TELECOM Nancy

Projet Pluridisciplinaire d'Informatique Intégrative

Projet P2I2

(Wordlove + Solveur)

Serrand Coralie
Tejedor Manon
Theisse Alexandre
Yebouet Antoine

Responsable de module :
Olivier Festor

Table des matières

Chapitre 1

Remerciements

Tout d'abord, nous souhaiterions remercier nos encadrants et professeurs Mr Olivier Festor et Mr Gérald Oster pour les cours de programmation en C, structure de données et de Flask dispensés sans qui nous n'aurions pu mener ce projet à bien ainsi que pour l'encadrement lors du projet. Nous souhaitons également remercier Mme Heurtel pour ses conseils en gestion de projet pour la réalisation de la matrice RACI et du diagramme de Gantt ainsi que Mr Rémi Bachelet.

Chapitre 2

Résumé

Quotidiennement, de nombreuses personnes jouent à Wordle ou à ses équivalents et cet intérêt grimpe de jours en jours.

Ce projet est constitué de deux parties. Nous avons tout d'abord implémenté une application permettant de jouer à un jeu de type Wordle que nous avons nommé WordLove. Autour du jeu en lui-même, nous avons pu créer d'autres fonctionnalitées afin d'améliorer l'application Web.

Puis, nous avons ensuite implémenté un solveur du jeu. Accessible depuis un terminal, il suffit de rentrer les informations reçues par le jeu pour que s'affiche dans le terminal le mot à soumettre à l'application. Ce solveur est codé en langage C. Nous avons également mis en place des méthodes de gestion de projet. Dimension indispensable pour améliorer l'efficacité du travail en équipe et le bon déroulé de l'avancement du projet.

Chapitre 3

Introduction

3.1 Contexte du projet

Ce projet a été réalisé dans le cadre du Projet Pluridisciplinaire d'Informatique Intégrative pendant le second semestre de première année du cycle ingénieur à Telecom Nancy.

Wordle est un jeu appartenant au magasine The New York Times qui l'a acquis début 2022. L'engouement pour ce jeu a augmenté de jours en jours avec toujours plus de joueurs. L'utilisateur peut quotidiennement tenter de deviner un mot de cinq lettres proposé par le site, et cela en un nombre d'essais limité. De nombreuses répliques sont présentes sur le web et disponibles en plusieurs langues. En tant qu'élèves ingénieurs dans le numérique, il nous est demandé dans un premier temps de créer notre propre application web du jeu de type "wordle". Puis dans un second temps, nous devons implémenter un solveur qui doit nous permettre de jouer à notre application. Le but étant pour ce dernier de trouver le bon mot en peu d'essais.

Notre travail est décomposé en deux grandes parties qui sont elles même décomposées en quelques sous-parties. Tout d'abord un état de l'art des applications Wordle a été réalisé. A partir de celui-ci nous avons imaginé notre propre application : WordLove avec ses fonctionnalités propres. Nous avons alors évalué les besoins requis pour l'implémentation. Dans un troisième temps, nous avons lancé la phase de programmation de notre application. À la fin de celle-ci, nous avons testé et corrigé les différentes fonctionnalités.
La deuxième partie a aussi commencé par un état de l'art des solveurs Wordle. Dans un second temps, un accord a été mis en place sur les algorithmes qui composent notre solveur et les structures de données nécessaires à leur implémentation. S'en suit la programmation du solveur accompagnée de tests sur le bon fonctionnement des algorithmes. Enfin nous avons corrigé et testé les performances de notre solveur. De plus, pendant l'intégralité de ce projet, des méthodes et outils de gestion de projet ont été déployés.

3.2 Organisation du document

Dans un premier temps, nous présenterons notre application Web. Nous commenceront par présenter notre état de l'art sur les applications Wordle que l'on peut retrouver sur le web. Puis nous détaillerons la conception et l'implémentation de l'application.
Dans un second temps nous présenterons notre solveur et, tout comme l'application, nous présenterons d'abord l'état de l'art puis la conception et enfin les tests de performance de notre solveur.
Enfin nous présenterons la gestion de projet mise en place pendant l'entièreté du projet, comprenant les outils et méthodes. Nous terminerons par une conclusion.

Chapitre 4

Application Web

4.1 État de l'art

4.1.1 Introduction

Trouver le mot. Voilà le but du jeu WORDLE. Cette application WEB créée par Josh Wardle propose un mot par jour composé de 5 lettres uniquement. Le joueur possède six chances de trouver le mot du jour choisi de manière aléatoire. Après chaque proposition de mot par la joueur, le jeu donne des indications sur les lettres grâce à un code couleur. Si le mot n'est pas dans la langue française, l'utilisateur doit entrer un nouveau mot.
Les lettres au bon emplacement sont colorées en rouge, celles qui sont comprises dans le mot mais placées au mauvais endroit sont en jaune et enfin celles qui sont exclues du mot sont grises.

Depuis quelques mois ce jeu fait fureur sur les réseaux sociaux avec plusieurs milliers de joueurs chaque jour. Il est en réalité très facile de partager ses résultats avec les autres. Comme il n'y a qu'un mot par jour et que celui-ci est le même pour tout le monde, chacun peut facilement comparer ses résultats, le nombre de tentatives pour trouver le mot, le temps qu'il aura fallu pour le trouver...

Wordle est disponible pour les mots anglais, mais il existe d'autres versions dans différentes langues. Par exemple, on peut trouver la version française Le Mot [2]. Il existe aussi d'autres applications similaires comme Motus ou d'autres encore qui proposent des modes de jeux différents comme Quadrus.

4.1.2 Définition des critères

Suite à nos recherches sur les différentes applications disponibles, nous avons remarqué qu'elles sont assez similaires dans leur fonctionnement. Le principe reste le même, il faut trouver le (ou les) mots. Mais quelques fonctionnalités sont néanmoins différentes d'une application à une autre. Afin de comparer les applications, nous avons décidé de retenir les critères suivants :

— Taille des mots
— Mode de jeu
— Historique des parties
— Nombre de coups possibles

— Nombre de mots à deviner

Le critère de "taille des mots" correspond à la taille des mots à deviner que propose chaque application. Celui-ci peut être fixe ou variable
Le critère "mode de jeu" note les différents modes de jeu proposés par l'application. Par exemple si l'application propose un mode de jeu quotidient (un seul mot disponible par jour) ou bien un mode entraînement qui permet de jouer autant de fois qu'on le veut.
Celui sur l'"historique des parties" dit simplement si l'application propose un historique des parties pour un joueur donné.
Le critère "statistiques" indique si l'application web donne ou non les statistiques et performances du joueur (pourcentage de réussite, nombre d'essais, etc...).
Le "nombre de coups possibles" indique le nombre d'essais que l'utilisateur possède afin de découvrir le mot secret.
Enfin, le critère "nombre de mots à deviner" indique le nombre de mots à deviner en une seule partie.

4.1.3 Présentation des applications étudiées

Motus [3] est une application web dans laquelle l'utilisateur peut quotidiennement tenter de trouver un mot de la langue française en 4 à 6 lettres. Pour ce faire, l'utilisateur a 6 chances possibles et lorsqu'il entre un mot de la langue française, l'application lui indique via un code couleur si les lettres sont bien placées (code couleur rouge), mal placées (code couleur jaune), ou absentes du mot à deviner(lettres de même couleur que le fond : bleu). La première lettre du mot est également indiquée à l'utilisateur.
Le sutom [5] étant un dérivé du Motus, celui-ci fonctionne exactement de la même manière.

Wordle [10] a un fonctionnement similaire à celui du Motus, à l'exception près que cette application web propose des mots de la langue anglaise en 5 lettres. Les lettres bien placées sont colorées en vert, les mal placées en jaune et les lettres absentes du mot restent de la même couleur que le fond c'est à dire gris. Un mode daltonien est aussi disponible et modifie les couleurs. Son équivalent français "Le mot" fonctionne de la même manière.
Un mode de jeu "Hardmode" est disponible dans la version anglaise. Il nous oblige, une fois des lettres vertes trouvées, à proposer des mots contenant ces mêmes lettres vertes à leur emplacement. Ce mode de jeu est difficile car il est plus compliqué d'avoir de nouvelles informations pour chaque essais. La stratégie commune est de proposer des mots avec des lettres différentes à chaque fois jusqu'à notre tentative finale afin d'avoir le plus d'informations possibles sur les lettres composant le mot.

Le quadrus [4] est une application web au fonctionnement un peu plus atypique. Proposant toujours un défi quotidien, celle-ci propose de découvrir quatre mots de 5 lettres en 9 essais. Pour ce faire, un essai indique comme précédemment les lettres du mot ou non mais pour chacun des quatre mots. Le code couleur reste le même que celui du Wordle.

4.1.4 Comparatifs des différents Wordle ou concurents

Nom des sites	Taille du mots	Mode de Jeu	Historique des parties	Statistiques	Nombre de coups possibles	Nombre de mots à deviner
Sutom	4-6 lettres	Quotidien/ Entrainement	Non	Oui	6	1
Wordle	5	Quotidien/ Hardmode	Non	Oui	6	1
Motus	4-6	Partie	Non	Non	7	1
Quaddrus	5	Quotidien/ Entrainement	Non	Oui	8	4

4.1.5 Conclusion

Malgré la fait que les applications sont toutes plus ou moins de même type, elles ont chacune quelques petites particularités.
Notre application web se démarquera par un plus large pannel de fonctionnalitées comme un mode de jeu qui offre un design original, authentification, possibilité de choisir le nombre d'essais en plus de la longueur du mot, ajout d'amis et possibilité de voir leurs statistiques et performances.

4.2 Conception et implémentation de l'application

4.2.1 Introduction

Notre application web a donc comme premier objectif de pouvoir permettre à un utilisateur de jouer à notre WordLove.

Premièrement, nous avons mis en place un système d'authenfication. Un utilisateur connecté peut ainsi enregistrer ses parties.

Notre application possède une fonctionnalité pour jouer à une partie à l'aide d'un clavier interactif. Les couleurs des lettres après chaque essais sont mises à jours sur ce clavier. Cela offre au joueur la visibilité des lettres qu'il a déjà proposées. L'entrée clavier du PC de l'utilisateur permet aussi de jouer à l'application. Un mode de jeu avec un design différent est aussi disponible.

Chaque utilisateur peut également avoir accès aux règles du jeu.
Un utilisateur connecté a accès à plus de fonctionnalités. Il peut demander d'autres joueurs en amis et accepter ou refuser les personnes qui l'ont demandé. Un moteur de recherche est disponible pour trouver des utilisateurs à partir de leur nom d'utilisateur afin d'envoyer des demandes d'amis. Il peut également avoir accès à la liste de ses amis et au classement de ses amis. Il est possible d'avoir accès au classement des amis de ses amis et demander en ami les amis de nos amis n'étant pas nos propres amis.

Un utilisateur peut également avoir accès à ses statistiques comprenant le nombre de parties jouées, le pourcentage de victoires, le nombre de victoires consécutives actuel,et le plus grand nombre de victoires consécutives.

Un autre onglet est dédié au compte de la personne avec un bouton de déconnexion.
Nous avions premièrement fait un schéma de conception de notre application web.

Celui-ci s'est avéré être assez juste dans sa globalité. Nous avons simplement ajouté quelques fonctionnalités telles que l'accès aux amis de ses amis et les stastistiques de ses amis.

Notre application est codée en langage python et utilise le module Flask. Dans le dossier Wordle de notre projet, on peut retrouver le fichier app.py (qui contient les routes de l'application via des blueprints) ainsi que le fichier database.db qui contient toutes nos bases de données. Pour implémenter cette base de données nous avons utilisé sqlite3. Dans le dossier Wordle nous pouvons retrouver les dossiers suivants :

- templates (contennant les fichiers html)
- static
  - css
  - dictionnaire
  - img
  - js
- python
  - database (pour créer les bases de données)
  - functions

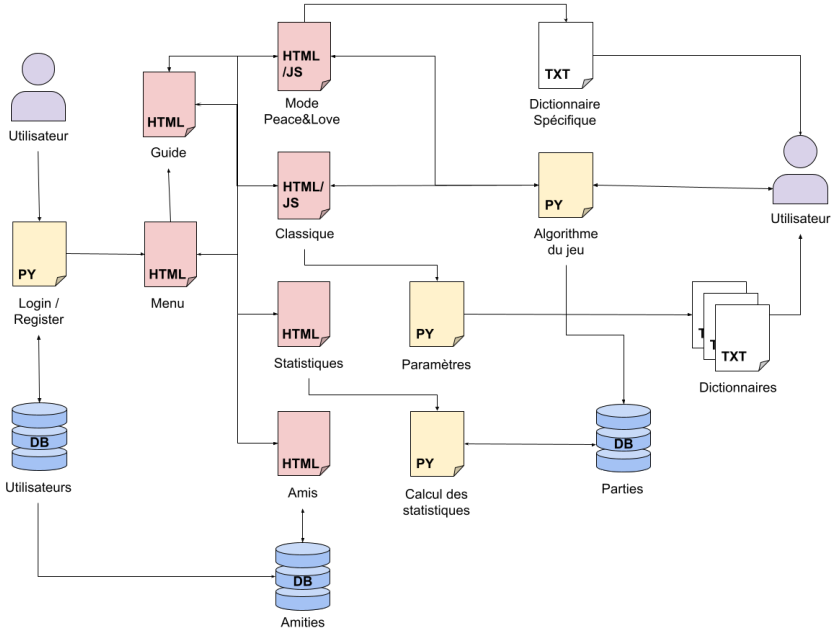


FIGURE 4.1 – Schéma de conception des intéractions

  — pages (liées au blueprint)
— data
— tests

Nous avons également utilisé du javascript qui permet de faire l'affichage interactif entre le clavier et la grille de jeu et que l'on peut retrouver dans le dossier static/js.

