AVIGNON UNIVERSITÉ

Wordle

Groupe3
Anthony Genti
Youssef Boudount
Axel Bertrand
Dylan Gely

16/01/2024

Licence d'Informatique
Ingénierie logicielle
UE Projet Programmation

Responsables
Noe Cecillon
Jarod Duret

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

CERI

CENTRE
D'**ENSEIGNEMENT**
ET DE **RECHERCHE**
EN **INFORMATIQUE**
ceri.univ-avignon.fr

Sommaire

1 Introduction

1.1 Wordle

1.1.1 Qu'est-ce que Wordle ?

Wordle est un jeu de réflexion basé sur les mots où le but est de deviner un mot cible à l'aide d'indices basés sur la couleur. Chaque guess correct rapproche le joueur du mot cible grâce à un système de feedback intuitif.

1.1.2 Règles du Jeu

- Le joueur doit trouver un mot a x lettres en 5 essais au plus.
- Chaque lettre du mot proposé reçoit un retour coloré :
  - Une lettre colorée en vert est correcte et bien placée.
  - Une lettre colorée en orange est correcte mais mal placée.
  - Une lettre colorée en rouge n'est pas dans le mot.

1.1.3 Déroulement d'une Partie

Voici comment se déroule une partie typique de Wordle :

1. Un mot cible est choisi aléatoirement par le jeu et caché au joueur.
2. Le joueur saisit une première proposition de mot.
3. Le jeu affiche des indices colorés pour chaque lettre du mot proposé.
4. Avec ces indices, le joueur continue à proposer des mots jusqu'à trouver le mot cible ou jusqu'à épuiser ses essais.
5. Si le joueur devine le mot avant d'avoir utilisé ses essais, il gagne ; sinon, le mot est révélé et le jeu se termine.

1.2 Présentation du projet

Ce rapport documente le développement d'un logiciel de jeu basé sur le concept de Wordle, spécifiquement adapté pour la langue française.

Notre version de Wordle est une application Java, qui utilise une architecture orientée objet pour une facilité de maintenance et d'extension.

Le logiciel se distingue par son approche innovante de l'assistance aux joueurs, exploitant le modèle Word2vec pour fournir des indices contextuellement pertinents.

Ce projet a pour but de combiner l'amusement des jeux de mots avec des technologies avancées de traitement de la langue naturelle.

1.3 Contexte et objectifs

L'engouement pour les jeux de mots et l'apprentissage de la langue a encouragé le développement d'applications ludiques stimulant l'esprit. Notre projet s'inscrit dans cette tendance, en proposant un défi linguistique quotidien qui est à la fois éducatif et divertissant.

L'objectif principal est de créer une application qui non seulement teste le vocabulaire des utilisateurs mais les aide également à l'élargir grâce à des indices générés par intelligence artificielle, offrant ainsi une expérience d'apprentissage enrichissante.

1.4 Méthodologie et répartition des tâches Intermediaire

Le projet a été conduit suivant une méthodologie agile, permettant une flexibilité et une adaptation continue face aux retours et aux exigences évolutives. L'équipe de développe-

ment est composée de quatre membres, chacun responsable d'une composante spécifique du logiciel :

- Dylan s'est concentré sur le développement de l'interface utilisateur, y compris la création du portail général et de la zone de jeu interactive.
- Antony a pris en charge le développement des fonctionnalités du menu, y compris les systèmes d'aide et de redémarrage de jeu.
- Youssef a travaillé sur la logique du jeu, s'assurant que les règles de Wordle étaient correctement implémentées et que les mots entrés par les utilisateurs étaient validés efficacement.
- Axel a été chargé d'intégrer le modèle Word2vec pour générer des indices et de mettre en place la communication entre le backend Python et l'application Java.

Chaque membre de l'équipe a activement participé à toutes les phases du cycle de développement, de la conception initiale aux tests finaux, en veillant à ce que leurs contributions s'intègrent harmonieusement dans l'architecture globale du logiciel.

