PPII2 2022

Rapport de projet de PPII2

Tom BENE
Alexandre LI
Camille MOUSSU
Guillaume RICARD

Responsable de module :
Olivier Festor

Table des matières

Introduction

Contexte

Ce projet a été réalisé dans le cadre de notre second semestre de première année du cycle ingénieur sous statut étudiant de TELECOM Nancy.
L'objectif est de concevoir notre propre instance du WORDLE, et également un solveur WORDLE, i.e. un programme qui reproduit le comportement d'un utilisateur et qui résout les puzzles WORDLE.
Le travail s'est décomposé en trois parties : la création d'un site similaire à WORDLE, l'implémentation en C d'un solveur pour ce site et la gestion du projet en équipe.

Plan

Dans le chapitre 1 , nous présenterons le concept du WORDLE ainsi que les stratégies possibles pour la mise en place d'un solveur.
Dans le chapitre 2, nous présenterons les éléments ainsi que les outils de gestion de projet que nous avons utilisé.
Dans le chapitre 3, nous présenterons la conception de notre jeu : Claudle ainsi que le solveur associé
Dans le chapitre 4, nous verrons les tests effectués et les performances de notre travail.
Dans le chapitre 5, nous réaliserons un bilan du projet, d'un point de vue personnel et global.

Chapitre 1

État de l'art

1.1 WORDLE

1.1.1 Principe

WORDLE est un jeu de lettres inspiré de l'émission américaine Lingo. Comme dans sa version française Motus, le joueur doit deviner un mot avec un nombre de lettres précis (dans WORDLE 5) en un nombre de tentatives fixé (dans WORDLE 6).

Le joueur écrit dans chaque ligne un mot avec un nombre de lettres précis. Si le mot correspond à un mot dans le dictionnaire de WORDLE, le jeu vérifie la disposition de chaque lettre par rapport à celles du mot à trouver et la colore selon 3 critères :

- La lettre n'est pas dans le mot : gris
- La lettre est dans le mot mais pas au bon endroit : jaune
- La lettre est dans le mot et au bon endroit : vert

À la fin de la partie, le jeu donne au joueur des statistiques sur les parties qu'il a joué, le nombre de victoires à chaque étape.

1.1.2 Implémentation du jeu

Puisque le code de Wordle n'est pas disponible au grand public, nous allons analyser l'implémentation de SUTOM, la version française de Wordle.
SUTOM est implémenté avec un front end en TypeScript, un sur-ensemble syntaxique strict de JavaScript. Cela permet de mettre à jour la page du jeu sans avoir à la recharger.
Le dictionnaire est stocké dans un fichier texte et accédé via un script JavaScript.
Le site est divisé en pages et en composants. Cela permet de coder chaque fonctionnalité dans un fichier à part, puis tout ré-assembler en une page et un site fonctionnel comme un puzzle.

1.2 Le solveur

1.2.1 Idée générale

Le principe du solveur est relativement simple, une approche naïve consiste à dire que, connaissant le nombre de lettres du mot recherché, il suffit au solveur de proposer des mots ayant ce bon nombre de lettre jusqu'à ce qu'il trouve le bon mot. Néanmoins, étant donné qu'il existe environ 16000 mots de 6 lettres dans la langue française, il est évident que cette solution n'est pas viable.

On va donc devoir utiliser les informations (Lettre bien placée, lettre mal placée, lettre pas dans le mot) pour affiner les choix que va faire le solveur à chaque tentative. Là encore une solution simple serait de lister tous les mots "matchant" le pattern proposé et d'en proposer 1 au hasard, néanmoins on devine bien que cela risque d'être très peu efficace.
Pour permettre au solveur de trouver en un temps "raisonnable" il convient de : [1] [3]

1. Optimiser le choix d'un mot proposé par le solveur en fonction des informations qu'il va pouvoir fournir
2. Quantifier la probabilité que chaque mot soit solution

Ces 2 points sont traités dans la section suivante.

1.2.2 Un peu de théorie de l'information

Pour pouvoir développer une stratégie, il nous faut d'abord définir certaines notions que l'on va utiliser pour décrire et quantifier les informations à notre disposition.

En théorie de l'information, l'information de Shannon est indispensable, elle permet de quantifier le niveau de "surprise" d'un évènement. De ce fait, moins l'évènement est probable, plus il est "surprenant" et donc apporte plus d'information.

Définition. *Soit X un évènement de probabilité p(X), l'information de Shannon est définie par :*

$$I(X) := -log_b\left(p(X)\right) = log_b\left(\frac{1}{p(X)}\right)$$

Le choix de la base b correspond aux différents unité de la quantité. Dans notre cas, $b = 2$, et l'unité pour ce dernier est le bit. [4]

Maintenant qu'on a une quantité pour décrire le niveau de surprise, on veut à présent se renseigner sur la "valeur moyenne d'information obtenu d'un évènement", aussi nommée entropie de Shannon. [4]

Définition. *Soit X une variable aléatoire discrète et pour tout évènements x_i de probabilité p_i, l'entropie de Shannon est définie par :*

$$H(X) := -\sum_i p_i \ log_2(p_i)$$

Dans notre cas, on dispose d'une variable aléatoire pour chaque mot, et le but est de calculer l'entropie de chaque mot pour ensuite trouver une stratégie gagnante.

1.2.3 Le fonctionnement du solveur et des idées de stratégies

Cette section va présenter différentes stratégies possibles pour un solveur de Wordle avec les avantages et inconvénients qui y sont associées.

L'alphabet

Un moyen simple d'obtenir de l'information consiste à utiliser toutes les lettres de l'alphabet dans nos tentatives, en utilisant des mots composés en priorité des lettres les plus "utilisées", on peut obtenir rapidement des informations et le solveur peut très vite réduire le champ des possibles et trouver la réponse en un temps correct. (Avec un mot de 5 lettres, on peut estimer qu'en 3-4 essais on a obtenu une quantité d'information suffisante en utilisant des "bons" mots.) Cette stratégie permet certainement de trouver le bon mot à la fin de la partie, mais elle est au final très peu efficace dans de nombreux cas où 3 essais seraient suffisants : par exemple, si le mot recherché est "CRADE", jouer "CRANE" puis "DOIGT" suffirait à gagner, mais le solveur perdrait du temps à faire 2 propositions dans le vide.

L'arbre des possibilités

Connaissant l'ensemble des mots possibles, on peut faire tourner un algorithme qui va étudier chaque partie possible, on peut ensuite faire des statistiques pour décider de quel mot est le meilleur à proposer selon le pattern renvoyé par le jeu. [2] Cette solution nous assure de gagner, mais pourvu que le dictionnaire soit très grand, le nombre de parties risque d'être trop grand pour que cette solution soit viable.

Avec la théorie de l'information

La méthode qui semble la plus efficace reste celle que nous évoquions précédemment dans la section 1.2.1 (*Idée générale*). Avant chaque proposition (sauf la 1ère qui au final restera toujours la même pour un nombre de lettres donné), on détermine l'ensemble des mots possible en fonction des indices, et on calcule l'entropie de chacun de ces mots. Le solveur renvoie alors le mot avec l'entropie maximale. Cette solution en théorie nous assure de trouver le mot, mais dans certains cas - si il existe "trop" de possibilités dans le cas où on a 4 lettres déjà bien placées par exemple - le solveur risque de ne pas gagner.

Chapitre 2

Gestion de projet

2.1 Équipe de projet

L'équipe se compose de quatre étudiants en première année :
- BENE Tom
- LI Alexandre
- MOUSSU Camille
- RICARD Guillaume

Camille a été désigné chef de projet, elle a eu la responsabilité d'animer les réunions et de suivre l'avancement du projet.

L'équipe s'est réunie régulièrement sur Discord le week-end mais aussi a l'école en semaine pour le suivi du projet, la définition des objectifs et la répartition des tâches.
Des séances de travail en groupe ont été organisées à chaque fin de partie pour assurer la finalité du travail et la coordination du groupe. Elles ont eu lieu à l'école et chez certains membres du groupe.

