



TELECOM Nancy

Projet P2I2

---

MOZA-RELA

---

CHANEL Valentin  
DEMANGEON Matéo  
FRACHE Nicolas  
VOISIN Victor

Responsable de module :  
FESTOR Olivier



# Table des matières

<b>1</b>	<b>Introduction au sujet</b>	<b>4</b>
1.1	Contexte du sujet . . . . .	4
1.2	Objectif . . . . .	4
1.3	Présentation de MOZA-RELA . . . . .	4
<b>2</b>	<b>État de l'art</b>	<b>5</b>
2.1	Principes de l'application WORDLE . . . . .	5
2.2	Étude des applications type WORDLE . . . . .	5
2.3	Étude des solveurs de WORDLE . . . . .	6
<b>3</b>	<b>Conception et implémentation de MOZA : Description fonctionnelle de l'ap- plication</b>	<b>7</b>
3.1	Liste des fonctionnalités du système . . . . .	7
3.2	Cas d'utilisation et descriptions textuelles . . . . .	7
3.3	Structure du site et maquettes . . . . .	8
<b>4</b>	<b>Conception et implémentation de MOZA : Mise en place technique</b>	<b>11</b>
4.1	Modèle relationnel de la base de données . . . . .	11
4.2	Interaction entre les couches de l'application . . . . .	12
4.2.1	Interaction serveur/client . . . . .	12
4.2.2	Dictionnaire . . . . .	12
<b>5</b>	<b>Conception et implémentation de RELA : algorithme de résolution</b>	<b>14</b>
5.1	Idée générale . . . . .	14
5.1.1	Principe de l'algorithme . . . . .	14
5.1.2	Quantification des informations apportées par une proposition . . . . .	14
5.1.3	Système de contraintes . . . . .	15
5.1.4	Boucle principale . . . . .	15
5.2	Structure de données . . . . .	16
<b>6</b>	<b>Tests et performances</b>	<b>17</b>
6.1	Tests sur les algorithmes . . . . .	17
6.2	Performances . . . . .	18
6.2.1	Calcul de complexités . . . . .	18
6.2.2	Temps d'exécution . . . . .	19
6.2.3	Nombre moyen de coups . . . . .	19
<b>7</b>	<b>Gestion de Projet</b>	<b>21</b>
7.1	Composition de l'équipe du projet . . . . .	21
7.2	Organisation . . . . .	21
7.3	Analyse de l'environnement - matrice SWOT . . . . .	21
7.4	Diagramme de Gantt . . . . .	22
7.5	Comptes-rendus des réunions . . . . .	22
7.5.1	16 Mars 2022 . . . . .	22
7.5.2	23 Mars 2022 . . . . .	23
7.5.3	20 avril 2022 . . . . .	24
7.5.4	9 mai 2022 . . . . .	25
7.5.5	24 mai 2022 . . . . .	26
7.5.6	07 juin 2022 . . . . .	27

<b>8</b>	<b>Bilan du projet</b>	<b>28</b>
8.1	Bilan global du projet . . . . .	28
8.2	Bilan du projet membre par membre . . . . .	28
8.2.1	VOISIN Victor . . . . .	28
8.2.2	CHANEL Valentin . . . . .	28
8.2.3	DEMANGEON Matéo . . . . .	29
8.2.4	FRACHE Nicolas . . . . .	29
8.3	Travail réalisé . . . . .	29

# Chapitre 1

## Introduction au sujet

### 1.1 Contexte du sujet

En ce début 2022, le monde du jeu numérique a été marqué par plusieurs annonces majeures dont l’acquisition par Microsoft d’Activision Blizzard pour près de 70 milliards de dollars. Cependant, la nouvelle qui a fait le plus de buzz, est l’acquisition par le New York Times du célèbre jeu WORDLE. Cette acquisition a généré une question existentielle pour des millions de joueurs quotidiens : WORDLE va-t-il rester libre ?

### 1.2 Objectif

Notre objectif est de sauver le monde des joueurs WORDLE en élaborant notre propre instance du jeu, mais aussi un solver WORDLE, i.e. un programme qui se prend pour un utilisateur et qui résout les puzzles WORDLE.

### 1.3 Présentation de MOZA-RELA

- MOZA : est notre application WORDLE de base. Elle est réalisée sur une architecture Python/Web/Base de données. Elle est paramétrable par le joueur (longueur des mots, nombre maximal d’essais, difficulté des mots), et les parties jouées sont sauvegardées dans une base de données.
- RELA : est notre solver WORDLE. Il est implémenté exclusivement en langage C, utilise des structures de données avancées pour être le plus efficace dans la résolution et dispose d’un mode interactif permettant de suivre son cheminement à l’exécution en pas à pas.

## Chapitre 2

# État de l’art

### 2.1 Principes de l’application WORDLE

Dans l’application WORDLE, l’objectif est de trouver un mot du dictionnaire caché par l’ordinateur. Pour cela, le joueur saisit à chaque étape un mot de la même longueur que le mot à trouver.

En réponse à cette proposition, l’ordinateur indique pour chaque lettre saisie :

- si elle est dans le mot recherché et si elle est à la bonne place,
- si elle est dans le mot recherché mais pas à place que vous avez choisie,
- si elle n’est pas dans le mot recherché.

Tout cela par l’intermédiaire de couleurs joyeuses.

Si l’on trouve le mot, le jeu s’arrête.

Si l’on ne pas trouve le mot, on peut en saisir un nouveau jusqu’à ce que l’on atteigne le nombre maximal d’essais autorisés.

Si l’on dépasse le nombre maximal d’essais, la partie est perdue.

### 2.2 Étude des applications type WORDLE

Nous avons trouvé plusieurs applications du même principe que WORDLE. Voici un extrait d’une liste [1] de quelques une d’entre elles :

Noms	Avantages	Désavantages
Wordle	Simple à comprendre et jouer	Manque de fonctionnalités, mots de longueur 5 seulement.
Motus	Superbe émission télévisuelle. Les mots sont très compliqués.	C'est fini.
Sutom	Facile à prendre en main. Présence de son et de statistiques.	Comme Worlde, on a aussi vite fait le tour.
Dordle	Présence de statistiques plus complètes, difficulté élevée	Plus compliqué à prendre en main, visuel moins intuitif.
Polydle	Modes de jeu différents, visuel intuitif.	Charte graphique discutable.
Wordle Off	Mode multijoueur. Possibilité de partager sa partie.	Mots de longueur 5 seulement.
Nerdle	Bien pour les matheux. Beaucoup de modes de jeu différents.	Visuel non intuitif.
Lordle Of The Rings	Pour les fans du seigneur des anneaux.	Aucun moyen de changer les paramètres.
Sweardle	Marrant, très simple à comprendre	Clavier visuel seulement. Manque de paramètres ajustables.
Horsle	Très très simple à prendre en main (le mot à deviner est toujours Horse)	C'est très vite fini

2.3 Étude des solveurs de WORDLE

Nous avons trouvé plusieurs solveurs de WORDLE. Voici une liste de quelques un d'entre-eux expliqués :

Noms	Avantages	Désavantages
3Blue1Brown's Wordle Slover [6]	Très complet, calcul d'une entropie approprié	La recherche en "2 steps" est compliquée.
Eldrow [7]	Compare la performance en fonction du langage informatique utilisé, devine les mots en moyenne au bout du 4/5ème essai.	Codé en Rust donc peut-être que les méthodes sont non adaptées au C.
Roy van Rijn's Algorithm [5]	Très simple, se base sur la fréquence des lettres dans la langue anglaise	Uniquement fait pour WORLDE donc les mots de longueur 5.

## Chapitre 3

# Conception et implémentation de MOZA : Description fonctionnelle de l'application

### 3.1 Liste des fonctionnalités du système

#### \* Définition de l'application

Notre application devra permettre de jouer au jeu Wordle classique, mais aussi donner la possibilité au joueur de personnaliser ses paramètres de jeu (nombre de tentatives, taille des mots, difficulté des mots). Ces paramètres seront sauvegardés via un système de compte, et un historique des parties sera conservé dans la base de données.

De plus, un tableau de score sera disponible pour permettre aux joueurs de comparer leurs performances.

On résume la liste des fonctionnalités principales ci-dessous.

