

Rapport du Projet P2I2 Flopple

Norman ASSING, Clément COUCHEVELLOU
Anna BIAUSQUE, Caroline WANG

8 juin 2022

Table des matières

1	Introduction au sujet	3
1.1	Contexte du projet	3
1.2	Précisions légales	3
2	Etat de l'art	4
2.1	Applications type WORDLE, Motus...	4
2.2	Compréhension d'algorithmes	6
3	Gestion de projet	7
3.1	Equipe de projet	8
3.2	Analyse du projet	8
3.3	Organisation	11
3.4	Outils de travail	13
4	Conception de l'Application web	14
4.1	Cahier des charges	15
4.2	Conception de l'Application web	15
5	Conception et implémentation du Solveur en C	23
5.1	Cahier des charges	24
5.2	Plusieurs stratégies pour le Solveur	24
6	Tests et performances du solveur	38
6.1	Tests unitaires	39
6.2	Complexité	40
6.3	Efficacité du solveur en nombre de tentatives	41
7	Bilan de projet	42
7.1	Bilan sur l'avancement du projet	43
7.2	Bilan global du projet	44
7.3	Bilan du projet par membre	44

8 Annexes	47
8.1 Comptes-rendus	48
8.2 Charte de projet	73

Chapitre 1

Introduction au sujet

Contents

1.1	Contexte du projet	3
1.2	Précisions légales	3

1.1 Contexte du projet

Ce projet a été réalisé dans le cadre du module P2I2 (Projet Pluridisciplinaire d’Informatique Intégrative) de la première année du cycle ingénieur sous statut étudiant de TÉLÉCOM Nancy.

1.2 Précisions légales

Ce projet n'est pas destiné à un usage commercial, ainsi, les images présentes, notamment les images de test, ne sont pas destinées à la publication et ne sont pas toutes libres de droit.

Cependant, le caractère strictement scolaire de ce projet nous autorise à les inclure en accord avec :

- Code civil : articles 7 à 15, article 9 : respect de la vie privée
- Code pénal : articles 226-1 à 226-7 : atteinte à la vie privée
- Code de procédure civil : articles 484 à 492-1 : procédure de réfééré
- Loi n°78-17 du 6 janvier 1978 : Informatique et libertés, Article 38

Chapitre 2

Etat de l'art

Contents

2.1 Applications type WORDLE, Motus...	4
2.2 Compréhension d'algorithmes	6
2.2.1 Python	6
2.2.2 Web	6

2.1 Applications type WORDLE, Motus...

Créé en 2021 par Josh Wardle, Wordle est un jeu de lettres en ligne gratuit. L'un des livrables demandés est une application similaire à Wordle, comportant globalement les mêmes fonctionnalités, tout en y ajoutant notre touche personnelle. Avant de passer à l'implémentation de notre “copycat” du jeu, nous avons trouvé nécessaire d'effectuer une étude préalable de Wordle, de sa situation sur le marché et de ses fonctionnalités. Nous avons décomposé notre recherche en plusieurs points :

1. Quel est le principe de Wordle ?
2. Qui va être intéressé par le produit ? Quelle est la cible ?
3. Comment le produit se situe-t-il par rapport aux produits existants sur le marché ?
4. Quelles sont les fonctionnalités critiques pour répondre aux besoins de façon à avoir un produit réussi ?
5. Quelles fonctionnalités “bonus” pourrions-nous ajouter ?

Le but du jeu est de deviner un mot spécifique de cinq lettres en un maximum de six tentatives, en tapant des lettres sur un écran de six lignes de cinq cases chacune. Après chaque proposition, les lettres apparaissent en couleurs : le fond gris représente les lettres qui ne se trouvent pas dans le mot recherché, le fond jaune représente les lettres qui se trouvent ailleurs dans le mot, et le fond vert représente les lettres qui se trouvent à la bonne place dans le mot à trouver. Un seul mot est proposé par jour, ce qui permet de “challenger” ses amis étant donné que le mot est le même pour tous.

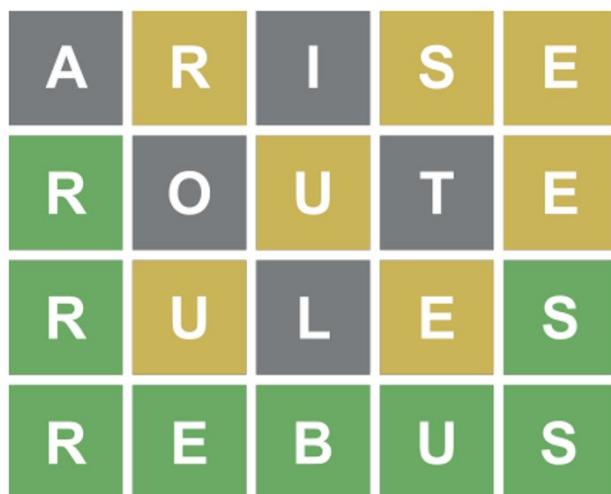


FIGURE 2.1 – Exemple de partie de Wordle

Le jeu s'adresse globalement à tout le monde de par sa gratuité, sa facilité d'accès, sa simplicité et l'absence de publicités. Il ne nécessite pas d'installation d'une application et propose également de partager ses résultats sur les réseaux sociaux, ce qui a permis d'augmenter sa popularité auprès de tous les publics.

Plusieurs jeux se basent sur le même principe que Wordle. Le plus populaire en France est certainement l'émission télévisée Motus (recherche d'un mot de 10 lettres en 6 essais), inspirée elle-même de l'émission américaine Lingo (recherche d'un mot de 5 lettres en 5 essais). Les versions françaises de Wordle en tant que jeu sur PC les plus connues sont Le Mot, qui comporte également un mode archive permettant de jouer les mots des jours d'avant, avec l'accès à des statistiques, ou encore Tusmo.

Afin d'obtenir un jeu similaire à Wordle, les fonctionnalités critiques imposées par le sujet et que nous avons nous-même déterminées sont les suivantes :

- un quadrillage rectangulaire de taille $n \times m$ avec n la longueur du mot recherché et m le nombre d'essais autorisés
- un clavier permettant au joueur d'entrer ses propositions de mots
- la vérification de l'existence du mot entré par l'utilisateur et le renvoi d'un message d'erreur si ce n'est pas le cas
- les lettres du mot proposé par le joueur doivent prendre une couleur différente selon qu'elles ne fassent pas partie du mot final ou qu'elles soient ou pas à la "bonne place" dans l'essai
- la possibilité pour le joueur de paramétriser sa session de jeu (choix de la longueur du mot et du nombre d'essais)
- la sauvegarde des parties précédemment jouées.

Quant aux fonctionnalités bonus que nous pourrions ajouter, il pourrait être intéressant d'instaurer un système de récompenses et de malus. La fonctionnalité "replay" est également à considérer ainsi que celle d'authentification et de gestion des utilisateurs.

2.2 Compréhension d'algorithmes

2.2.1 Python

Pour la partie python de l'application web, nos connaissances préalables et un peu de réflexion nous ont permis de mettre au point tous les algorithmes nécessaires sans avoir besoin d'analyser les algorithmes utilisés par les autres applications déjà en place.

2.2.2 Web

Pour la partie site, nous avons vite compris qu'il nous faudrait une technologie autre que le couple HTML/CSS, si nous voulions avoir un résultat dynamique et fluide. Nos recherches ont toutes plus ou moins convergé vers JavaScript, qui était l'outil parfait pour rendre notre site plus agréable à utiliser et réellement interactif.

Chapitre 3

Gestion de projet

Contents

3.1	Equipe de projet	8
3.2	Analyse du projet	8
3.2.1	Définition du projet	8
3.2.2	Définition des lots	9
3.2.3	Matrice SWOT	10
3.3	Organisation	11
3.3.1	Diagramme de Gantt	11
3.3.2	Répartition des tâches	11
3.3.3	Réunions	12
3.4	Outils de travail	13
3.4.1	IDE	13
3.4.2	Outils de communication	13
3.4.3	Outils de travail collaboratif	13
3.4.4	Outils de mise en forme	13

3.1 Équipe de projet

L'équipe se compose de quatre étudiants en première année :

- ASSING Norman
- COUCHEVELLOU Clément
- BIAUSQUE Anna
- WANG Caroline

Norman, en tant qu'initiateur du projet, est désigné comme chef de projet.
Ce sera lui qui suivra l'avancement du projet.

3.2 Analyse du projet

3.2.1 Définition du projet

L'idée est de développer une instance, ainsi qu'un solveur de Wordle.

D'une part, l'objectif est de créer une application Web interactive qui reproduira le fonctionnement du jeu. Elle doit être réalisée sur une architecture Python/Web/Base de données et sauvegarder les données des parties jouées. Le joueur doit également pouvoir paramétriser la longueur des mots et le nombre maximal d'essais possibles.

D'autre part, le solveur WORDLE doit être implémenté exclusivement en langage C et utiliser les structures de données maximisant l'efficacité de la résolution. Il doit en outre proposer un mode d'exécution pas à pas pour suivre la résolution.

Le sujet du projet est donc précis et se concentre en partie sur un concept déjà existant : celui du Wordle. Ses contraintes sont ainsi assez élevées et sont décrites dans la charte de projet (fournie en Annexe). Le profil du projet peut être illustré par la figure suivante :

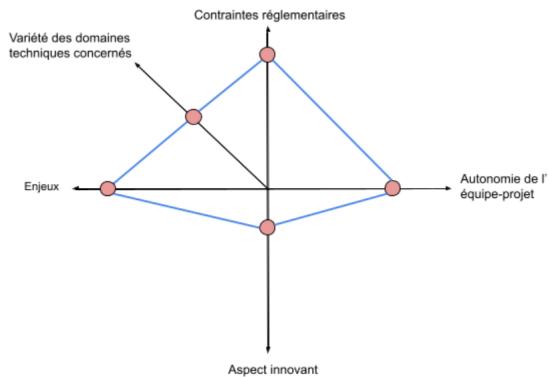


FIGURE 3.1 – Profil projet

3.2.2 Définition des lots

Les lots ont été définis à l'aide de la méthode **SMART** : “Spécifique, Mesurable, Atteignable, Réaliste, délimité dans le Temps”. Notons tout de même que le dernier critère n'a pas toujours été respecté, étant donné nos connaissances en informatique toujours en développement.

3.2.3 Matrice SWOT

Voici les forces, faiblesses, opportunités et menaces qui se sont ou auraient pu se présenter à nous durant ce projet :



FIGURE 3.2 – Matrice SWOT

3.3 Organisation

3.3.1 Diagramme de Gantt

Le diagramme de Gantt ci-dessous a servi de calendrier pour ces jalons, et représente également les écarts entre le temps prévu pour chaque tâche et la réalité.

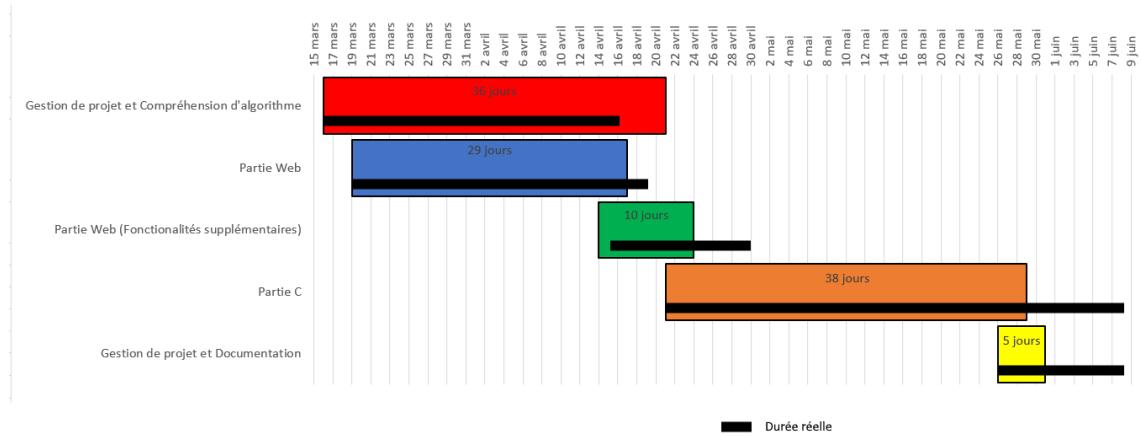


FIGURE 3.3 – Diagramme de Gantt

3.3.2 Répartition des tâches

Pour mieux visualiser la répartition des tâches, l'équipe a utilisé Google Sheet pour faire une matrice **RACI** : **R**esponsible , **A**ccountable , **C**onsulted, **I**nformed. Comme tout le groupe était constamment informé de l'avancement de chacun et que le contrôle des lots se faisait principalement par tous en réunion, la matrice RACI est principalement remplie de R.