2.2 Analyse du projet

2.2.1 Définition des objectifs

Après quelques parties de Wordle, nous avons choisi de faire un application qui permette à ses utilisateurs de choisir plusieurs dictionnaires dont le Claudle avec des mots des chansons de Claude François.
Notre Wordle suit la même logique que le jeu d'origine. Le code couleur et le concept seront les mêmes. Cependant, les joueurs pourront jouer dès qu'ils accèdent au site. S' ils souhaitent garder des statistiques personnelles de jeu, ils pourront créer un compte. Sinon, leurs statistiques seront stockées de façon commune pour donner une vision globale des performances des joueurs.

Pour le solveur, nous avons d'abord choisi de nous focaliser sur une stratégie en 6 essais qui permet de parcourir tout l'alphabet. Suite à cette stratégie peu efficace, nous nous sommes tourné vers la théorie de l'information, plus efficace.

2.2.2 Analyse des risques : Matrice SWOT

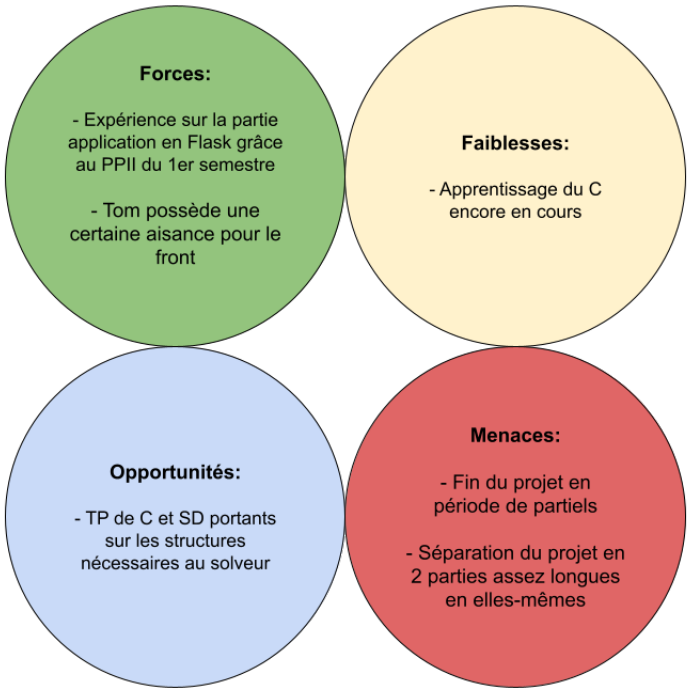


FIGURE 2.1 – Matrice SWOT

Interprétation

Nous avons évalué les risques liés au développement du projet ainsi que les potentielles menaces mais également les forces et les opportunités.
Cette matrice permet de décider de l'organisation du travail. Comtpe tenu des connaissance de Tom pour le frontend, il s'est donc chargé du bon déroulement du développement du front.
Puisque personne dans le groupe n'avait jamais fait du C auparavant, les travaux pratiques de C/SD étaient une grande opportunité de développer nos compétences.
Sur la fin du projet, notre groupe devra faire attention à finaliser les livrables avant le début des partiels pour éviter de devoir travailler dans l'urgence.

2.3 Organisation du projet

Le projet prend place du mois de mars 2022 jusqu'à la fin du mois de mai 2022 avec un premier rendu pour le site le 31 mars 2022. Nous avons donc séparé le travail en 2 partie, site et solveur.
Premièrement, pour le site, nous nous sommes réparti les fonctions nécessaires à son fonctionnement et les composants du font-end équitablement.
Ensuite, pour le solveur nous avons essayé au mieux de répartir équitablement les fonctions utilisées et la conception de la structure de données.

2.3.1 Répartition des taches : RACI & WBS

Nous avons mis au point une matrice RACI et un WBS (en figure 5.1) pour répartir les tâches équitablement.

	Tom	Camille	Alexandre	Guillaume
Gestion de projet	I	RA	I	I
Affichage page d'accueil	R	I	I	I
Algorithmes de traitment des mots en Python	I	RA	R	R
Page de stats	C	RA	R	R
Page de Register	A	I	R	I
Page de profil	R	R	I	C
Interactivité	RA	R	I	I
SD	C	R	R	R
Algorithme en 6 essais	I	R	C	C
Base de Données	I	I	R	RA
Théorie de l'information	I	R	RA	RC
Input/Output	RA	I	I	I
Fonctions de traitement des mots en C	R	C	R	R

2.3.2 Diagramme de Gantt

A partir de la répartition des tâches du WBS, nous avons créé un diagramme de Gantt prévisionnel, représenté ci dessous, ainsi qu'en annexe en figure 5.2.

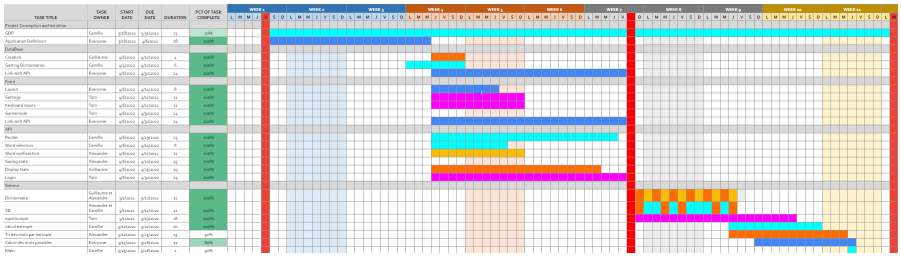


FIGURE 2.2 – Diagramme de Gantt prévisionnel

L'avancement a été respecté sur la partie jeu, mais la partie solveur a nécessité plus de temps que prévu, ce que montre le Gantt final (Figure 5.3 en annexe)

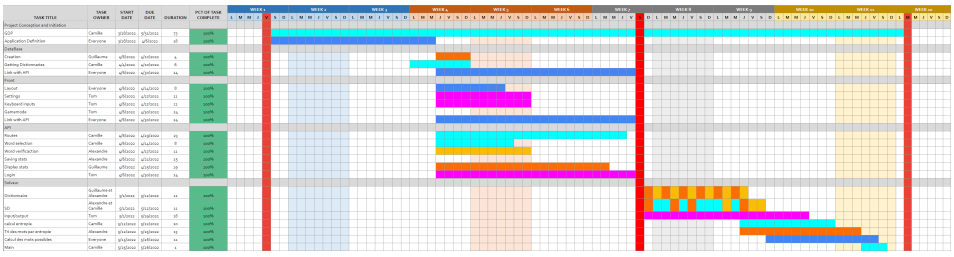


FIGURE 2.3 – Diagramme de Gantt final

2.4 Outils de travail

IDE :

L'ensemble de l'équipe a travaillé sur VisualStudio Code.

Partage du travail :

L'équipe a majoritairement utilisé le dépôt git fourni par l'école pour échanger le travail mais avait aussi un drive dédié pour certains fichiers de la gestion de projet et pour partager des documents lors de la mise en place du projet.

Rédaction du rapport :

Le rapport a été rédigé sur Leaf pour permettre à tous les membres de compléter leurs éléments simultanément.

Chapitre 3

Conception et réalisation

3.1 La base de données

La base de données a été faite en utilisant SQLAlchemy. C'est un ORM permettant de simplifier la gestion de base de données et de réduire le temps de développement des fonctionnalités liées. Nous avons créé un fichier python déclarant le modèle de la base de données, puis créé un fichier en .db contenant des données de test, nous permettant de réinitialiser rapidement les données.

3.1.1 Structure de la base

La base de donnée permet de stocker les utilisateurs et l'historique de leurs parties.

- La table *users* contient un ID unique, clé primaire de la table, pour identifier l'utilisateur. On y retrouve également son pseudo et son mot de passe (sous forme hashée par sha256).