#### \* Gestion compte

- Accès à l'application en mode anonyme sans compte
- Création d'un compte à l'aide d'un formulaire
- Connexion au compte
- Déconnexion du compte actuel
- Possibilité de rester connecté d'une session à l'autre sur le même appareil
- Possibilité de se connecter à plusieurs appareils avec le même compte
- Modification des informations d'un compte

#### \* Utilisation normale

- Choix du nombre de lettres dans le mot
- Choix de la difficulté
- Réinitialisation des paramètres de jeu pour revenir aux valeurs par défauts
- Saisie des lettres au clavier
- Saisie des lettres avec le clavier virtuel
- Validation du mot avec la touche entrée ou le bouton de l'interface
- Consultation du leaderboard (tableau des scores)
- Consultation de son historique de partie
- Continuer la partie en cours sur un autre appareil connecté au même compte
- Commencer une nouvelle partie et abandonner automatiquement la partie en cours s'il y en a une

### 3.2 Cas d'utilisation et descriptions textuelles

#### \* Inscription

— Précondition :

L'internaute accède à l'application à partir du site web sur mobile ou PC.

— Postcondition :

L'utilisateur a maintenant un compte avec lequel il pourra se connecter ultérieurement.

— Déroulement normal :

1. Un utilisateur, souhaite s'inscrire sur MOZA, pour cela il se rend sur l'application web via son navigateur.

2. Il se rend sur la rubrique "Inscription".
3. Le site lui demande alors d'entrer un pseudo, une adresse e-mail et un mot de passe.
4. Suite à la validation de ses informations, celui-ci va recevoir un message de confirmation.
5. Le compte est créé

— Variantes :

- (3) → Au moment de la validation le site indique à l'utilisateur un des messages d'erreur suivant :
  - \* Mot de passe non conforme
  - \* Nom d'utilisateur déjà utilisé
  - \* Adresse e-mail non-valide ou déjà utilisée
  - \* Erreur de communication avec les serveurs, merci de réessayer ultérieurement

#### \* Connexion

— Précondition :

L'utilisateur est inscrit dans l'application web.

— Postcondition :

L'utilisateur accède à son espace personnel.

— Déroulement normal

1. Un internaute souhaitant se connecter sur MOZA se rend sur l'application web via son navigateur.
2. Il se rend sur la rubrique "Connexion".
3. Le site lui demande alors son adresse e-mail ainsi que son mot de passe afin de pouvoir s'identifier.
4. L'utilisateur se trouve sur son espace personnel sinon il devra de nouveau donner ses identifiants jusqu'à validation par le système.

— Variantes :

- (3) → si le mot de passe est faux ou le nom d'utilisateur inexistant le système demande à l'utilisateur de rentrer de nouveau ses identifiants.

#### \* Reprendre une partie en cours

— Précondition :

L'utilisateur est connecté.

Il commence une partie de jeu peu importe ses réglages

— Postcondition :

L'utilisateur peut continuer sa partie.

— Déroulement normal

1. L'utilisateur accède à l'application avec n'importe quel appareil.
2. Il se connecte à son compte.
3. Le système détecte qu'une partie est en cours :
  - a. Les paramètres de la partie sont chargés comme les paramètres actuels (nombre de lettres, de tentatives...)
  - b. La partie en cours est chargée comme la partie actuelle
4. L'utilisateur reprend sa partie.

— Variantes :

- (2) → L'utilisateur était préalablement connecté, il le reste donc en accédant à l'application.
- (4) → L'utilisateur change les paramètres de partie, la partie en cours est donc supprimée.
- (4) → L'utilisateur préfère commencer une nouvelle partie, la partie en cours est donc écrasée par la nouvelle.