	Norman	Anna	Clément	Caroline
Partie Web				
Adapter la partie Flask de l'ancien projet			R	
Conception du Jeu Wordle			R	
Clavier (CSS)		R		
Clavier (Raccorder au JS)		R		
Interactivité JS	R			
Colorer le clavier	R			
CSS (game + account system)	R		R	
CSS (stats + profile)				R
Connexion JS-Flask via Cookies	R			
Implémentation JS-Flask-CSS	R			
Partie BD (user)			R	
Partie BD (stats)				R
Partie BD (partie en cours)				R
Partie GdP/conception				
Charte projet			R	R
Profil projet		R		
CR réunion		R		
Document de conception Web	R	R	R	R
SWOT		R		R
Mockups		R		
Liste des tâches	R			
Calendrier/Gant	R			
Rapport	R	R	R	R
Partie C				
Interface	R			
Solveur Listes Classiques				R
Solveur Listes Doublement chaînées	R			
Solveur Arbres		R	R	
Tests Unitaires Listes Classiques				R
Tests Unitaires Listes Doublement chaînées	R			
Tests Unitaires Arbres		R	R	
Mesure Performance (temps)				R
Mesure Performance (nbr d'essais)				R

FIGURE 3.4 – Matrice RACI

3.3.3 Réunions

Ce projet n'aurait pu voir le jour sans organisation et une forte communication au sein du groupe. Il a été rythmé par les jalons qu'ont été les réunions, effectuées toutes les semaines. Ces réunions d'avancement, au nombre de 15, ont duré 1h et ont fait l'objet de la rédaction de comptes-rendus. Elles se sont déroulées en présentiel et en visioconférence.

Les comptes-rendus se présentent sous la forme suivante :

- Nom du/de la secrétaire
- Liste des membres présents
- Liste des membres absents
- Récapitulatif des travaux à faire depuis la dernière réunion
- Point sur l'avancement de chacun dans ses tâches
- Ordre du jour
- Traitement point par point des sujets à aborder
- Répartition des tâches à accomplir avant la prochaine réunion

Des exemples de comptes-rendus sont disponibles en annexe.

3.4 Outils de travail

3.4.1 IDE

L'ensemble du groupe a travaillé sur Visual Studio Code.

3.4.2 Outils de communication

Les réunions se déroulaient généralement via Discord, mais étant donné les agendas de chacun, il était difficile de tout faire rentrer lors de ces réunions, ainsi beaucoup de décisions et de répartition des tâches ont été faites via la conversation Messenger du projet. Certains comptes rendus, inachevés, ont pu être complétés en reprenant les messages de la conversation.

3.4.3 Outils de travail collaboratif

Pour synchroniser nos travaux, nous avons utilisé le GitLab de l'école pour tout l'aspect programmation. L'aspect gestion de projet a lui plutôt été géré sur Google Drive, puis déposé sur le GitLab.

3.4.4 Outils de mise en forme

L'équipe a utilisé diverses solutions pour illustrer ses idées, notamment Canva pour les présentations et la matrice SWOT, ainsi qu'Excel pour le diagramme de Gantt. Le rapport a été rédigé via LateX.

Chapitre 4

Conception de l'Application web

Contents

4.1	Cahier des charges	15
4.1.1	Contexte	15
4.1.2	Cadre technique : Jeu de WORDLE	15
4.2	Conception de l'Application web	15
4.2.1	Utilisateur	15
4.2.2	Mockups	16
4.2.3	WORDLE en Python	17
4.2.4	Intégration web	17
4.2.5	Partie Base de données	17
4.2.6	Particularité de l'application	21
4.2.7	Détail des fonctionnalités supplémentaires	21

4.1 Cahier des charges

4.1.1 Contexte

L'objectif est la conception d'un solveur de WORDLE, WORDLE que nous devons nous-même réaliser de A à Z, avec de multiples contraintes techniques.

4.1.2 Cadre technique : Jeu de WORDLE

La première étape est la conception du jeu de WORDLE, ici prénommé FLOPPLE, basée sur une architecture Web/Python/Base de données. L'application doit être finalisée pour le 31 avril, pour cela nous avons donc accès à Python et Flask, pour la partie interactive, SQL pour le stockage et la gestion des données ainsi que HTML et CSS pour la partie statique. Le jeu devra être paramétrable, en effet le joueur doit pouvoir choisir le nombre d'essais qu'il aura à sa disposition pour trouver le mot, mais il doit également pouvoir choisir la longueur du mot en question, qui sera compris entre 2 et 25, car limité par le dictionnaire français. Afin qu'un joueur ne tombe pas plusieurs fois sur le même mot nous devons également sauvegarder dans une base de données les mots déjà trouvés par un joueur.

4.2 Conception de l'Application web

4.2.1 Utilisateur

Chaque utilisateur peut créer un compte sur l'application, ce qui lui permet notamment de sauvergarder les mots qu'il a déjà réussi à trouver afin de ne pas retomber dessus en jouant. Les informations relatives aux utilisateurs sont stockées dans notre base de données qui sera détaillée plus loin.

4.2.2 Mockups

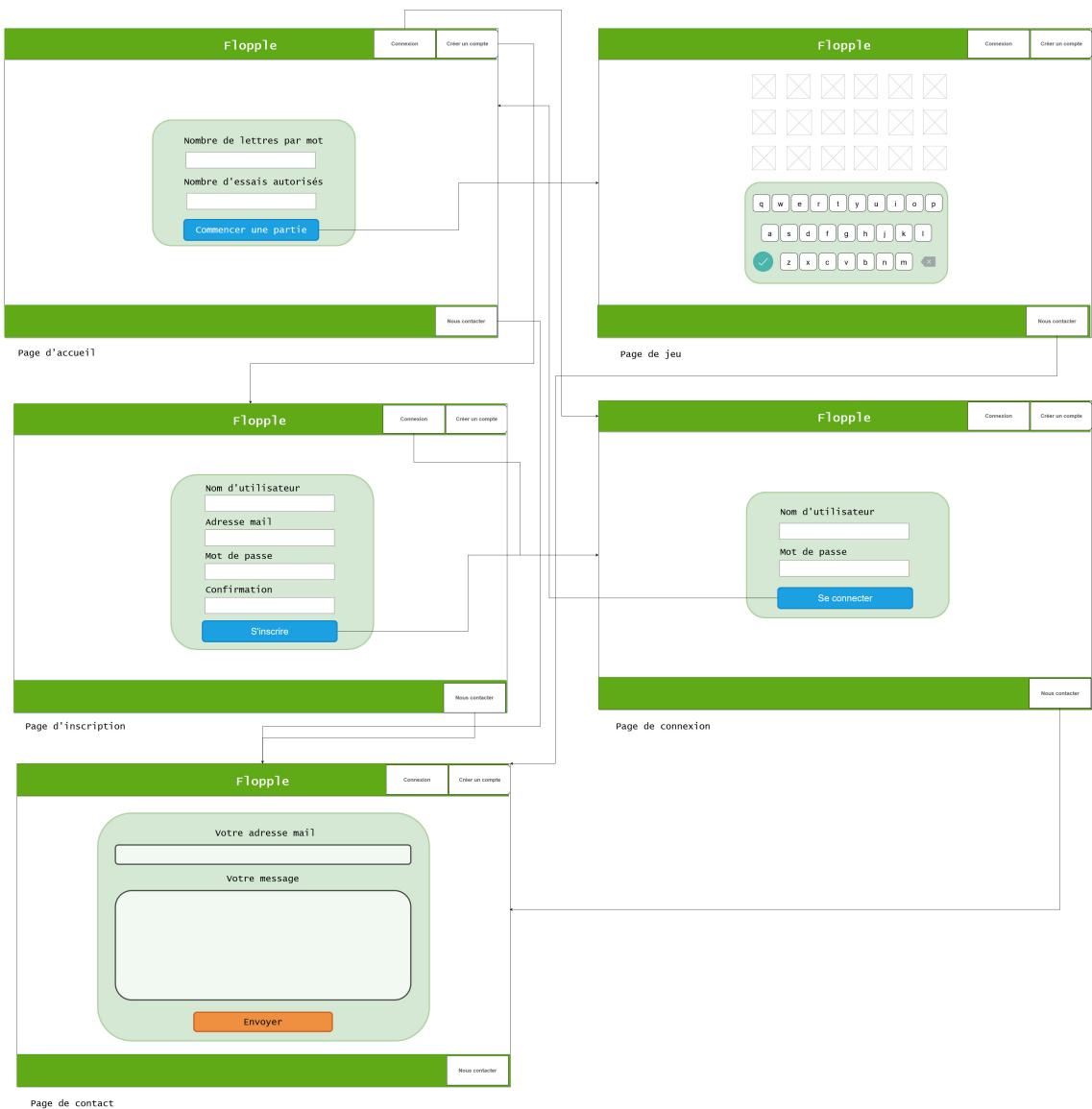


FIGURE 4.1 – Mockups de l'application

4.2.3 WORDLE en Python

Le fichier Python sélectionne de manière aléatoire un mot de la bonne longueur parmi les mots stockés dans la base de données. On fait ensuite interagir grâce à Flask la page HTML et le fichier Python afin de récupérer puis d'analyser les mots entrés par l'utilisateur, en les comparant avec le mot cible. A chaque nouveau mot entré, le fichier Python renvoie les similarités sous forme de couleurs associées à chaque lettre, ainsi que les mots précédemment entrés et leurs similarités avec le mot cible. Est enfin renvoyé le nombre d'essais restant.

4.2.4 Intégration web

La base du site est réalisé via HTML/CSS, couplé à Flask via un fichier Python pour rendre le site interactif. Afin de rendre le site plus agréable, dynamique et ergonomique possible, nous utiliseront également JavaScript pour l'affichage dynamique et la gestion des cookies.

4.2.5 Partie Base de données

Pour notre base de données, nous utilisons SQL, plus précisément SQLite3 sur Python. Le choix de SQLite3 pour gérer nos données sous Python a été motivé majoritairement pour la performance du SQL pour gérer de grandes quantités de données, mais également par le fait que SQLite3 permette de gérer très intuitivement des bases de données directement dans notre fichier Python.

Notre base de données est constituée de plusieurs Tables, la table userInfo qui regroupe toutes les informations concernant les utilisateurs.

Les informations d'un compte utilisateur sont conservées dans la table userInfo. Un compte utilisateur est identifié par un identifiant *id* (de type *integer*) et est obligatoirement associé à un unique nom d'utilisateur *username* (de type *text*), une unique adresse mail *email* (de type *text*), un mot de passe *password* (de type *text*), la série de victoires en cours *currentStreak* (de type *int*) et la série maximum de victoires du joueur *maxStreak* (de type *int*).

Ainsi que la table words qui contient toutes les informations relatives aux mots que le jeu reconnaît.

Chaque mot de la langue est conservé dans la table words. Un élément de cette table est identifié par un identifiant *id_word* (de type *integer*) et représente obligatoirement et sans redondance un mot *word* (de type *text*) d'une longueur *length* (de type *int*) donnée.

La gestion de l'historique des parties du joueur requiert également une table games.

Les données de chaque partie sont conservées dans la table games. Une partie est identifiée par un identifiant *id game* (de type *integer*) et les informations conservées sont : le nombre d'essais maximum *tries* (de type *int*) de la partie, les tentatives *guesses* (de type *text*) du joueur (chaque nouvelle tentative est concaténée aux mots précédents, eux-mêmes concaténés dans l'ordre chronologique d'entrée), les résultats *results* (de type *text*) correspondant à chaque tentative (cette chaîne de caractère est obtenue de la même manière que *guesses*), le statut *status* (de type *int*) de la partie (perdue, gagnée ou en cours), le nombre d'essais *leftT* (de type *int*) restant à la fin de la partie. Une partie terminée est associée à un mot par son identifiant #*id_word* (de type *integer*) et à un compte utilisateur par son identifiant #*id* (de type *integer*).

Les tables utilisées sont donc les suivantes :

```
userInfo(id, username, email, password, currentStreak, maxStreak)
words(id word, word, length)
games(id game, tries, guesses, results, status, leftT, #id_word, #id)
```

Les contraintes attachées au schéma de la base de données sont énumérées ci-dessous :

- Les clés primaires sont soulignées.
- Les clés étrangères sont identifiées par #.
- Le nom d'utilisateur *username* et l'adresse mail *email* d'un joueur ne peuvent avoir la valeur *null* et doivent être uniques dans la table.
- Le mot de passe *password* ne peut être *null*.
- La série de victoires courante et la série de victoires maximale d'un joueur sont des entiers naturels.
- Le mot *word* n'est pas *null*
- La longueur d'un mot *length* est un entier naturel non nul.
- Le nombre maximum d'essais possibles *tries* est un entier naturel non nul.
- Les attributs *guesses* et *results* ont pour valeur par défaut " lorsqu'un joueur lance une nouvelle partie.
- Le statut *status* d'une partie est restreint aux valeurs 0 si la partie a été perdue, 1 si elle a été gagnée, 2 si elle est encore en cours.

- Le nombre d'essais restants d'un joueur pour une partie $leftT$ est un entier compris entre 0 et $tries$.
- L' $\#id_word$ de la table **games** fait référence à l'identifiant id_word de la table **words**.
- L' $\#id$ de la table **games** fait référence à l'identifiant id de la table **userInfo**.

Ces tables sont illustrées par la figure suivante.