- La table *partie* continent quand à elle un identifiant unique (sa clé primaire) ; puisque les mots sont tirés au hasard, il est possible qu'un utilisateur ait plusieurs parties avec le même mot (surtout l'user 0 qui sert pour les joueurs non connectés), l'ID permet de les différencier pour s'assurer que les statistiques obtenues avec la base de données soient justes.
La table contient également le mot recherché et l'ID du joueur en train de jouer. Dans l'optique de pouvoir afficher l'historique d'un joueur, cette table stocke également le nombre d'essais qu'il a fallu à un joueur (ou -1 s'il ne trouve pas) ainsi que les mots qu'il a proposé pendant sa partie, séparés par le caractère "/"

Pour les utilisateurs non connectés, ils n'ont pas accès à leur statistiques de jeu mais elles sont toutes regroupées dans la table d'ID égal à 0. Ainsi, leurs statistiques sont quand même comptées dans les statistiques globales.

3.1.2 Schéma de la base de données

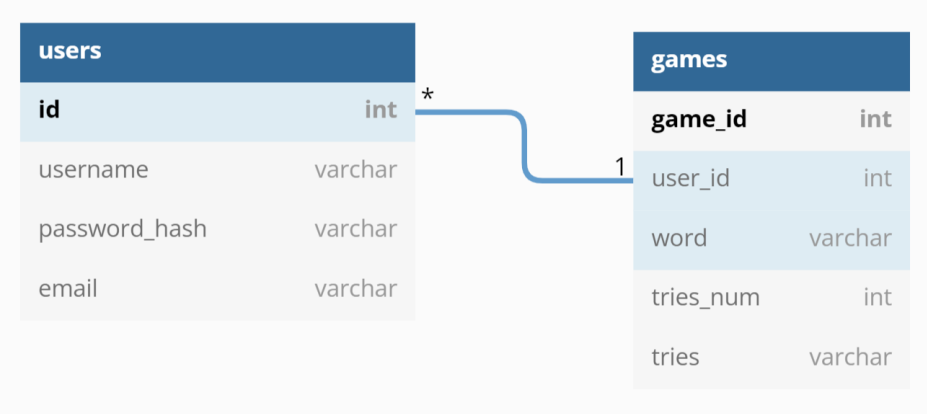


FIGURE 3.1 – Schéma relationnel

3.1.3 Le lien entre l'API et la base de données

Dans notre fichier python de l'application, on importe le module Flask-SQLAlchemy. La connexion à la base de données ainsi que les requêtes sont directement faites avec le module.

3.2 Le Claudle

Nous avons choisi de nommer notre Wordle Claudle en référence à Claude François. Nous avons donc en référence à cet artiste fait une version avec un dictionnaire de mots présents dans les titres de ses chansons.

3.2.1 Front End

Nous avons choisi de faire notre site avec un front end en utilisant ReactJS. Cela nous a permis de rendre plus facilement le site réactif, sans avoir à rafraîchir la page pour actualiser les données affichées.

Le frontend communique avec le backend par le biais de la librairie Axios, qui permet de faire des requêtes HTTP à l'API de façon asynchrone. Les données récupérées sont ensuite stockées côté client dans le *state* interne de l'application. Ce *state* est géré à l'échelle de l'application grâce à l'API *Context* de React, notamment les paramètres de jeu tels que la longueur du mot, le nombre d'essais ou la disposition du clavier.

Les pages

Il y a 4 pages sur le site :

La page principale qui donne accès directement au jeu, une page de statistiques globales qui permet à tout le monde de regarder comment les joueurs performent, une page pour créer un compte et une page pour regarder ses statistiques personnelles une fois connecté.

Les différents modes de jeu sont accessibles sur l'URL racine "/".

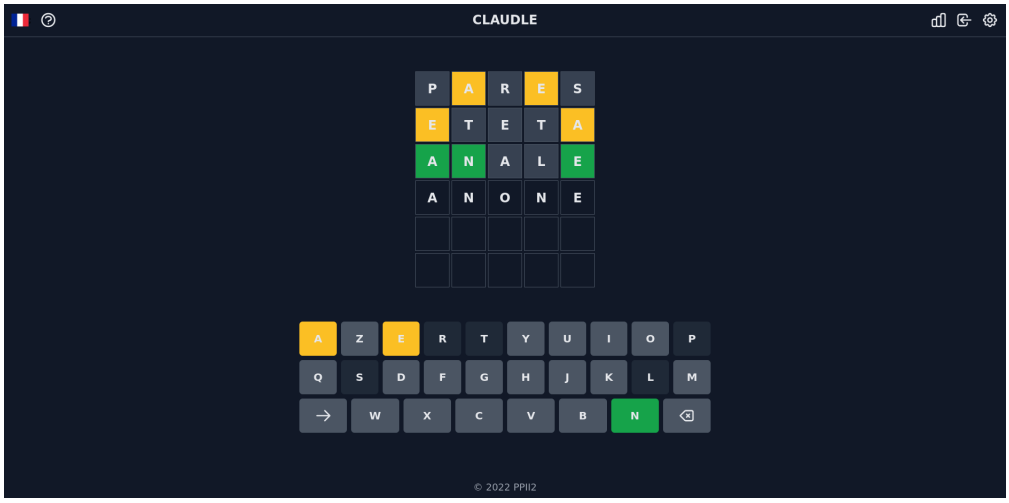


FIGURE 3.2 – Page d'accueil du site en cours de partie

La page de création de compte sera sur l'URL "/register"

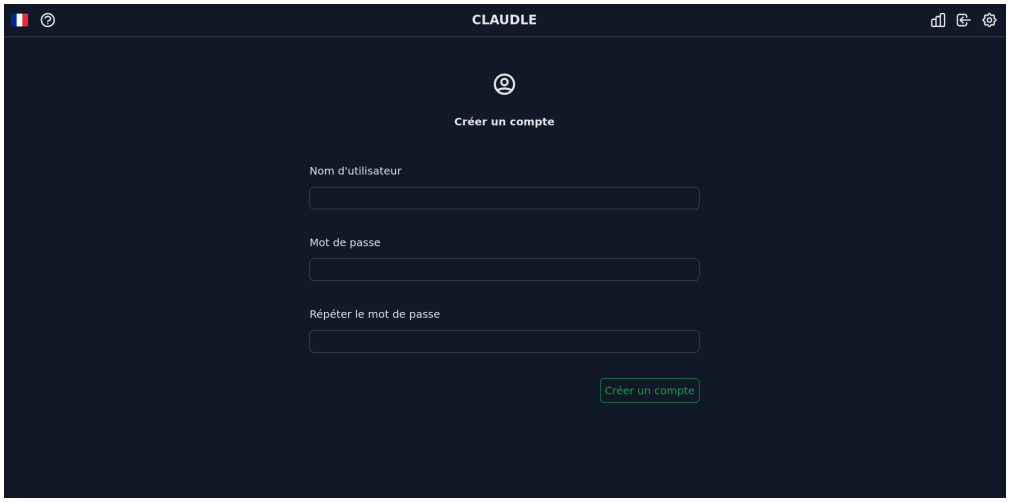


FIGURE 3.3 – Page de création de compte

La page de profil sur "/profile"

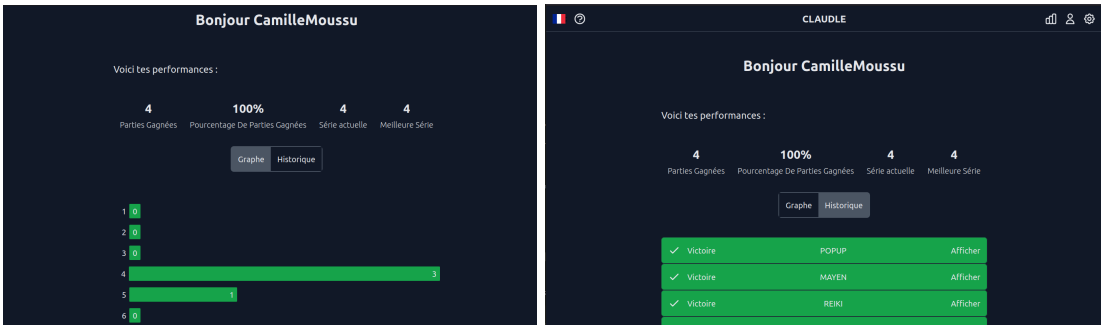


FIGURE 3.4 – Page de profil

La page de statistiques globales sur “/stats”.