4.2.2 Web

Depuis le menu, il y a quatre boutons permettant d'accéder aux différentes fonctionnalités. Sur toutes les pages et notamment sur le menu, le layout constitué de 5 boutons (cf figure ci-dessous) permet aussi d'accéder aux différentes pages.

FIGURE 4.2 – Barre de navigation "layout"

Le premier bouton du menu, "Classique", permet d'accéder à une page permettant de lancer le jeu. Si l'utilisateur n'est pas connecté, la partie n'est pas enregistrée. La page de lancement demande à l'utilisateur le nombre d'essais qu'il veut avoir (compris entre 3 et 8), et la taille du mot qu'il veut deviner (compris entre 4 et 8).

L'utilisateur peut ensuite lancer le jeu et tombe sur une page avec un clavier et une grille de jeu correspondant aux paramètres entrés précédemment. L'utilisateur peut entrer des mots dans le dictionnaires français jusqu'à ce qu'il ait découvert le mot ou bien que le nombre d'essais soit utilisé.

Le clavier et la grille se mettent à jour au niveau des couleurs après chaque entrée de mot, c'est à dire que les lettres bien placées sont en vert, les mal placées en jaune et celles qui ne sont pas dans le mot sont en gris.
Le deuxième bouton du menu, "Peace&Love", redirige vers la même page de lancement que "Classique". Ensuite l'utilisateur a accès au même jeu à l'exception près que les lettres bien placées sont des coeurs rouges, les mals placées sont des coeurs roses et les mal placées restent grises.
Les deux autres boutons du menu ont les mêmes fonctionnalités que certains boutons du layout expliqués ci-dessous.

Le premier bouton du layout sous forme de " ?" permet d'avoir accès à une fenêtre dite "pop-up" avec les règles du jeu.

Le deuxième (équivalent du bouton "amis" du menu) permet au joueur connecté de voir ses amis sous forme de classement mais également leurs statistiques. Il peut aussi avoir accès à ses invitations envoyés, ses demandes d'amis reçues en attente et à une barre de recherche d'amis via différents onglets. Via la page qui affiche ses amis, l'utilisateur connecté a également accès au classement des amis de ses amis. Si un joueur n'est pas connecté il sera redirigé vers la page de connexion.

Le bouton "WordLove" du layout donne accès au menu.
Le quatrième bouton du layout permet à un utilisateur connecté d'avoir accès à son nom de compte et à un bouton de connexion. Sinon l'utilisateur est redirigé vers la page de connexion.

Le dernier bouton du layout donne accès aux statistiques du joueur connecté. Sinon de même que les autres pages, le joueur a accès à la page de connexion.

Sur la page de connexion, l'utilisateur peut simplement se connecter avec son mot de passe et son adresse mail. Sinon l'utilisateur peut avoir accès à la page d'enregistrement qui lui demande :

— Son nom d'utilisateur (qui doit être différent de ceux déjà existants)
— Son adresse email (qui doit être différente de celles déjà existantes)
— Son mot de passe
On utilise une fonction de hachage pour stocker les mots de passe dans la base de données.

4.2.3 Algorithmes de traitement

Afin de permettre le bon usage de notre application web nous avons dû implémenter un certain nombre d'algorithme de traitements sous le langage python.
Dans cette partie, nous détaillerons le fonctionnement de certaines de nos fonctions.

Recherche d'amis

Sur la page des amis d'un utilisateur connecté, nous avons créer un onglet qui permet de rechercher des amis.
Pour cela nous avons créer une fonction recherche d'amis qui va rechercher dans la base de données des utilisateurs qui ont dans leur nom le même item que celui tapé dans la barre de recherche.

Classement des amis

Toujours sur la page des amis d'un utilisateur connecté, nous pouvons voir ses amis classés du meilleur au plus faible.
Afin de réaliser ceci, la fonction va chercher dans la base de données les statistiques des amis de l'utilisateur connecté. Pour chaque amis, on attribue un nombre de points à l'ami en fonction du nombre d'essais qu'il fait pour deviner un mot pour chacune de ses parties.
Les amis apparaissent ensuite sur la page par ordre décroissants de points.

Jeu

Cette fonction permet de comparer le mot que l'utilisateur a proposé et de le comparer au mot à deviner.
Sachant que le mot à deviner se génère sur le serveur de l'application web, la fonction fait des requêtes au serveur afin de comparer le mot. La fonction compare ensuite lettre par lettre et fait appel à du javascript pour mettre les couleurs à jour.

4.2.4 Base de données

Nous avons implémentés plusieurs tables sur une même base de données. Elles sont implémentées comme ci-dessous :

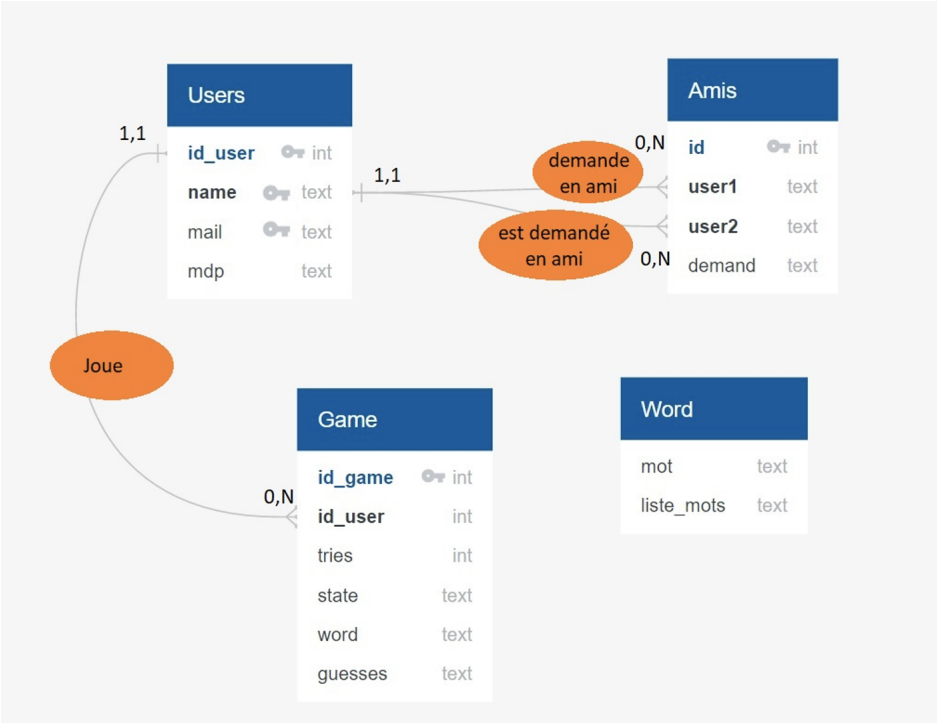


FIGURE 4.3 – Schéma relationnel de la base de données

Chapitre 5

Solveur

5.1 État de l'art

5.1.1 Introduction

Le Solveur peut être considéré comme un moyen de tricher au jeu Wordle. En effet, le solveur propose à l'utilisateur des mots à entrer qui sont optimisés pour arriver le plus vite possible au mot résultat.

5.1.2 Définition des critères

Suite à nos recherches, nous avons décidé de retenir les critères suivants :

— placer les lettres où l'on veut
— commence/termine par
— choix du dictionnaire
— choix du nombre de lettre du mot

Le critère "placer les lettres où l'on veut" précise si l'on peut indiquer l'emplacement des lettres qui sont bien ou mal placées dans le mot.
Celui nommé "commence/termine par" indique si l'on peut dire au solveur qu'un mot commence par telle ou telle lettre.
Le critère de choix du dictionnaire indique que le solveur laisse le choix parmi les dictionnaires disponibles, notamment en distinguant ceux des États-Unis et de Grande-Bretagne.
Enfin "choix du nombre de lettre du mot" dit s'il est possible de choisir la taille du mot que l'on veut deviner.

5.1.3 Présentation des solveurs étudiés

Pour notre état de l'art nous avons décidé d'étudier les solveurs suivants :
— WordFinder [8]
— The Word Finder [9]
— Solveur de 3blue1brown [7]

WordFinder

WordFinder est un solveur sur lequel on peut rentrer les lettres que l'on veut dans un mot. On peut également dire si le mot commence ou fini par certaines lettres ou encore si il contient certaines lettres.
Il propose ensuite une liste de mots qui est classée selon un ordre de points attribués aux mots. Les mots avec le plus de points sont les mots qui ont le plus de chance de correspondre.
Il ne propose pas de résolution complète du jeu, il est seulement une aide.

The Word Finder

The Word Finder est un solveur qui, pour les mots de 3 à 12 lettres, propose de rentrer les lettres bien placées que nous avons obtenues. De même il est possible de donner les lettres mal placées. Seulement il n'est possible de mettre qu'une lettre mal placées à un seul endroit.
Il est possible de donner les lettres non présentes dans le mot.
Le solveur propose ensuite, comme l'autre solveur, une liste de mots possibles, seulement ils ne sont pas classés par ordre de décroissance selon les probabilités.
Dans ce solveur sont également données la liste des voyelles les plus probables et des lettres des consonnes des plus probables.

Solveur de 3blue1brown

Le solveur de 3blue1brown (ou Grant Sanderson) est un solveur, qui en entrant en entrée la taille du mot, vous propose le choix qui permettra de vous donner le plus d'information et ce jusqu'à l'obtention du mot à deviner.
Dans ce solveur, l'information que donne le mot envoyé est calculée par le calcul de l'entropie du mot. Ce nombre chiffre l'information que nous apporte un mot sur la solution.

5.1.4 Comparatifs des différents solveurs

Nom des sites	placer les lettres où l'on veut	commence/ termine par	choix du diction- naire	choix du nombre de lettre du mot	lettres exclues	Lettres mal placées
Wordfinder	non	oui	oui	Oui	oui	Une lettre mal placée mais on indique pas l'endroit
The Word Finder	oui	oui	Non	Oui	oui	Lettre mal placée mais on indique qu'un endroit
Solveur 3blue1brown	non	oui	oui	oui	oui	Lettre mal placée

5.1.5 Conclusion

D'après cet état de l'art, on peut s'aperçevoir que tous les solveur se ressemblent et qu'il n'y a pas de différence majeure à première vue malgré leur différence de forme. La difference se fait dans l'implementation du solveur et dans les structures utilisées. On décide de mettre en place un solveur qui permet de résoudre l'entiereté de la partie. On se base sur la théorie de l'information afin de proposer un mot qui nous donne à chaque fois un maximum d'informations. Dans la communauté scientifique, il est d'avis que cela reste une des meilleures solutions pour résoudre un problème du type wordle.

5.2 Conception et implémentation du solveur

5.2.1 Introduction

Le Solveur va permettre trouver le mot à deviner de manière semi-automatisée, c'est-à-dire, le solveur communiqueras avec l'utilisateur via le terminal pou donner le mot qui va lui permettre d'obtenir le plus d'information. L'utilisateur renvoie via le terminal le modèle (pattern) qu'il a obtenu. L'ensemble du solveu est implémenté sous le langage C.

5.2.2 Explication du fonctionnement du solveur

Notre solveur se base sur la théorie de l'information [1] [6]. Le but est de calculer, à chaque étape de la partie, le mot qui nous permet d'avoir un maximum d'informations. Il nous faut donc chiffrer l'information qu'un mot est susceptible de nous donner. Pour ce faire on va chercher,pour chaque mot, à calculer son entropie. Plus celle-ci est grande, plus le mot est susceptible de nous donner de l'information. On note E l'entropie. x est un modèle (ou pattern).

$$E(mot) = \sum_{x} p(x) \times log_2(\tfrac{1}{p})$$

Ainsi pour calculer l'entropie d'un mot, il nous faut calculer pour chaque pattern le nombre d'informations que le mot associé au pattern nous donne. Ainsi pour x, un pattern, on a la proportion p :

$$p(x) = \tfrac{NombreDeMotsEncorePossible}{NombreDeMotsTotal}$$

Le nombre de mots encore possible est le nombre de mot qui correspondent avec le modèle associé au mot. Et le nombre de mots total est le nombre de mots qu'on a initialement avant ce traitement.