1.5 Méthodologie et répartition des tâches Finales

Après l'évaluation intermédiaire, notre équipe a continué à suivre une approche agile, toutefois, des disparités dans la répartition des tâches sont apparues. Jusqu'à l'évaluation intermédiaire, tous les membres ont travaillé de manière équitable, mais par la suite, certains ont pris des responsabilités supplémentaires tandis que d'autres ont été moins impliqués.

- Youssef a joué un rôle prépondérant dans cette phase. Il a travaillé à réadapter le code du package View pour en améliorer la lisibilité et la maniabilité, facilitant ainsi l'ajout de nouveaux jeux. Youssef a également pris en charge les petites modifications et améliorations de l'interface graphique, suite aux commentaires du professeur lors de l'évaluation intermédiaire.
- Anthony s'est concentré sur le développement de la classe Score. Il a réalisé toutes les tâches liées aux scores, incluant leur calcul, affichage et enregistrement.
- Axel a significativement amélioré son module d'indices et de recherche d'indices, ce qui a contribué à enrichir l'expérience utilisateur du jeu.
- Dylan, malheureusement, n'a pas contribué autant que les autres membres de l'équipe. Une plus grande implication de sa part aurait été appréciée, notamment pour équilibrer la charge de travail au sein de l'équipe.

Cette répartition des tâches reflète la dynamique de travail de l'équipe dans sa phase finale. Chaque membre, à l'exception de Dylan, a apporté une contribution significative à la réussite du projet, en veillant à ce que le produit final réponde aux attentes des utilisateurs et des parties prenantes.

1.6 Organisation interne et flux de travail

Cette section explique nos conventions de travail et les commandes Git utilisées. Nous avons pris soin d'assurer que chacun travaille dans sa propre branche pour éviter tout conflit et permettre une revue de code efficace avant l'intégration dans la branche principale.

Pour une gestion efficace du projet, nous avons établi un guide d'organisation clair, "ORGANISATION.md", que nous avons mis à disposition dans la branche principale dès la création du dépôt.

Voici le processus que chaque membre de l'équipe a suivi :

1.6.1 Clonage du projet

1. Ouvrez un terminal (invite de commande) sur votre ordinateur.
2. Naviguez vers le répertoire où vous souhaitez cloner le projet.

3. Utilisez la commande `git clone https://gitlab.com/ceri-projet-programmation-2023/groupe-3.git`.

1.6.2 Structure des branches

Chaque membre du groupe dispose d'une branche dédiée pour le développement, permettant un travail indépendant sans interférence :

- main
- axel-bertrand
- youssef-boudount
- dylan-gely
- anthony-genti

1.6.3 Travailler dans sa propre branche

1. Après avoir cloné le projet, naviguez dans le répertoire du projet avec la commande `cd`.
2. Basculez sur votre branche personnelle avec la commande suivante :

```
git checkout `nom-de-votre-branche-personnelle`
```

1.6.4 Ajout de modifications

1. Travaillez sur vos fichiers locaux dans votre branche.
2. Ajoutez toutes les modifications en préparation pour le prochain commit avec la commande suivante :

```
git add .
```

1.6.5 Commit des modifications

1. Committez vos modifications avec un message explicatif en utilisant la commande suivante :

```
git commit -m "Votre message descriptif"
```

1.6.6 Pousser les modifications sur GitLab

1. Poussez vos changements sur votre branche GitLab avec la commande suivante :

```
git push origin `nom-de-votre-branche-personnelle`
```

2. Assurez-vous d'utiliser le nom correct de votre branche personnelle.

1.6.7 Gestion des identifiants GitLab

Pour faciliter le processus de poussée, utilisez un gestionnaire de trousseaux avec la commande suivante :

```
git config --global credential.helper manager
```

2 Conception

2.1 Architecture générale du logiciel

Notre projet s'appuie sur une architecture Modèle-Vue-Contrôleur (MVC) qui divise l'application en trois composants interconnectés. Cette séparation permet de gérer les responsabilités de manière efficace, en isolant la logique métier de l'interface utilisateur et du contrôle de l'application.

