite pour marquer l'emplacement sur lequel il pourrait y avoir des mines potentielles. Le nombre de drapeaux est limité au nombre de mines. Pour finir, le joueur a comme indication le nombre de mines qu’il y a été introduit.

# IV. Planning de réalisation des scénarios

## IV.1 Planning prévue

Voici la liste d’itération prévue.

Itération 1 : créer le site Création d'une maquette et un enchaînement entre les activités.  
Itération 2 : mise en place du premier jeu démineur. Création de la grille et possibilité de marquer un cas.  
Itération 3 : mise en place des différentes mines et indices dans la grille. Ajout des propagations lors du déminage lorsque l'utilisateur a choisi un cas non miné. Afficher les indices autour pour sonder les mines. Et mettre fin au jeu en cas de fin de jeu.  
Itération 4 : mise en place du deuxième jeu : mots fléchés Création de la grille. Entrer des lettres dans chaque cas. Ajout d'indice sur le mot à trouver ainsi que son sens. Et mettre fin au jeu en cas de fin de jeu.  
Itération 5 : ajout des vérifications sur les mots Vérification de chaque caractère par case de la grille.  
Itération 6 : ajout de niveau de difficulté sur les différents jeux  
Itération 7 : ajout des scores/réalisations Afficher lorsque le joueur gagne une partie, afficher son historique, ajouter  
Itération 8 : ajouter les règles et solutions de chaque partie dans chaque jeu.

## IV.2 Planning réalisée

Cependant les itérations n’ont pas été suivies à la lettre. En effet, la réalisation du premier jeu démineur aura pris plus de temps que nécessaire. Ainsi, nous nous sommes tournés vers le développement des jeux et de leurs logique sans contrainte de temps. Puis une partie de la navigation et l’implémentation des scores. Ensuite, l’implémentation des niveaux. Enfin, l’ajout d’une connexion.

Ainsi voici le planning des réalisations.

Semaine 1: création des maquette et scénarios  
Semaine 2-3: développement du jeu démineur  
Semaine 4-5-6: développement du jeu Mots Fléchés et un début de navigation  
Semaine 7: ajout des scores sur les deux jeux et complétion de la navigation  
Semaine 8: ajout de la connexion et correction des bugs.

# V. Techniques utilisées ( js , css, ...)

Nous avons utilisé beaucoup de techniques de programmation tout au long du projet:

* Pour les diagrammes nous avons utiliser diagrams.net <https://app.diagrams.net/>,un logiciel en ligne pour réaliser des diagrammes .
* Pour le CSS nous allons utiliser Bootstrap <https://getbootstrap.com/> pour faciliter l’apparition de notre site et qu’il soit conception réactive.Nous nous sommes tourné vers cette technique car elle était plus rapide ,simple et plus efficace.
* Pour le niveau facile des mots fléchés nous avons repris les indices et solutions du site [https://www.notretemps.com/jeux/jeux-en-ligne/mots-fleches-force-1-fevrier-2021-gratuits-23,i235504#](https://www.notretemps.com/jeux/jeux-en-ligne/mots-fleches-force-1-fevrier-2021-gratuits-23,i235504). Nous nous sommes beaucoup inspirés de ce site car on trouve que visuellement et au niveau conception il correspondait à l'idée que nous nous faisions du jeu.
* Pour démineur on s’est inspiré de : <http://demineur.hugames.fr/#level-3>.Nous nous sommes beaucoup inspirés de ce site également car on trouve que visuellement et au niveau conception il correspondait à l'idée que nous nous faisions du jeu.
* On a aussi utilisé certaines parties du code du TP cookies de M. Buffa et d’autre partie du TP de M. Renevier.Ces codes nous ont beaucoup aidé au départ notamment pour avoir une base de jeu qui soit solide et prendre une bonne direction dès le début.

# 

# VI. Comparaison et explication des différences entre ce qui était prévu et ce qui a été fait

Les itérations qu'avions prévues n’ont pas été respectées, car le développement des jeux était plus long que prévu, nous avions des problèmes sur les fonctions implémentées. De plus, l’apprentissage du modèle PAC nous a ralenti en début de projet ce qui nous a fait prendre du retard sur nos itérations. Ainsi, nous avons comme objectif de terminer les jeux et ensuite d’implémenter la navigation.

Nous avions prévu pour les maquettes de garder un style assez sobre pour le design du site finalement, nous avons amélioré sa conception en utilisant Bootstrap et nous a également permis de gagner du temps sur le positionnement des éléments dans la page. Nous avons un site un peu plus beau et meilleur dans sa conception.

Au niveau structure du code, nous sommes restés sur le modèle PAC comme nous l'avons vu en cours avec M. Renevier. Cependant sur l’implémentation des niveaux nous avons pensé qu’il aurait fallu mettre le niveau au constructeur du contrôleur et non sur la présentation du démineur ni dans l’abstraction des mots fléchés, tout comme l’implémentation du joueur connecté en début des fichiers Grille. Js, la connexion aurait été notifiée par le contrôleur de la grille et non pas sur une variable global, sinon créer un autre agent PAC pour la connexion. Nous aurions pu également améliorer la navigation en ne mettant qu’une page de sélection de niveaux qui s'adapterait au jeu cliqué précédemment.

En jeu, nous avions prévu de mettre les règles en bas de page. Cependant, nous avons jugé que la fenêtre aurait été chargée.

Nous avions prévu aussi de mettre des sons et des petites animations notamment sur le démineur, mais nous avons été pris par le temps à cause de notre manque d’organisation. Nous voulions privilégier un code qui marche et le plus complet possible quitte à rester simple dans la présentation.

Nous avons aussi voulu faire un démineur avec différents niveaux dont le nombre de lignes et de colonnes varient en fonction du niveau choisi, cependant, il aurait fallu refaire une partie du css lié aux dimensions de la grille afin qu’il s’adapte et l’implémentation aurait été compliquée de ce fait.

Nous avons de plus rajouté un onglet connexion qui permettra au joueur de se connecter et d’enregistrer ces scores pour augmenter son expérience. Cette implémentation n’était pas prévue cependant nous avions pensé qu’ajouter une page de connexion permettra de récupérer les données liées aux parties plus facilement en jeu.

Finalement, nous pouvons ajouter que le joueur obtient des succès après un certain nombre de parties. Il pourra avoir 2 succès selon son nombre de parties.