Le nombre de bits d'information est alors donné par le logarithme en base 2 de p. On a pour un pattern x :

$$I(x) = log(\tfrac{1}{p})$$

Dans l'exemple, ci-dessous, on voit l'impact d'un pattern qui garde seulement 25% de la liste initial. (On peut par exemple imaginer qu'on garde tous les mots terminant par 'ER') Puis l'impact d'un deuxième pattern qui garde lui aussi seulement 25% de la liste précédente. A chaque fois on calcule le nombre de bits d'informations.

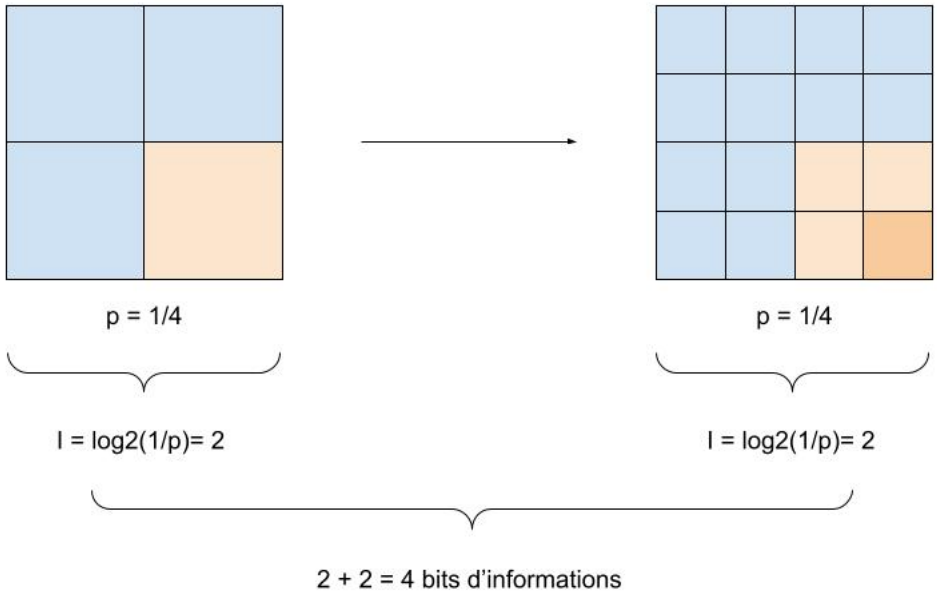


FIGURE 5.1 – Schéma explicatif de calcul du nombre de bit d'information

Ainsi dans notre calcul d'entropie, certains patterns vont certe donner beaucoup d'informations (I très grand), mais la proportion p, elle, sera petite. Et inversement.

Un pattern étant un tableau de taille égale au nombre de lettres composé de 0, 1 ou 2, on a pour un mot de taille m, 3^m pattern possible. Pour calculer l'entropie d'un seul mot il nous faut donc faire un calcul de nombre de mot pour chacun des 3^m modèle. Cela peut vite faire beaucoup de calculs.

5.2.3 Structures de données

Afin de mettre en oeuvre notre solveur, des structures de données étaient nécessaires. On stocke nos différents patterns dans une liste de pattern. Mais afin de limiter la complexité de nos algorithmes, on stocke nos mots dans un arbre. Ainsi lorsque l'on veut diminuer la liste de mot, on coupe les branches d'un arbre. Cela nous évite de parcourir des mots dont l'on sait très vite qu'ils ne correspondent pas au modèle. Par ailleurs, pour chaque mot, on doit effectuer une coupure différente d'un arbre pour chaque modèle. Afin d'éviter de devoir copier l'arbre initial pour le modifier, on calcule directement le nombre de mots générés par l'arbre en simulant des coupures de branches d'arbre. Ainsi, on effectue une réelle coupure seulement lorsqu'on obtient le modèle envoyé par l'utilisateur après qu'il ai rentré le mot proposé.

Nous avons donc implémenté les structures dans des fichiers ".h" comme suivant :

mot{
char* val ;
double entropy
}

La structure mot est composée d'une chaîne de caractère et de l'entropie du mot en question.

arbre_mots{
noeud *root ;
int nb_mots
}

Nous avons décidé de mettre en place une structure d'arbre qui contient l'ensemble des mots. La particularité de notre arbre est que la racine est une liste de noeuds et non un noeud tout seul comme pour un arbre classique. En effet, seulement la liste de fils du premier noeud nous intéresse. On stocke aussi le nombre de mots générés par l'arbre dans la variable nb_mots. Ce nombre est en fait égal au nombre de feuilles de l'arbre. Cela nous permet de calculer plus facilement l'impact d'un pattern sur un arbre sur le nombre de mots généré par celui-ci.

list{

noeud* head ;
int nb_mots ;
}

La structure list permet de lister l'ensemble des fils possible d'un noeud de l'arbre. Ce qui correspond à l'ensemble des lettres suivantes possibles. Elle pointe vers la tête de la liste 'head'. C'est une liste doublement chaînée composée de noeuds.

noeud{
noeud* prev ;
noeud* suivant ;
noeud* parent ;
list* liste_fils ;
int nb_mots ;
}

La structure noeud implémente les noeuds de l'arbre. Elle contient un pointeur vers la liste de ses fils (structure ci-dessus). La structure liste que nous implémentons est une liste doublement chaînée. Ainsi, le noeud qui est donc un élément de la liste possède un pointeur vers le noeud suivant et un autre un vers le noeud précédent. Le noeud possède aussi un pointeur vers le noeud parent afin de simplifier le parcours de remontée d'un arbre. Enfin le nombre de mots généré à partir du mot est stocké dans la variable nb_mots.

```
    pattern{
int *tab;
double entropy;
int size
}
```

La structure de pattern permet d'avoir l'entropie d'un modèle (pattern) pour un mot donné, la taille du mot et enfin elle contient un pointeur vers le tableau qui donne le modèle en question.

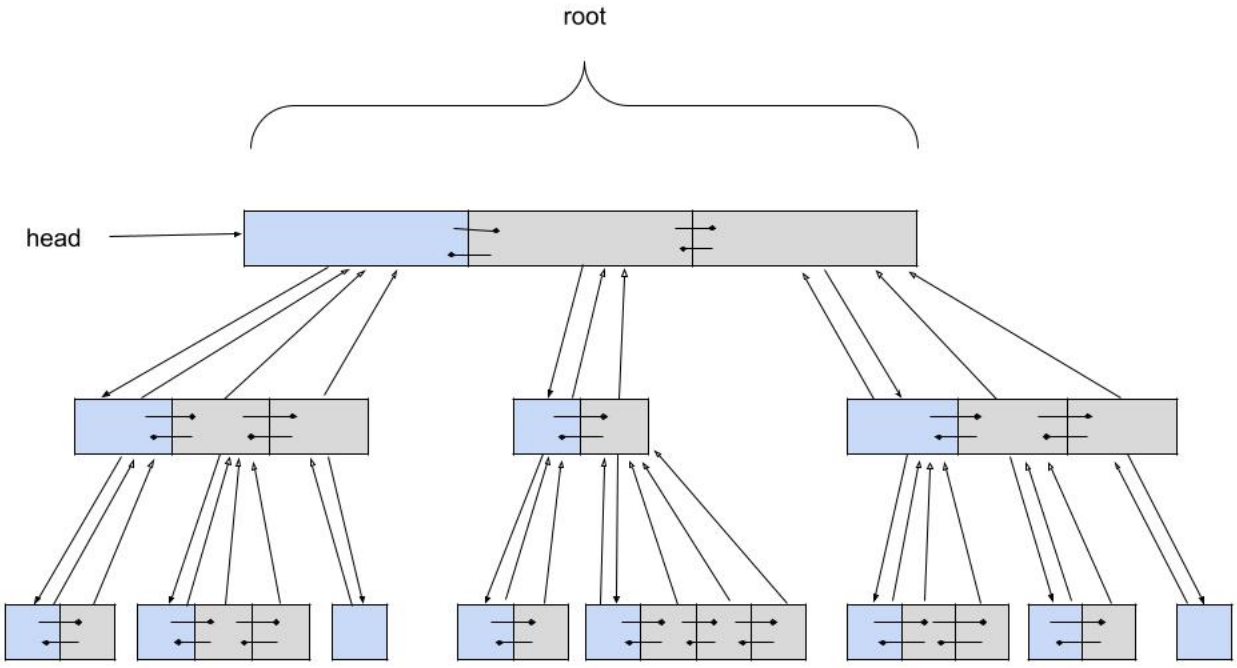


FIGURE 5.2 – Schéma de notre structure de données

5.2.4 Fonctions

Sachant que nous devons modifier beaucoup d'éléments, nos fonctions fonctionnent principalement par effet de bords et prennent en paramètre des pointeurs.

Main

Dans Arbre.c :

arbre_mots *arbre_init()

Cette fonction permet de créer un arbre vide de type arbre_mots et retourne un pointeur vers cet arbre.

noeud *node_init(char val,noeud* previous,noeud* parent)

Cette fonction permet de créer un noeud avec sa valeur, un pointeur vers noeud qui le précède, un pointeur vers le noeud parent et retourne un pointeur qui pointe vers ce noeud.

list *list_init()

Cette fonction permet de créer une liste doublement chainée vide de type list et retourne un pointeur vers la liste.

void lecture_fichier(arbre_mots* arbre,int taille)

Cette fonction permet de lire le fichier "liste_78k.txt" qui contient l'ensemble des mots de notre dictionnaire. Au fur et à mesure, les mots composées de "taille" lettres sont ajoutés dans l'arbre pris en paramètre. Elle fonctionne par effet de bords.

void arbre_append_mot(arbre_mots* arbre,char* mots)

Cette fonction permet d'ajouter une chaîne de caractère "mot" dans l'arbre pris en paramètre. Elle fonctionne par effet de bords.

void arbre_update(arbre_mots* one_arbre, mot* m, pattern *pat)

Cette fonction permet d'actualiser l'arbre pris en paramètre en supprimant les mots en fonction d'un pattern et d'un mot associé. Cette fonction fait appel à la fonction ci-dessous sur chacun des noeuds de sa racine.

Pour chaque lettre du mot associé à un 0 dans le pattern, on supprime tous les mots contenant cette lettre à cet emplacement. On fait aussi attention qu'un 1 ou un 2 n'est tout de même pas déjà associé à cette lettre, sinon il nous faut couper la branche si le nombre d'occurence de la lettre dans le mot est supérieur au nombre d'occurence connu dans la solution. Pour les lettres mal placées, on supprime tous les mots qui ont cette lettre placée au même endroit. Enfin, pour les lettres du pattern bien placées, on supprime tous les mots qui n'ont pas cette lettre au même endroit tout en faisant attention de garder les mots qui peuvent avoir plusieurs fois cette lettre.

void node_update(noeud* node, mot* m, pattern *pat,int depth)

Cette fonction récursive sur les noeuds permet d'actualiser l'arbre en supprimant les noeuds en fonction d'un pattern et d'un mot pris en paramètre. Pour un noeud, on parcourt le pattern pour savoir si on doit le supprimer ou non. Depth correspond à la profondeur de l'arbre pendant notre parcours.

Si le caractère du mot à la profondeur depth correspond à l'étiquette du noeud, on vérifie le

chiffre du pattern associé. Si c'est un 2, on le garde, si c'est un 0 on coupe la branche et on enlève le noeud.

Si les profondeurs ne correspondent pas, un 0 ne signifie pas forcément coupure, il nous faut calculer le nombre d'occurences de la lettre dans le mot parcouru et le mot réel. On coupe seulement si le premier est supérieur à l'autre.

Pour les 1, si les profondeurs correspondent en plus de la valeur, on coupe car la lettre est mal placée. Si on tombe sur une feuille, on compte le nombre d'occurence de la lettre dans le mot caché. On vérifie ensuite en parcourant les parents succesifs du noeud que le mot contient assez de la lettre en question.

La fonction est ensuite appelée de façon récursive sur les noeuds de sa liste_fils jusqu'à ce que la profondeur dépasse la taille des mots de l'arbre.

void noeud_remove(arbre_mots *arbre,noeud *node)

Cette fonction récursive sur les noeuds permet de supprimer un noeud de l'arbre ainsi que ses descendants. Elle est surtout utilisée dans arbre_destroy ou dans la suppression de noeuds dans arbre_update. La particularité de cette fonction est qu'elle prend en compte si une branche est toujours valide, c'est-à-dire si elle génère toujours des mots avec la bonne taille.