- **Modèle (Model)** : Comprend les classes qui représentent les données du jeu et la logique de traitement de ces données. Le dossier *model* contient des classes telles que *FrenchWordChecker*, *RandomWordSelector*, *WiktionaryScraper* et *WordList*, qui ensemble gèrent la sélection des mots, la validation des mots français et l'interaction avec les ressources externes comme le Wiktionnaire.
- **Vue (View)** : Constituée des classes qui définissent comment les données sont présentées à l'utilisateur. Le dossier *view* inclut *CustomDialog* et *Main*, qui s'occupent de l'affichage des dialogues personnalisés et de la fenêtre principale de l'application.
- **Contrôleur (Controller)** : Sert de pont entre le modèle et la vue. Le dossier *controller* comprend *WordleGame* qui gère les interactions entre l'utilisateur et le système.

2.2 Diagrammes de conception (UML)

Les diagrammes suivants présentent une vue d'ensemble de la répartition et de la gestion des tâches au sein de notre équipe de développement.

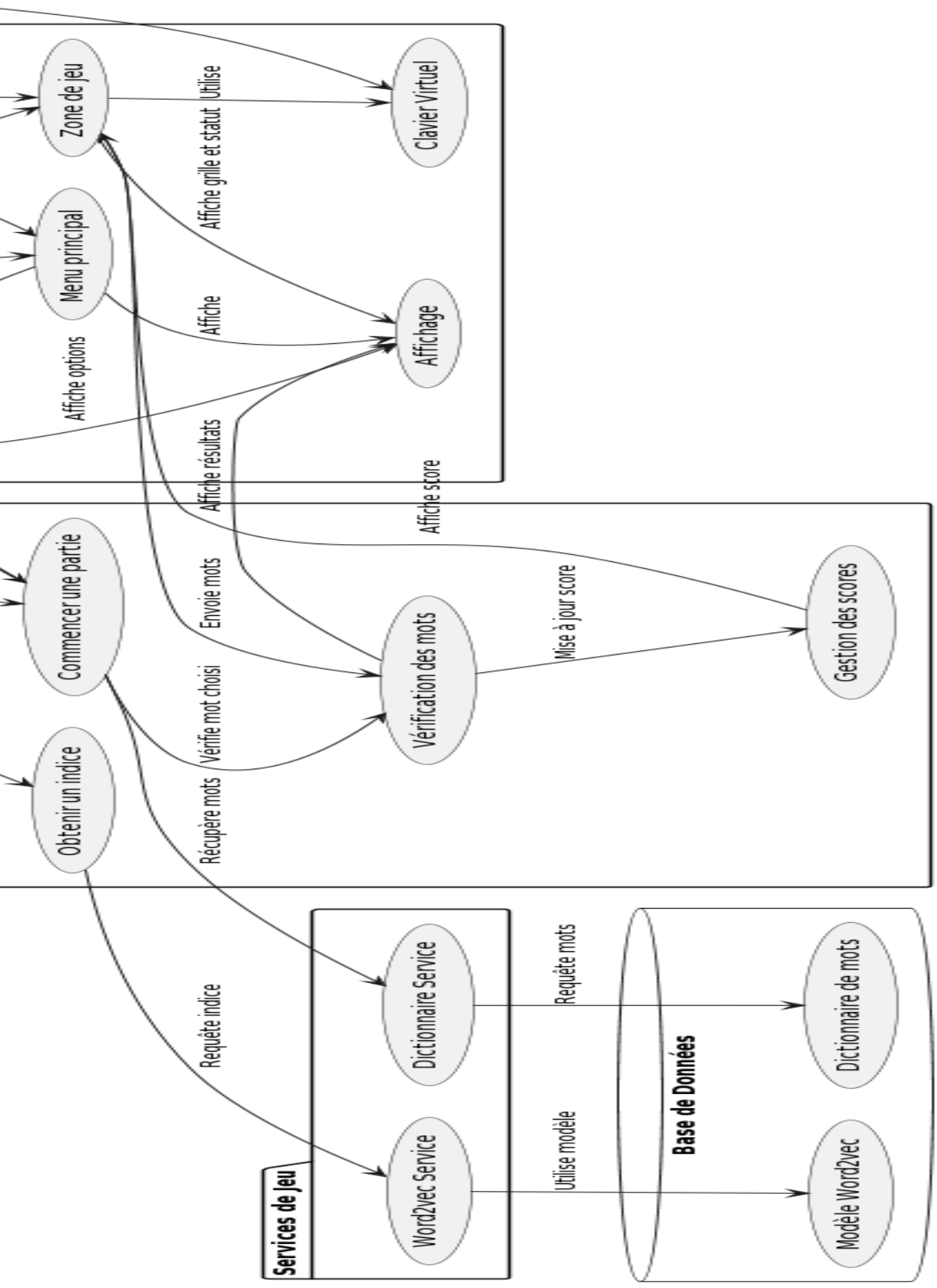

Figure 1. Diagramme UML représentant la répartition générale des tâches.