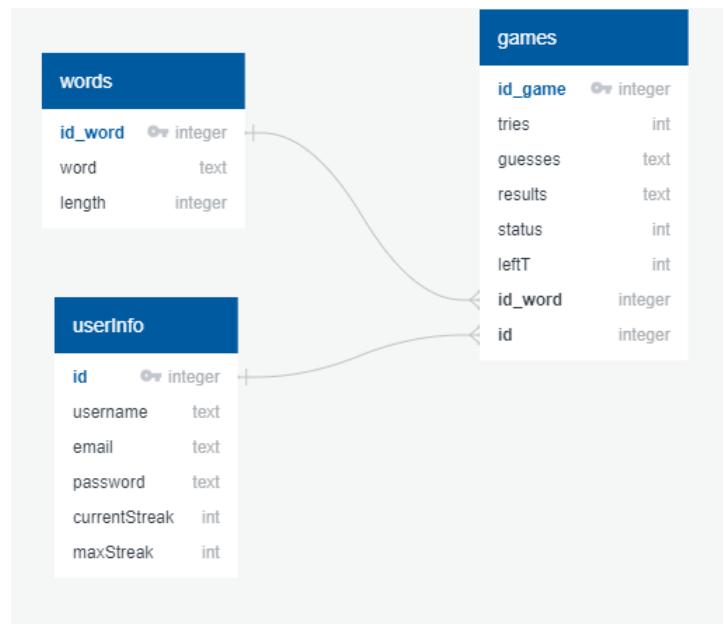


FIGURE 4.2 – Schéma de la base de données

La mise sous troisième forme normale (3NF) des tables décrites ci-dessus donne les tables suivantes :

```

userInfo(id, #username, currentStreak, maxStreak)
userInfo_username(username, #email)
userInfo_email(email, password)

words(id_word, #word)
words_word(word, length)

games(id_game, #tries, #guesses, #id_word, #id)
games_triesANDguesses(tries, #guesses, status, leftT)
  
```

games_guesses(guesses, results)

Le schéma de la base de données sous troisième forme normale est alors celui de la figure 2.2 :

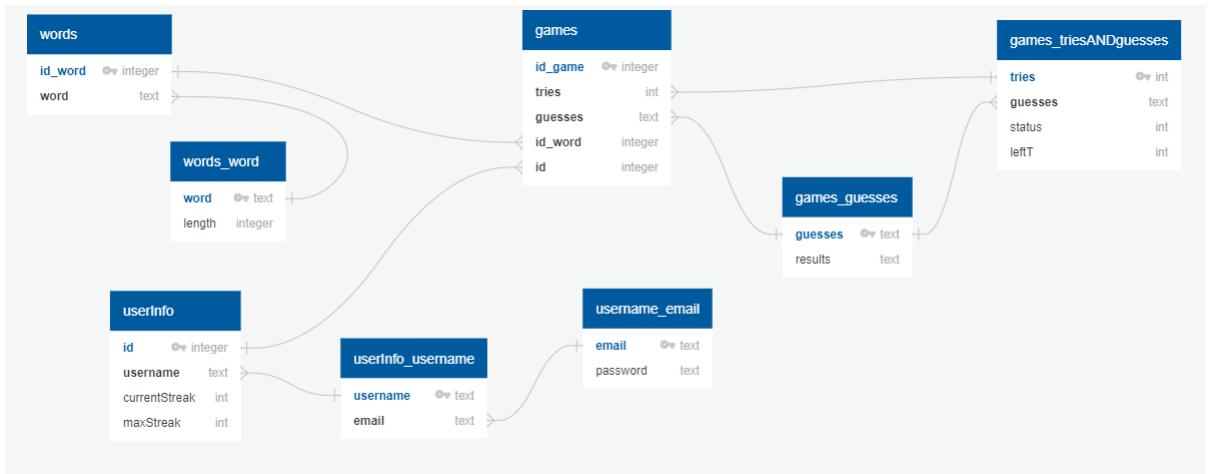


FIGURE 4.3 – Schéma de la base de données mise sous forme 3NF

4.2.6 Particularité de l'application

Plutôt que de créer un simple clone en français de Wordle, nous avons décidé d'ajouter des fonctionnalités ludiques dans le but de donner un réel intérêt à notre application. Ainsi, Flopple serait une version de Wordle où l'on collecterait des images autour du thème du même de Big Floppa.



FIGURE 4.4 – L'image la plus célèbre de Big Floppa

Big Floppa est le nom donné à plusieurs caracals, un genre de la famille des félins. Le plus célèbre est Gosha, dont le compte Instagram dépasse les 150 000 abonnés.

4.2.7 Détail des fonctionnalités supplémentaires

Lorsqu'un joueur trouve un mot, il obtient un Floppa Coin, qui lui permet de faire un tirage dans un distributeur afin d'obtenir des images de Big Floppa. Un système de rareté fera que certaines images seront plus difficiles à obtenir que d'autres.

Cependant, les parties ne seront pas sans risque. S'il perd trop souvent, le joueur verra son compte se faire supprimer, et il n'aura plus d'autre choix que de tout recommencer depuis le début. Cette mécanique de jeu est introduite via le plus grand ennemi de Floppa, un chat de la race des Sphynx nommé Bingus.

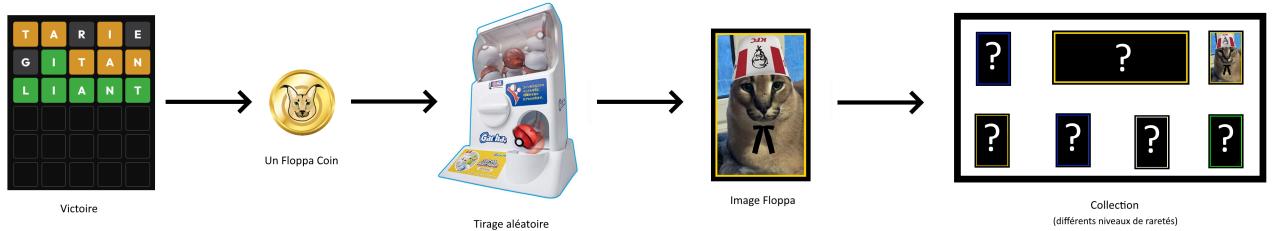


FIGURE 4.5 – Schéma expliquant le système de collection d’images



FIGURE 4.6 – Bingus

Chaque partie perdue donne au joueur un fragment de Bingus, et au bout de trois fragments, le compte du joueur est définitivement supprimé. En revanche, si lors d'un tirage au distributeur, le joueur reçoit une image étant déjà dans sa collection, tous ses fragments seront retirés.

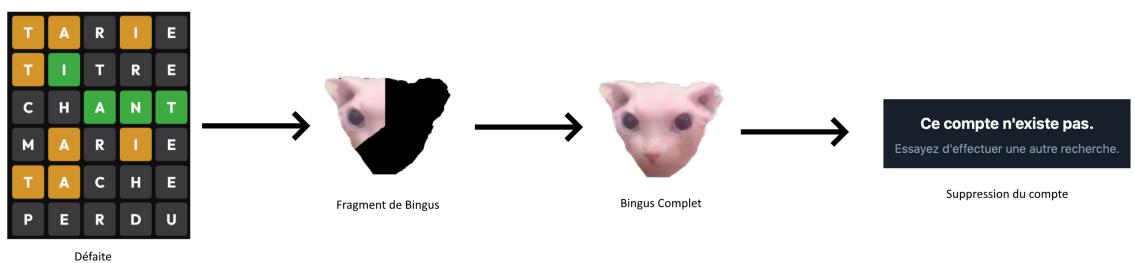


FIGURE 4.7 – Schéma expliquant le processus de suppression de compte

Chapitre 5

Conception et implémentation du Solveur en C

Contents

5.1	Cahier des charges	24
5.1.1	Cadre technique	24
5.1.2	Besoin en structures de données	24
5.2	Plusieurs stratégies pour le Solveur	24
5.2.1	Première stratégie se basant sur l'entropie	24
5.2.1.1	Détail de la stratégie	24
5.2.1.2	Structures de données	25
5.2.1.3	Détail de l'implémentation	26
5.2.2	Seconde stratégie se basant sur l'entropie	27
5.2.2.1	Détail de la stratégie	27
5.2.2.2	Structures de données	29
5.2.3	Stratégie basée sur la fréquence des lettres et des bigrammes	29
5.2.3.1	Détail de la stratégie	29
5.2.3.2	Structures de données	31
5.2.3.3	Algorithmique & organisation du code . .	34

5.1 Cahier des charges

5.1.1 Cadre technique

Pour la partie solveur, nous devons uniquement utiliser le langage C, avec une focalisation très importante sur la partie "Structure de données" afin d'optimiser notre algorithme aussi bien en temps qu'en utilisation de la mémoire, mais l'algorithme doit surtout être efficace en terme de nombre de mots rentrés. Il faut également que le cheminement de l'algorithme soit visible en temps réel dans la console.

5.1.2 Besoin en structures de données

Pour réaliser le solveur, il nous faut au minimum implémenter un dictionnaire de référence, qui est très long, régulièrement appelé et entièrement parcouru par l'algorithme

5.2 Plusieurs stratégies pour le Solveur

5.2.1 Première stratégie se basant sur l'entropie

5.2.1.1 Détail de la stratégie

L'algorithme présenté dans cette section repose sur le calcul de l'information que l'on espère obtenir après avoir essayé un certain mot. Cette information espérée est aussi appelée entropie.

L'idée est de déterminer cette quantité pour chaque mot potentiellement correct, et de jouer le mot dont l'entropie est la plus élevée.

Pour calculer l'entropie d'un mot m relative à un dictionnaire donné, on applique l'algorithme suivant :

1. Lister les motifs que le mot peut engendrer, un motif étant une suite de 0, 1 ou 2 de même longueur que le mot et indiquant quelles lettres du mot essayé sont présentes ou non dans le mot caché et le cas échéant, si elles sont à la bonne place ou non
2. Pour chaque motif, compter le nombre de mots m' différents de m du dictionnaire sont compatibles avec le motif. Un mot m' est dit compatible avec le motif du mot m lorsque :
 - les lettres de m numérotées 2 dans le motif sont présentes au même endroit dans m'

— les lettres de m numérotées 1 dans le motif sont présentes dans m' ,

mais pas au même endroit que dans m

— les lettres de m numérotées 0 dans le motif sont absentes de m'

3. A partir de ce nombre, on calcule l'entropie E dudit mot :

$$E = \sum_{motif} p(motif) * I(motif), \quad (5.1)$$

avec :

$$p(motif) = \frac{\text{nombre de mots compatibles avec le motif}}{\text{nombre de mots dans le dictionnaire}} \quad (5.2)$$

et

$$I = -\log_2 (p(motif)) \quad (5.3)$$

L'algorithme de résolution pour une partie jouée avec un mot de longueur n est alors :

1. On initialise le dictionnaire à tous les mots de longueur n
2. Tant que le mot n'est pas trouvé ou que la partie n'est pas terminée :
 - Calculer l'entropie de chaque mot du dictionnaire
 - Jouer le mot dont l'entropie est la plus élevée
 - Modifier le dictionnaire pour ne garder que les mots qui sont compatibles avec le motif obtenu à l'étape précédente

5.2.1.2 Structures de données

L'implémentation de cet algorithme requiert une structure de données permettant de stocker chaque mot du dictionnaire. Dans cette sous section, on s'intéresse à un dictionnaire de la forme suivante :

```
typedef struct dictionary dictionary ;
struct dictionary
{
    int length ;
    int nb_words ;
    list_str* list_words ;
    int nb_patterns ;
    list_str* list_patterns ;
};
```

FIGURE 5.1 – Structure de données utilisée pour le dictionnaire

Le dictionnaire contient les informations suivantes :

- La longueur des mots du dictionnaire : *int length*
- Le nombre total des mots contenus dans le dictionnaire : *int nb_words*
- Un pointeur vers la liste chaînée des mots de ce dictionnaire : *list_str* list_words*
- Un pointeur vers la liste chaînée des motifs pouvant être produits par un mot de longueur *length* : *list_str* list_patterns*

Pour cela, une structure de listes chaînées a été implémentée :

```

typedef struct element_str element_str ;
struct element_str
{
    char* string_value ;
    element_str* next ;
} ;

typedef struct list_str list_str ;
struct list_str
{
    element_str* head ;
} ;

```

FIGURE 5.2 – Structures de données utilisées pour les listes chaînées

5.2.1.3 Détail de l’implémentation

L’implémentation de cette stratégie s’est faite à partir de différents modules, chacun composé d’un fichier .h, qui contient la liste des fonctions implémentées et son objectif, et un fichier .c :

- *list_str* (*list.h* et *list_str.c*), qui définit la structure de données des listes chaînées (figure 5.2) et contient toutes les fonctions relatives à la gestion de celles-ci.
- *patterns* (*patterns.h* et *patterns.c*), qui contient les fonctions de construction de la liste des motifs possibles pour une longueur de mots données.
- *dictionary* (*dictionary.h* et *dictionary.c*), qui définit la structure de données des dictionnaires (figure 5.1) et les fonctions relatives à la création, libération de la mémoire utilisée et affichage d’un dictionnaire.
- *compatible* (*compatible.h* et *compatible.c*), qui définit les fonctions relatives à l’analyse de la compatibilité des mots pour la recherche du mot idéal.
- *bestGuess* (*bestGuess.h* et *bestGuess.c*), qui définit les fonctions relatives à la recherche effective du mot idéal.
- *files* (*files.h* et *files.c*), qui définit les fonctions relatives à la gestion

des fichiers (lecture et écriture pour wsolf et pour le stockage des mots d'ouverture et de la liste des motifs possibles pour une longueur données).