En effet, si un noeud est supprimé alors qu'il était le dernier élément de sa liste (liste_fils de son parent), la fonction fait un appel récursif sur son parent.

int arbre_nb_mot(arbre_mots *arbre, mot *m, pattern *pat)

Cette fonction calcule et renvoie le nombre total de mots qui correspondent avec le modèle pat et le mot m. Elle est ainsi utilisée dans arbre_proba pour calculer la proportion de mots restant dans la liste.

int noeud_nb_mot_coupe(noeud *node,mot *m,pattern *pat,int depth,char *str)

Cette fonction récursive sur les noeuds est appelée par la fonction ci-dessus. Le parcours d'arbre effectué est le même que dans celui de noeud_update. La différence étant qu'au lieu de rééellement couper les branches en supprimant les noeuds, on compte le nombre de mots générés par le noeud qu'on souhaite enlever. On rappelle uniquement la fonction sur les descendants des noeuds qu'on ne coupe pas. Comme on parcourt chaque noeud de l'arbre qu'une seule et unique fois, on s'assure que l'on n'enlève pas plus de mots qu'il ne faut.

double arbre_proba(arbre_mots* arbre, mot *m, pattern* one_pattern)

Cette fonction renvoie un double correspondant au quotient du nombre de mot restant dans l'arbre après mise-à-jour par rapport au pattern et au mot et le nombre total de mot dans l'arbre initial. Cette fonction est appelée pour calculer les bits d'information que donne chaque modèle ou pattern.

bool noeud_est_feuille(noeud *n)

Cette fonction permet de savoir si un noeud est une feuille.

bool liste_est_vide(liste *l)

Cette fonction permet de savoir si une liste est vide.

void arbre_destroy(arbre_mots *arbre)

Cette fonction permet de détruire l'arbre à la fin de l'algorithme pour éviter les fuites de mémoire.

void noeud_destroy(noeud *node)

Cette fonction récursive sur les noeuds permet de supprimer les noeuds d'un arbre. Elle est utilisée dans arbre_destroy. Elle fait plusieurs appels récursifs sur ses descendants et libère l'espace mémoire alloué au noeud ainsi qu'à sa liste de fils.

void printTree(arbre_mots *arbre)

Cette fonction permet d'afficher l'arbre. Elle était surtout utilisée pour trouver les bugs de nos algorithmes. En effet, il était plus simple de visualiser l'arbre de cette façon. On pouvait donc s'assurer du bon fonctionnement de nos fonctions.

void printNTree(noeud* n, bool flag[], int depth, bool isLast)

Cette fonction permet d'afficher l'arbre de manière verticale en utilisant la fonction printTree et nous a permis de trouver les bugs existants et de les résoudre.

void arbre_init_nb_mots(arbre_mots *arbre)

Cette fonction permet de calculer le nombre de mot dans l'arbre en comptant son nombre de feuilles. Elle initialise ainsi la variable arbre->nb_mot de l'arbre.

int noeud_init_nb_mots(noeud *node)

Cette fonction récursive sur les noeuds permet de calculer le nombre de mot générés par un noeud. On renvoie 1 pour le cas de base (le noeud est une feuille). Elle fait des appels récursifs sur les descendants du noeud en paramètre sinon. Comme elle retourne la valeur entière correspondant au nombre de mot, il suffit d'initialiser la variable nb_mots du noeud à la somme des nb_mots de ses descendants.

Dans mot.c :

mot *mot_create(char *str) ;

Cette fonction permet de créer en fonction d'un mot le mot de type mot avec sa valeur et son entropie.

void mot_destroy() ;

Cette fonction permet de détruire le mot de type mot en libérant l'espace mémoire alloué au mot.

void mot_arbre_explore(arbre_mots *arbre, noeud *node,int depth,char* str,mot *best, pattern **patterns)

Cette fonction permet d'explorer l'arbre afin de définir si la chaîne de caractère de type char* a une meilleur entropie que le meilleur mot best préalablement trouvé. Elle prend en paramètre un pointeur vers une liste de pattern contenant tous les modèles possible.

int mot_occurences_pattern(mot *m,pattern *p,char c)

Cette fonction permet de calculer le nombre d'occurences minimale du caractère dans le mot à trouver en fonction du pattern et du mot m. C'est-à-dire qu'on compte le nombre de 1 et de 2 associé au caractère. Contrairement à str_occurences, elle ne compte pas la lettre si il y a un 0. Cette fonction est utilisée pour les parcours d'arbre en fonction du pattern dans arbre_update et arbre_nb_mots.

int str_occurences(char* str,char c)

Cette fonction permet de calculer le nombre d'occurence d'un caractère dans le mot.

void mot_generate_best(arbre_mots *one_arbre,mot *m,int taille_mot)

Cette fonction permet de définir le meilleur mot à utiliser en fonction de l'arbre et en comparant tous les mots restant dans l'arbre entre eux pour définir le mot avec la plus grande entropie.

void mot_arbre_explore(arbre_mots *one_arbre,mot *m,int taille_mot)

C'est donc un parcours d'arbre classique qui calcul l'entropie dès qu'elle trouve un mot complet c'est-à-dire une feuille.

Dans pattern.c :

pattern** pattern_init_all(int taille)

Cette fonction permet de générer tous les patterns en fonction de la taille du mot.

double entropy_pat(pattern* one_pattern, mot* m,arbre_mots *arbre)

Cette fonction permet de générer l'entropie en fonction du pattern, du mot et de l'arbre .

Cette fonction se sert d'une formule pour calculer l'entropie d'un pattern. Pour cele, elle fait appel à une autre fonction que nous avons créée : arbre_proba(arbre_mots* arbre, mot *m, pattern* one_pattern). Celle-ci permettant d'avoir accès au nombre de mot coupés sur un arbre pour un mot donné, on peut faire le calcul pour avoir l'entropie.

double _entropy(pattern** patterns, mot* m,arbre_mots *arbre, int taille)

Cette fonction permet de calculer l'entropie d'un mot en parcourant tous les modèles de la liste patterns.

pattern *pattern_from_input(int* tab,int taille)

Cette fonction permet de récupérer le pattern du mot envoyé précedement dans le Wordle.

void pattern_print(pattern *one_pattern)

Cette fonction permet d'afficher un pattern.

void pattern_destroy(pattern* one_pattern)

Cette fonction permet de supprimer le pattern placé en argument.

void patterns_destroy(pattern** patterns)

Cette fonction permet de supprimer tous les patterns préalablement générées.

int base3(int n)

Cette fonction permet de calculer le nombre de patterns à faire en fonction de la taille du mot.

5.3 Tests et performances

5.3.1 Complexité

On étudie la complexité de l'algorithme mot_arbre_explore(). Cette fonction parcourt l'arbre et calcule l'entropie pour chaque mot. Elle fait appel à d'autres fonctions et représente le coeur de l'algorithme de notre solveur. On note m le nombre de lettre d'un mot.

Notre arbre est un arbre-N-aire lexicographique. Ainsi dans le pire cas, pour accéder à l'information on doit effectuer 26^m opérations. En effet, à chaque niveau de l'arbre (il y'en a m), Le solveur parcourt les 26 lettres de la liste fils à chaque fois. Cela correspond au mot "zzzzz" pour $m = 5$. Dans le meilleur cas, on effectue m opérations. Pour $m = 5$, c'est donc le mot "aaaaa". On a donc une complexité moyenne en

$$O(m) = \frac{m+26^m}{2}$$

En réalité, toutes les noeuds de l'arbre ne possèdent pas 26 lettres fils. Les mots cités précédemment ne font pas parti du dictionnaire. Par ailleurs, on peut noter que peu de mots commencent par x,y,z. Certes, beaucoup de mots vont commencer ou contenir des a et des e. Mais ces lettres sont en début de liste.

Lors du calcul d'entropie, il nous faut parcourir la totalité de la liste de pattern. Elle est de taille 3^m. En effet, pour chaque lettre du mot on peut avoir soit un 1, un 0 ou un 2. On multiplie donc notre complexité :

$$O(m) = \frac{m+26^m}{2} \times 3^m$$

De plus, pour chaque pattern, on parcourt l'arbre à nouveau pour calculer le nombre de mots restants dans l'arbre correspondant au pattern et mot associé. Au final notre complexité est donc :

$$O(m) = \frac{m+26^m}{2} \times 3^m \times \frac{m+26^m}{2}$$

Soit,

$$O(m) = \frac{(m+26^m)^2}{4} \times 3^m$$

FIGURE 5.3 – Graphique de la compléxité en fonction du nombre de lettres

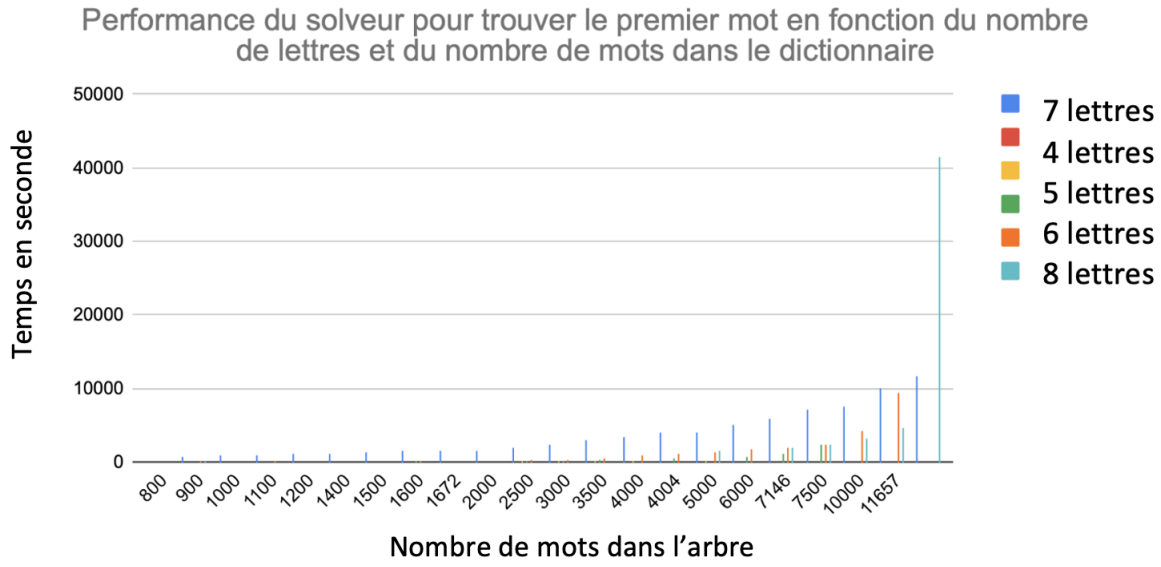
On remarque que la complexité augmente de façon critique à partir de 8 lettres (cf.figure 5.3). Cependant, si l'on garde à l'esprit que nos arbres contiennent un certain nombre de mots, à chaque fois qu'on diminue le nombre de mots dans l'arbre, les tailles des listes fils diminuent aussi. Si on note n la taille moyenne d'une liste de noeuds. On a alors

$$O(m, n) = n^{2\times m} \times 3^m$$

Ainsi, on peut observer l'influence du nombre de mots dans l'arbre sur la complexité. Ce nombre dépend du dictionnaire utilisé initialement. Il diminue aussi fortement au fur et à mesure des mise-à-jours de l'arbre.

5.3.2 Résultats et observation des performances

Nous avons ensuite effectué différents tests pour illustrer l'impact de la complexité des algorithmes et les performances du solveur.

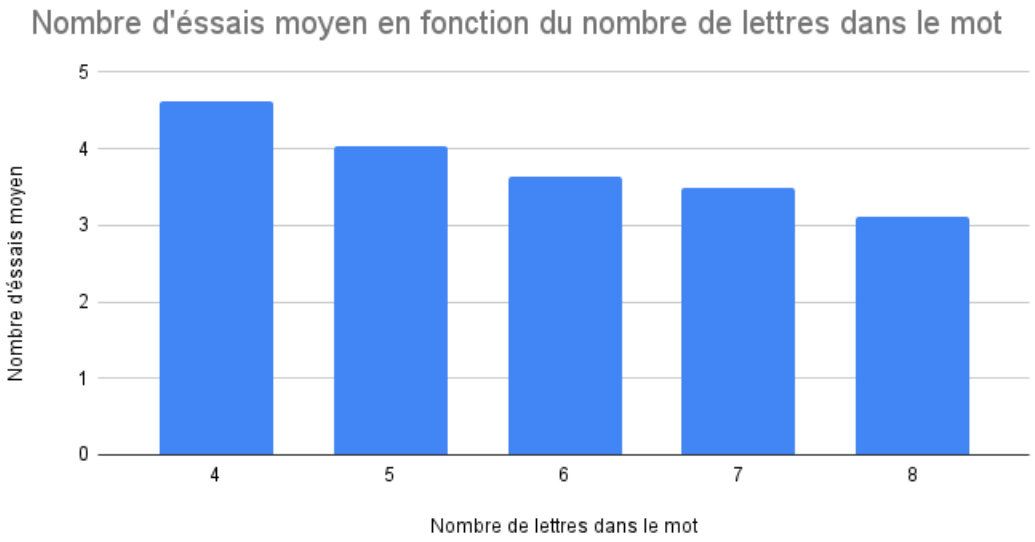


Ce graphique montre l'évolution des performances du solveur. Le solveur met de plus en plus de temps à calculer le mot le plus optimal au fur et à mesure que le nombre de lettre augmente. Cela est en accord avec le calcul de complexité précédent.