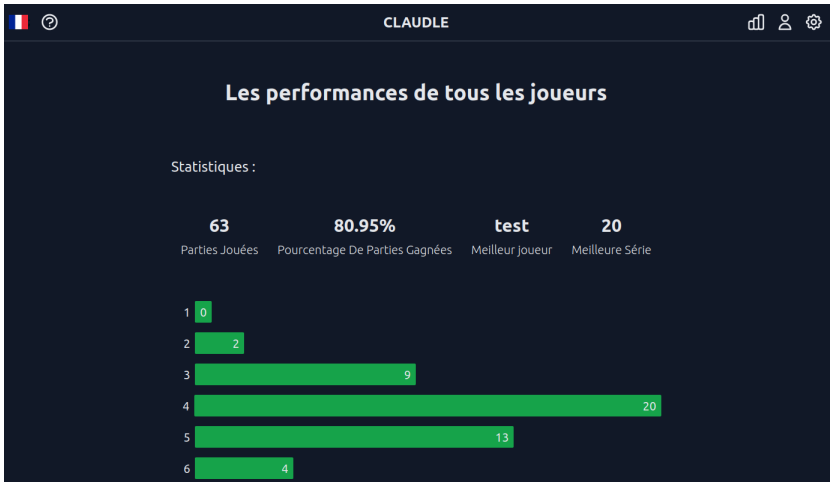


FIGURE 3.5 – Page de statistiques globales

Si on possède déjà un compte, le bouton login sur la barre de navigation présente sur chaque page ouvrira un pop-up qui permettra de se connecter tout en restant sur la même page.

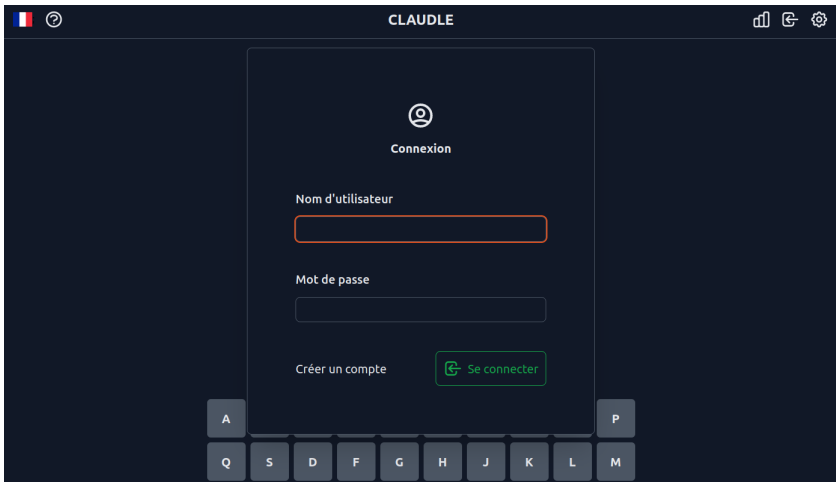


FIGURE 3.6 – Popup de login

3.2.2 Back End

Pour la mise en place du back-end, on a utilisé un serveur web avec Flask. Toutes les routes et les fonctions correspondantes sont stockées dans le fichier app.py.
Puisque notre site a un front-end en ReactJS, on accède uniquement au port 3000 pour jouer. Cependant, pour que le jeu puisse fonctionner, il faut faire appel à des fonctions en python. Nous avons donc un site en flask qui est lancé sur le port 5000. Ce site renvoie des données au format *JSON*, il permet au front d'obtenir les données nécessaires selon les routes appelées.

Les routes

Puisque le front et l'API étaient séparés, nous avons fait en sorte que l'API ait une route pour chaque fonctions nécessaires a fonctionnement du Claudle.

/api/validate
Renvoie les résultats de 2 fonctions.
Une qui prend en entrée un mot et vérifie qu'il est bien dans le dictionnaire.
Une autre qui prend en entrée deux mots, le mot à trouver et le mot à tester qui renvoie une suite de 0, 1 ou 2 selon la correspondance des deux mots.

/api/getword
Renvoie le résultat d'une fonction qui a en entrée la taille du mot à renvoyer et qui choisi un mot aléatoirement dans le dictionnaire. C'est le mot qu'il faudra trouver pendant la partie.

/api/saveStats
Permet d'enregistrer les statistiques de partie, elle est appelée a chaque fin de partie.

/api/endgame
Permet à l'API de savoir que la partie est finie et de lancer l'enregistrement des données.

/api/register
Prend en entrée les données nécessaires pour créer un nouvel utilisateur dans la base de données.

/api/login
Vérifie que la combinaison du pseudo et du mot de passe en entrée correspond à un utilisateur de la base de donnée.

/api/logout
Déconnecte un utilisateur en supprimant les cookies de connexion.

/api/profile
Renvoie le résultat d'une fonction prenant en entrée l'ID du joueur et renvoie son nom.

/api/stats
Renvoie le résultat d'une fonction prenant en entrée l'ID du joueur et renvoie le nombre de parties jouées, le pourcentage de victoire, le combo actuel et le meilleur combo depuis la création du compte.

/api/history
Renvoie le résultat d'une fonction prenant en entrée l'ID du joueur et renvoie la liste de toutes les parties jouées avec les essais.

/api/delUser
Supprime un compte utilisateur de la base de données.

3.3 Le solveur

3.3.1 Implémentation

Dans le cadre du projet PPII-2, il a été demandé de faire l'implémentation du solveur en C. Nous avons utilisé C99.

3.3.2 Structure de données

Étant donné la taille des dictionnaires, nous avons décidé d'utiliser une table de hash pour stocker les mots. Pour cela, il nous a fallu une fonction de hash, le SipHash a été choisi pour ses performances et ses collisions. Avant d'implémenter la table de hash, il faut d'abord avoir une structure de liste, ici, nous avons utilisé une liste simplement chaînée. On donne ici les spécifications algébriques des principales structures utilisées.

Element

$$\begin{aligned} element\_create &: Mot, Entropy, Score \to Element \\ list\_get\_key &: Element \to Mot \\ list\_get\_entropy &: Element \to Entropy \\ list\_get\_score &: Element \to Score \end{aligned}$$

Axiomes :

$$\begin{aligned} &list\_get\_key(element\_create) = \text{'}\backslash 0\text{'} \\ &list\_get\_entropy(element\_create) = 0.0 \\ &list\_get\_score(element\_create) = 0 \end{aligned}$$

Liste

$$\begin{aligned} list\_create &: void \to Liste \\ list\_copy &: Liste \to Liste \\ list\_is\_empty &: Liste \to bool \\ list\_append &: Liste, Mot, Entropy, Score \to Liste \\ list\_prepend &: Liste, Mot, Entropy, Score \to Liste \\ list\_find &: Liste, Mot \to Entropy \\ list\_contains &: Liste, Mot \to bool \\ list\_remove\_key &: Liste, Mot \to Liste \\ list\_remove\_first &: Liste \to Liste \\ list\_get\_size &: Liste \to Nombre \\ list\_max\_entropy &: Liste \to Element \end{aligned}$$

Axiomes :