- *solver* (*solver.h* et *solver.c*), qui définit les fonctions relatives à l'exécution du solveur : commencer ou recommencer une partie, avancer d'une étape, terminer une partie.

Étant donnée un nombre de lettres, le mot d'ouverture reste le même pour chaque partie. Nous avons donc décidé de stocker ce mot d'ouverture dans un fichier texte afin de réduire le temps d'exécution du solveur pour lancer une partie. Le même choix a été fait pour stocker la liste des motifs possibles.

Ainsi, les fichiers *patterns_of_length3*, *patterns_of_length4*, ..., *patterns_of_length8* sont des fichiers textes qui, pour chaque partie de longueur 3 à 12, contient dans sa première ligne le mot d'ouverture correspondant, puis un motif par ligne.

À son lancement depuis l'exécutable *test_solver*, le solveur va simplement lire le fichier *patterns_of_length* correspondant à la longueur de mot souhaitée, en extraire le mot d'ouverture pour le proposer, et remplir l'attribut *list_patterns* du dictionnaire grâce aux lignes du fichier ouvert.

Dans le but de simplifier la compilation et l'exécution de ces modules, une makefile a été écrit.

Ainsi, les commandes suivantes permettent de lancer le solveur :

```
make test_solver  
./test_solver
```

5.2.2 Seconde stratégie se basant sur l'entropie

5.2.2.1 Détail de la stratégie

Cette stratégie est assez similaire à celle précédente mais elle intègre plusieurs modifications notables qui en font une stratégie à part entière.

Il y a trois changements principaux : l'ajout de structures de données, l'utilisation de deux dictionnaires distincts et une entropie un peu modifiée.

Concernant les structures de données, on s'est basé sur une liste doublement chaînée qui permet de supprimer des éléments du dictionnaire facile-

ment. Ces structures sont détaillées dans la section suivante.

Le calcul de l'entropie a été modifié afin de prendre en compte toutes les informations possibles que l'on peut apporter, c'est à dire les nouvelles lettres vertes, oranges et grises.

Une pondération était donc nécessaire pour optimiser la recherche, ce qui nous a mené au système suivant :

$$E = \sum_i \frac{score_{input_solution_i}}{\text{taille du dictionnaire}} \quad (5.4)$$

avec :

$$score_{input_solution_i} = \frac{n_{vertes} + 0.5n_{oranges} + 0.2n_{grises}}{\text{taille du mot}} \quad (5.5)$$

où E est l'entropie et $n_{[couleur]}$ est le nombre de lettres de cette couleur. $score_{input_solution_i}$ est le score de l'input calculé pour le i-ème mot du dictionnaire des solutions.

Finalement, à cause des nombreuses contraintes que cela impliquait et surtout par manque de temps, cette formule a été modifiée pour ne prendre en compte que les lettres vertes, avec une pondération en fonction de si on savait déjà que la lettre était verte ou non :

$$score_{input_solution_i} = \frac{2n_{nouvelles_vertes} + n_{anciennes_vertes}}{\text{taille du mot}} \quad (5.6)$$

Ces poids ont été choisis arbitrairement. Il aurait certes été possible de déterminer les meilleurs coefficients avec des milliers et des milliers de simulations, mais le manque de temps ne nous a pas permis de faire ce genre d'expérience.

Pour obtenir le meilleur mot, on calcule l'entropie comme précédemment ; on doit donc parcourir une fois le dictionnaire, puis une seconde fois pour chaque mot du premier, afin de calculer son score. On aura alors une complexité en $O(n^2)$, et l'objectif est de réduire cette complexité au minimum. Cependant, en enlevant à chaque fois les mots qui ne conviennent pas du dictionnaire, on risque de passer à côté de mot qui nous auraient fourni davantage d'information que les autres, et c'est pourquoi nous avons décidé de considérer deux dictionnaires au lieu d'un.

Le premier que l'on parcourt, c'est le dictionnaire des mots que le solveur va proposer ; on le nommera *dictionnaire des inputs* : c'est celui qui diminuera le moins en taille. En effet, il sera tout de même possible de supprimer

quelques mots de ce dictionnaire en enlevant ceux ayant un score inférieur à une valeur seuil. Encore une fois, ce seuil a été choisi arbitrairement, et une simulation aurait pu nous donner une valeur plus optimale.

Lorsque l'on parcourt le second dictionnaire pour chaque mot du dictionnaire des inputs, il n'est pas nécessaire d'utiliser les mots qui ne conviennent pas pour calculer l'entropie.

Le second dictionnaire sera donc le *dictionnaire des solutions possibles*. Il diminuera fortement en taille à chaque essai car il ne conserve que les solutions possibles, d'où le nom plutôt ingénieux. Il a d'ailleurs un autre avantage : en ne considérant pas les mots qui ne conviennent pas dans le calcul de l'entropie, on obtiendra un résultat plus affiné, permettant ainsi de diminuer potentiellement le nombre d'essais.

5.2.2.2 Structures de données

Pour faire un dictionnaire sous forme d'une liste doublement chaînée, la structure de base est Dictionary. Il ne contient que l'adresse du premier mot du dictionnaire.

La seconde structure est LinkedWord, qui contiendra le mot sous forme de pointeur de char (*value*), ainsi que l'adresse des mots suivants et précédents.

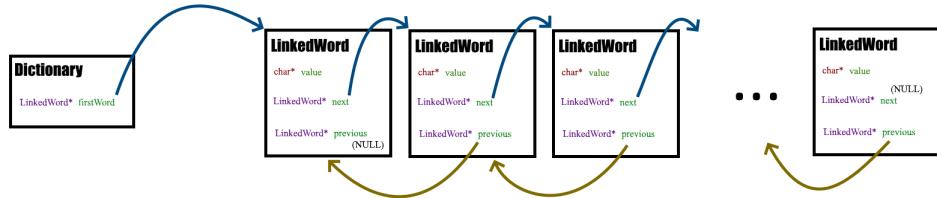


FIGURE 5.3 – Structure du dictionnaire

5.2.3 Stratégie basée sur la fréquence des lettres et des bigrammes

5.2.3.1 Détail de la stratégie

Nous avons décidé, afin d'appliquer notre stratégie en ayant le maximum d'information possible au départ, d'imposer 2 mots à rentrer en début de

partie. Ils ne sont pas choisis arbitrairement mais à la suite de calculs de diversité et de fréquence des caractères les composant.

La suite se déroule en 2 étapes principales. Tout d'abord, on récupère un dictionnaire des mots de la langue française, auquel on retire tous les mots qui ne sont pas de la bonne longueur. On effectue ensuite un tri récursif à chaque mot entré dans Wordle. Ce tri permet de minimiser le nombre de tests effectués dans la suite de la stratégie, et consiste en une suppression dans le dictionnaire actuel de tous les mots ne correspondant pas au "pattern" renvoyé par Wordle lors du dernier mot rentré.

La suite de la stratégie nécessite d'avoir accès à une liste des lettres de l'alphabet et des bigrammes (chaînes de caractères de longueur 2) de la langue française, respectivement munis de leurs fréquences d'apparition dans cette même langue. On va ensuite considérer le couple (mot, pattern) renvoyé lors de l'entrée du dernier mot dans le Wordle, que l'on va décomposer sous forme d'un arbre binaire comme celui ci :

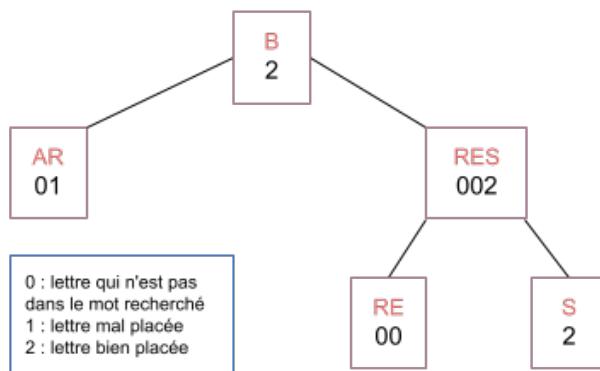


FIGURE 5.4 – Exemple d'arbre formé à partir d'un mot et de son pattern associé

La méthode utilisée afin de décomposer un couple (mot, pattern) en arbre est détaillée dans la partie "Structure de données".

On effectue un pré-tri de la liste des bigrammes et des lettres à chaque nouveau pattern renvoyé par le jeu : pour tout "0" dans le pattern renvoyé, on supprime la lettre correspondante dans la liste des lettres ou les bigrammes la contenant dans la liste des bigrammes.

On va ensuite parcourir l'arbre jusqu'à ses feuilles et effectuer un tri sur l'existence et la fréquence de cette feuille dans un mot du dictionnaire préalablement trié, utilisant la méthode du backtracking. Ainsi, sur chaque feuille, on vérifie que le bigramme ou la lettre la composant :

- soit présent à cet emplacement dans au moins un mot du dictionnaire
- ait une fréquence non nulle dans la langue française.

On en retire ainsi une liste des n bigrammes/lettres étant les plus fréquents dans la langue française et compatibles avec le dictionnaire déjà trié.

On concatène les possibilités de feuilles ayant le même parent, et pour chaque concaténation possible, on répète les étapes précédentes sur le bigramme formé par le dernier caractère du fils gauche et le premier du fils droit.

A chaque concaténation, on obtient une "fréquence globale" en calculant la moyenne des fréquences des deux éléments fils et du bigramme formé par le dernier caractère du fils gauche et le premier du fils droit. On peut classer les possiblités pour l'élément parent par ordre décroissant de fréquence globale et réitérer le processus jusqu'à atteindre la racine.

5.2.3.2 Structures de données

Afin d'implémenter notre stratégie, nous aurons besoin de créer au préalable différentes structures de données. Elles sont toutes détaillées dans le fichier **architecture.h**.

- La structure d'un noeud

```
typedef struct Node Node;
struct Node
{
    int length;
    int position;
    char* letters;
    char* pattern;
};
```

FIGURE 5.5 – Structure d'un noeud

La structure d'un noeud est telle qu'on ait accès au maximum d'informations sur le noeud sans avoir à parcourir l'arbre pour les retrouver. Un noeud est composé de :

- une partie du mot précédemment entré dans la Wordle

- la partie du pattern qui lui est associé
 - la position de sa première lettre dans le mot
 - sa longueur
- La structure d'un arbre

```
typedef struct Tree Tree;
struct Tree
{
    Node *node;
    Tree *tleft;
    Tree *tright;
    Tree *parent;
};
```

FIGURE 5.6 – Structure d'un arbre

La structure utilisée pour l’arbre est classique : à un noeud on associe un arbre gauche, un arbre droit et un arbre parent. Sa particularité repose sur la manière dont il est construit à partir d’un mot et de son pattern associé.

Pour ce faire, on utilise la fonction *buildTree(Tree* beginTree)* qui prend en argument l’arbre de départ, construit grâce à la fonction *beginTree(Node* racine, char* mot, char* pattern)*. Le pivot est choisi comme étant la lettre bien placée la plus proche du milieu du mot (obtenue avec les essais de mots prédéfinis).

De là, on va créer un arbre avec pour racine le pivot, et feuilles les parties gauche et droite du mot. On les décompose ensuite en fonction de leurs parités respectives :

- si la feuille est de longueur paire, elle devient parent de deux feuilles paires de même longueur.
- si la feuille est de longueur impaire, elle devient parent d’une feuille paire et d’une feuille impaire, avec $\text{length}(\text{feuille_paire}) = \text{length}(\text{feuille_impaire}) + 1$

La partie gauche du noeud à décomposer devient toujours son fils gauche et la partie droite son fils droit.

On répète l’opération jusqu’à avoir des feuilles composées uniquement de lettres et de bigrammes (longueur 1 ou 2).

- La structure d’une lettre ou d’un bigramme

```
typedef struct Frequency Frequency;
struct Frequency
{
    char* elemt;
    double freq;
    Frequency* next_elmt;
    Frequency* prev_elmt;
};
```

FIGURE 5.7 – Structure d’une lettre ou d’un bigramme

Il nous est apparu nécessaire de créer une structure **Frequency** afin de stocker les lettres/bigrammes associés à leur fréquence d’apparition dans la langue française. Elle se présente sous la forme ci-dessus.

Petite précision quant aux fréquences : il nous était apparu correct au départ de baser notre étude sur des fréquences de lettres et de bigrammes trouvées sur Internet. Nous nous sommes rendu.e.s compte par la suite que le calcul de ces fréquences était basé sur leur fré-

quence d'apparition dans un corpus, et donc dépendait également de la fréquence d'utilisation des mots dans la langue française. Nous avons alors décidé de recalculer ces fréquences par nous-même à l'aide du dictionnaire utilisé dans la partie **Flopple**.