On remarque aussi que pour un même nombre de lettres, l'augmentation du temps de calcul se fait de manière exponentielle en fonction du nombre de mots. Notamment, lors de l'observation de la longueur du temps d'execution pour les mots de 8 lettres. On remarque que ce temps est assez long surtout lorsque l'arbre possède tous les mots initiaux. Néanmoins, ce premier calcul est déjà effectué au préalable lorsqu'un utilisateur lance le solveur. En effet le premier mot proposé par le solveur est toujours le même. Par ailleurs, notre solveur propose toujours les mêmes mots pour un mot solution déterminé.

En effet, une fois que le premier mot est calculé et que l'on obtient son pattern. Beaucoup de mots sont coupés pendant la mise à jour de l'arbre. Cela entraîne une augmentation de la vitesse d'éxécution pour des mots de 8 lettres.

Afin d'évaluer les performances en terme de nombre d'essais, on simule les parties pour trouver tous les mots de la liste et on note le nombre d'essais nécessaire au solveur pour trouver la solution. On calcule ensuite le nombre d'essais moyen en fonction du nombre de mots. On obtient les valeurs ci-dessous.



On observe que lorsqu'on augmente le nombre de lettres, le nombre d'essais diminue. La performance est donc meilleure. Le solveur a donc plus du mal pour les petits mots. En effet, il est plus difficile de gagner de l'information sur les lettres contenus dans le mots à chaque essais.
De plus, il s'avère qu'en moyenne une personne met généralement entre 3 et 4 coups pour trouver le mot. Ainsi notre solveur est au moins aussi performant qu'un Homme.
Cependant, nous avons pu observer que notre solveur ne trouvera jamais certains mots dans la limite du nombre d'essais maximal. Par exemple, pour le mot "RACE", le solveur trouve la solution en 11 essais. Heureusement, c'est le cas pour peu de mots.
D'un autre côté, pour des longs mots, notre solveur est plutôt performant.

Chapitre 6

Gestion de Projet

6.1 Équipe de projet

Nous sommes un groupe de projet constitué de quatre étudiants de première année :

— Serrand Coralie
— Tejedor Manon
— Theisse Alexandre
— Yebouet Antoine

Tout comme le précedent projet de Coralie et Manon, nous avons opté pour une structure fonctionnelle,c'est-à-dire sans chef de projet. Vis-à-vis de la taille du groupe et de la dynamique nous avons jugé que cela n'était pas nécessaire.

6.2 Outils de travail

Durant le projet, nous avons tenu plusieurs réunions afin d'échanger,de débloquer certains points et connaître l'avancée du travail de chacun. Les comptes-rendus peuvent être retrouvées en annexe.

Lorsque nous étions tous présents à Nancy, les réunions se tenaient à l'école. Pendant les vacances, nos réunions ont eu lieu sur la plateforme Discord. La plateforme Messenger a également servie afin d'échanger sur les heures des réunions qui étaient modulables notamment pendant les vacances de chacun ou bien encore pour s'entraider sur certains problèmes.0.2cm En fin de réunion, nous faisions des To-Do lists afin de nous répartir le travail et que le projet avance au mieux. De plus le diagramme de GANTT (que nous abordons plus tard) nous a permis de nous projetter dans le temps même si quelques problèmes ont été rencontrés.

Lors de la phase de dévellopement de l'application web et du solveur, nous nous sommes servi du git de l'école afin de déposer nos travaux respectifs. Nous nous sommes notamment beaucoup servi des branches de l'école. Nous avons décidé que chacun aurait la sienne afin de ne pas avoir de problèmes de merge sur le master et de ne pas se retrouver bloqués.
Le logiciel Visual Studio Code nous a également servi pour implémenter nos fonctions et structures. La rédaction du rapport s'est faite via la plateforme overleaf.

6.3 Déroulement

Afin de se répartir au mieux la charge de travail, nous nous sommes servis de certains éléments de gestion de projets.

6.3.1 Brainstorming

Au début de chaque grande phase du projet, application et solveur, nous avons fait une séance de brainstorming. Celle-ci nous a permi de mettre en commun nos idées pour et de discuter sur ce que nous pouvions faire. Cela nous a également permis de nous mettre d'accord sur les structures de données que nous allions utilisées pour le solveur.
Les craintes de chacun ont également été dissipées et chacun a pu comprendre les structures que nous voulions mettre en place. Ce brainstorming a été un moment privilégié pour chacun car il a permis à chacun de s'exprimer sur le sujet.

6.3.2 Matrice RACI, diagramme de GANTT

Afin d'implémenter le solveur et l'application web, nous avons dû réaliser une liste de tâches. Pendant les réunions nous avons nommées les tâches auquelles nous pensions, puis quelq'un a été chargé de les mettres en commun sur un fichier. C'est comme ça que nous avons, en réunion, attribuées les tâches de chacun, ce qui a donné la matrice RACI (Réalise,Autorité,Conseille,Informe). Sur cette matrice RACI, nous nous sommes dit les tâches à faire en priorité afin que nous ne nous retrouvions pas bloqués à cause des tâches liées entre elles. Les tâches ont également été réparties selon les domaines. 0.2cm Nous avions été assez clairvoyants sur ce que nous allions faire. Nous avons tout de même oublié quelques fonctionnalités que nous avons rajoutées ensuite.
Afin de communiquer et de s'entraider si besoin, nous avons utilisé l'application Discord. Cela nous a donné la possibilité d'échanger notamment au niveau du javascript, langage qu'Antoine avait déjà utilisé contrairement au reste du groupe.

Nous avons ensuite à partir de la matrice RACI réalisé le diagramme de GANTT, ce qui nous a permis de remettre en ordre les tâches que nous avions définies comme prioritaires et de s'organiser autour de cela .
Ce processus a été répeté pour l'application web et pour le solveur.
Les diagrammes de GANTT et matrices RACI sont disponibles en annexes. 0.2cm

Une particularité de ce projet était la phase de programmation du solveur. En effet, avant la phase de programmation, nous nous étions mis d'accord sur les structures et les fonctions nécessaires. Le langage C fonctionne avec plusieurs fichiers. Afin de ne pas empiéter les uns sur les autres, nous avions attribué un ensemble de fichier autour d'une structure à chaque personne du groupe. Puis nous avions mis en commun notre travail.

6.3.3 Jalons et PDCA

Afin d'éviter certains blocages que nous aurions pu rencontrés, nous avons décidé de mettre en place des réunions assez régulièrement, c'est à dire de mettre en oeuvre plusieurs PDCA.
Dans un premier temps, nous prévoyons les tâches à faire pour la prochaine fois, ceci grâce au diagramme de GANTT. Nous entamions ensuite la phase de dévellopement. Pendant cette phase, l'entraide était au coeur de notre groupe. Une personne était très souvent disponible si quelqu'un

avait des questions.
Puis lors de la réunion suivante, chacun annonçait ce qu'il avait fait et nous ajustions le diagramme de GANTT.
Nous réitérions ensuite le PDCA.

6.3.4 Matrice Swot

Une évaluation de nos forces et de nos faiblesses ainsi qu'une analyse des menaces et des opportunités ont été regroupées dans une matrice SWOT (ci-dessous).

Nous avons ainsi essayé d'éviter certaines menaces et de palier à nos faiblesses. Au niveau du C et de la SD nous avons travaillé ces matières de nos côtés afin d'avoir moins de problèmes lors du projet. Manon a emmené son ordinateur à l'étranger et a essayé de travailler là-bas.
Pourtant certaines choses n'ont pas pu être évitées. Nous avons effectivement moins travaillé les semaines où nous avions des partiels, et malgré le fait que Manon ait emporté son ordinateur à l'étrangé, il lui a été plus difficile de travailler pendant cette période. Les fêtes de pâques ont également ralenti le rythme de travail.
De plus nous avons été confrontés à des problèmes lors de la mise en commun du solveur. Les structures que nous avions mises en place n'ont pas toujours été suivies et cela a donc créer des problèmes pour les autres fonctions.

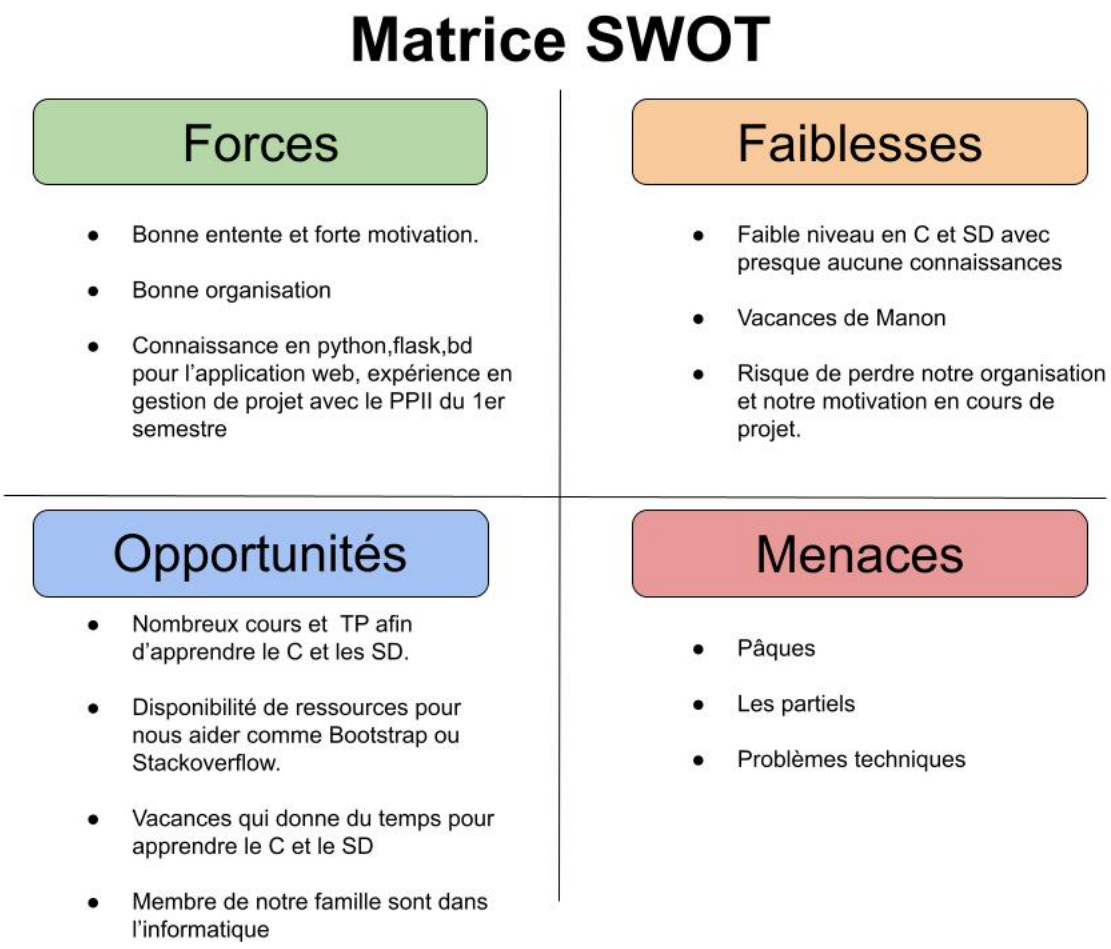


FIGURE 6.1 – Matrice SWOT

Chapitre 7

Bilan du projet

7.1 Conclusion

À l'issu de ce projet, nous avons donc implémenté une application web de type Wordle et un solveur du même jeu. Cela permettra aux joueurs d'avoir une nouvelle plateforme de jeu et de les aider si jamais ils bloquent pour certains mots.
Notre application permet de visionner leurs statistiques et de les comparées à celles de leurs amis. Elle propose également un jeu plus ludique au niveau du css.
Le solveur peut permettre à un utilisateur d'améliorer ses statistiques ou bien de le débloquer si il ne connaît pas le mot à deviner et donc potentiellement améliorer son vocabualire.