### 3.3 Structure du site et maquettes



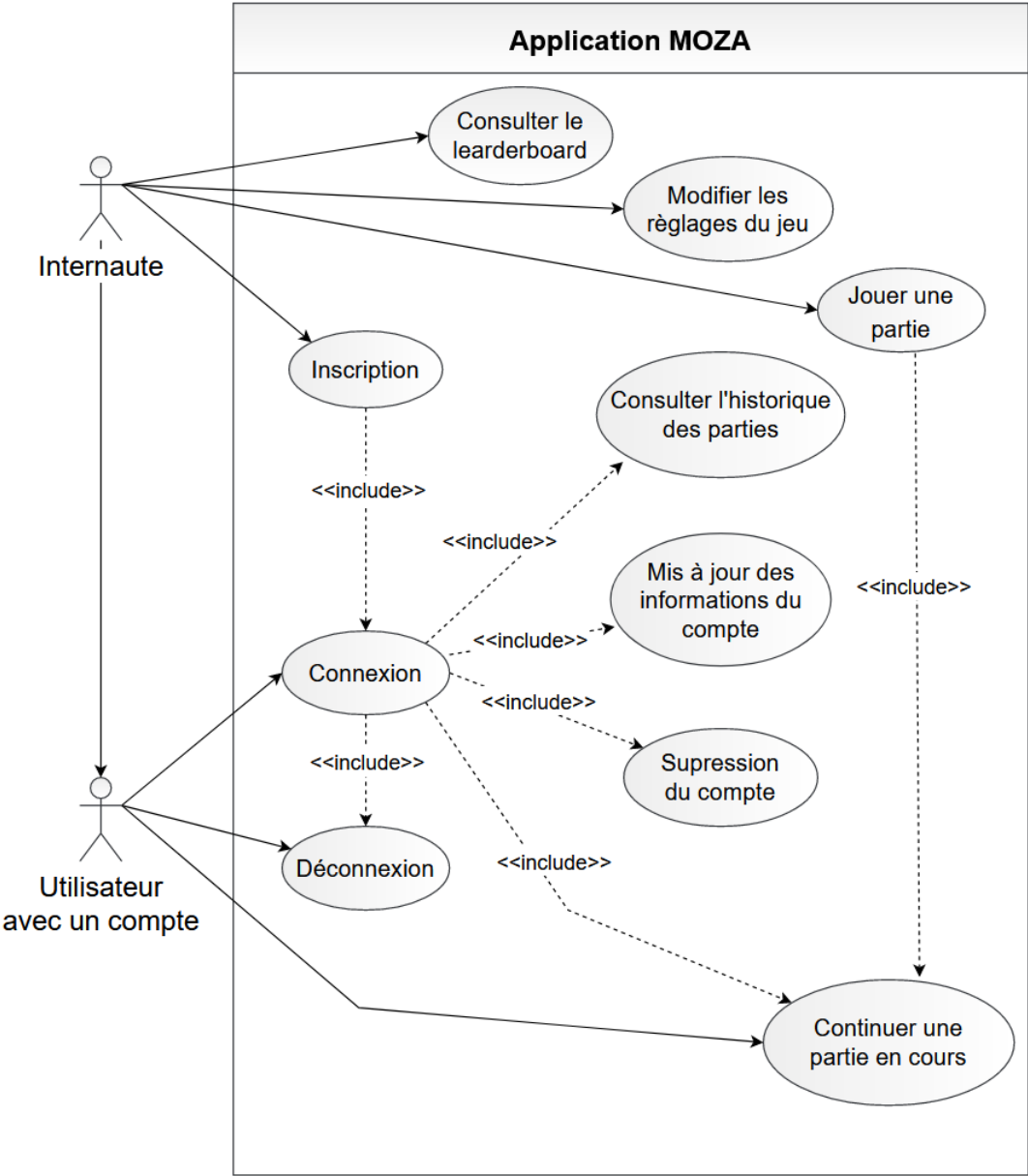


FIGURE 3.1 – Diagramme de cas d’utilisation général de l’application

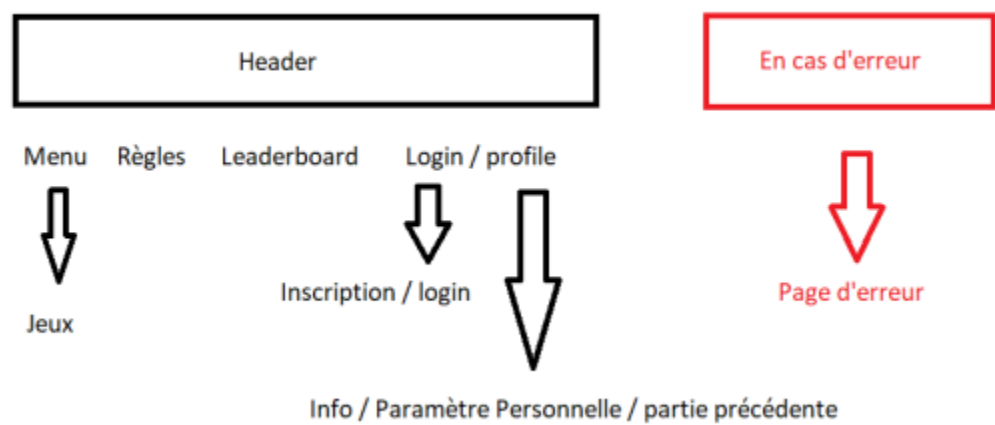


FIGURE 3.2 – Structure du site

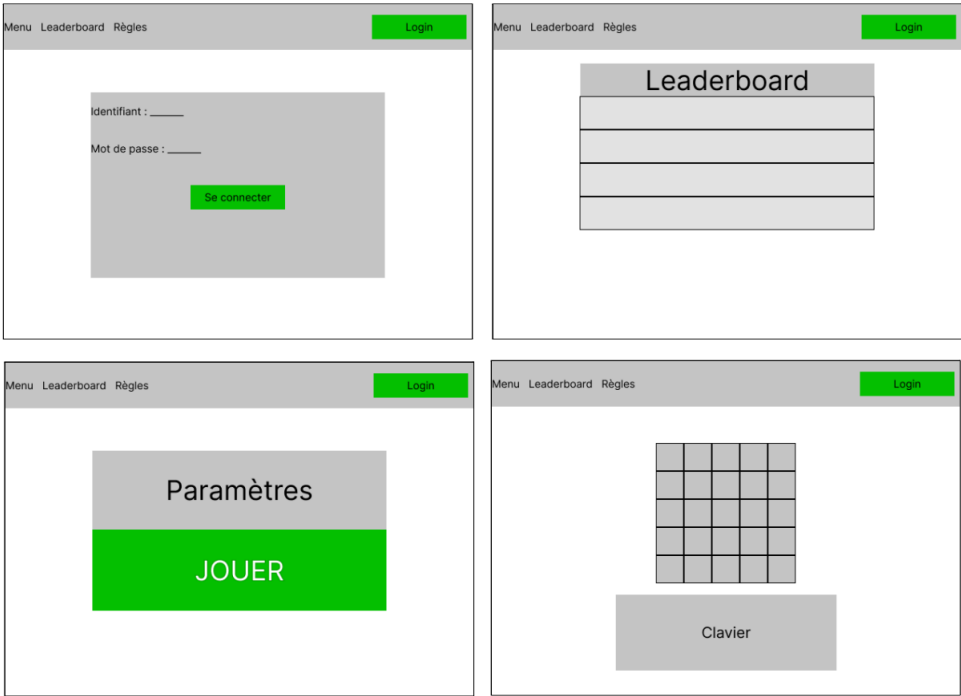


FIGURE 3.3 – Maquette de l’interface utilisateur

Menu

Leaderboard

Règles

Historique

MOZA

Logout of nico

id	Nombre essais	Nombre de lettres	Solution	Etat	Score	Difficulté	Date
1	6	7	arroger	Gagnée	30	2	2022-04-29 20:51:21
2	6	7	uniment	Gagnée	30	2	2022-04-29 20:51:44
3	4	7	mariner	Perdue	0	2	2022-04-29 21:02:55
4	4	7	🔍	En cours	-	2	2022-04-29 21:03:30

MOZA

Voisin Victor

Chanel Valentin

Frache Nicolas

Demangeon Matéo




FIGURE 3.4 – Ajout d’une page historique des parties, visible seulement si on est connecté

# Chapitre 4

## Conception et implémentation de MOZA : Mise en place technique

### 4.1 Modèle relationnel de la base de données

Une fois la liste des fonctionnalités de notre application mise en place, nous avons cherché à en déduire un modèle relationnel dont la représentation schématique est présentée ci-dessous [4.1].

Nous avons volontairement omis le cas d’un utilisateur qui se connecte sans compte car il aura accès à un nombre de fonctionnalités restreint et rien ne sera stocké à son sujet sur la base de données.

Une partie a pour attributs non seulement le mot du dictionnaire qui représente la solution, mais également les paramètres qui lui sont appliqués et l’utilisateur qui y joue. De plus, une partie est constituée d’une liste de tentatives de la part du joueur. Ces tentatives contiennent donc un mot du dictionnaire, et sont liées à une partie.

Le système de paramètres (nombre de lettres et nombre d’essais possible) est conçu de manière à ce que chaque partie soit liée à une configuration de paramètres données. Et tant que l’utilisateur ne change pas ses paramètres, toute les nouvelles parties sont liées au même objet paramètre. Ce lien est permis par la sauvegarde des derniers paramètres utilisés par l’utilisateur avec l’attribut “parametreDernierePartie”.

Passons maintenant à la reprise de la partie en cours. Pour intégrer cela, on commence par lier l’utilisateur à sa dernière partie jouée avec l’attribut “partieEnCours”. Puis on vérifie si cette partie est ou non terminée avec l’attribut “estEnCours”. Si la partie est terminée, le système peut ainsi en proposer une nouvelle. Et sinon, le système récupère l’ensemble des tentatives liées à cette partie pour permettre au joueur de continuer.

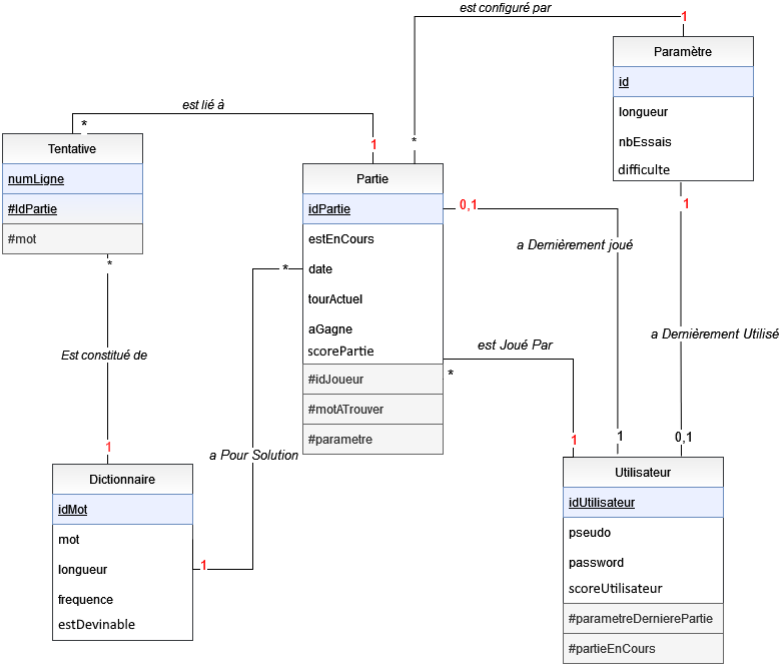


FIGURE 4.1 – Schéma du modèle relationnel de base de données

## 4.2 Interaction entre les couches de l'application

### 4.2.1 Interaction serveur/client

Techniquement pour faire cela le principe est le même que pour le chargement de n'importe quelle page de notre site. Côté back, l'application Flask va recevoir une requête HTTP sur une route. Sauf qu'au lieu de renvoyer une chaîne de caractère contenant le html (généré avec le moteur de template Jinja2), nous allons ici envoyer une réponse en XML ou JSON avec les informations nécessaires.

Côté front, nous utilisons la technologie de requêtes AJAX (asynchronous JavaScript and XML) qui permet à notre application javascript d'envoyer une requête HTTP à un url du serveur. Concrètement, pour cela nous allons utiliser l'API Fetch qui offre une syntaxe plus simple et fonctionne avec un système de promesses.

Le diagramme de séquence ci-dessous [4.2] permet de représenter chronologiquement les interactions entre les différents acteurs et couches du système pour un cas d'utilisation particulier. Ici, on s'intéresse à la situation d'un utilisateur qui joue une partie et qui saisit un mot valide (dans le dictionnaire) mais non-solution de la partie. On peut noter que le diagramme ci-dessous a été réalisé avec l'outil open-source PlantUML qui permet de générer des diagrammes à partir d'un texte source. Le code correspondant à ce diagramme est donc disponible dans le répertoire documents de notre dépôt git.