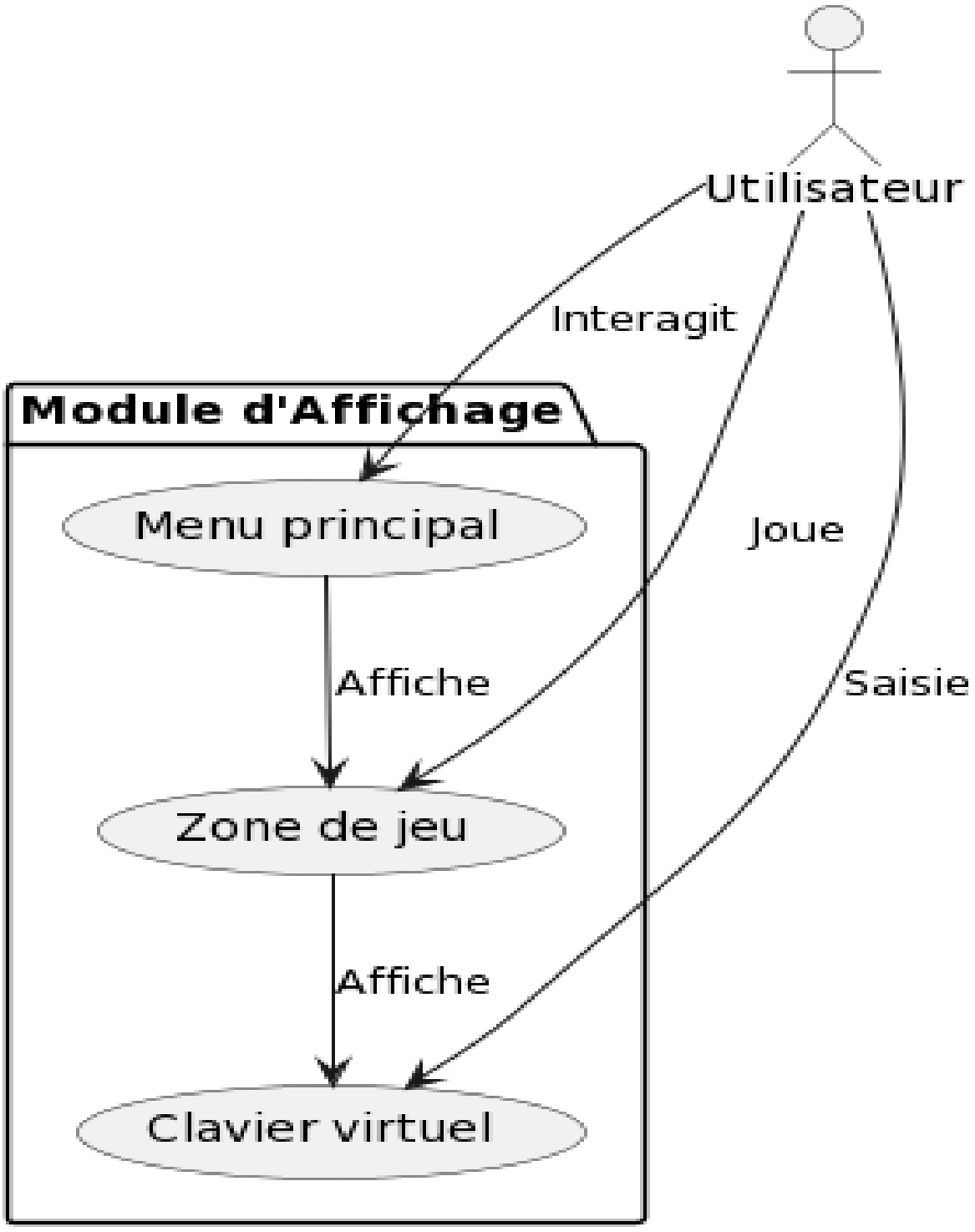

Figure 2. Diagramme UML détaillant les tâches attribuées à Dylan Gely.