Afin de mener à bien ce projet nous avons utilisés des méthodes de gestion de projet que nous avions apprises au premier semestre.
Nous avions notamment mis en place des PDCA qui nous ont permis de rester en contact et de ne pas avoir de zones de flous pendant le durée totale du projet et d'organiser les grandes étapes du projet. En parralèle, les matrice RACI élaborées pour le solveur et l'application web nous ont permis de répartir les tâches équitablement sur des tâches préalablement définies. Celles-ci ont pu être réparties dans le temps grâce au diagramme de GANTT.

Toutefois, certains problèmes sont tout de même survenus.
Il était initialement prévu que nous finissions plus tôt notre application web. Nous avons finalement décaler la date afin de pouvoir peaufinner le rendu et ajouter quelques fonctionnalités auquelles nous n'avions pas pensé. Au départ, nous voulions que le mode de jeu Peace&Love ait un dictionnaire différent. Cependant, cela n'a pas pu être possible.
Quant au solveur, nous nous étions mis d'accord sur une certaine structure à implémenter avant la phase de programmation. Cependant,pendant les périodes de partiels, lors de la mise en commun, nous avons remarqué que quelques points avaient été échangé. Cela a donc créer quelques problèmes de concordance entre les différentes versions de structures.
Les échanges sur cette partie auraient donc dû être plus approfondis ou plus fréquents afin d'éviter d'avoir plusieurs fonctions déjà implémentées à recoder.

Malgrè ces quelques problèmes, notre application possède la très grande majorité des fonctionnalités prévues initialement. De même, le solveur est capable de faire ce pourquoi il a été conçu c'est à dire deviner un mot de taille donnée.
Toutefois, quelques points restent à améliorer. Notre application web aurait pu proposer plus de fonctionnalités comme peut-être la possibilité de partager ses scores avec des amis ou bien encore la possibilité d'échanger avec d'autres joueurs. De même le solveur pourrait être plus optimisé en changeant les structures ou bien encore en analysant les mots qui reviennent le plus souvent dans la langue française. Cela permettrait d'utiliser notre solveur pour d'autres applications.

7.2 Conclusions personelles

7.2.1 Serrand Coralie

Dans un premier temps, j'aimerais remercier l'équipe pédagogique pour avoir rédiger et imaginer un sujet intéressant et ludique. En effet, j'ai beaucoup apprécié travailler sur ce projet. Je me suis mise à jouer au wordle et j'ai aimé l'implémenter moi-même.

Dans un second temps, j'aimerai remercier Grant Sanderson pour sa vidéo sur Wordle et sur la théorie de l'information. J'ai l'ai trouvé très intéressante. Implémenter un solveur utilisant cette théorie était plutôt formateur. Je pense que ce projet m'a permis de progresser en C et de monter en compétence en structures de données. L'utilisation de pointeur et de croisement entre arbre et liste doublement chaînée m'a permi de bien distinguer les particularités de chaque structure. Par ailleurs, je remercie aussi Antoine pour les conseils apportés lors de l'utilisation du javascript lors du codage de l'application Web.

Néanmoins, j'ai trouvé compliqué de gérer période de partiel et projet. En effet, nous avons commencé la partie solveur juste après l'évaluation de l'application. Et au même moment, nous avions plusieurs partiels par semaine. Nous aurions probablement du débuter la partie solveur plus tôt.

En outre, j'ai particulièrement apprécié la partie algorithmique de ce projet qui était plus importante que dans le projet du premier semestre.

7.2.2 Tejedor Manon

J'aimerais premièrement remercier mes camarades pour ce projet.
Notre groupe a réussi à se lancer assez rapidement au début du projet afin de faire l'application web mais a connu plus de difficultés au niveau du solveur. Nous nous sommes lancés assez rapidement lors de cette partie mais nous n'avons pas fait assez de jalons et des quiproquos se sont fait ressentir lors de la mise en commun.

Le projet s'est toutefois bien déroulé dans sa globalité.

À titre personnel je pense avoir progressé sur certains points par rapport au précédent projet. J'ai notamment moins stréssé au moment de se mettre au projet par peur de manque de compétences. De même la partie dévellopement de l'application web a été plus rapide que sur le projet précédent car nous avions déjà acquis certaines compérences.

Également, je pense avoir acquis de nouvelles compétences en langage C, en javascript, et en Latex. Toutefois, quelques points pourraient encore être améliorés. N'ayant fait que le "main" en C, certaines compétences me manquent, qui ont été compensées par un plus gros travail que les autres membres du groupe en gestion de projet.

7.2.3 Theisse Alexandre

Ce projet m'as permis d'apprendre à coder de nouveaux languages de programmation telle que le C et le Javascript. Je pense m'être plus investi que le projet du semestre dernier car ce projet était à la fois plus dur et plus interresant.

Néanmoins, je pense pouvoir m'améliorer sur quelque point, notamment, dans ma productivitée pendant certaines périodes qui a un peu vascillé.

Enfin, je tiens à remercier mes camarades qui m'auront supporté durant tout le projet.

7.2.4 Yebouet Antoine

J'ai beaucoup apprécié participé à ce projet. J'ai trouvé que le groupe était très soudé, et malgré quelques ralentissements pour le solveur, tout était fluide au niveau de la communication et la répartition des tâches. C'était une expérience agréable, même si compliquée avec la période des partiels.

Ce projet m'a permis d'approfondir mes connaissances et ma pratique en structures de données et en language C particulièrement, mais aussi de confirmer l'acquisition des comptétences en programmation web avec flask. J'ai pris conscience de l'importance de la gestion de projet, beaucoup plus que lors du premier projet.

En outre j'ai trouvé ce sujet très intéressant, et remercie l'équipe pédagique de nous l'avoir proposé. Je remercie aussi Grant Sanderson (3Blue1Brown) qui grâce à sa vidéo, m'a permis d'en savoir plus sur la théorie de l'information et son application pour le jeu Wordle.

Chapitre 8

Annexes

8.1 Comptes-rendus des réunions

8.1.1 Mardi 22 Mars 2022

Compte-rendu n°1 - PPII : groupe E.8

Type de réunion : Réunion technique	Lieu : Terasse de l'école
Présents : — Serrand Coralie — Tejedor Manon — Theisse Alexandre — Yebouet Antoine	Le 22/03/22 De 16H00 à 16H40 Durée de la réunion : environ 0h40

Ordre du jour :

— État de l'art
— Que veut-on faire ?

État de l'art :

Il nous faut commenter chacun des sites de Wordle que nous connaissons et faire un tableau les comparants selon des critères que nous devons définir.

Nous connaisons ces sites que nous pourrions comparer :

— Sutom
— Motus

— Quadrus

Que veut-on faire ? :

Nous voulons premièrement nous concentrer sur un mode de jeu (en attendant de pouvoir faire le solveur), que nous complèterons avec d'autres modes de jeux et du solveur.

Alexandre partage les dessins d'une première visualisation de l'application. Il est décidé de les garder et de les compléter au fur et à mesure.

Nous voulons faire un mode de jeu solo classique qui apparaît dans tous les jeux de type Wordle. Les idées sont d'implémenter deux modes de jeu multijoueurs sur le concept du mastermind ou bien en s'affrontant pour découvrir le mot le plus rapidement possible (les critères seraient le temps ou bien le nombre de lignes utilisées pour trouver le mot).

Il serait possible également de choisir la taille du mot à deviner.
Il faudrait que l'application mette en couleur les lettres bien placées, mal placées et non contenues dans le mot rentré préalablement.
Pour cela, Antoine précise que l'on peut utiliser du css combiné avec du javascript (notamment la librairie p5 qui permet de créer un canvas).

Un premier planning est donné, réparti en trois catégories :

— La conception de l'application (2 semaines et quelques)
— Le solveur (5 semaines)
— Le rapport et les tests (2 semaines)
Il nous faudra également une base de données avec un schéma relationnel assez complet pour ne pas avoir à changer de base en cours de route.
Il faudra nottament une base de donnée pour tous les mots existants (fichiers .txt existants), l'hisotrique des parties (les statistiques) et l'historique de la partie en cours.

Nous avons choisi un thème pour notre application : les Hippies. Si cela est possible, nous mettrons un peace pour une lettre contenue dans le mot mais placée au mauvais endroit, un coeur rouge pour une lettre au bon endroit, et un coeur brisé pour une lettre qui n'appartient pas au mot.
Notre application s'appellera Worldlove.

Prochaine réunion :

Fixée pour le 29/03/22

TO DO LIST :

Tâches à faire	Par qui ?	Pour quand ?
État de l'art	Alexandre Antoine Coralie Manon	Prochaine réunion
Matrice SWOT	Alexandre	Prochaine réunion
Matrice RACI Diagramme de GANTT	Antoine	Prochaine réunion
Croquis de l'application	Coralie Manon	Prochaine réunion
CR	Manon	Prochaine réunion

8.1.2 Mardi 29 Mars 2022

Compte-rendu n°2 - PPII : groupe E.8

Type de réunion : Réunion d'avancement	Lieu : Salle 1.15
Présents : — Serrand Coralie — Tejedor Manon — Theisse Alexandre — Yebouet Antoine	Le 29/03/22 De 16H00 à 16H50 Durée de la réunion : environ 0h50

Ordre du jour :

— Vérification de ce qui a été fait
— Validation de la première étape
— Esquisse des idées pour le solveur

Vérification de ce qui a été fait :

La matrice SWOT a bien été faite et chacun a pu la regarder et y ajouter des idées.
Le croquis de l'application a également été fait et nous avons abordé ses différentes parties pour vérifier si il ne manquait rien.
De même, l'état de l'art a été fait et chacun doit vérifier le contenu afin de pouvoir valider la première étape du projet.
Coralie a également fait la partie conception de l'application que chacun est chargé de relire.
Un début de matrice RACI a été fait mais il faut se mettre d'accord sur la répartition des tâches et les tâches exactes à faire. Il faut donc la continuer. Le diagramme de Gantt dépendant de la matrice RACI n'a pas encore été fait.
Antoine a de plus arrangé le git avec différents dossiers pouvant servir à notre programmation.

Validation de la première étape :

Afin de faire valider la première étape de notre projet, la gestion de projet doit être faite. Nous nous mettons donc d'accord pour que chacun relise la gestion de projet pour envoyer le mail. Nous fixons une date avant la prochaine réunion afin que tout le monde relise.

Esquisse des idées pour le solveur :

Des rélexions sur la conception du solveur sont menées.
Il faudrait premièrement un dictionnaire trié par rapport à la taille des mots.
Nous nous disons que le solveur doit dans un premier temps proposer des mots qui permettent d'éliminer un maximum de lettres et donc de trouver les lettres qui vont et ne pas chercher un mot qui correspond à tout prix. Seulement il faut fixer une limite et que le solveur propose à un moment un mot qui correpond. Une réponse évidente est que s'il l'on arrive à la dernière proposition, le solveur doit obligatoirement proposer une réponse qui correspond et ne pas tenter de deviner simplement les lettres présentes.
Pour la recherche d'un mot, nous proposons que 3 listes soient mises en place, une avec les lettres bien placées, une avec celles mal placées et enfin une avec les lettres qui ne sont pas présentes dans le mot. Seulement, un problème se pose la question des lettres doublons. Si une lettre est bien ou mal placée elle peut également être à un autre endroit. Cette idée de liste ne prend pas cette considération en compte. Il faudras réfléchir à une autre méthode.

Prochaine réunion :

Fixée pour le 06/04/2022

TO DO LIST :

Tâches à faire	Par qui ?	Pour quand ?
Relecture de la gestion de projet	Tous	04/04/2022
Envoie du mail	Alexandre	04/04/2022
CR	Manon	Dans les 48H

8.1.3 Mercredi 06 Avril 2022

Compte-rendu n°3 - PPII : groupe E.8

Type de réunion : Réunion d'avancement	Lieu : Salle 1.17
Présents : — Serrand Coralie — Tejedor Manon — Theisse Alexandre — Yebouet Antoine	Le 06/04/22 De 18H00 à 18H30 Durée de la réunion : environ 0h30

Ordre du jour :

— Tâches réalisées
— Répartition des tâches à faire

Tâches réalisées :

Chacun a relu la partie gestion de projet ce qui a permis à Alexandre d'envoyer le mail afin de faire valider la première étape du projet. Alexandre a également push la matrice SWOT sur notre git.
Coralie a push le schéma de l'application, la conception et a également proposé un logo pour l'application.