$list\_is\_empty(list\_create()) = true$
$list\_is\_empty(list\_copy(one\_list)) = list\_is\_empty(one\_list)$
$list\_is\_empty(list\_append(one\_list, one\_char, one\_double, one\_int))$
$= \begin{cases} true & \text{si} \quad one\_char = \text{'}\backslash 0\text{'} \text{ et} \quad one\_double = 0.0 \quad \text{et} \quad one\_int = 0 \\ false & \text{sinon} \end{cases}$
$list\_is\_empty(list\_prepend(one\_list, one\_char, one\_double, one\_int))$
$= \begin{cases} true & \text{si} \quad one\_char = \text{'}\backslash 0\text{'} \text{ et} \quad one\_double = 0.0 \quad \text{et} \quad one\_int = 0 \\ false & \text{sinon} \end{cases}$
$list\_is\_empty(list\_remove\_key(one\_list, one\_char)) = \begin{cases} true & \text{si} \quad list\_get\_size(one\_list) = 1 \\ false & \text{sinon} \end{cases}$
$list\_is\_empty(list\_remove\_first(one\_list) = \begin{cases} true & \text{si} \quad list\_get\_size(one\_list) = 1 \\ false & \text{sinon} \end{cases}$
$list\_contains(list\_create(), one\_char) = \begin{cases} true & \text{si} \quad one\_char = \text{'}\backslash 0\text{'} \\ false & \text{sinon} \end{cases}$
$list\_get\_size(list\_create()) = 0$
$list\_get\_size(list\_copy(one\_list)) = list\_get\_size(one\_list)$
$list\_get\_size(list\_append(one\_list, one\_char, one\_double, one\_int)) = list\_get\_size(one\_list) + 1$
$list\_get\_size(list\_prepend(one\_list, one\_char, one\_double, one\_int)) = list\_get\_size(one\_list) + 1$
$list\_get\_size(list\_remove\_key(one\_list, one\_char) = list\_get\_size(one\_list) - 1$
$list\_get\_size(list\_remove\_first(one\_list) = list\_get\_size(one\_list) - 1$

Dictionnaire

$$table\_create : Nombre \to Table$$
$$table\_copy : Table \to Table$$
$$table\_is\_empty : Table \to bool$$
$$table\_indexof : Table, Mot \to Nombre$$
$$table\_add : Table, Mot \to bool$$
$$table\_contains : Table, Mot \to bool$$
$$table\_remove : Table, Mot \to void$$
$$table\_max\_entropy : Table \to Element$$

Axiomes :

$$table\_is\_empty(table\_create(n)) = true$$
$$table\_is\_empty(table\_copy(one\_table) = table\_is\_empty(one_table)$$
$$table\_contains(table\_create(n), key) = \begin{cases} true & \text{si} \quad key = \text{'}\backslash 0\text{'} \\ false & \text{sinon} \end{cases}$$
$$table\_add(one\_table, key) = \begin{cases} true & \text{si} \quad table\_contains(table, key) = false \\ false & \text{sinon} \end{cases}$$

Choix des mots

Le choix du premier mot lorsque l'on lance le jeu a été préalablement calculé. Ensuite, selon les indices renvoyés par le joueur, le solveur recalcule l'entropie de tout les mots possible, et renvoie celui avec la plus grande entropie.
Il y a donc :

- Pour 4 lettres : SEAU
- Pour 5 lettres : PARES
- Pour 6 lettres : CASTRE
- Pour 7 lettres : CASTREE
- Pour 8 lettres : ORIENTES
- Pour 9 lettres : ENTOUREES
- Pour 10 lettres : CLAIREMENT

Les mots possibles

Pour le calcul des mots possible, le solveur parcourt le dictionnaire du tour précédent et sélectionne dans un nouveau dictionnaire les mots trouvables avec les indices actuels. Nous avons fait le choix de créer un nouveau dictionnaire à chaque étape car plus rentable lorsque l'on doit élaguer beaucoup de mots.
Pour la sélection des mots, on met d'abord un marqueur aux indices des lettres qui sont les mêmes que celles du mot notées à 2. Ensuite, il vérifie que les lettres avec la note 1 sont dans le mots mais pas au même endroit que celles de bases en utilisant encore le marqueur. Finalement, il vérifie que les lettres notées à 0 sont bien absente du mot.
Le dictionnaire recréé, la boucle du jeu continue.

Interaction solveur-joueur

Puisque le solveur n'interagit pas avec le jeu Claudle, il est demandé au joueur de mettre le score des mots proposé par le solveur dans la ligne de commande, après avoir entré le score du mot proposé, le solveur recalcule le nouveau mot a proposer.

```
Importation des mots...
Dico importé
Mot Proposé : orientes
Entrez les indices donnés par le jeu : 10111102
Mot Proposé : testions
Entrez les indices donnés par le jeu : 22002222
Mot Proposé : tendions
Entrez les indices donnés par le jeu : -1
Jeu terminé, à bientôt :)
```

FIGURE 3.7 – Partie classique avec victoire

Chapitre 4

Tests et performances

4.1 Tests

Claudle

Les tests ont été fait indépendemment dans des fichiers séparés. Les tests vérifient la validité des mots, le score des mots proposés comparés au mot recherché et le calcul des pourcentage de victoire.

Solveur

Pour le solveur, les tests ont permis de traiter les cas limites des fonctions.
Par exemple le comportement de la comparaison des mots à l'indice associé au mot précédent lorsqu'il y a plusieurs lettres identiques avec une note différente.

4.2 Performance

Claudle

Toutes les fonctions du Claudle sont de complexité linéaire au maximum.
De plus, étant donné que la longueur des mots est très petites devant le nombre de mot, la vérification des mots sont négligeable.

Solveur

Dans l'hypothèse qu'on ait peu de conflit, toutes les fonctions de la structure de données sont, au pire en complexité linéaire.
Grâce à la fonction de hash que nous avons choisi, qui est le SiphHash, nous pouvons bien faire l'hypothèse précédente, car cette fonction est connu pour sa rapidité et sa distribution plus ou moins uniforme.

	4	5	6	7	8	9	10
Nombre de partie	48	191	100	39	29	21	19
Nombre de victoire	45	175	100	39	29	21	19
Pourcentage de victoire (%)	93.7	91.6	100	100	100	100	100

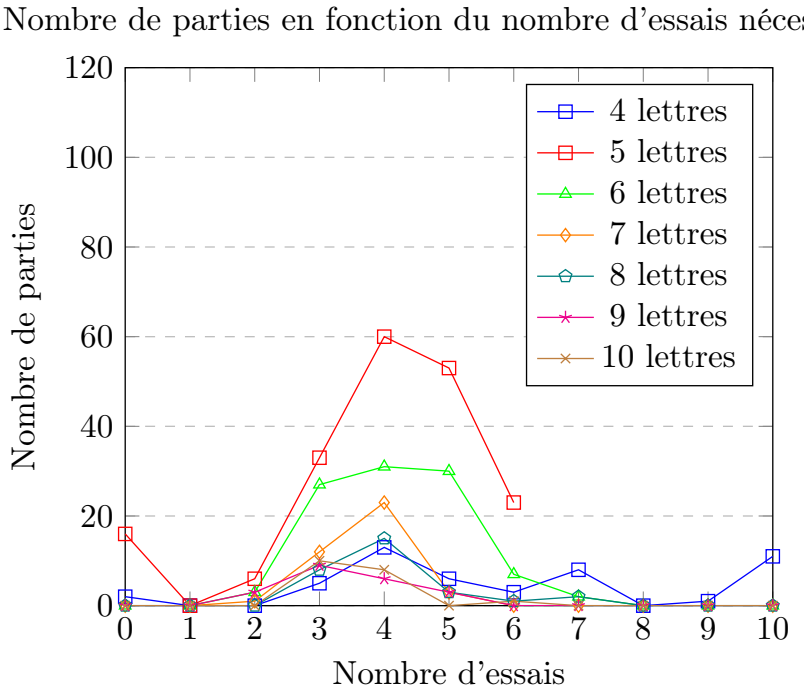
Puisque le solveur joue en "mode difficile", c'est-à-dire est qu'il est obligé d'utiliser les indices donnés pour le prochain mot proposé, il y a des cas où le solveur ne peut pas gagner s'il y a des mots qui sont similaire à une lettre près.

Piste d'amélioration des tests

Les tests que nous avons effectué sont loin d'être idéaux, un point où nous pouvons améliorer serait d'utiliser la bibliothèque *snow* pour faire les tests unitaires.