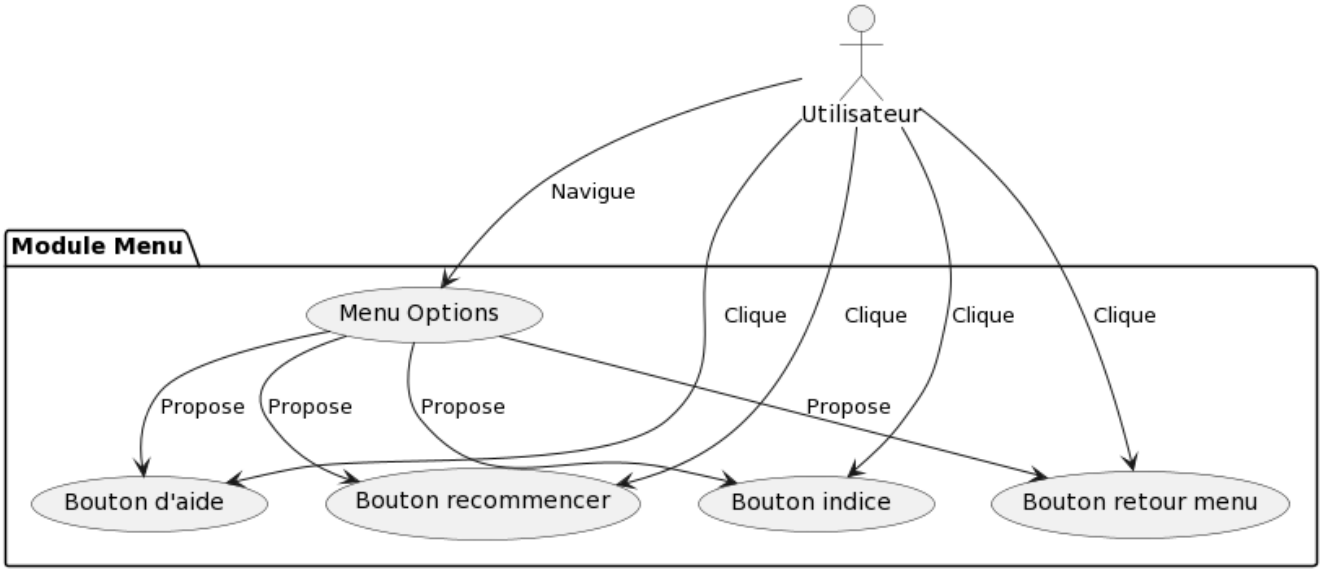

Figure 3. Diagramme UML détaillant les tâches attribuées à Anthony Genti.