### 4.2.2 Dictionnaire

#### \* Front to back

La réalisation de l'état de l'art nous a permis de constater que la majorité des applications similaires existantes demandent au client (ordinateur de l'utilisateur) de télécharger un fichier contenant l'ensemble du dictionnaire, puis le front en javascript le manipule à partir de là.

Ici, nous avons fait un choix différent. Le dictionnaire n'est pas sous forme de fichier, mais intégré au sein de notre base de données. Ainsi, l'application javascript va interagir avec notre back pour demander par exemple si un mot est dans le dictionnaire.

#### \* Recherche de dictionnaire adéquat

Nous avons aussi trouvé un dictionnaire [2] très complet et approprié pour trier les mots et évaluer la difficulté de ceux-ci. Cependant, il a été nécessaire de le traiter pour le simplifier.

*Note : la difficulté des mots est basée sur leur fréquence d'apparition dans la langue française. Cette fréquence était donnée dans ce dictionnaire.*

#### \* Injection dans la base de données

Avant d'injecter notre dictionnaire dans notre base de donnée SQLite3, nous avons réalisé quelques traitements sur les mots qui le composent :

- Suppression des colonnes inutiles de la base
- Suppression des mots avec des longueurs inadéquates
- Suppression des mots composés ou avec un espace (exemple : "a-capella")
- Tous les accents sont retirés
- Tous les doublons dus à ces manipulations sont supprimés

#### \* Deux dictionnaires

Nous nous sommes rendu compte, en jouant à notre jeu, que les mots au pluriel ou que les verbes conjugués étaient très difficiles à deviner. Cela était très inconvenient car la fréquence des verbes conjugués et des mots au pluriel était élevée, et qu'ils se retrouvaient dans nos parties "faciles".

C'est pour cette raison que nous avons décidé de faire 2 dictionnaires. Un dictionnaire des mots qui peuvent être devinés, et l'autre, plus large, pour tous les mots que l'utilisateur peut proposer. Ces deux dictionnaires seront utilisés de cette façon dans l'application web mais aussi dans le solver, qui peut proposer des mots non devinables mais finit toujours par proposer un mot potentiellement proposé par le jeu.

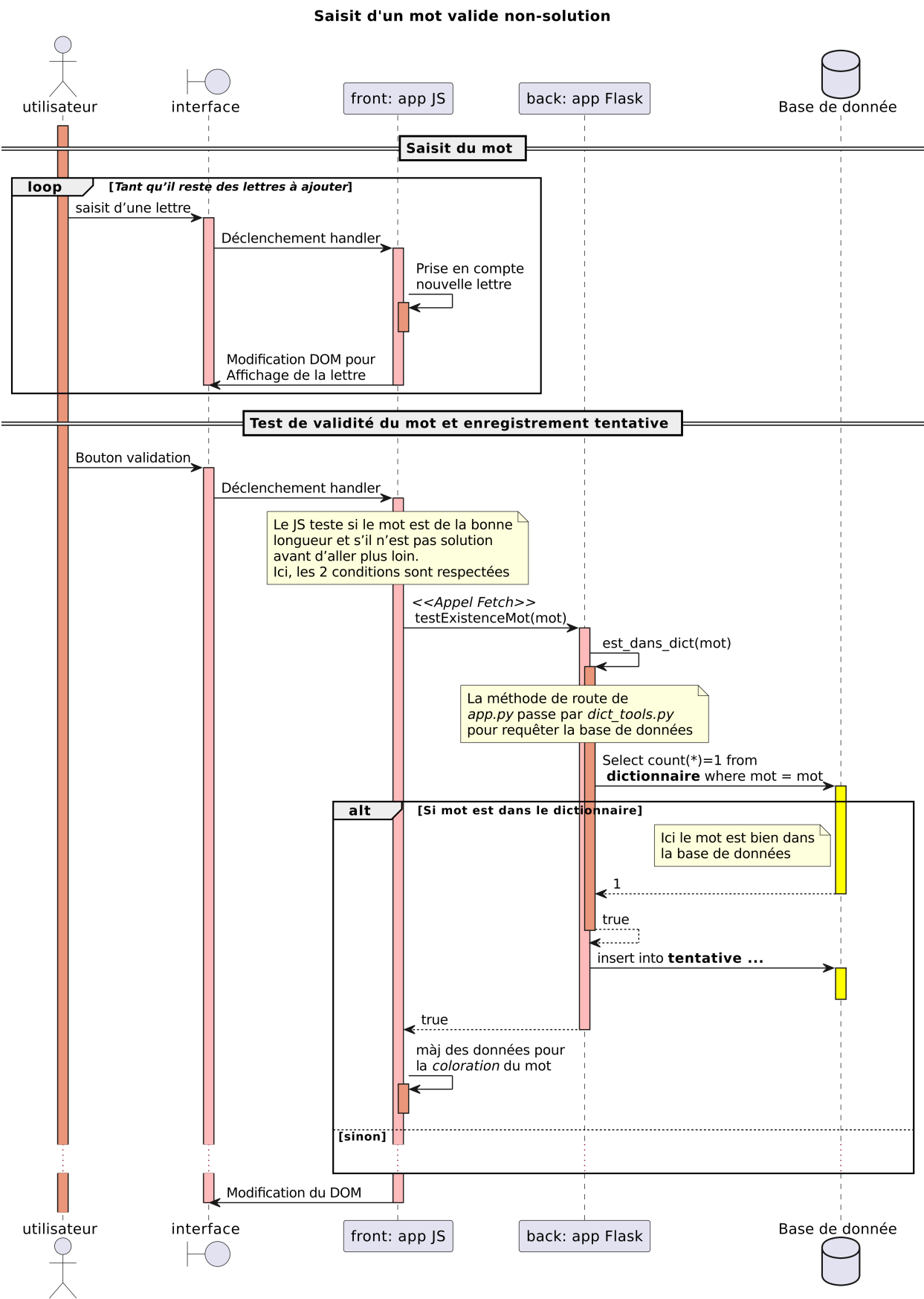


FIGURE 4.2 – Diagramme de séquence correspondant à la saisie d'un mot par le joueur puis à sa validation par le système pour déterminer s'il appartient ou non au dictionnaire.

## Chapitre 5

# Conception et implémentation de RELA : algorithme de résolution

### 5.1 Idée générale

Notre algorithme doit permettre de trouver le mot mystère en un nombre minimum de tentatives. Pour cela, il doit faire des propositions de mots pertinentes pour récupérer un maximum d'informations.

#### 5.1.1 Principe de l'algorithme

**\* Étape d'algorithme**

L'algorithme doit à chaque étape :

- Demander les informations obtenues après l'essai précédent
- Calculer la pertinence des mots à proposer
- Proposer le mot le plus pertinent pour éliminer un maximum d'autres mots

Les premiers mots pertinents sont les mêmes, pour chaque longueur. Il a donc été décidé de fixer les mots à proposer à chaque première étape, ce qui évite une perte de temps de recherche.

#### 5.1.2 Quantification des informations apportées par une proposition

À chaque tour, le solver doit choisir le meilleur mot à jouer. Pour cela, on va calculer l'entropie de chaque tentative selon la formule suivante :

$$E[I] = \sum_x p(x) \cdot \log_2(1/p(x)) \quad (5.1)$$

avec  $x$  parcourant les  $3^n$  patterns de couleur potentiellement répondus par le jeu et  $p$  la probabilité que ce pattern apparaisse en jouant la proposition :

$$p(x) = \frac{\text{nombre de mots éliminés par le pattern } x}{\text{nombre de mots restants}} \quad (5.2)$$

Ainsi, on quantifie l'information apportée par chaque proposition, et on joue la proposition qui en apporte le plus.