Comparaison entre le nombre de lettres et le nombre d'essais nécessaires

Avant d'analyser les résultats, notons qu'une partie ayant 0 pour nombre d'essais signifie que la partie a été perdue et que pour les parties à 5 lettres on jouait avec les vraies règles du WORDLE en se limitant à 6 essais.



On remarque qu'en moyenne, toutes lettres confondues, le solveur trouve en 4 essais. Ce qui reste satisfaisant quand on sait que sur le wordle officiel trouvent en moyenne en 3-4 essais [1]. Il est intéressant de noter que les parties en 4 lettres sont les seules qui peuvent être perdantes en 10 tentatives, alors qu'il s'agit du plus petit dictionnaire. Cela peut s'expliquer par la présence dans le dictionnaire de 4 lettres de mots possédant la même structure. (cf. figure 4.1)
On peut regretter de ne pas avoir une moyenne avec le solveur plus basse, mais cela peut s'expliquer par le fait que notre solveur fonctionne en mode "difficile" ; l'obligation de proposer des mots respectant exactement le pattern ne permet pas de proposer un mot qui permettrait de réduire encore plus la liste des mots possibles et donc de trouver en moins d'essais.

1. Source : `https://twitter.com/WordleStats`

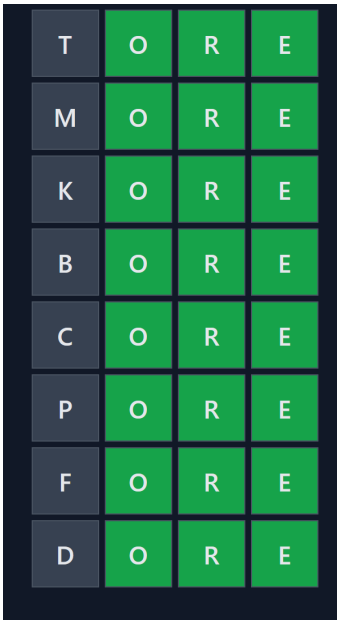


FIGURE 4.1 – Exemple d'une structure problématique à 4 lettres. *Le mot recherché était GORE*

Chapitre 5

Bilan du projet

Bilan global du projet

Travail attendu	**Travail réalisé**
- Instance de jeu Wordle - Paramètres du jeu (longeueur mot, nombre essai, ...) - Architecture Python/Web/BD - Algorithme de vérification des mots pour le jeu - Sauvegarder les parties des joueurs dans la base de données - Implémentation du solveur en C - Structure de données avancées - Interaction solveur-joueur - Réalisation de la documentation lié à la gestion du projet	-Instance de jeu Wordle - Paramètres du jeu - Architecture Python/Web/BD - Algorithme de vérification des mots pour le jeu - Sauvegarder les parties des joueurs dans la base de données - Performance globale de tout les joueurs - Historique des parties des joueurs inscrits - Implémentation du solveur en C - Table de hash et liste chaînée - Interaction solveur-joueur - Réalisation de la documentation lié à la gestion du projet

	Points positifs	Points négatifs	Expérience
Web	- Connaissances d'un des membres ont permis de répartir efficacement les taches sur la conception du front	- Manque d'expérience des autres membres a fait que le site n'a pas atteint son plein potentiel	- Apprentisage du Javascript et de ReactJS
Solveur	- Progression en équipe et bien ordonnée	- Beaucoup d'incompréhensions lors de la mise en place des structures de données	- Beaucoup d'expérience en C et de gestion des erreurs
Gestion de projet	- Prise de note des réunions efficace - Réunions régulières	- Focalisation sur des points peu pertinents pour le solveur - Manque de clarté de la conception du solveur	- S'assurer que tout ce qui va être fait est efficacement défini pour tous les membres du projet - Éviter la confusion en préparant un dossier de conception

Bilan du projet par membre

Tom BÉNÉ

Points positifs	- Bon ambiance au sein du groupe et motivation de l'ensemble du groupe - Bonne cohésion au sein du groupe
Difficultés rencontrées	- Gestion du state de l'application React - Prise en main du C pour le solveur
Expérience personnelle	- Amélioration de mes compétences en Typescript et avec le framework React - Apprentissage de l'utilisation d'API REST - Approfondissement de mes connaissances en C
Axes d'amélioration	- Meilleure gestion des deadlines vis-à-vis de ma vie personnelle

Alexandre LI

Points positifs	- Groupe motivé et spontané - Entraide en cas de difficulté
Difficultés rencontrées	- Début d'apprentissage lente du TypeScript
Expérience personnelle	- Apprentissage du TypeScript - Approfondissement des connaissances en C - Appréciation du C
Axes d'amélioration	- Répartition du travail personnel en tenant compte des contraintes extérieur (partiels, révisions, ...)

Camille MOUSSU

Points positifs	- Répartition des taches efficaces - Groupe très motivé
Difficultés rencontrées	- Des problèmes personnels d'un membre du groupe ont ralenti son travail et donc la progression globale - Complexité du langage C et apprentissage d'un nouvel outil pour le front
Expérience personnelle	- La partie Claudle m'a appris à faire du javascript. - Le solveur m'a permis de beaucoup m'entraîner au C et à faire des SD - J'ai globalement apprécié travailler sur ce projet
Axes d'amélioration	- Accélérer le temps de définition de début de projet - Mieux definir les deadlines

Guillaume RICARD

Points positifs	- Groupe motivé - Bonne répartition du travail entre les membres du groupe
Difficultés rencontrées	- Difficultés à travailler efficacement en C
Expérience personnelle	- Apprentisage d'un nouveau langage (TypeScript) - Entrainement au C et à la construction de structures de données
Axes d'amélioration	- Mieux définir les taches pour optimiser le temps de travail de tous - Mieux commenter/documenter mes codes pour faciliter le travail en équipe

Bilan des heures par personnes

Étapes	Tom	Alexandre	Guillaume	Camille
État de l'art	1h	2h	2h	3h
Gestion de projet	1h	1h	1h	8h
Claudle				
Base de donnée	2h	1h	4h	4h
Front	42h	20h	22h	19h
API	18h	15h	19h	20h
Solveur				
Structure de donnée	25h	30h	26h	25h
Théorie de l'information	24h	24h	28h	35h
Dictionnaire	1h	1h	2h	2h
Secrétariat				
Comptes-rendus de réunion	3h	3h	5h	2h
Rédaction du rapport	10h	11h	11h	10h
TOTAL	127h	108h	121h	128h

Conclusion

En conclusion, nous avions prévu de faire un WORDLE. Pendant la réalisation du projet, l'ensemble de l'équipe a travaillé en coopération pour assurer une mise en place efficace des fonctionnalités du site et du solveur. Notre but a été atteint, le Claudle fonctionne comme un wordle et propose même plus que le site original et le solveur est résoud le Claudle correctement.
Il aurait pu être plus efficace mais par manque de temps et de connaissances, nous n'avons pas pu l'optimiser. D'un point de vue plus personnel, cette expérience nous a permis de renforcer et acquérir nombre de compétences techniques en web et en C.
Nous en garderons globalement une bonne expérience.

Annexe

Comptes-rendu des réunions

16 mars 2022

16h35 – Salle 1.15

Présent	Absent
Tom BENE Alexandre LI Camille MOUSSU Guillaume RICARD	Personne

Ordre du jour

1. Première réunion du groupe
2. Élection du chef de projet
3. Définition du projet
4. Début de la répartition des tâches pour la définition du projet

Élection du Chef de projet

Camille est candidate, un vote est effectué et Camille est élue avec 3 votes pour et une abstention

Définition du projet

Wordle

Le groupe décide d'adopter différents moyens pour la mise en place du site :

1. JavaScript pour le frontend (plus précisément React) et Tailwind pour la mise en page
2. La partie API en Flask avec SQLalchemy pour les appels à la base de données
3. La base de donnée en SQlite

Le groupe discute aussi des fonctionnalités à faire :

1. Jeu paramétrable par le joueur (longueur des mots, nombre max d'essais, choix du dictionnaire)
2. Sauvegarder dans une base de données les parties jouées (pour chaque joueurs et en général)
3. Mode difficile (on utilise forcément les indices)
4. Plusieurs dictionnaires (Anglais, Français, Langues régionales)

Solveur

Le groupe est assez hésitant sur cette partie puisqu'il y n'a pas encore eu de cours sur le C et qu'aucun d'entre nous n'a d'expérience.
Le groupe envisage quand même une petite fonctionnalité annexe si le solveur arrive a être mis en place sans trop de difficulté : une version automatique du solveur.