Répartition des tâches à faire :

Le projet étant lancé, nous avons continué d'établir la matrice RACI. Pour cela nous avons fait une liste plus détaillée des tâches à faire, puis nous nous les sommes réparties selon les compétences nécessaires à leurs réalisations afin que chacun acquérisse des compétences dans tous les domaines.
Une question est posée concernant la langue utilisée pour la conception et pour l'utilisation du site. Nous décidons donc de mettre les variables et la conception du site en anglais mais de faire le site en français.
Nous définnissons également qu'il faudrat hacher le mot de passe dans la base de donnée afin de permettre une certaine confidentialité.
Concernant l'affichage d'une partie, il est choisit de mettre un clavier virtuel à l'écran permettant de donner les lettres bien placées, mal placées et celles qui ne sont pas présentes dans le mot.
Pour le programme permettant de comparer le mot entré au mot à deviner, une liste sera crée avec des 0, 1 et 2 en fonction de si la lettre est dans le mot secret ou non ce qui permettra d'attribuer une couleur à chaque chiffre (comme nous avions précédemment défini pour le solveur). Il y'aura une liste avec les lettres de l'alphabet pour cela.
Le clavier virtuel doit également pouvoir comprendre que l'on tape sur le clavier et afficher la lettre lorque l'on clique dessus.
Nous réalisons qu'un algorithme vérifiant que le mot est dans la langue française va également devoir être créer.
Nous reprécisons pour tout le monde que la base de données ne doit pas être push sur le git car elle est souvent source de conflits, seules les fonctions qui permettent de les créer et les modifier doivent être push.

Notre application proposera également de rechercher une personne pour l'ajouter à ses amis et comparer ses statistiques. Pour cela il nous faudra une algorithme de recherche permettant de trouver une personne par son nom.
Nous créerons donc une base de données permettant de réferencer les amitiées.
En premier lieu nous nous concentrerons sur les créations de pages et de bases de données qui permettrons de faire avancer au mieux le projet et de tester nos fonctionnalités.

Prochaine réunion :

Fixée pour le 11/04/2022

TO DO LIST :

Tâches à faire	Par qui ?	Pour quand ?
Avancer la programmation	Tous	Pour la prochaine réunion
CR	Manon	Dans les 48H

8.1.4 Lundi 11 Avril 2022

Compte-rendu n°4 - PPII : groupe E.8

Type de réunion : Réunion d'avancement	Lieu : Distanciel sur Discord
Présents : — Serrand Coralie — Tejedor Manon — Theisse Alexandre — Yebouet Antoine	Le 11/04/22 De 16H00 à 17H00 Durée de la réunion : environ 1h

Ordre du jour :

— Travail de chacun
— Merge des branches
— Mode Peace&Love

Travail de chacun :

Chacun a fait une partie du travail que nous avions noté comme prioritaire par rapport à notre application. Chacun cite à tour de rôle ce qu'il a fait et donc ce qu'il lui reste à faire en priorité. Alexandre a créé le layout qui reste à modifier, la page de jeu, qu'il a commencé à travailler, la fonction et la page qui permettent de rechercher des amis, d'où la création de la DataBase Friends

et enfin il a importé le dictionnaire pour le jeu.
Antoine a créé la page "home" et a commencé à la mettre en forme, la plupart des objets que nous devions y mettre étant déjà créés.
Manon a commencé à travailler sur la connexion au site et Coralie travaille sur la page d'enregistrement, d'où la création de la DataBase Users.
Nous devons également réfléchir sur comment il est possible de mettre un seul mot par jour.

Merge des branches :

N'ayant jamais utilisé de branches, nous avons ensemble vu comment faire et résoudre les conflits. Il est décidé que nous garderons ce mode de fonctionnement jusqu'à la fin du projet vu la facilité d'utilisation, et ce, afin d'éviter de gros conflits.
Antoine ayant déjà fait des branches pour son précédent projet, il a pour cette première réunion fait les merges des branches de chacun et nous a expliqué comment cela fonctionnait afin que chacun puisse merge ses branches à l'avenir.

Mode Peace&Love :

Il est décidé que le mode Peace&Love ne sera qu'une modification du css. En effet lors de la recherche de dictionnaires, il s'est avéré qu'aucun n'était vraiment pertinent avec le thème ou alors trop vulgaire et nous avons décidé qu'il n'était pas nécessaire de passer beaucoup de temps sur la constitution d'un dictionnaire à la main.
L'idée d'un dictionnaire spécial est donc abandonnée.

Prochaine réunion :

Fixée pour le 14/04/2022

TO DO LIST :

Tâches à faire	Par qui ?	Pour quand ?
Avancer sur les priorités de l'application	Tous	Pour la prochaine réunion
CR	Manon	Dans les 48H

8.1.5 Jeudi 14 Avril 2022

Compte-rendu n°5 - PPII : groupe E.8

Type de réunion : Réunion d'avancement	Lieu : En distanciel sur Discord
Présents : — Serrand Coralie — Tejedor Manon — Theisse Alexandre — Yebouet Antoine	Le 14/04/22 De 20H00 à 20H50 Durée de la réunion : environ 0h50

Ordre du jour :

— Travail de chacun
— Merge des branches

Travail de chacun :

Chacun a avancé sur le travail prioritaire qu'il avait à faire.
Antoine a avancé sur les statistiques de jeu, sur le clavier du jeu et a fait des modifications sur le menu.
Manon a avancé sur la page de connexion.
Alexandre a fait des modifications sur le layout qui reste tout de même à modifier notamment le css et les icônes présentes sur la page.
Coralie a fait des modifications sur son travail précédent et a modifié la page de jeu. Elle a également ajouté le logo de l'application.
De même qu'à la dernière réunion, chacun sait ce qu'il doit faire grâce à la matrice RACI et avancer le plus qu'il le peut.

Merge des branches :

Antoine s'est encore occupé pour cette réunion de merger les branches car nous avons décidé que le faire sur un seul ordinateur était plus simple et que le faire en réunion permettait de résoudre les conflits plus facilement.
Ainsi chaque branche a été merge.

Prochaine réunion :

Fixée pour le 18/04/2022

TO DO LIST :

Tâches à faire	Par qui ?	Pour quand ?
Avancer sur les priorités de l'application	Tous	Pour la prochaine réunion
CR	Manon	Dans les 48H

8.1.6 Lundi 18 Avril 2022

Compte-rendu n°6 - PPII : groupe E.8

Type de réunion : Réunion d'avancement	Lieu : Distanciel sur Discord
Présents : — Serrand Coralie — Tejedor Manon — Theisse Alexandre — Yebouet Antoine	Le 18/04/22 De 18H00 à 18H40 Durée de la réunion : environ 0h40

Ordre du jour :

— Travail de chacun
— Merge des branches
— Gantt

Travail de chacun :

Chacun cite le travail qu'il a fait.
Manon a rectifié des problèmes qu'elle avait trouvés notamment dans le fichier app et les statistiques, et a terminé le login. Elle a également mis à jour la base de données amis pour correpsondre a ce qu'elle avait à faire et a donc créé la page "mes amis".
Coralie a travaillé sur le clavier, a modifié le css de certaines pages, et a modifié la page "register".
Antoine a fait des correctifs sur le déroulement d'une partie et a également fait des mises à jour sur le clavier. De même, il a protégé l'url du jeu et a créé de quoi sauvegarder les données du jeu.
Il faut maintenant avancer notamment sur les statistiques, les amis, les deux modes de jeu, et le layout, c'est à dire mettre le titre, les icônes, et avoir accès à son compte et à un guide.

Merge des branches :

Le merge des branches a été fait de tel sorte à ce qu'il n'y ait pas de problème de merge et ceux qui étaient présents ont été résolus.

Gantt :

Antoine qui est chargé de faire le Gantt proprement demande si des applications sont connues pour le faire. Il est proposé de prendre un créateur de Gantt en ligne.
Antoine doit donc faire le Gantt pour la prochaine fois.

Prochaine réunion :

Fixée pour le 20/04/2022

TO DO LIST :

Tâches à faire	Par qui ?	Pour quand ?
Diagramme de GANTT	Antoine	Prochaine réunion
Avancer sur la matrice RACI	Tous	Prochaine réunion
CR	Manon	Dans les 48H

8.1.7 Mercredi 20 Avril 2022

Compte-rendu n°7 - PPII : groupe E.8

Type de réunion : Réunion d'avancement	Lieu : Distanciel sur Discord
Présents : — Serrand Coralie — Tejedor Manon — Theisse Alexandre — Yebouet Antoine	Le 20/04/22 De 14H00 à 14H55 Durée de la réunion : environ 0h55

Ordre du jour :

— Travail de chacun
— Merge des branches

Travail de chacun :

Chacun a dit ce qu'il avait fait depuis la fois précedente.
Antoine a réparé des erreurs sur la base de données du jeu et sur les statistiques. Il a travailé sur le Gantt également.
Alexandre a fini la page statistiques, et a fait du css pour la page recherche d'amis.
Manon a modifié des erreurs sur la page d'enregistrement et sur la page du login. Elle a crée un algorithme de comparaison du mot qui a est entré par l'utilisateur avec le mot à deviner.
Coralie a également modifié la fonction de login et le css de la page correspondante. Elle a fait une fonction de hachage pour que le mot de passe n'aparaisse pas en clair dans la base de données. Un onglet de demande d'amis a été ajouté à la page des amis.
Alexandre a créé une version améliorée du layout qui est proche de la version que nous souhaitions.

Merge des branches :

Comme à notre habitude lors des réunions, les branches ont été merges.

Prochaine réunion :

Fixée pour le 23/04/2022

TO DO LIST :

Tâches à faire	Par qui ?	Pour quand ?
Avancer sur la matrice RACI	Tous	Pour la prochaine réunion
CR	Manon	Dans les 48H

8.1.8 Samedi 23 Avril 2022

Compte-rendu n°8 - PPII : groupe E.8

Type de réunion : Réunion d'avancement	Lieu : En distanciel sur Discord
Présents : — Serrand Coralie — Tejedor Manon — Theisse Alexandre — Yebouet Antoine	Le 23/04/22 De 16H30 à 17H20 Durée de la réunion : environ 0h50

Ordre du jour :

— Travail de chacun
— Merge des branches

Travail de chacun :

Chacun a cité le travail qu'il avait fait.
Alexandre a posté une nouvelle version du layout.
Antoine a ajouté des modifications aux parties et aux bases de données.
Coralie a fait des corrections sur les demandes d'amis, a mis en place le classement des amis et a ajouté certaines fonctions pour la recherche d'amis.

Merge des branches :

Les branches ont été merges comme à chaque réunion.

Prochaine réunion :

Fixée pour le 25/04/2022

TO DO LIST :

Tâches à faire	Par qui ?	Pour quand ?
Avancer sur la matrice RACI	Tous	Prochaine réunion
CR	Manon	Dans les 48H

8.1.9 Lundi 25 Avril 2022

Compte-rendu n°9 - PPII : groupe E.8

Type de réunion : Réunion technique	Lieu : Cafétéria de l'école
Présents : — Serrand Coralie — Tejedor Manon — Theisse Alexandre — Yebouet Antoine	Le 25/04/22 De 14H00 à 18H00 Durée de la réunion : environ 4h00

Ordre du jour :

— Mise en commun
— Reste à faire

Mise en commun :

Antoine a fini le clavier, il commence la base de données jeu, et enregistre les statistiques.
Manon travaille encore sur le login car quelques problèmes sont survenus. Elle a également mis les couleurs sur le calvier correspondantes aux résulats, modifié des liens dans le layout et fait des modifications sur les amis.
Alexandre a terminé le layout, a arrangé les routes et a un peu regardé la game. Il a ajouté quelques fonctions également.
Coralie a fait la grille pour le jeu, corrigé des choses dans le layout, a fait des corrections de certaines choses et fait du css.

Reste à faire :

Il faut finir la gestion des parties.
De nouvelles fonctionnalités ont été pensées telles que avoir accès aux classement des amis de ses amis et pouvoir ajouter ceux avec qui nous ne sommes pas amis.

Prochaine réunion :

Fixée pour le 03/05/2022

TO DO LIST :

Tâches à faire	Par qui ?	Pour quand ?
Résoudre les derniers bugs	Tous	Avant samedi
CR	Manon	Dans les 48H

8.1.10 Mardi 05 Mai 2022

Compte-rendu n°10 - PPII : groupe E.8

Type de réunion : Réunion technique	Lieu : Une salle de l'école
Présents : — Serrand Coralie — Tejedor Manon — Theisse Alexandre — Yebouet Antoine	Le 03/05/22 De 14H00 à 15H00 Durée de la réunion : environ 1h00

Ordre du jour :

— Technique pour le solveur

Technique pour le solveur :

Il est dit qu'une vidéo explicative de comment faire un solveur existe et des membres du groupe l'ayant déjà visionné, ils ont pu expliquer aux autres le principe. Nous choisissons donc qu'une enthropie sera définie pour chaque mot c'est à dire qu'elle évaluera l'information qu'un mot peut potentiellement apporter à l'aide de patterns possibles selon le mot réponse. Il faut un arbre avec tous les mots possibles qui diminue à chaque fois. Il faut également une fonction qui génère tous les patterns.
Nous créerons donc les structures suivantes :
Pattern
MotLongueur, Liste pattern, Esperance bits, Charactères
Arbre de recherche Etiquette, Liste de fils, *prochain fils
Un main doit également être créés et sera l'interface avec l'utilisateur.