Par exemple, les meilleurs premiers mots de longueur 3 à 7 sont respectivement **rai**, **aria**, **aerer**, **errera** et **recre**. Ils sont joués systématiquement au premier tour (sans être recalculés à chaque fois). À titre indicatif, les 10 meilleurs premiers mots de 5 lettre au premier tour sont :

1. **aerer**
2. **raire**
3. **erres**
4. **terre**
5. **creer**
6. **aeres**
7. **serre**
8. **recre**
9. **rirai**
10. **narre**

5.1.3   Système de contraintes

A partir des informations obtenues (croisement des mots proposés et des patterns obtenu), on créé un objet contrainte qui permet de calculer si un mot est possiblement le bon. Les contraintes prisent en compte sont :

- pour chaque emplacements : l'éventuelle lettre dont on est sûr et les lettres interdites
- pour chaque lettres : le nombre d'occurrences et si ce nombre d'occurrence est exacte.

5.1.4   Boucle principale

A chaque itération, on va calculer le mot avec le plus d'entropie, le proposer, écouter la réponse, calculer les contraintes pour enfin mettre à jour la liste des mots qui sont actuellement possibles. La liste des mots possibles étant strictement décroissante, on est sûr que le résolveur converge vers la bonne réponse (avec l'assomption que le mot est dans le dictionnaire).

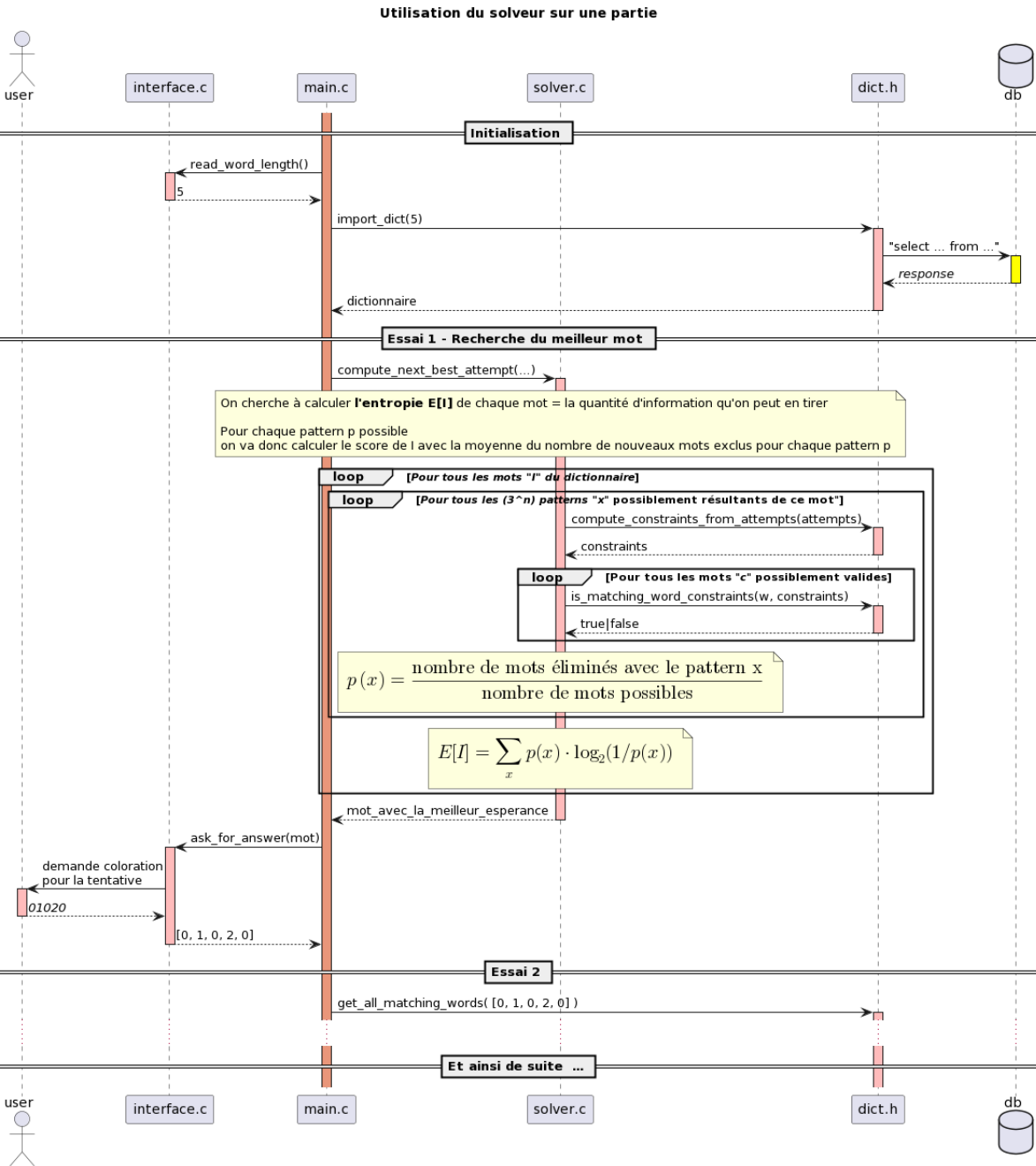


FIGURE 5.1 – Diagramme de séquence du solveur.

## 5.2 Structure de données

Nous avons du faire plusieurs choix dans la manière de représenter les données. Nous avons retenu la structure de liste doublement chaînée cyclique pour représenter les mots possibles. Cela nous permet d’avoir une certaines flexibilités dans la manière d’itérer sur les mots tout en gardant des conditions d’arrêts pertinentes.

Ensuite, pour la liste des tentatives nous avons choisi une liste simplement chaînée car cette structure est simple mais efficace et correspond à la manière dont on veut manipuler ces données.



# Chapitre 6

## Tests et performances

### 6.1 Tests sur les algorithmes

Afin de progresser dans le développement sans accumuler des erreurs, nous avons effectués de nombreux tests unitaires sur les différentes fonctions de notre programme grâce à la librairie C Snow [3].

#### **Right bicep**

Nous allons suivre la philosophie du Right - BICEP (extrait de cours de CS54 [4]) :

- Right : Are the results right ?
- B : Are all the boundary conditions CORRECT ?
- I : Can you check inverse relationships ?
- C : Can you cross-check results using other means ?
- E : Can you force error conditions to happen ?
- P : Are performance characteristics within bounds ?

#### **Environnement de tests**

La librairie Snow [3] nous permet de définir des fonctions exécutables avant et après chaque test de manière automatique. On utilise ce système pour définir correctement des variables ainsi que des exemples au début de tests mais aussi pour libérer les espaces mémoires utilisés pour éviter les fuites de mémoire.

Nous avons ainsi pu tester les opérations de bases sur les structures définies plus haut et corriger quelques cas d'erreurs notamment pour des cas marginaux ou incongrus.

On peut ainsi voir la sortie des tests unitaires 6.1 :

```

dict.o attempts_tools.o solver.o -l sqlite3  && mv *.o obj/
evele@TNCY-Linux:~/Documents/GitLab/project2-E32/solveur$ cd bin/
evele@TNCY-Linux:~/Documents/GitLab/project2-E32/solveur/bin$ ./test

Testing test_elementaire:
✓ Success: It's working ! (10.01µs)
test_elementaire: Passed 1/1 tests. (180.91µs)

Testing attempts_tools:
✓ Success: creation (7.08µs)
✓ Success: insertion (23.19µs)
✓ Success: access (26.86µs)
✓ Success: deletion (12.94µs)
✓ Success: remove_element (22.95µs)
✓ Success: create and compute result (8.06µs)
attempts_tools: Passed 6/6 tests. (166.99µs)

Testing Dict:

Testing words_list:
✓ Success: creation (11.96µs)
✓ Success: append (6.84µs)
✓ Success: remove element (16.11µs)
words_list: Passed 3/3 tests. (78.12µs)

Testing import_dict:
✓ Success: import of dict from db (20.60ms)
import_dict: Passed 1/1 tests. (20.71ms)

Testing constraints:
✓ Success: Constraints struct initialization (44.92µs)
✓ Success: compute_constraints_from_attempts (31.98µs)
✓ Success: update_current_possible_with_attempt (370.85µs)
constraints: Passed 3/3 tests. (489.01µs)

Testing match:
✓ Success: match with no attempt (9.03µs)
✓ Success: match with one attempt (239.01µs)
✓ Success: match with two attempts (43.95µs)
match: Passed 3/3 tests. (319.09µs)

Dict: Passed 10/10 tests. (21.61ms)

Total: Passed 17/17 tests. (21.99ms)

evele@TNCY-Linux:~/Documents/GitLab/project2-E32/solveur/bin$

```

FIGURE 6.1 – Réussite de tests unitaires utilisant la librairie Snow.