Répartition des taches :

— Tom : définition du front end
— Alexandre : pré-définition du solveur
— Camille : API et BD
— Guillaume : pré-définition du solveur

Réunion suivante *(Fin de définition)* : mardi 22/03/2022 à 16h

Fin de réunion à 17h10

22 mars 2022

16h15 – salle 1.15

Présent	Absent
Tom BENE Alexandre LI Camille MOUSSU Guillaume RICARD	Personne

Ordre du jour

1. Progression dans la définition
2. To-do list pour la prochaine réunion

Progression

Partie wordle
Un schéma du site (non fonctionnel) a été effectué en React par Tom. Camille a de son côté listé les différentes fonctions nécéssaires pour l'API. Les dictionnaires sont mieux stockés sur des API externes (Type datamuse) plutôt que dans une base de donnée que l'on aurait créée. Dans le cas d'un wordle utilisant un dictionnaire particulier, il sera simplement donné dans un fichier texte.

Partie solveur
Guillaume et Alexandre ont réalisé un "état de l'art" sur le solveur où ils donnent l'idée générale de son fonctionnement ainsi que différentes stratégies. Un "complément" sur les bases de la théorie de l'information a également été réalisé afin que l'ensemble du groupe soit au point sur ces notions et pour fixer les conventions.

To-do list

— Tom : Finaliser le schéma du site et aider Camille avec l'API
— Camille : Finir le listing pour l'API
— Alexandre : Réfléchir sur les structures de données utilisées par le solveur
— Guillaume : Réfléchir sur les structures de données utilisées par le solveur
— Tous : Remplir le document de conception

Réunion suivante *(Relecture du document de conception)* : mardi 29/03/2022 à 16h

Fin de réunion à 17h00

29 mars 2022

16h25 – Salle 1.15

Présent	Absent
Tom BENE Alexandre LI Camille MOUSSU Guillaume RICARD	Personne

Ordre du jour

1. Proposition de structures de données
2. Retour sur la conception de l'API
3. Validation de la conception du projet par M. Festor

Proposition de structures de données

Guillaume et Alexandre ont réfléchi à quelle structure de données implémenter pour stocker le dictionnaires et les mots candidats. La première idée est celle des tables de hachage. Une autre idée est d'utiliser des arbres. Cependant, cette implémentation semble plus complexe et ne semble pas présenter d'avantage concret pour notre cas de figure, c'est donc les structure de table de hachage qui est retenue.

Retour sur la conception de l'API

Camille et Tom énumèrent l'esnsemble des routes de l'API qu'il sera nécessaire d'avoir. Ces routes sont les routes "/", "/login", "/register", "/profile" et "/stats".
Il est décidé d'aborder la conception de l'API de manière modulaire afin de mieux pouvoir structurer notre code.

Validation de la conception du projet

Le groupe a prévu d'envoyer le document de conception complété à M. Festor. Si possible, nous prévoyons de nous entretenir avec lui jeudi avant ou après le CM.

Réunion suivante : mardi 20/04/2022 à 14h

Fin de réunion à 17h00

20 avril 2022

14h10 – Discord

Présent	Absent
Tom BENE Alexandre LI Camille MOUSSU Guillaume RICARD	Personne

Ordre du jour

1. Avancement sur l'application

Avancement sur l'application

— *Connexion des utilisateurs*
Tom a commencé à implémenter le système de connexion et le terminera prochainement.
— *Utilisation d'Axios*
Pour réaliser les différentes requetes HTTP, il est décidé par le groupe d'utiliser *Axios*.
— *API*
L'API est quasiment finie, il reste quelques fonctions à finaliser ou à réécrire pour utiliser correctement *SQLAchemy* par exemple.
— *Base de données*
Un des problèmes soulevé par M.Festor lors de la validation du projet est la gestion de l'historique des parties. Une proposition simple venant de Camille est d'avoir un fichier texte où tous les essais sont récuperés. Néanmoins le caractère pratique de cette solution est questionné. Une 2ème option est proposé par Tom d'avoir une colonne dans la base de donneés sur les parties qui soit une longue chaine de caractères contenant toutes les propositions faites lors d'une parties, séparées par un séparateur. Le groupe remarque alors que cela pose problème pour le respect des différentes formes normales. Cette solution est retenue pour le moment, faute de mieux.

Répartition des taches :

— Tom : Finir le système de connexion
— Alexandre : Finir les fonctions dans l'API
— Camille : Finir les fonctions dans l'API
— Guillaume : Réécrire les fonctions en *SQLAchemy*
— Tous : Apprendre à utiliser *Axios* et commencer à réaliser les requetes HTTP

Réunion suivante : lundi 04/05/2022 à 9h

Fin de réunion à 14h45

05 Mai 2022

9h00 – Salle 1.19

Présent	Absent
Tom BENE Alexandre LI Camille MOUSSU Guillaume RICARD	Personne

Ordre du jour

1. Retour sur la soutenance concernant l'application
2. Le solveur

Retour sur la soutenance concernant l'application

Le groupe est satisfait de la soutenance, et est prêt à se lancer dans la grande aventure qu'est le solveur.

Solveur

— *Implementation de la structure de donnée pour le dictionnaire*
Une première solution proposée pour stocker nos dictionnaires en C est de réaliser une table de hash. Cela a été vu en TP ce qui facilitera son implémentation pour le projet.
— *Affichage et récupération des entrées utilisateurs*
Cette partie relativement simple et rapide à concevoir sera réalisée par Tom en suivant les consignes du sujet.
— *Stratégie*
Pour "s'échauffer" le groupe va réaliser une première version de solveur où l'on propose à chaque fois les 5 mêmes mots au jeu, et selon les résultats, le 6ème mot doit être le bon. Dès que des résultats satisfaisants arriveront sur cette solution. Le groupe travaillera sur un solveur faisant appel à la théorie de l'information.

Répartition des taches :

— Guillaume : Récupération du mot dans le dictionnaire
— Alexandre : Définition de la structure de données pour le dictionnaire
— Camille : Stratégie en 6 mots
— Tous : Réfléchir à la stratégie "théorie de l'information" et commencer à implémenter les fonctions de base (entropie, etc...)

Réunion suivante : vendredi 13/05/2022 à 14h

Fin de réunion à 9h30

13 mai 2022

14h10 – Discord

Présent	Absent
Tom BENE Alexandre LI Camille MOUSSU Guillaume RICARD	Personne

Ordre du jour

1. Gestion de projet : Rapport
2. Avancement sur le solveur
3. Proposition de stratégies

Gestion de projet : Rapport

— *Le rapport*
Tout le monde rédigent le rapport lorsque le temps le permet. Les comptes-rendus des réunions sont à mettre au propre.

Avancement sur la structure de données

— *Input / output*
Tom a rencontré des difficultés pour les saisies de l'utilisateur dans le terminal.
— *Structure de données*
Pour la structure de données, nous avons décidé d'implémenter une table de hash. Ayant déjà une implémentation des listes suite aux travaux pratiques de structures de données, il faudrait juste faire l'implémentation de la table.
— *Stratégie autre que la "méthode classique"*
La stratégie en 6 essais, avance sans trop de difficulté. Pour finir l'implémentation, il faut la structure de données
— *Structure de données*
L'implémentation de la structure de données est quasiment bouclée, mais Alexandre rencontre quelques difficultés.