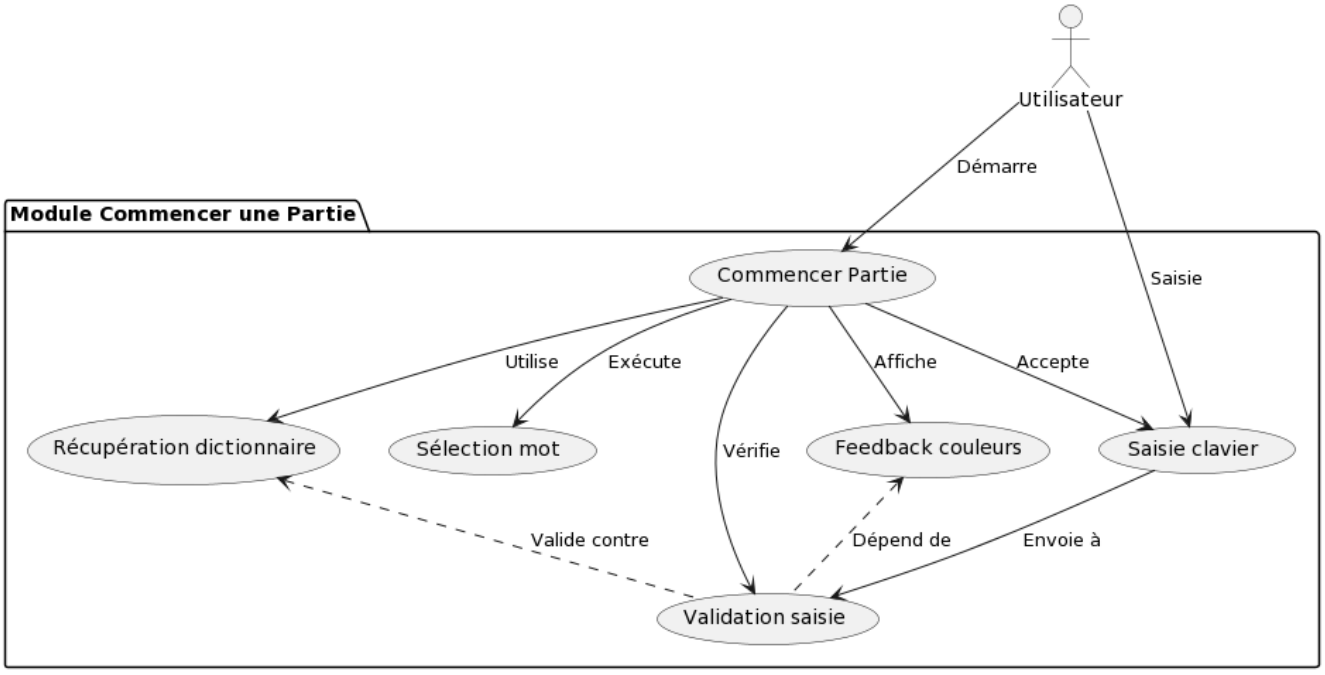

Figure 4. Diagramme UML détaillant les tâches attribuées à Youssef Boudount.