## 6.2 Performances

### 6.2.1 Calcul de complexités

Comme nous l'avons précédemment expliqué, le fonctionnement de notre solveur repose sur le calcul d'une entropie en bits pour chaque mot de la longueur donnée  $L$ .

Le calcul de l'entropie d'un mot implique de calculer, pour chaque pattern possible, une entropie partielle. Un pattern est une combinaison de 3 couleurs, on en a donc  $3^L$  possibles.

Enfin, pour chaque pattern, conformément à la formule de calcul présentée plus tôt [5.1], nous avons besoin de tester la compatibilité de ce pattern avec chaque mots encore possible à cette étape du fonctionnement du solveur. Pour le calcul de la complexité ci-dessous nous approximations le nombre de mots possible avec le nombre de mots  $n$  du dictionnaire

$$\text{Complexité} = n^2 \cdot \text{nombre\_pattern\_possible} = n^2 \cdot 3^L \quad (6.1)$$

Ainsi la complexité de l'algorithme est en  $O(n^2 \cdot 3^L)$ . On remarque que le nombre de calculs explose quand la longueur des mots  $L$  augmente. C'est le facteur qui nous limite dans le temps d'exécution du solveur.

### 6.2.2 Temps d'exécution

#### Outils de mesure

- Les deux principales indicateurs de performance retenus ont été
- le temps d'exécution mesuré avec la commande `bash time`
  - le nombre d'essai moyen pour arriver au résultat.

#### Exécution

L'équipe a choisi de rédiger les fonctions au plus simple puis de les optimiser une fois qu'elles étaient fonctionnelles. Un travail important a été dédié à la fonction calculant le prochain mot le plus pertinent étant donné qu'elle est la plus lourde. Nous avons essayé de :

- réduire le nombre d'opérations.
- réfléchir à leur nature d'un point de vu complexité.
- réfléchir à leur nature d'un point de vu précision dans l'arithmétique des flottants.
- éviter les allocations mémoires inutiles et/ou redondantes.

L'équipe a ainsi testé différent scénario tel que le suivant :  
Nous devons calculer, pour chaque boucle :

$$p = match \cdot total \tag{6.2}$$

$$info = \log_2(1 \div p) \tag{6.3}$$

$$entropie+ = p \cdot info \tag{6.4}$$

Où total est indépendant de l'itération.

Un membre a proposé le calcul alternatif :

$$p = match \tag{6.5}$$

$$info = \log_2(total) - \log_2(match) \tag{6.6}$$

$$entropie+ = p \cdot info \tag{6.7}$$

Plus, à la fin de toutes les boucles, une unique division :

$$entropie/ = total \tag{6.8}$$

Cet exemple de changement n'a pas améliorer le temps d'exécution il n'a pas donc été retenu mais nos optimisations diverses lors des différentes étapes de l'algorithme ont réduit le temps de calcul du premier mot avec 5 lettres par un facteur 4.

```
> 5033/5037
> 5034/5037
> 5035/5037
> 5036/5037  garee
real    7m58.185s
user    7m57.330s
sys     0m0.780s
```

FIGURE 6.2 – Temps d'exécution avant optimisation

### 6.2.3 Nombre moyen de coups

Nous avons écrit un programme permettant de calculer le nombre moyen de tentatives nécessaire pour deviner un mot. Malheureusement, nous avons du faire face à des contraintes de structure, d'organisation, de temps mais principalement de complexité nous rendant difficile d'avoir des données chiffrable.

En effet, le cahier des charges a orienté notre application vers une organisation rendant difficile le test de tous les mots. De plus, le dictionnaire choisit est particulièrement volumineux ce qui rend tous calculs extrêmement lourd sachant que la complexité est exponentielle par rapport aux nombres de mots possibles.

```
Nombre de lettre : 5
Nombre de mots possibles : 2264

Il reste 2264 mots possibles

• Mot avec la plus grande entropie: aerer
Saisissez le résultat de la proposition
>

real    1m51.302s
user    1m49.431s
sys     0m0.471s
```

FIGURE 6.3 – Temps d'exécution après optimisation

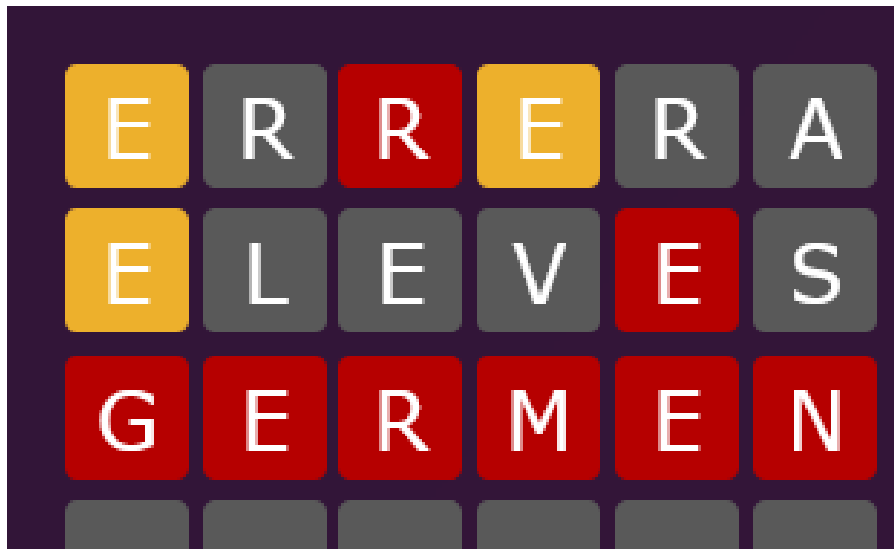


FIGURE 6.4 – Exemple d'un mot de longueur 6 lettres trouvé en 3 coups

[illegible]

FIGURE 6.5 – Interface du solveur

# Chapitre 7

## Gestion de Projet

### 7.1 Composition de l’équipe du projet

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

- VOISIN Victor
- CHANEL Valentin
- DEMANGEON Matéo
- FRACHE Nicolas

Nicolas est nommé chef de projet, il sera responsable de suivre et de superviser le projet.

L’équipe s’est réunie dans les locaux de TELECOM Nancy pour définir les objectifs, répartir le travail et aussi le réaliser. Lors des vacances scolaires, l’équipe s’est réunie à travers la plateforme Discord.

Les éléments de programmation ont été créés, partagés et utilisés à partir du Git fourni par l’école. De même, pour la rédaction du rapport, l’équipe a utilisé le Leaf fourni par l’école.

### 7.2 Organisation

Au vue des délais restreints pour la réalisation de notre livrable, en particulier le solver, nous avons décidé d’opter pour la méthode AGILE. En effet, celle-ci permet entre autres de découper le projet en itérations (sprint) à l’issue desquelles nous aurons un livrable fonctionnel (même si incomplet dans l’état).

Ainsi, nous avons organisé des réunions régulièrement dans le but de planifier le prochain sprint, alors découpé en lots de travail.

### 7.3 Analyse de l’environnement - matrice SWOT

La matrice SWOT permet d’évaluer la pertinence et l’environnement de notre projet en identifiant les points positifs et négatifs qui s’imposent à nous. Notre projet étant découpé en deux parties (une traitant de développement web, l’autre du développement d’une application en langage C), il était important de l’utiliser pour déceler à l’avance les principales difficultés.