Répartition des taches :

— Tom : Finir la partie saisie de commande des joueurs
— Alexandre & Tous : Boucler la structure de données
— Camille & Guillaume : Fonction pour le calcul d'entropie
— Tous : Rédaction du rapport

Réunion suivante : Jeudi 19/05/2022 à 14h (reporté à Dimanche 22/05/2022 14h)

Raison : Problème organisationnel

Fin de réunion à 14h40

22 Mai 2022

14h00 – Discord

Présent	Absent
Tom BENE Alexandre LI Camille MOUSSU Guillaume RICARD	Personne

Ordre du jour

1. Solveur et théorie de l'information
2. Rapport

Solveur et théorie de l'information

Le groupe rencontre des difficultés sur certaines parties du solveur. Globalement, le système d'interaction fonctionne - il ne manque que la lecture du fichier texte avec la longueur du mot à trouver -. Le calcul sur l'entropie a été complètement implémenter. Les structures de données essentielles au solveur ont été implémentées en totalité.
Il ne reste qu'à réaliser la partie récupération des mots et de choix du mot à jouer.

Rapport

Le rapport est bien avancé en particulier sur la partie introduction et état de l'art du solveur. Il faudrait mettre plus d'image pour illustrer les différentes pages (par exemple, /login, ...).

Répartition des taches :

— Tous : Rédaction du rapport
— Tous : Finir le solveur
Fin de réunion à 14h30

WBS

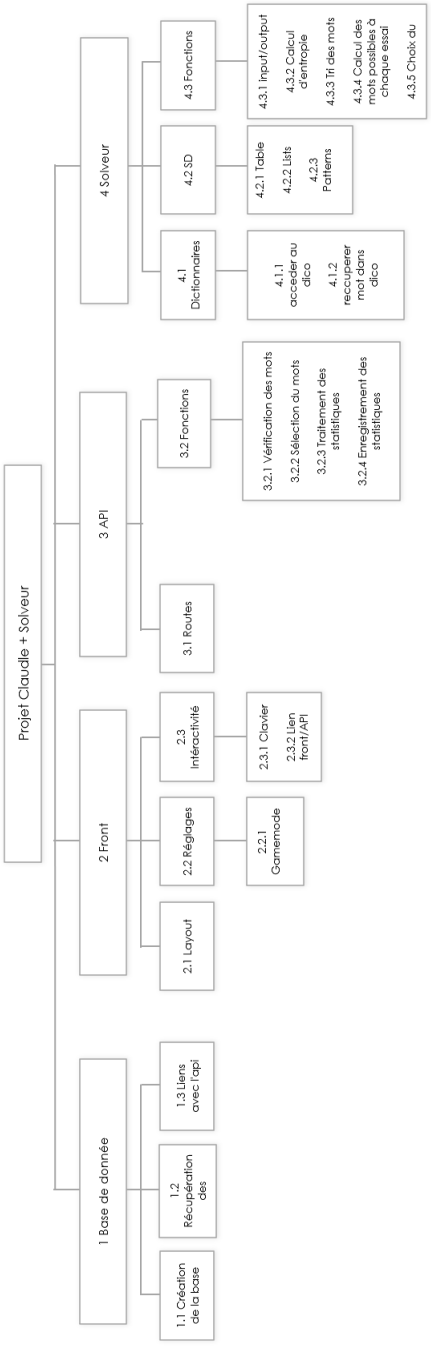


FIGURE 5.1 – WBS

Diagramme de Gantt

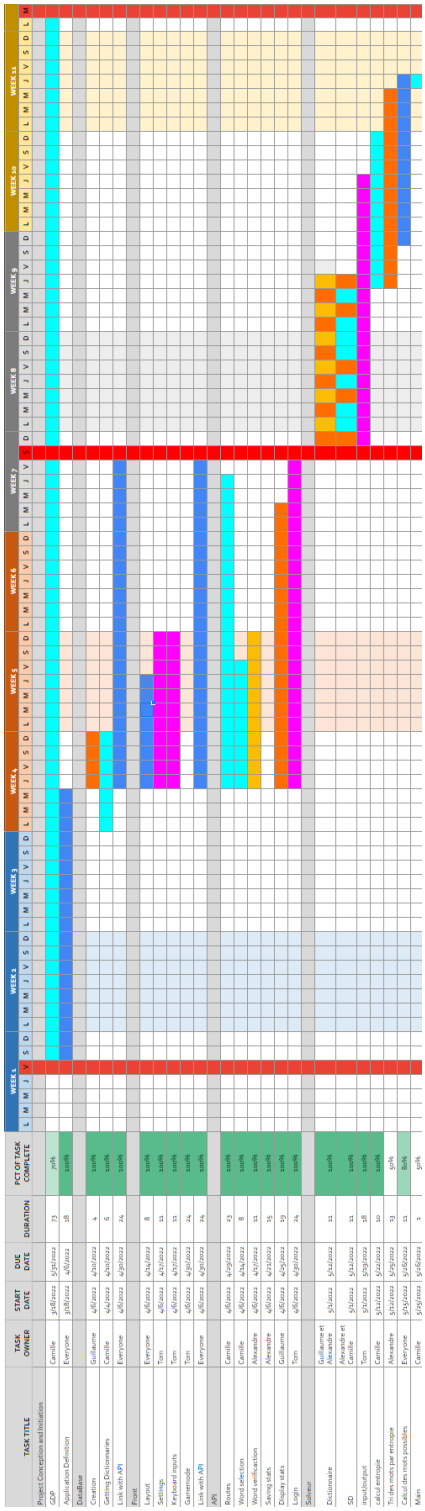


FIGURE 5.2 – Diagramme de Gantt prévisionnel

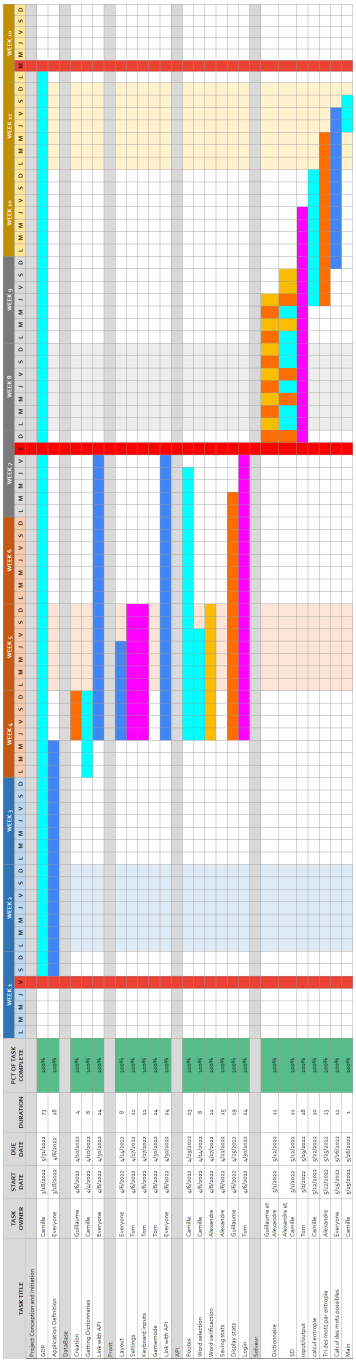


FIGURE 5.3 – Diagramme de Gantt final

Bibliothèques Python utilisées

— Flask
— SQLAlchemy
— Hashlib : sha256 (pour l'encodage du mot de passe)
— Datetime
— Pickle
— Timeit
— Pytest

Bibliographie

[1] 3 Blue 1 Brown - Grant Sanderson. Solving wordle using information theory. https://youtu.be/v68zYyaEmEA, 2022.

[2] Jonathan Olson. Optimal wordle solutions. https://jonathanolson.net/experiments/optimal-wordle-solutions, 2022.

[3] Andrew Steele. The best wordle strategy – according to science. https://www.youtube.com/watch?v=YEoCBnQwdzM, 2022.

[4] Wikipedia contributors. Information theory — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Information_theory&oldid=1071824496, 2022. [Online ; accessed 14-February-2022].