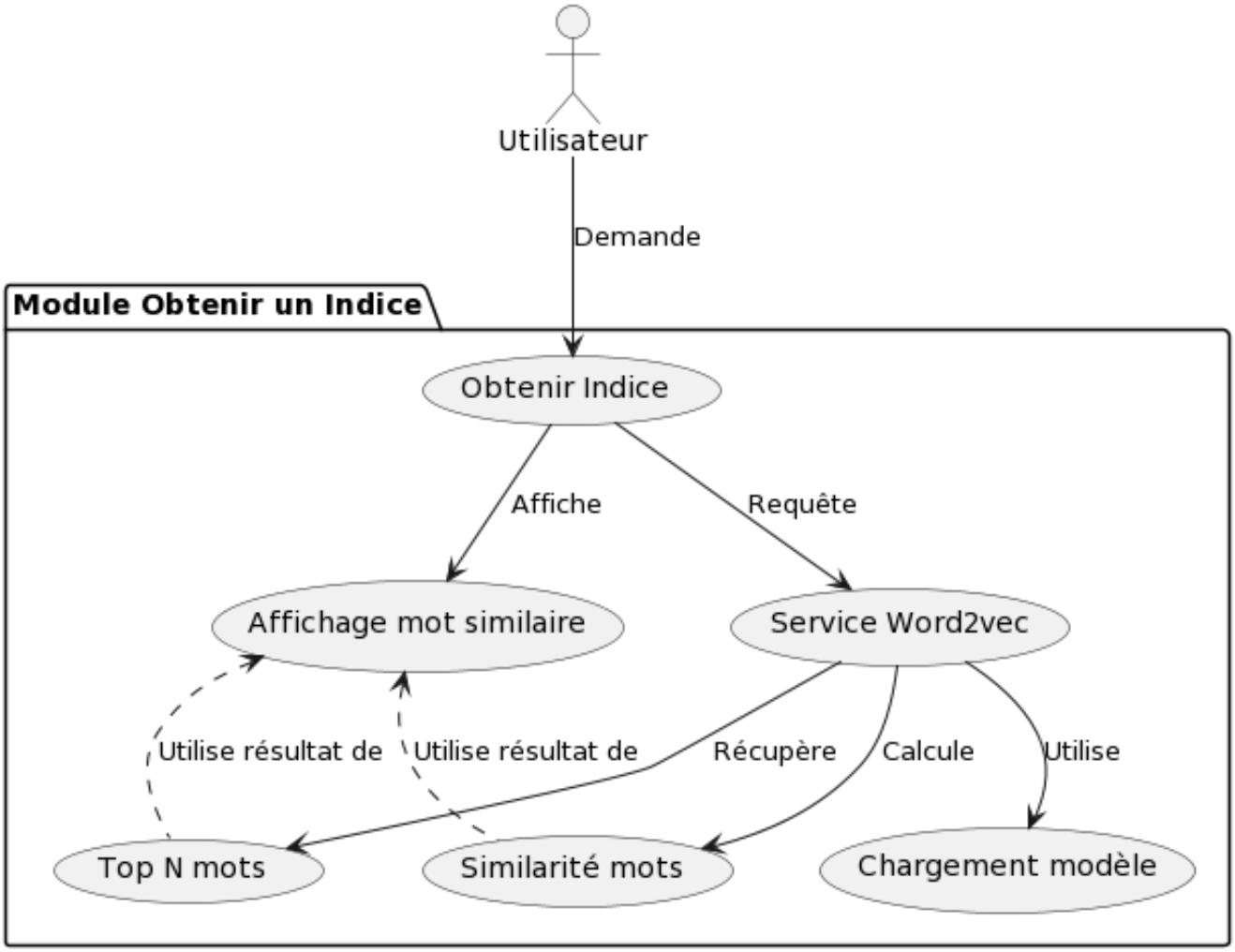

Figure 5. Diagramme UML détaillant les tâches attribuées à Axel Bertrand.

2.3 Interface utilisateur et expérience utilisateur

L'interface utilisateur a été conçue pour être intuitive et réactive. L'application offre une expérience utilisateur fluide et agréable en minimisant la complexité visuelle et en fournissant des retours clairs à l'utilisateur après chaque interaction. L'esthétique visuelle s'aligne avec la simplicité du jeu Wordle, permettant aux utilisateurs de se concentrer sur le gameplay sans distraction.

3 Développement Intermediaire

3.1 Développement de l'Interface Utilisateur (Dylan)

3.1.1 Parcours de développement

Je devais réaliser les taches du projet suivantes :
1.1 Affichage
1.2 Menu principal
1.2.1 Choisir le jeu à lancer
1.3 Zone de jeu
1.3.1 Afficher la grille de jeu
1.3.2 Afficher un clavier virtuel

Les contraintes du développement de l'interface étaient d'utiliser Javafx, la bibliothèque graphique de Java, en travaillant sur Eclipse et en réalisant un code source commenté selon le standard Javadoc : commentaires anglais (j'ai d'abord fait les commentaires en français puis je les ai traduit en anglais après relecture des règles du projet), identificateurs en anglais, commentaires explicatifs pour chaque champs, classes et méthodes.