	Positif	Négatif
Interne	Compétences dans le langage Python, Connaissances dans la gestion de base de données, Expérience préalable dans le développement web (projet du premier semestre)	Nécessité de se former en JavaScript, Peu d’expérience dans le langage C, Organisation du projet de solver compliquée
Externe	Application WORDLE récemment devenue populaire sur Internet Application web apportant de nouvelles fonctionnalités par rapport aux différentes variantes de WORDLE. Différents solvers du jeu WORDLE déjà existants (algorithme déjà plus ou moins connu)	Délai restreint pour réaliser le solver en langage C

## 7.4 Diagramme de Gantt

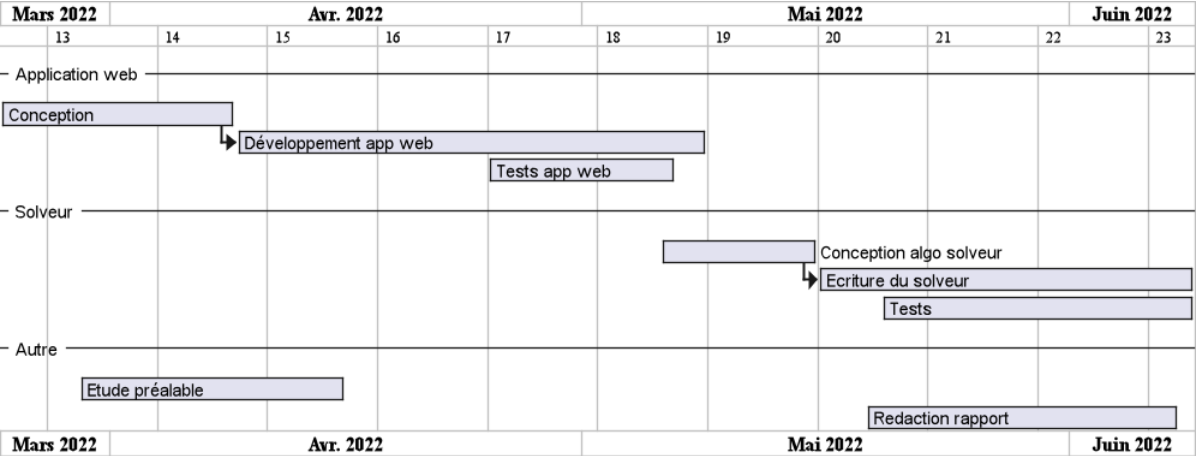


FIGURE 7.1 – Diagramme de Gantt réalisé en PlantUML

## 7.5 Comptes-rendus des réunions

### 7.5.1 16 Mars 2022

Motif : Démarrage du projet		Lieu : TELECOM Nancy en salle 1.18	
Présent	Absent	Durée : 1h	
VOISIN Victor CHANEL Valentin DEMANGEON Matéo FRACHE Nicolas			

#### Ordre du jour

- Mise en place du projet
- Réflexion sur le sujet

#### Informations échangées

- Mise en place et réflexion
  - Planification des réunions*

L'équipe souhaite se réunir au minimum une fois par semaine. La communication des réunions se fera à travers la plateforme discord et en réel.
  - Objectif mi-projet*

L'équipe projet se fixe comme objectif d'obtenir d'ici 3 semaines une application fonctionnelle, puis ensuite de développer l'aspect esthétique, pour finalement passer à la partie 2.
  - Réflexion sur les outils utilisés*

L'équipe réfléchit aux outils qu'elle veut utiliser comme Pycharm, SQLite, IntelliJ ou Docker.

#### TO-DO LIST

- Établir l'état de l'art sur Google Drive pour le mardi 21 mars.
- Jouer au jeu pour en comprendre toutes les subtilités.
- Trouver un bon dictionnaire Python.
- Prochaine réunion fixée à mercredi 23 mars.

7.5.2 23 Mars 2022

Motif : Réunion d’avancement		Lieu : TELECOM Nancy, locaux de TNS
Présent	Absent	Durée : 1h30
VOISIN Victor CHANEL Valentin DEMANGEON Matéo FRACHE Nicolas		

Ordre du jour

1. Avancement des objectifs de la dernière réunion
2. Discussions de choix sur les fonctionnalités et sur la structure des données.

Informations échangées

- Avancement des objectifs de la dernière réunion

*Le jeux motus*  
L’équipe discute de ce qu’elle a trouvé sur le jeux, ses différentes formes et elle discute de certaines subtilités dans la conception.

*Le Solver*  
L’équipe projet discute des différentes méthodes observées, elle réalise la complexité de la tâche. On met pour l’instant le solver en pause en attendant d’avoir fait la première partie.
- Discussions de choix sur les fonctionnalités et sur la structure des données.

*Fonctionnalités*  
Après beaucoup d’échange d’idées et de débats, on se met d’accord sur ce que l’on veut faire : une page d’accueil avec les réglages de la partie, une page de jeux, une page / pop-up de connexion / inscription, une page de leaderboard et une page de règles. On note la présence des fonctionnalités Session et Login de Flask qui pourrait nous aider. On discute également des différents paramètre possibles, pour l’instant on ne retient que la taille du mot et le nombre d’essai.

*Organisation du code*  
On utilise le tableau des locaux de TNS pour faire l’arborescence des pages et le schéma relationnelles de la BD. On décide de stocker les parties en cours dedans plutôt que dans les cookies après un long débat . On réfléchit également à l’interaction Front-Back de notre application, de l’utilisation de JavaScript ou CSS pour certaines choses.
- Définition des objectifs

*Pour la prochaine réunion* L’équipe se met d’accord pour les prochains objectifs et fixe une séance de travaille de groupe pour le samedi 26/03.

TO-DO LIST

- Val : Faire le compte rendu
- Victor : Se renseigner sur les formules pour le leaderboard.
- Nicolas : Mettre en place l’environnement de travail.
- Matéo : Se renseigner sur Flask session et Flask Login.

- Envoyer le mail aux responsables projet samedi.

7.5.3 20 avril 2022

Motif : Réunion d’avancement		Lieu : Discord
Présent	Absent	Durée : 1h30
VOISIN Victor		
CHANEL Valentin		
DEMANGEON Matéo		
FRACHE Nicolas		

Ordre du jour

1. Rétrospective sur les dernières semaines
2. Avancement sur l’implémentation
3. Discussions sur le futur du projet

Informations échangées

- Rétrospective sur les dernières semaines
  - Le mail à Mr Festor*

Le mail a été envoyé un peu tard suite à une relance de Mr Festor, le groupe regrette qu’il y ai eu une confusion sur l’origine des schéma : ils ont été réalisé par l’équipe mais Mr Festor a cru que nous les avions pris sur internet sans créditer les auteurs originaux.
  - La communication*

La communication a été bonne, que ce soit pour les réunions pause café ou sur le serveur discord, il n’y a pas eu de problèmes et le travail a été efficace.
- Avancement sur l’implémentation.
  - Ce qui a été fait*

Une grosse partie du jeu a été créé, notamment l’architecture Flask, les fonctions pythonns d’interaction avec la BD et autres, les pages HTML, le javascript ...
  - Les difficultés rencontrées*

L’interaction JS / BD est un peu plus difficile que prévu et il y a un problème lors de l’inscription d’un joueur bloquant le développement de beaucoup de fonctionnalités.
  - Fonctionnalités*

L’équipe débat sur certains point et décide d’ajouter un paramètre difficultés se basant sur la fréquence d’utilisation des mots. Le groupe réfléchit aussi à créer un dictionnaire plus épurée pour éviter de faire devenir des mots conjuguées ou terminant par des "s".
- Définition des objectifs
  - Pour la prochaine réunion* L’équipe se met d’accord pour les prochains objectifs.

TO-DO LIST

- Val : Résoudre les problèmes d’inscription
- Victor : Réaliser le leaderboard
- Nicolas : Travailler sur l’interaction Back / Front



- Matéo : Implémenter le système de difficulté
- Libre : Gérer les cas de victoire / défaite
- Libre : Travailler sur le système de paramètre

7.5.4 9 mai 2022

Motif : Début de la partie 2		Lieu : TELECOM Nancy hall d'entrée	
Présent	Absent	Durée : 30min	
VOISIN Victor CHANEL Valentin DEMANGEON Matéo FRACHE Nicolas			

Ordre du jour

1. Organisation du projet
2. Réflexion sur le sujet

Informations échangées

- Organisation du projet

Organisation

Nicolas a proposé des squelettes .c et .h à remplir. Cela simplifiera l'écriture des fonctions. Matéo a soulevé l'idée d'écrire les fonctions en pseudo code pour aussi accélérer le processus. Cela permettra d'avoir plus de temps pour tester les fonctions ultérieurement. L'équipe à aussi pensé à écrire des tests et voudrait tester chaque fonction avant d'en écrire une nouvelle pour éviter le trop de debugging. Victor à aussi proposé d'écrire le rapport en parallèle du reste pour encore gagner en efficacité.