- La structure d'un dictionnaire

```
typedef struct Dictionary Dictionary ;
struct Dictionary
{
    // La Longueur des mots du dictionnaire
    char* word;
    // Un pointeur vers Le mot suivant
    Dictionary* next ;
    // Un pointeur vers Le mot précédent
    Dictionary* previous ;
} ;

typedef struct First_word First_word;
struct First_word
{
    Dictionary* dico;
};
```

FIGURE 5.8 – Structure d'un dictionnaire

La structure utilisée est la même que pour le solveur précédent.

5.2.3.3 Algorithmique & organisation du code

Le code est divisé en trois parties principales :

- les fichiers **architecture.h**, **architecture.c** et **architecture_test.c** qui définissent les structures (fréquences, arbres, dictionnaires etc) et les fonctions de manipulation de ces structures.
- les fichiers **solveur.h**, **solveur.c** et **solveur_test.c** qui contiennent le "dur" du solveur.
- les fichiers **.txt frequencies_bigrammes.txt**, **dictionnaire.txt** et **frequencies_lettres.txt** qui regroupent les mots, lettres et bigrammes utiles à l'utilisation du solveur.

Les deux fonctions principales sont :

- la fonction de construction d'un arbre *buildTree* :

```

void buildTree(Tree* beginTree)
{
    bool finished_l = false;
    bool finished_r = false;

    Tree* tleft_ = beginTree->tleft;
    Tree* tright_ = beginTree->tright;

    Tree* leftside_l= malloc(sizeof(Tree));
    Tree* rightside_l= malloc(sizeof(Tree));
    Tree* leftside_r = malloc(sizeof(Tree));
    Tree* rightside_r = malloc(sizeof(Tree));

    tleft_->tleft=leftside_l;
    leftside_l->parent=tleft_;
    tleft_->tright=rightside_l;
    rightside_l->parent=tleft_;
    tright_->tleft=leftside_r;
    leftside_r->parent=tright_;
    tright_->tright=rightside_r;
    rightside_r->parent=tright_;
}

```

FIGURE 5.9 – Début de la fonction buildTree

On initialise des flags à **false** qui nous aideront par la suite à "arrêter" la construction récursive de notre arbre. On alloue ensuite la mémoire nécessaire à la construction de notre arbre et on "relie" les branches enfants et parents.

```

if (tleft_>node->length == 1 || tleft_>node->length == 2){
    tleft_>tleft=NULL;
    tleft_>tright=NULL;
    finished_l= true;

}
else
{
    if(isEven(tleft_>node->letters))
    {
        lpivot= (tleft_>node->length)/2-1;
    }

    else
    {
        lpivot=(tleft_>node->length+1)/2-1;
    }

    char* cleft = access_chunk(tleft_>node->letters, 0, lpivot+1);
    char* pattern_left = access_chunk(tleft_>node->pattern, 0, lpivot+1);
    char* cright = access_chunk(tleft_>node->letters, lpivot +1, strlen(tleft_>node->letters)-(lpivot+1));
    char* pattern_right = access_chunk(tleft_>node->pattern, lpivot +1, strlen(tleft_>node->letters)-(lpivot+1));
    tleft_>tleft=new_tree( new_node(strlen(cleft),tleft_>node->position,cleft,pattern_left));
    tleft_>tright=new_tree( new_node(strlen(cright),strlen(cleft),cright,pattern_right));
}

```

FIGURE 5.10 – Construction de la branche gauche

On passe ensuite à la construction de la branche gauche de l’arbre. On construit chaque noeud avec les informations qu’il doit contenir : partie du mot, pattern associé, longueur de la partie, position de la partie dans le mot global. Si la longueur du noeud considéré est égale à 1 ou 2, notre flag passe à **true**.

```

if (tright_>node->length == 1 || tright_>node->length ==2){
    tright_>tleft=NULL;
    tright_>tright=NULL;
    finished_r = true;
}

else {
    if (isEven(tright_>node->letters))
    {
        rpivot= tright_>node->length/2 -1;
    }
    else
    {
        rpivot= (tright_>node->length+1)/2 -1;
    }

char* cleft2 = access_chunk(tright_>node->letters, 0, rpivot);
char* pattern_left2 = access_chunk(tright_>node->pattern, 0, rpivot);
char* cright2 = access_chunk(tright_>node->letters, rivot, strlen(tright_>node->letters)-strlen(cleft2));
char* pattern_right2 = access_chunk(tright_>node->pattern, rivot +1, strlen(tright_>node->letters)-strlen(cleft2));
leftside_r=new_tree( new_node(strlen(cleft2),tright_>node->position,cleft2,pattern_left2));
rightside_r=new_tree( new_node(strlen(cright2),strlen(cleft2),cright2,pattern_right2));
}

```

FIGURE 5.11 – Construction de la branche droite

On fait de même pour la branche droite.

```

if (!finished_l)
{
    buildTree(tleft_);
}
if (!finished_r)
{
    buildTree(tright_);
}
if (finished_r && finished_l)
{
    return ;
}

};


```

FIGURE 5.12 – Construction récursive

Tant que les flags ont la valeur false, on construit récursivement les deux branches en prenant en argument de buildTree respectivement l'arbre droit et l'arbre gauche du noeud considéré.

Chapitre 6

Tests et performances du solveur

Contents

6.1	Tests unitaires	39
6.2	Complexité	40
6.2.1	Solveur basé sur la première stratégie avec l'entropie	40
6.2.2	Solveur basé sur la deuxième stratégie avec l'entropie	41
6.2.3	Solveur basé sur la fréquence des lettres et des bigrammes	41
6.2.3.1	Complexité de builTree	41
6.3	Efficacité du solveur en nombre de tentatives	41

6.1 Tests unitaires

Les tests unitaires ont été réalisés au cours de l'implémentation :

- Pour le solveur basé sur la première stratégie avec l'entropie, les tests de chaque fonction de chaque module ont été écrits dans un fichier correspondant préfixé de "*test_*", terminant par le nom du module testé. Par exemple, les tests des fonction de *list_str* ont été faits dans le fichier *test_list_str.c*. Le makefile permet alors de créer l'exécutable puis de le lancer grâce aux commandes suivantes :

```
make test_list_str  
./test_list_str
```

- Pour le solveur basé sur deux listes différentes, tous les tests pour chaque fonction ont été écrits dans un unique fichier *solveur_linked_test.c*. Il a été compilé avec l'extension *C/C++ Compile Run* par daniel-pinto8zz6 sur Visual Studio Code.

```
PS C:\Users\norma\Documents\Code\project2-E9\solveur_linked> & .\"solveur_li  
Tests for LinkedWords  
LinkedWords : 5 test(s) passed out of 5  
  
Tests for MakeDictionary  
MakeDictionary : 7 test(s) passed out of 7  
  
Tests for RemoveFromDictionary  
RemoveFromDictionary : 4 test(s) passed out of 4  
  
Tests for RefreshFromDictionary  
RefreshDictionary : 6 test(s) passed out of 6  
  
Tests for getScore  
getScore : 7 test(s) passed out of 7  
  
Tests for getGreenLetters  
getGreenLetters : 3 test(s) passed out of 3  
  
Tests for getBestWord  
getBestWord : 3 test(s) passed out of 3  
-----  
Final results : 35 test(s) passed out of 35
```

FIGURE 6.1 – Tests

Toutes les fonctions ont pu être testées mis à part *GetFirstBestWord* qui récupère le premier meilleur mot.

— A COMPLETER ...

6.2 Complexité

6.2.1 Solveur basé sur la première stratégie avec l'entropie

Pour le calcul de complexité des fonctions, on note N le nombre de mots dans le dictionnaire. On considère une longueur de mot n fixée, comprise entre 3 et 12 (alors, si la complexité d'une fonction dépend de n, on la considère

comme étant constante).

On détermine la complexité de la fonction *compute_bestGuess* dans le pire des cas. On effectue un parcourt exhaustif de la liste de mots, les opérations suivantes sont donc effectuées N fois :

- un appel à la fonction *compute_entropy* (de complexité linéaire en N)
- si $E_{max} < E$: un appel à la fonction *strcpy* et deux affectations (complexités constantes)
- une addition (complexité constante)

Ainsi, la complexité de cette fonction est quadratique.

6.2.2 Solveur basé sur la deuxième stratégie avec l'entropie

L'étude de la compléxité n'a pas pu être menée à temps.

6.2.3 Solveur basé sur la fréquence des lettres et des bigrammes

L'étude de la compléxité n'a pas pu être menée à temps.

6.2.3.1 Complexité de builTree

Tout d'abord, calculons la complexité temporelle de notre algorithme de construction d'arbres. Soit n la longueur du mot considéré et la longueur de la chaîne extraite en cours de traitement. On a :

- 16 affectations en début de code
- si $i_l = 1$ ou 2 : +3 affectations
- sinon :
 - + appel *isEven(i_l)*
 - + i_l
 - + appel *new_tree* (5 affectations)

Au final, on devra étudier la suite $u_n = 2u_{\lfloor n/2 \rfloor}$

6.3 Efficacité du solveur en nombre de tentatives

Chapitre 7

Bilan de projet

Contents

7.1	Bilan sur l'avancement du projet	43
7.1.1	Fonctionnalités implémentées	43
7.1.1.1	Partie Web	43
7.1.1.2	Partie C	43
7.1.2	Fonctionnalités non réalisées	43
7.1.2.1	Partie Web	43
7.1.2.2	Partie C	43
7.2	Bilan global du projet	44
7.2.1	Bilan de l'écriture du code	44
7.2.2	Bilan de la gestion de projet	44
7.3	Bilan du projet par membre	44
7.3.1	ASSING Norman	44
7.3.2	BIAUSQUE Anna	45
7.3.3	COUCHEVELLOU Clément	45
7.3.4	WANG Caroline	46

7.1 Bilan sur l'avancement du projet

7.1.1 Fonctionnalités implémentées

7.1.1.1 Partie Web

- Système de compte : création de compte, connexion, collection de récompense.
- Jeu entièrement fonctionnel, plusieurs paramètres modifiables pour les parties, historique des parties et affichage des statistiques
- Fonctionnalités ludiques autour du thème de Big Floppa : loterie, collecte d'images de différentes raretés, suppression de compte en cas de nombreuses défaites ...

7.1.1.2 Partie C

- Solveur basé sur l'entropie
- Solveur basé sur l'entropie et une liste de mots qui contient les solutions potentielles et une liste de mots qui diminue au mieux la première liste.
- Solveur basé sur une structure d'arbre, des listes doublement chaînées et des fréquences de bigrammes

7.1.2 Fonctionnalités non réalisées

7.1.2.1 Partie Web

- Statistiques avancées : Utilisateurs les plus actifs, statistiques par jour/semaine/mois, ...
- Créer un algorithme de machine learning, voir un réseau de neurone, afin de trouver une stratégie à partir des données des utilisateurs du site.

7.1.2.2 Partie C

- Implémenter plus de stratégies, dont des stratégies mises au point par les meilleurs joueurs.
- Mettre au point un algorithme performant de comparaison des solveurs, afin de trouver la meilleure stratégie pour les différentes tailles de mots.

7.2 Bilan global du projet

7.2.1 Bilan de l'écriture du code

- Expérience commune
- Progression en C et structures de données, mais aussi en Algorithmique et résolution de problèmes.
- Les points positifs
 - Une meilleure maîtrise de Flask, HTML, CSS, SQL
 - Un résultat assez satisfaisant par rapport à nos attentes initiales
 - Une bonne gestion de projet au départ qui a entraînée une productivité importante au début
- Les points négatifs
 - Les contraintes de temps liées au partiels qui ont été sous-estimées
 - Adaptation à la manière de coder de chacun - pas toujours évident de travailler à partir de code écrit par un autre membre

7.2.2 Bilan de la gestion de projet

- Des échanges quotidiens qui ont empêché l'effet tunnel, ainsi que des réunions régulières
- La gestion du temps difficile avec les partiels, mais encadrée grâce à différents outils.

7.3 Bilan du projet par membre

7.3.1 ASSING Norman

- Les points positifs
 - Amélioration de mes capacités en SASS/CSS
 - Meilleure compréhension du langage C
- Les difficultés rencontrées
 - Gestion du temps très difficile avec les partiels
 - Le déboggage en C qui n'est pas toujours simple
- L'expérience personnelle
 - J'ai trouvé ce projet particulièrement ludique et intéressant mais c'est un peu dommage que l'on ait pas eu l'occasion d'aller plus

loin sur le solveur à cause des partiels qui nous prenaient la plupart de notre temps sur la fin.