À la prochaine réunion il faudrat faire les fichiers ".h" afin de se répartir le travail. L'état de l'art doit également être fait pour cette partie du projet.

Prochaine réunion :

Fixée pour le 07/05/2022

TO DO LIST :

Tâches à faire	Par qui ?	Pour quand ?
Réfléchir sur le solveur et regarder la vidéo	Tous	Prochaine réunion
CR	Manon	Dans les 48H

8.1.11 Samedi 07 Mai 2022

Compte-rendu n°11 - PPII : groupe E.8

Type de réunion : Réunion technique	Lieu : Cafétéria de l'école
Présents : — Serrand Coralie — Tejedor Manon — Theisse Alexandre — Yebouet Antoine	Le 22/03/22 De 11H30 à 12H30 Durée de la réunion : environ 1h00

Ordre du jour :

— Coder les fichier ".h" du solveur
— Répartition du travail

Coder les fichier ".h" du solveur :

L'ensemble des fichiers ".h" ont été codés et ont été répartis selon trois fichiers :

— arbre.h
— mot.h
— pattern.h
Les fonctions associées ont également été signées dans les fichiers ".h".

Répartition du travail :

Il est décidé qu'Antoine codera les fonctions associées aux patterns, Coralie celles associées aux mots, Alexandre celles associées aux arbres et Manon fera le main.

Prochaine réunion :

Fixée pour le 13/05/2022

TO DO LIST :

Tâches à faire	Par qui ?	Pour quand ?
Avancer ses fonctions	Tous	Pour la prochaine réunion
CR	Manon	Dans les 48H

8.1.12 Vendredi 13 Mai 2022

Compte-rendu n°12 - PPII : groupe E.8

Type de réunion : Réunion technique	Lieu : Cafétéria de l'école
Présents : — Serrand Coralie — Tejedor Manon — Theisse Alexandre — Yebouet Antoine	Le 22/03/22 De 10H00 à 12H00 Durée de la réunion : environ 2h00

Ordre du jour :

— Ce que chacun a fait

Ce que chacun a fait :

Chacun a avancé sur son travail et des questions ont été posées au sein du groupe afin de lever les incompréhensions et les incertitudes. Notamment sur les liens entre les fonctions de chacun et la façon dont devait s'afficher le main par exemple.

Prochaine réunion :

Fixée pour le 26/05/2022

TO DO LIST :

Tâches à faire	Par qui ?	Pour quand ?
Finir ses fonctions	Tous	Prochaine réunion
CR	Manon	48H

8.2 Gestion de projet

		Coralie	Manon	Alexandre	Antoine	**Légende :**
Layout	Layout *			R		R = Réalise
Menu	Faire la page du menu *				R	C = Conseil
	Faire les boutons pour aller aux différentes pages *				R	
Guide	Faire l'icone du guide		R		C	
	Afficher le texte du guide		R		C	
Log in / Sign in	Faire l'icone s'enregistrer / se connecter	R				
	Faire la page de connexion	C	R			
	Faire la page d'enregistrement	R				
	Faire un hacher pour le mdp			R		
	Faire la BD Users *	R				
Jeu	Faire la page du jeu*			R		
	Afficher le clavier en bas de page				R	
	Lier l'entrer clavier aux cases	R				
	Update du clavier après chaque étape algo (griser)		R			
	Demander les paramètres en début de partie *			R	C	
	Afficher les cases et lignes en fonction des params *			R	C	
	Trouver les dictionnaires *		R			
	Algo d'existence du mot				R	
	Algo de comparaison entre réponse et mot	R	R	R	R	
	Afficher les couleurs après algo	R				
Statistiques	Faire la page de stats*				R	
	Afficher les stats				R	
	Faire les fonctions de calculs de stats			R		
	Faire la BD Game *				R	
Amis (Friends)	Faire la page des amis*		R			
	Faire la BD AddFriends *		R			
	Faire la BD Friendship *			R		
	Faire l'onglet rechercher des amis *			R		
	Faire le form de recherche avec nom du user			R		
	Algo de recherche amis	R				
	Afficher les Amis sur la page	R				
	Faire l'onglet demande d'amis (reçue) *		R			
	Boutons accepter et refuser amis	R				
	Faire l'accès aux stats de tes amis		R			
	Afficher les amis de ton ami				R	
	Classement amis	R				

FIGURE 8.1 – Matrice RACI de l'application Web

		Coralie	Manon	Alexandre	Antoine	**Légende :**
Main			R			R = Réalise
Arbre.h	Proba			R		C = Conseil
	Arbre_init			R		
	Arbre_update			R		
	Taille_arbre			R		
Mot.h	Mot_generate_best	R				
	Mot_get_val	R				
Pattern.h	Pattern_init				R	
	Entropy_pat				R	
	List_entro				R	
	Pattern_from_input				R	

FIGURE 8.2 – Matrice RACI du solveur

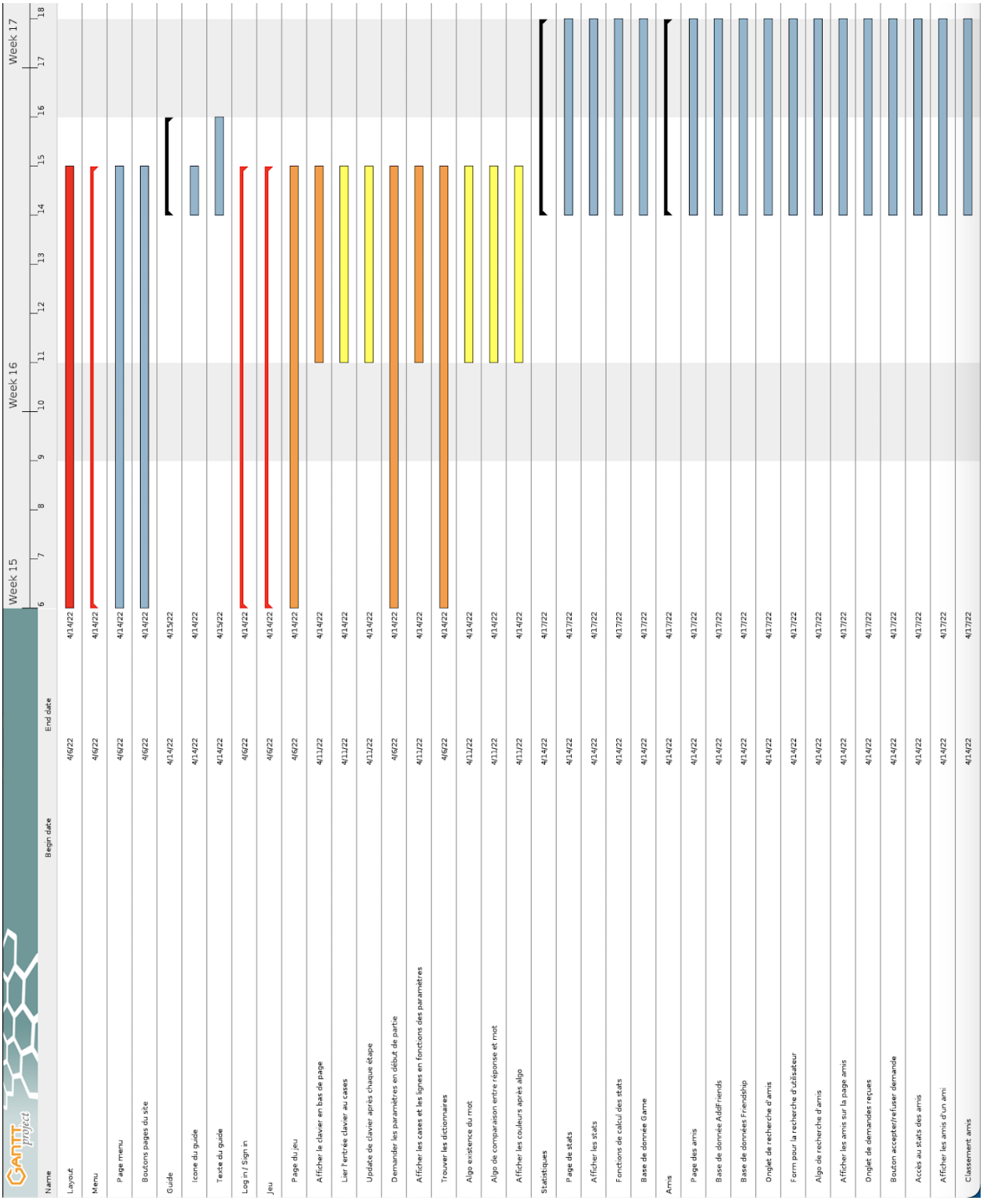


FIGURE 8.3 – Diagramme de GANTT de l'application Web

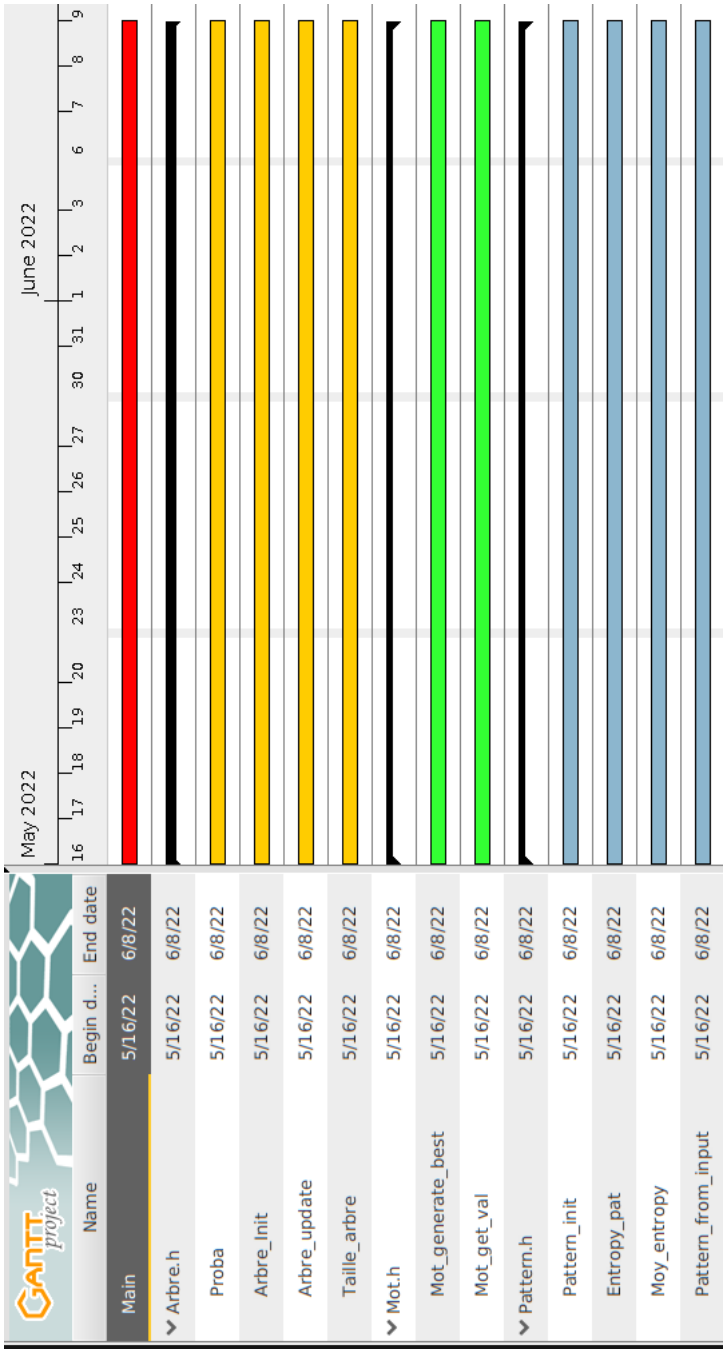


FIGURE 8.4 – Diagramme de GANTT du Solveur

Bibliographie

[1] Explication de la théorie de l'information, wikipédia. https ://fr.wikipedia.org/wiki/Théorie$_d e'_l information$.

[2] Le mot. https ://wordle.louan.me.

[3] Motus. https ://motus.absolu-puzzle.com.

[4] Quadrus. http ://quadrus.guigro.com//.

[5] Sutom. https ://sutom.nocle.fr.

[6] Théorie de l'information. https ://culturemath.ens.fr/thematiques/probabilites/qu-est-ce-que-la-theorie-de-l-information.

[7] Vidéo de la chaîne youtube 3blue1brown. https ://www.youtube.com/watch ?v=v68zYyaEmEA.

[8] Wordfinder. https ://wordfinderx.com/.

[9] The wordfinder. https ://www.thewordfinder.com/.

[10] Wordle. https ://www.nytimes.com/games/wordle/index.html.