Réflexion sur le sujet et les outils à utiliser

L'équipe réfléchit à l'utilisation de CLion mais ce dernier semble compliqué à mettre en place et ne va pas être préféré à la machine virtuelle. Valentin a proposé de suivre les indications et la vidéo suggérée par les professeurs. Il a aussi été décidé de faire le solveur pour notre jeu, avec le même dictionnaire et donc pas d'indication sur la première lettre.

TO-DO LIST

- Écrire les fonctions en pseudo-code
- Commencer l'écriture de tests.
- Commencer à écrire les fonctions.
- Faire la première partie du rapport
- Suivre les discussions Discord pour rester à jour.

7.5.5 24 mai 2022

Motif : Réunion d’avancement		Lieu : Bureau de TNS
Présent	Absent	Durée : 15min
VOISIN Victor CHANEL Valentin DEMANGEON Matéo FRACHE Nicolas		

Ordre du jour

1. Avancement sur l’implémentation
2. Discussions sur la fin du projet

Informations échangées

- Avancement sur l’implémentation

*Interface et solver*

La partie interface a été complétée ainsi que beaucoup des fonctions. Le système de rédiger la signature de toutes les fonctions avant de les implémenter a été efficace. Il faut maintenant les assembler pour avoir un solver complet.
- Discussions sur la fin du projet

*Le rapport*

La fin du projet arrivant vite, l’équipe commence a discuter du rapport et Victor propose de l’avancer en mettant en place des templates

*Les tests et la performance*

La performance n’est pas la priorité actuelle mais l’équipe discute déjà de manière potentielle pour l’améliorer. Nicolas propose d’en tester plusieurs.

TO-DO LIST

- Faire la deuxième partie du rapport
- Continuer l’écriture du solver pour qu’il soit fonctionnel (au moins pour les mots de longueur 5)

7.5.6 07 juin 2022

Motif : Réunion d’avancement		Lieu : Bureau de TNS
Présent	Absent	Durée : 30min
VOISIN Victor CHANEL Valentin DEMANGEON Matéo FRACHE Nicolas		

Ordre du jour

- 1. Le mail de Mr.Festor
- 2. Finitions du solver
- 3. Arrivée de la date de rendu

Informations échangées

- Finitions du solver

*Date de rendu décalé*  
La date de rendu a été décalée, l’équipe est soulagée et va utiliser le temps donnée pour de petite optimisation, pour la rédaction du rapport et pour éradiquer les derniers bugs survivants.
- Finition du solver

*Sloveur fonctionnel*  
Le solver est complet et livrable pour les mots plus petits que 7. Victor a soulevé l’idée des 2 premiers mots codés en dur pour le mots de longueur 8 et 9. Il a été décidé de ne pas aller plus loin car la recherche est trop longue.

*Les tests et la performance*  
Valentin a écrit une fonction pour estimer le nombre moyen de tentatives pour deviner un mot mais elle ne marche pas. L’équipe discute des contraintes de la fonction, de celle au quelle on fait face et décide de ne pas mettre la priorité sur cette fonction.
- Arrivée de la date de rendu

*Le rapport*  
Le rapport est quasi-complet. Victor souligne l’importance de bien remplir les parties personnelles du rapport et demande des captures écran de manière à illustrer des passages parfois long dans le rapport.

TO-DO LIST

- Compléter le chapitre 6 du rapport
- Compléter la partie bilan du rapport

# Chapitre 8

## Bilan du projet

### 8.1 Bilan global du projet

Travail attendu	Travail réalisé
Application utilisant des algorithmes python, une base de donnée et une interface web pour jouer à Wordle Solver C calculant les meilleurs mots à jouer obéissant à des codes d'interactions avec l'entrée et la sortie standard	Toutes les fonctionnalités du cahier des charges sont disponibles et plus encore.  Le résolveur fonctionne et est optimisé. Certaines fonctionnalités auraient pu être ajoutées.

### 8.2 Bilan du projet membre par membre

#### 8.2.1 VOISIN Victor

<b>Points positifs</b>	Le projet m'a permis d'approfondir mes connaissances, notamment en utilisant de nouveaux logiciels, mais aussi dans l'organisation des choses (l'organisation de Nicolas était très inspirante).
<b>Difficultés rencontrées</b>	Mon bas niveau en C m'a beaucoup ralenti et j'ai du demander de l'aide plusieurs fois. Cela à entraîné une petite perte de motivation au début de la rédaction du solver.
<b>Expérience personnelle</b>	J'ai beaucoup apprécié le projet qui était clair. Cette équipe est super, merci à eux.
<b>Axes d'amélioration</b>	Devenir meilleur en C.

#### 8.2.2 CHANEL Valentin

<b>Points positifs</b>	Ce projet ma donné l'opportunité de m'intresser à CSS, de participer à un projet C de taille moyenne mais aussi de travailler avec une structure / organisation de projet très propre.
<b>Difficultés rencontrées</b>	L'apprentissage du C en parallèle du développement du solver a créé certaines difficultés.
<b>Expérience personnelle</b>	La bonne dynamique du projet soutenu par un sujet intéressant m'a permis de progresser et de me faire de nouveaux amis.
<b>Axes d'amélioration</b>	On aurait pu mieux anticiper la problématique de rejouer et structurer nos fonctions différemment.

8.2.3 DEMANGEON Matéo

Points positifs	Le projet m’a permis d’apprendre de façon approfondie le langage Javascript. De plus, il m’a fait progresser dans ma compréhension du langage C.
Difficultés rencontrées	Le langage C était difficile à appréhender au début. Par ailleurs, il était compliqué de s’organiser dans un projet comme le solver en C.
Expérience personnelle	J’ai bien aimé la problématique du projet, en particulier la partie solver.
Axes d’amélioration	Avec un peu plus de temps, on aurait pu mieux quantifier les performances de notre solver, par exemple en calculant le nombre moyen d’essais nécessaires pour résoudre le jeu.

8.2.4 FRACHE Nicolas

Points positifs	C’était pour moi l’occasion de mettre en pratique mes connaissances du C dans un premier vrai projet.
Difficultés rencontrées	J’ai consacré beaucoup de temps à diagnostiquer et corriger des erreurs (parfois obscures) liées à la gestion de la mémoire en C. C’est particulièrement décourageant de ne pas toujours bien saisir l’origine d’une erreur alors qu’on utilise un debugger ligne par ligne.
Expérience personnelle	Le contexte de période de partiels m’a empêché de me consacrer autant que je l’aurai voulu à ce projet pour pouvoir ajouter des fonctionnalités supplémentaires ou encore mettre en ligne l’application web sur un serveur.
Axes d’amélioration	figlet aucun   cowsay -n

8.3 Travail réalisé

Étapes	Valentin	Victor	Matéo	Nicolas
État de l’art/Document de Conception	8h	9h	?h	15h
Conception et Implémentation de MOZA	56h	68h	?h	52h
Conception et Implémentation de RELA	49h	16h	?h	55h
Conception du rapport	9h	26h	?h	5h
TOTAL	122h	119h	?h	127h



FIGURE 8.1 – Mozzarella-di-bufala

# Bibliographie

- [1] Dave CHILD. *Wordle Alternatives*. 2022. URL : <https://aloneonahill.com/blog/wordle-alternatives>.
- [2] Boris New et CHRISTOPHE PALLIER. *Lexique*. 1999. URL : <http://www.lexique.org/>.
- [3] Martin DØRUM. *Snow*. 2022. URL : <https://github.com/mortie/snow>.
- [4] Olivier FESTOR. *Testing*. 2022. URL : [https://arche.univ-lorraine.fr/pluginfile.php/2204234/mod\\_resource/content/1/Testing\\_Lecture.pdf](https://arche.univ-lorraine.fr/pluginfile.php/2204234/mod_resource/content/1/Testing_Lecture.pdf).
- [5] Roy van RIJN. *An algorithm for Wordle*. 2022. URL : <https://www.royvanrijn.com/blog/2022/01/wordle-bot/>.
- [6] Grant SANDERSON. *Solving Wordle using information theory*. 2022. URL : <https://youtu.be/v68zYyaEmEA>.
- [7] Simon ZENG. *Meus recogitare et excogitare, Eldrow*. 2022. URL : <https://xsznix.wordpress.com/2022/01/29/eldrow/>.