- Les axes d'amélioration
 - Si ce n'est la base de données qui aurait pu être mieux implémentée, je trouve que l'équipe a vraiment fait de son mieux pour ce projet et qu'il aurait difficilement pu être meilleur.
- Temps passé sur le projet
 - Plus de 150h

7.3.2 BIAUSQUE Anna

- Les points positifs
 - Gain d'expérience en CSS et C
 - Equipe-projet motivée et agréable
- Les difficultés rencontrées
 - Difficulté du langage C (compréhension de la mémoire avec les segmentation faults etc)
 - Gestion du temps avec les partiels
- L'expérience personnelle
 - Ce projet a été pour moi très enrichissant. Le premier P2I2 m'ayant déjà donné une certaine aisance quant au travail en groupe sur des projets de ce type, j'ai pu tirer pleinement parti de cette expérience, qui m'a notamment permis de gagner en autonomie et en maturité dans mes prises de décisions. Je pense que ce projet a été pour moi la meilleure manière de découvrir et de développer des compétences en C et SD.
- Les axes d'amélioration
 - La gestion du temps aurait toujours pu être meilleure mais je pense que l'équipe-projet a fait de son mieux, en considérant les périodes de partiels.
- Temps passé sur le projet
 - Plus de 150h

7.3.3 COUCHEVELLOU Clément

- Les points positifs
 - Amélioration de mon efficacité en programmation python
 - Nombreuses connaissances en C/SD acquises
- Les difficultés rencontrées
 - Manque de temps sur la fin
- L'expérience personnelle

- Ce projet était agréable, en grande partie grâce au reste de l'équipe, mais également pour les connaissances que j'ai acquises.
- Les axes d'amélioration
 - Mieux anticiper les périodes de manque de temps
 - Passer plus de temps à théoriser sur la BDD.
- Temps passé sur le projet
 - Plus de 150h

7.3.4 WANG Caroline

- Les points positifs
 - Approfondissement de connaissances en CSS
 - Développement de connaissances en C/SD
- Les difficultés rencontrées
 - Processus de création de la base de données
 - Débuts du code en C difficiles (cours et TP de C/SD pas encore passés)
- L'expérience personnelle
 - L'équipe a été dynamique le long du projet, les réunions régulières ont permis une avancée homogène.
 - L'implémentation du solveur m'a appris à effectuer des tests unitaires plus spontanément.
- Les axes d'amélioration
 - Être plus rigoureux sur la création des bases de données
- Temps passé sur le projet
 - Plus de 150h

Chapitre 8

Annexes

Contents

8.1	Comptes-rendus	48
8.2	Charte de projet	73

8.1 Comptes-rendus

CR Réunion n°1 (16/03/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- BIAUSQUE Anna
- COUCHEVELLOU Clément
- WANG Caroline

Ordre du jour :

- Présentation du projet
- Mail à O. Festor
- Résumés des projets du 1er semestre
- Calendrier

Présentation du projet

Norman commence par présenter le projet. La date de rendu est le 31 Mai.

La partie implémentation de Wordle représente 30% du projet, la partie solveur 50% et la partie GdP 20%.

Chacun doit contribuer aux sections suivantes :

- GdP
- Etat de l'art et compréhension d'algorithme
- Code en C
- Documentation

Mail à O. Festor

On a envoyé le mail à O. Festor.

Résumé des projets du 1er semestre

Projet Norman (et Clément) :

Le site est assimilable à Twitter avec l'utilisation de messages audios.

Ce qu'on en retient :

Il serait intéressant de faire un schéma des fonctionnalités (cf Drive). Au niveau GdP, nous aurons besoin d'un GANTT, matrice RACI. Il pourrait être pertinent d'utiliser Discord et Trello pour l'organisation et le séquencement des tâches.

Projet d'Anna :

Le site permet de répondre à un questionnaire politique et de comparer ses résultats avec ceux de candidats politiques donnés.

Ce qu'on en retient :

Il peut être judicieux de faire un profil projet au début de la partie GdP. Quant à l'implémentation de la base de données, il faut la créer théoriquement en 3nF au préalable (utiliser les notions de cours). Pour le design du site, il peut être intéressant de faire l'architecture des pages Web.

Projet de Caroline :

Site d'informations et de communication avec les candidats et leur liste pendant les campagnes municipales.

Ce qu'on en retient : La partie GdP et état de l'art du projet était très développée. Nous pourrons nous en inspirer pour celui-ci. Les réunions fréquentes sont nécessaires.

Diversité des compétences de chacun :

Norman :

-points forts : CSS et JavaScript, graphisme
-points faibles : motivation pour la partie compréhension d'algorithmes

Clément :

-points forts : Flask, Javascript et Python
-points faibles : GdP

Caroline :

-points forts : GdP, rigueur, motivation pour la partie compréhension d'algorithmes
-points faibles : C, BD

CR Réunion n°2 (22/03/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- BIAUSQUE Anna
- COUCHEVELLOU Clément
- WANG Caroline

Ordre du jour :

- Point sur l'avancement
- Répartition des tâches
- Calendrier
- Idée du siècle

Depuis la dernière fois :

Objectifs personnels :

-Anna : CR, profil projet, état de l'art
-Caroline : compréhension d'algorithmes, charte de projet
-Norman : état de l'art, voir ce qui est adaptable en GdP
-Clément : compréhension d'algorithmes, charte projet

Point sur l'avancement

Quid du dépôt Git ?

Norman prend la parole pour faire un point sur l'avancement. Le Wordle est quasiment fini, il manque quelques "détails" (clavier en CSS etc..) et à "connecter" la partie python de Clément et le JS/CSS de Norman.

Il a créé un chemin /reload pour actualiser directement le site à partir des mises à jour du dépôt Git. La discussion s'oriente sur le panel des lettres que l'on devrait proposer à l'utiliser. Nous décidons de proposer des mots de 2 à 25 lettres.

La liste des mots à implémenter en BD se trouve à l'URL suivant : <http://www.lexique.org>

Au niveau gestion de projet, Caroline et Clément ont commencé la rédaction de la charte de projet, et Norman celle de l'état de l'art. Le profil projet est fini.

Caroline a commencé à regarder des vidéos sur la partie solveur et a pris des notes sur celles-ci.

Il est évoqué qu'il faudrait formaliser les schémas de la base de données. Caroline se propose de le faire avec l'aide de Clément qui lui expliquera la structure de celle-ci.

Le débat s'oriente vers les tests et leurs utilité dans les différentes parties du projet. On en conclut qu'ils seront le plus utile dans la partie solveur.

Répartition des tâches

Norman présente l'excel de répartition des tâches. Chacun remplit les tâches qu'il a à faire et celles qu'il a déjà faites.

Calendrier

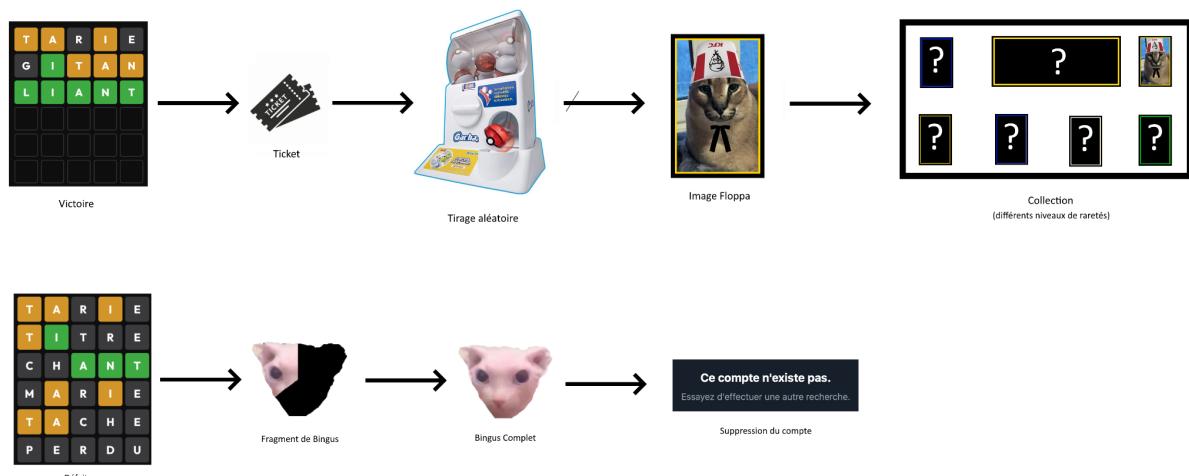
On fait un point sur la mise à jour du calendrier. La partie WEB est déjà bien avancée et peut donc être terminée avant les vacances.

On prévoit le rendu du document de conception avant le 31 Mars.

Quant aux fonctionnalités supplémentaires pour le Wordle, celles proposées par le sujet sont les suivantes : gestion des utilisateurs, sauvegarde des jeux par joueur et replays, récompenses, apprentissage de stratégies des joueurs.

Ce qui nous amène à l'idée de Norman quant à ces fonctionnalités bonus.

Idée du siècle



Avant la prochaine réunion :

Objectifs personnels :

- Anna : état de l'art, SWOT, clavier, vérification de l'existence des mots, CR
- Caroline : schéma BD
- Norman : document de conception
- Clément : actualiser la BD, document de conception

CR Réunion n°3 (05/04/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- BIAUSQUE Anna
- COUCHEVELLOU Clément
- WANG Caroline

Ordre du jour :

- Document de conception
- Calendrier
- Solveur

Depuis la dernière fois :

Objectifs personnels :

-Anna : état de l'art (fait), SWOT (fait), clavier (en cours), vérification de l'existence des mots (fait), CR (fait)

-Caroline : schéma BD (description textuelle faite, table pour les statistiques des parties faite, réécriture des tables en 3NF en cours, schéma non fait)

-Norman : document de conception

-Clément : actualiser la BD, document de conception

Document de conception

Nous commençons par faire un point sur le document de conception. L'état de l'art est terminé. Pour la suite du document, il faut le structurer en différentes parties (base de données, web, fonctionnalités, mockups..) en se référant au sujet.

Il reste également à mettre à jour la charte-projet.

Calendrier

Quelques changements sont apparus dans le calendrier : le document de conception doit être rendu au plus vite. On se donne jusqu'à la semaine prochaine pour le finaliser.

Il serait idéal que d'ici le milieu des vacances le Wordle soit finalisé sans les fonctionnalités supplémentaires, afin de pouvoir avancer sur la partie solveur.

Solveur

Clément souhaite parler des différentes stratégies de solving du Wordle : l'idéal serait d'énumérer les stratégies possibles, de les comparer et d'implémenter la meilleure. Elles devront toutes être mentionnées sur le document de conception.

Avant la prochaine réunion :

Objectifs personnels :

- Anna : clavier, document de conception, CR, mockups
- Caroline : réécriture et schéma des tables en 3NF, description de l'algorithme du solveur.
- Norman : Finalisation de la partie CSS/Javascript, document de conception Wordle, relecture de l'état de l'art
- Clément : Avancer le document de conception, finalisation de la partie Flask/Javascript, avancer sur les structures de données en C.

CR Réunion n°4 (12/04/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- BIAUSQUE Anna
- COUCHEVELLOU Clément
- WANG Caroline

Ordre du jour :

- Réunion d'urgence
- To-do list

Réunion d'urgence

Norman a demandé à ce qu'on fasse une réunion d'urgence. Il est inquiet quant au temps qu'il nous reste pour la réalisation du projet. A partir de la rentrée, nous avons des partiels chaque semaine, notamment le 31 Mai, date de rendu du projet. On doit donc s'organiser afin de faire le maximum pendant les vacances.

Etant donné que nous n'avons pas une idée concrète du temps qu'il nous faudra pour la partie C, nous devons prioriser le solveur et nous verrons après pour les fonctionnalités supplémentaires du Wordle.

Quant au document de conception, il doit être rendu le plus rapidement possible : nous allons donc séparer la partie Web et C dans sa rédaction. Il faudrait le rendre dans l'idéal demain soir (en ayant réglé les problèmes avec les images et les schémas de BDD) et terminer les fonctionnalités essentielles d'ici jeudi (la liste des fonctionnalités essentielles restantes est à la racine du drive). Pour les structures de données, nos connaissances ne sont pas suffisantes pour l'instant pour pouvoir faire le solveur dans sa globalité. La priorité sera donc de rendre un code facilement lisible pour pouvoir le reprendre dans un mois si besoin.

To-do list

Document de conception:

- schémas 3NF (Caroline)
- description de la BD (Caroline)
- mise en page (images etc) (Anna)

Web :

- statistiques de joueurs (Clément/Caroline)
- correction des bugs graphiques JS (Norman)
- raccord Python/JS

CR Réunion n°5 (13/04/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- BIAUSQUE Anna
- WANG Caroline

Membre(s) absent(s) :

- COUCHEVELLOU Clément

Ordre du jour :

- Document de conception
- Web
- C
- Point réunions et avancement

Depuis la dernière fois :

Objectifs personnels :

- Anna : clavier (fait), document de conception (fait), CR (fait), mockups (fait)
- Caroline : réécriture et schéma des tables en 3NF pour le document de conception (fait), description de l'algorithme du solveur (en cours), affichage statistiques du joueur (en cours).
- Norman : Finalisation de la partie CSS/Javascript (fait), document de conception Wordle (fait), relecture de l'état de l'art (fait)
- Clément : Avancer le document de conception, finalisation de la partie Flask/Javascript, avancer sur les structures de données en C.

Document de conception

Norman : La partie sur les fonctionnalités supplémentaires a été incluse.

Anna : Les mockups ont été inclus et les problèmes avec les images réglés.

Caroline : Les schémas de BDD et la description des tables avec la liste des contraintes ont été inclus.

On décide de rendre le document à la fin de la réunion.

Web

Norman a fait les fonctions en JS pour que tous les mots soient bien affichés et celle qui colore les lettres et le clavier, il attend Clément pour relier le JS au Python.

Caroline s'occupe des statistiques de résultats. Elle a trouvé une fonction "toute faite" pour les statistiques et se demande s'il est judicieux de l'utiliser ou s'il vaut mieux tout refaire "à la main".

Etant donné que les statistiques sont une fonctionnalité supplémentaire, cela ne nous semble pas

dérangeant d'utiliser la fonction "toute faite", d'autant plus que le design est intéressant. Sinon, elle attend le raccord de Clément entre le JS et Python, et évoque la possibilité d'un menu déroulant pour choisir le nombre de lettres et d'essais pour accéder aux statistiques.

Anna attend également le raccord pour pouvoir intégrer le dictionnaire à la partie Web.

C

Etant donné notre peu de connaissances en SD, la pré-implémentation peut aller vite. Pour l'instant, nous ne pouvons utiliser que les structures de liste.

Point réunions et avancement

Réunions :

- demain à 15h sur le C et bilan du Python
- samedi matin à 10h
- mardi prochain à 14h
- jeudi prochain

Avant les deux prochaines réunions :

Objectifs généraux :

- relire les cours de C, les éventuels algorithmes

Objectifs personnels :

- Anna : dictionnaire, CR
- Caroline : affichage des statistiques
- Norman : régler les bugs graphiques avec Clément
- Clément : finir les modifications de l'application pour utiliser plus de BD, fonctions de stats, structures de données en C

CR Réunion n°6 (14/04/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
 - COUCHEVELLOU Clément
 - BIAUSQUE Anna
 - WANG Caroline

Ordre du jour :

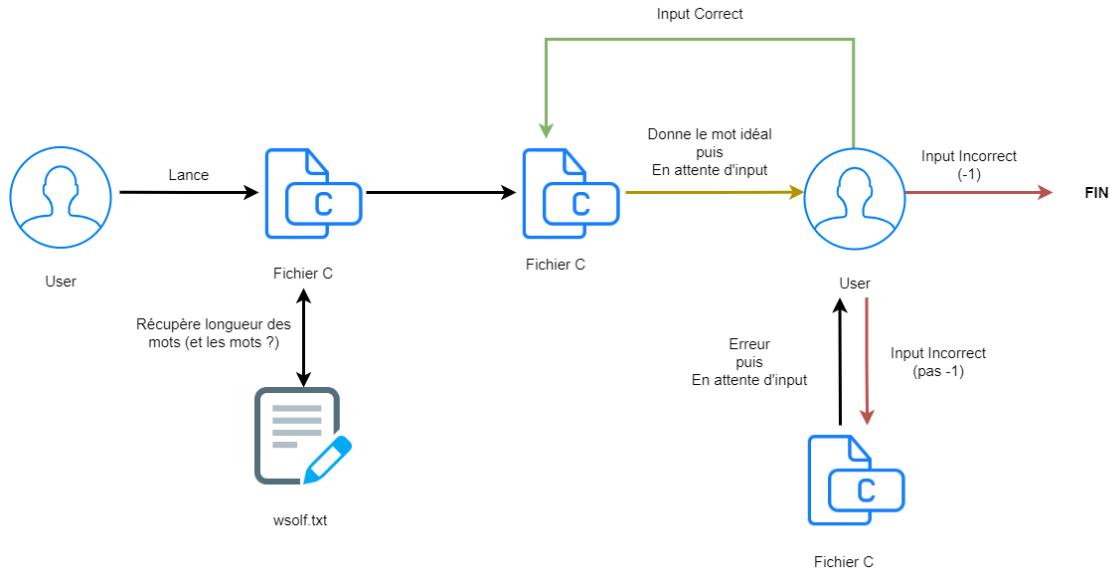
- Web
 - C

Web

Nous nous répartissons les tâches restantes pour la partie Web. Clément va push ce qu'il a fait, nous notons les tâches déjà faites et faisons un point sur ce qui a été dit à la réunion précédente. Les tâches qui ne sont pas attribuées seront réalisées plus tard.

- Empêcher les joueurs pas connectés de jouer
 - Il faut sauvegarder les essais précédents
 - Il ne faut pas les sauvegarder dans session mais dans la BDD :
 - Les envoyer dans session
 - Les envoyer dans la BDD
 - pop les infos de la session
 - Bug quand on joue une nouvelle partie mais que y'a déjà des infos dans la session
 - On devrait pas pouvoir lancer une nouvelle partie si on a pas fini la dernière
 - Ajout du dictionnaire au jeu (Anna)
 - Restreindre les mots à des mots qui existent
 - Page index -> Bouton Jouer
 - Une partie en cours ? -> Retour à la partie
 - Pas de partie en cours ? -> Choix du nombre de lettres et d'essais

C



Au niveau de la flèche jaune se situe le “plus gros” du projet. Le reste devrait être assez rapide à implémenter.

D’ici samedi, Anna finit la bibliothèque, Norman s’occupe de la partie “input” et Clément et Caroline avancent sur la partie “mot idéal”.

Avant la prochaine réunion :

Objectifs personnels :

- Anna : dictionnaire, CR
- Caroline : mot idéal
- Norman : input en C
- Clément : finir les modifications de l’application pour utiliser plus de BD, mot idéal

CR Réunion n°7 (16/04/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- COUCHEVELLOU Clément
- BIAUSQUE Anna
- WANG Caroline

Ordre du jour :

- Web
- C

Depuis la dernière fois :

Objectifs personnels :

- Anna : dictionnaire (non fait)
- Caroline : solveur (mot idéal) (pour la stratégie avec l'entropie : liste des fonctions principales à coder, pseudo-code)
- Norman : solveur (input) (fait mais à debugger), web (intégration de la partie JS en cours)
- Clément : solveur (mot idéal), web (correction des bugs etc) (fait)

Web

Clément explique les modifications qu'il a faites sur la DB, notamment la suppression de la modalité "perdu" pour une partie.

Il reste l'intégration de la partie JS à debugger (Norman) et le dictionnaire (Anna).

C

Clément n'a pas eu le temps d'avancer sur la partie C. Caroline nous explique ce qu'elle a fait.

On utilise l'entropie (probabilité d'obtenir un certain motif pour un mot donné) et on calcule l'information qu'on peut obtenir pour ce motif là.

Les fonctions principales sont déterminées et elle a commencé le pseudo-code.

Caroline consulte le reste de l'équipe-projet quant à une idée qu'elle a eue : au fur et à mesure de l'élimination de certains mots dans la recherche de la solution, on pourrait attribuer à un mot 0 ou 1 en fonction de si oui ou non on le garde. Faudrait-il l'écrire dans le fichier txt ou plutôt créer des tuples ? L'option des tuples semble plus judicieuse mais nous allons y réfléchir.

Avant la prochaine réunion (19/04 à 14h) :

Objectifs personnels :

- Anna : dictionnaire, CR, existence des mots
- Caroline : débuter l'implémentation pour le mot idéal, document de conception
- Norman : Débogage C + Web
- Clément : Avancer dans la partie C + finaliser l'application web essentielle

CR Réunion n°8 (19/04/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- COUCHEVELLOU Clément
- BIAUSQUE Anna
- WANG Caroline

Ordre du jour :

- Web
- C

Depuis la dernière fois :

Objectifs personnels :

- Anna : dictionnaire (fait), CR (fait), existence des mots (fait)
- Caroline : implémentation pour le mot idéal (en cours), document de conception (section sur l'algorithme avec entropie : faite)
- Norman : Débogage C (fait) + Web (fait + fonctionnalités suppl faites)
- Clément : Avancer dans la partie C (pas fait) + finaliser l'application web essentielle (fait)

Web

L'intégration du dictionnaire dans la DB est terminée, ainsi que la fonction de vérification de l'existence des mots entrés par l'utilisateur. Anna a push ces modifications sur une branche, il reste à merge.

Les fonctionnalités "Galerie" et "Fragments de Bingus" ont été rajoutées par Norman.

Quant à la page galerie : elle est pour l'instant vide, donc il faut rajouter les images, les flèches pour naviguer etc..

Pour les fragments de Bingus, l'image est générée avec Jinja.

En ce qui concerne l'affichage des statistiques de jeu : Caroline a tout mis dans une nouvelle branche.

Il reste à vérifier que l'affichage du graphe se fait correctement, en remplaçant un peu la DB.

Elle a rencontré les problèmes suivants : le clavier se superpose au pop up et l'icône de statistiques ne s'affiche pas quand on passe par /tryWord.

C

Beaucoup d'informations essentielles se trouvent dans le document Notes pour l'algorithme.

Caroline a implémenté la plupart des fonctions essentielles. Elle évoque l'option de faire 1 dictionnaire pour chaque longueur de mots : ça serait l'idéal.

Elle a également commencé les tests (compilation de dictionary.c notamment) et rencontre certains problèmes, notamment le message "leak de bits", mais ne sait pas à quoi ça correspond. Norman, Clément et Anna essaieront de l'aider à résoudre ces bugs et comprendre les warnings.

Il faut également tester les autres modules mais ceux-ci dépendent les uns des autres : ceux sur dictionary doivent être terminés au préalable.

Quid du doc de conception en C ? Caroline a ajouté une section avec son algorithme.

Avant la prochaine réunion (jeudi 21 à 15h):

La démo étant le 30 Avril, Anna, Norman et Clément vont se concentrer sur l'implémentation du Wordle jusqu'à la prochaine réunion, et aider Caroline pour les bugs. Caroline continue avec le C.

Norman :

- affichage des stats (régler le bug du clavier + bouton stat)
- ~~Répertorier les assets graphiques (machine Floppa, Floppa coin ...)~~
- Galerie (JS : Flèches suivantes / précédentes)

Clément :

- Flask, implémentation fonctionnalités supplémentaires
- Parties perdues

Caroline :

- affichage des stats : graphique de distribution
- mot idéal (C)

Anna :

- Loterie HTML
- Loterie CSS
- Galerie
- CR

CR Réunion n°9 (21/04/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- COUCHEVELLOU Clément
- BIAUSQUE Anna
- WANG Caroline

Ordre du jour :

- Web
- C

Depuis la dernière fois :

Norman :

- affichage des stats (régler le bug du clavier + bouton stat)
- Répertorier les assets graphiques (machine Floppa, Floppa coin ...)
- Galerie (JS : Flèches suivantes / précédentes)

Clément :

- Flask, implémentation fonctionnalités supplémentaires (commencé)
- Parties perdues
- Correction et optimisation de l'appli web

Caroline :

- affichage des stats : graphique de distribution (en attente de pouvoir gagner/perdre des parties)
- Problème ieône bouton stats
- mot idéal (C)

Anna :

- Loterie HTML
- Loterie CSS (en cours)
- Galerie (en cours)
- CR

Web

On fait le point sur ce qu'il nous reste à faire dans la partie Web :

- galerie : le JS pour l'animation est terminé, il reste à mettre les images et les autres éléments. Clément s'en charge.
- loterie : le HTML est terminé, il reste le CSS à finaliser.
- il faut afficher dans la navbar le nombre de Floppa Coins que possède le joueur. Norman s'en charge.

Clément parle des bugs qu'il a rencontré lors de l'implémentation des fonctionnalités supplémentaires et des changements qu'il a dû effectuer pour les régler.

Au niveau du dictionnaire, Anna a rencontré des difficultés avec les accents : elle a changé la liste de mots en une liste ne comportant initialement pas d'accents.

Anna demande des précisions quant à la loterie et aux valeurs données aux images.

Caroline a réglé les bugs graphiques des statistiques : tout fonctionne, il ne reste qu'à tester l'affichage graphique en faisant plusieurs parties, ce qu'elle n'a pas pu faire à cause des bugs du jeu.

To do list Web :

- afficher le nombre de Floppa coins de l'utilisateur
- ajouter images etc.. dans la galerie
- affichage des statistiques (ajouter dans le code de la branche master et jouer des parties pour tester le graphique)
- gagner des floppa coins quand on gagne des parties
- gagner des Bingus fragments quand on perd
- régler bugs du chargement infini + couleurs

C

Caroline a fini les tests unitaires et réglé tous les bugs. Elle réorganise son code afin de le rendre plus lisible. Clément, Norman et Anna vont lire son code et se concentrer sur leurs solveurs respectifs.

Avant la prochaine réunion (lundi 25 à 13h):

Norman :

- afficher les floppa coins
- galerie (grid etc...)
- Corriger CSS revenir à l'écran d'accueil (bug)

Caroline :

- test affichage des stats (ajouter dans le code de la branche master et jouer des parties pour tester le graphique)
- organiser les modules du solveur

Clément :

- implémenter solveur
- commenter le code
- flask/python derrière la galerie/floppa coins
- chasse au bugs en web

Anna :

- gagner des floppa coins quand on gagne des parties
- gagner des Bingus fragments quand on perd
- implémenter solveur
- commenter le code
- changer les probabilités de la loterie

CR Réunion n°10 (25/04/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- COUCHEVELLOU Clément
- BIAUSQUE Anna
- WANG Caroline

Ordre du jour :

- Web

Depuis la dernière fois :

Norman :

- afficher les floppa coins
- galerie (grid etc...) (en cours)
- Corriger CSS revenir à l'écran d'accueil (bug)

Caroline :

- test affichage des stats (ajouter dans le code de la branche master et jouer des parties pour tester le graphique)
- organiser les modules du solveur

Clément :

- implémenter solveur
- commenter le code
- flask/python derrière la galerie/floppa coins (en cours)
- chasse au bugs en web (en cours)

Anna :

- gagner des floppa coins quand on gagne des parties
- gagner des Bingus fragments quand on perd
- implémenter solveur
- commenter le code
- changer loterie probas (en cours)

Web

Il reste à raccorder la partie probabilités de la loterie avec le reste de l'algorithme.

Nous avons constaté un bug : dès que la partie est terminée, le site plante. Le problème se situe au niveau de la fonction nbTries : Caroline et Clément s'en chargent.

Anna demande l'aide de Norman pour finir le CSS de la partie loterie.

Pour la galerie, il faut mettre toutes les images à la main, avec des images noires pour celles non débloquées. Il faut également raccorder le Flask. Anna s'en charge.

Loterie :

- raccord le Flask
- css

Galerie:

- placer les images
- raccorder le Flask

Avant la prochaine réunion (lundi 25 à 18h):

Anna :

- raccorder le Flask loterie / galerie
- CR

Clément :

- raccorder le Flask loterie
- C / aider Norman

Caroline :

- raccorder le C

Norman :

- CSS loterie
- galerie

CR Réunion n°11 (27/04/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- COUCHEVELLOU Clément
- WANG Caroline

Membre absent :

- BIAUSQUE Anna

Ordre du jour :

- Web
- Soutenance

Depuis la dernière fois :

Anna :

- raccorder le flask loterie
- raccorder le flask galerie
- CR

Clément :

- raccorder le flask loterie
- C

Caroline :

- raccorder C

Norman :

- CSS loterie
- galerie

Web

Les bugs de la dernière fois sont réglés. On fait la To Do List globale avant la soutenance.

Prochaine réunion : jeudi 18H

Norman:

- Lien vers la Page Gallery
- Rendre les phrases du site plus cohérentes
- Ajuster la charte graphique du site
- Rendre la Loterie plus jolie

- Animation de la loterie
- Ajouter des sons sur le site

Anna:

- Gallery relier au Flask

Caroline:

- Remplir Gallery
- Vérifier ce qu'il se passe quand on utilise une seule image noire
- Bouton Statistiques à re-regarder (changements des deux menus déroulant pour un seul)

Clément:

- Bouton Statistiques à re-regarder
- Destruction du compte
- Boutons pour la taille des mots / nbr d'essais (menu déroulant car buttons trop complexe)

Fonctionnalités essentielles restantes

- Mettre des images noir avant de débloquer les images

Fonctionnalités possibles restantes

- Page index -> Bouton Jouer
- Une partie en cours ? -> Retour à la partie
- Partager les games (en texte)
- Replays

Ajouts mineurs restants

- Rendre la Gallery plus jolie
- Adapter CSS Pour Demi Ecran / Mobile

CR Réunion n°12 (29/04/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- COUCHEVELLOU Clément
- BIAUSQUE Anna
- WANG Caroline

Ordre du jour :

- Présentation
- ToDo list

Depuis la dernière fois :

Norman:

- Lien vers la Page Gallery
- Rendre les phrases du site un peu mieux
- Modifier texte : Rejoignez Kleps
- Enlever text-decoration de Perdu chez
- Couleur de texte sur les différentes pages contraste pas ouf
- Rendre la Loterie plus jolie
- Animation loterie
- Ajouter des sons sur le site

Anna:

- Gallery relier au Flask

Caroline:

- Remplir Gallery
- Vérifier ce qu'il se passe quand on utilise une seule image noire
- Bouton Statistiques à re-regarder (changements des deux menus déroulant pour un seul)

Clément:

- Bouton Statistiques à re-regarder
- Destruction du compte
- Boutons pour la taille des mots / nbr d'essais (menu déroulant car buttons trop complexe)

Présentation

On fait un point sur la soutenance. Chacun doit être certain de s'être approprié le app.py.
On se répartit les points à aborder pendant l'oral : Anna explique le thème du site et Norman ses fonctionnalités. Norman amène son PC pour la présentation.

ToDo List

Norman :

- Animation 3 fragments de bingus
- CSS pour game.html
- fonctionnalité "partager les games en txt"

Anna :

- CSS ajouter un mot au dico

Clément :

- page replay qui reste à faire

Caroline :

- remplacer le logo Kleps (pages connexion et créer un compte)

CR Réunion n°13 (05/05/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- BIAUSQUE Anna
- COUCHEVELLOU Clément
- WANG Caroline

Ordre du jour :

- C
- ToDo list

C

Clément a push ses avancées sur le solveur sur le Git.

Norman pense qu'il aura bien avancé d'ici la semaine prochaine.

Anna a travaillé sur sa stratégie, en travaillant avec la fréquence des lettres et des bigrammes.

Il reste à Caroline à raccorder son solveur avec l'interface faite par Norman.

La discussion s'oriente sur les tests et performances. Il nous semble compliqué de faire des tests qui conviennent pour tous les solveurs, il nous apparaît donc plus judicieux de faire des tests de performances (en temps/mémoire). Il faudra faire attention car les tests de performance temporelle dépendent de la machine utilisée. Il faudra également travailler sur la complexité.

Nous aurons au final 3 ou 4 solveurs avec différentes stratégies.

ToDo (Prochaine réu : Mercredi 11/05)

Norman :

- terminer sa version et raccorder avec l'interface

Caroline :

- raccorder version avec l'interface

Clément :

- avancer sur sa version
- analyse des structures de données dans les versions déjà implémentées

Anna :

- travailler sur sa version
- aider Caroline pour le raccord

CR Réunion n°14 (11/05/2022)

Secrétaire :

- BIAUSQUE Anna

Membre(s) présent(s) :

- ASSING Norman
- BIAUSQUE Anna
- COUCHEVELLOU Clément
- WANG Caroline

Ordre du jour :

- C
- GP
- ToDo list

C

Chacun fait un point sur son avancée.

- Norman : les fonctions qu'il lui reste à implémenter sont les suivantes : score d'un mot, raccord avec l'interface. Il doit ensuite traduire son pseudo-code.
- Caroline : elle a fini les fonctions de raccord avec l'interface, il reste les tests à faire, qui seront fait d'ici la prochaine réunion.
- Anna et Clément (qui mettent en commun leurs stratégies et leurs solveurs) :
 - ❖ Anna : structures sont implémentées, les fonctions de navigation dans les arbres et de construction sont en cours. Il reste à faire le débogage, les tests et les fonctions de calcul des fréquences.
 - ❖ Clément : il a implémenté les dictionnaires en listes doublement chaînées. Les fonctions sont faites, le traitement des patterns est en cours.
Il faudra raccorder les travaux d'Anna et Clément, et regarder la vidéo d'explication de Norman quant à l'interface.

On soulève le problème du temps qu'il nous reste et des partiels à venir (SIC, Matlab, TP MSED) : nous avons conscience que nous ne pourrons avancer autant qu'on le voudrait d'ici la semaine prochaine.

Nous pensons commencer les tests après le partielle de MO et faire le maximum au niveau GdP et implémentation d'ici là.

GdP

On fait le point sur ce qu'il nous reste à faire en GdP. Caroline parle du WBS, qui lui avait été utile lors du projet précédent. Le GANTT reste également à formaliser (un GANTT par partie).

To do list

Chacun doit avancer comme il le peut sur l'implémentation de son solveur.

8.2 Charte de projet

CHARTE DE PROJET

Projet	Projet Pluridisciplinaire d'Informatique Intégrative 2 - Flopple
Chef de projet	Norman Assing

Historique		
Date	Auteur	Description
16.03.2022	Clément Couchevellou	Premier jet du document
17.03.2022	Caroline Wang	Ajout de contenu et mise en page
31.03.2022	Anna Biausque	Mise à jour de la matrice SWOT

TABLE DES MATIERES

I.	Cadrage.....	1
a.	Présentation et motivation du projet.....	1
b.	Livrables attendus.....	2
II.	Déroulement du projet.....	2
a.	Principaux jalons.	2
b.	Forces et opportunités.....	3
c.	Chef de projet.....	3
d.	Parties prenantes du projet.....	3

CADRAGE

PRESENTATION ET MOTIVATION DU PROJET

Ce projet s'inscrit dans le cadre du module du projet pluridisciplinaire d'informatique intégrative.

Il est centré sur le jeu WORDLE, dont le principe est le suivant : l'ordinateur cache un mot du dictionnaire d'une longueur donnée et le but du joueur est de trouver ce mot avant d'avoir épuisé son nombre d'essais. Si je joueur parvient à deviner le mot avant d'atteindre la limite maximale de son nombre d'essais, il gagne et sinon, il perd.

L'idée est d'en développer une instance, ainsi qu'un solveur WORDLE.

D'une part, l'objectif est de créer une application Web interactive qui reproduira le fonctionnement du jeu. Elle doit être réalisée sur une architecture Python/Web/Base de données et sauvegarder les données des parties jouées. Le joueur doit également pouvoir paramétriser la longueur des mots et le nombre maximal d'essais possibles.

D'autre part, le solveur WORDLE doit être implémenté exclusivement en langage C et utiliser les structures de données maximisant l'efficacité de la résolution. Il doit en outre proposer un mode d'exécution pas à pas pour suivre la résolution.

LIVRABLES ATTENDUS

- Un état de l'art des applications de jeu similaires à WORDLE et des algorithmes de résolution.
- Une documentation de conception comprenant une présentation du jeu WORDLE, une présentation textuelle et détaillée de l'instance du jeu et de son solveur et une description précise et argumentée des structures de données utilisées par le solveur. Ce document de conception sera compris dans une documentation plus complète.
- Un serveur Flask composé d'un ou plusieurs fichiers Python et des templates html mis en page grâce à des fichiers CSS, le tout en communication avec une base de données SQL.
- Un algorithme de résolution implémenté en langage C et utilisant des structures de données permettant d'optimiser la résolution en terme de mémoire et/ou de temps (...)
- Un rapport de projet rédigé en LaTeX synthétisant le travail de conception et d'implémentation des applications et consacrant notamment une partie aux tests et performances, ainsi d'une section décrivant la gestion de projet effectuée par le groupe (matrice SWOT, matrice RACI, diagramme de Gantt...).
- Une présentation en vue de la soutenance orale.
- Une démonstration des produits du projet.

DÉROULEMENT DU PROJET

PRINCIPAUX JALONS

Jalon	Echéance	Description
Etape 1 : étude préalable du sujet	05.04.2022	Etat de l'art et conception des applications (avec validation du document de conception par les encadrants)
Etape 2 : organisation du travail	10.04.2022	Préparation de la phase d'implémentation à l'aide des outils de gestion de projet (charte de projet, matrice SWOT/RACI, WBS, diagramme de Gantt)
Etape 3 : implémentation de l'application WORDLE	30.05.2022	Réalisation de l'instance du jeu par implémentation de l'application Web WORDLE
Etape 4 : implémentation du solveur WORDLE	29.05.2022	Réalisation du solveur WORDLE
Etape 5 : rédaction du rapport	31.05.2022	Synthèse du travail effectué précédemment
Etape 6 : préparation à la soutenance	10.06.2022	Réalisation de la présentation et des démonstrations, concentration sur le solveur

RESSOURCES

Temps limité, fermeture du GitLab le 31 Mai 2022, 23h00 CEST

FORCES ET OPPORTUNITES

	Positif	Négatif
Origine interne	<p>Forces :</p> <ul style="list-style-type: none"> • Équipe-projet complémentaire • Bonne entente de l'équipe-projet • Expérience dans l'utilisation des outils de gestion de projet 	<p>Faiblesses :</p> <ul style="list-style-type: none"> • Utilisation de grid en CSS, malgré le peu de connaissances • Peu d'expérience en travail de groupe
Origine externe	<p>Opportunités :</p> <ul style="list-style-type: none"> • Période de vacances d'Avril • Disponibilité des professeurs en cas de besoin 	<p>Menaces :</p> <ul style="list-style-type: none"> • Périodes de partiels • Apprentissage du C en parallèle de l'avancement du projet

CHEF DE PROJET

Norman Assing a été désigné chef de projet.

PARTIES PRENANTES

Equipe projet
Norman Assing
Anna Biausque
Clément Couchevellou
Caroline Wang

Parties prenantes	Responsabilité	Influence sur le projet
Prénom Nom		
Olivier Festor	Responsable du module	Elevée