N'aillant peu de connaissances en Java, et aucune connaissances en la bibliothèque Javafx, Il m'a fallu d'abord regarder un tutoriel sur youtube pour travailler avec sur Eclipse, puis regarder d'autres tutoriels, forum, site internet, exemples de projet en javafx pour y comprendre son fonctionnement, ses fonctions, les noms et rôles de ses différents containers, du rôles de la Stage (fenêtre) et des scènes, comment styliser, positionner et configurer des objets et container.

3.1.2 Création du portail général et du choix des jeux

J'ai d'abord commencer par réaliser l'interface graphique du portail de jeux.

Le portail de jeux contient un titre qui accueil l'utilisateur et deux boutons : l'un pour jouer à Wordle, l'autre pour un deuxième choix de jeu qui reste indisponible pour le moment.

Pour sa réalisation, j'ai voulu representer un portail de jeu classique, simple et intuitif avec un fond de couleur grise et uni, 1 titre central qui accueil l'utilisateur au-dessus de 2 gros boutons central qui permettent soit de jouer à Wordle, soit à un autre jeu (indisponible pour le moment). Le tout est placé dans un container pour organiser et positionner facilement. Lorsqu'on clique sur le bouton du jeu, les méthodes de l'interface du jeu vont crée un container qui contient tout les éléments.

Cette partie couvre 5 méthodes de mon code :

Méthode : start

La méthode start est déjà présente lorsque qu'on crée un projet javafx. Comme elle s'execute en premier, j'ai rajouté l'appel des fonctions initialiseStage et createAndSetCenterContent qui vont permettre la configuration de la fenetre et de la scene. Puis après l'appel, la fonction affiche la fenetre qui aura pour effet d'afficher la scene et donc le portail de jeux

Méthode : intializeStage

Cette méthode initialise et configure la fenêtre du logiciel et la scene de la fenêtre, elle donne à la fenetre le titre "Portail de jeux", sa scene, et une taille minimal. La scène, elle, possède une taille initial et est placé dans un container pour lui donner une couleur de fond grise pour être sobre.

Méthode : createAndSetCenterContent

Dans cette méthode, on crée le titre du portail de jeux qui accueil les utilisateurs sous forme d'un label, puis un style lui est donné (police de 24, en gras, en blanc) pour l'embellir. Puis elle crée nos 2 boutons de choix de jeux grâce à la fonction createGameButton. Le premier, Wordle, a pour effet, en cliquant dessus, d'appeler la fonction showWordleGame qui

affiche le jeu. Le deuxième est un bouton supplémentaire qui mène, en cliquant dessus, à un pop up qui informe que d'autres jeux seront proposés plus tard. Pour organiser les boutons, j'ai décidé de les mettre dans un container VBox (vertical box) qui organise ses éléments verticalement en les espaçant de 20 pixels verticalement et en les centrant au milieu de la VBox. Je décide de recréer une deuxième VBox qui va contenir la VBox des boutons, et le label pour donner un espacement vetical différent et disposer tout les éléments au centre.

Méthode : createGameButton

Cette méthode sert à créer les boutons du portail de jeux de manière à créer optimalement les mêmes boutons, de même style (police de 30, largeur de 200px, hauteur de 100px)

Méthode : showPopupMessage

Cette derniere méthode utile au portail de jeux est celle appelé lorsqu'on clique sur le bouton supplémentaire du portail, celui des autres jeux, pour pouvoir afficher un pop up informons de la disponibilité d'autres jeux que Wordle. J'ai choisi que la fenêtre du pop up soit de style UTILITY vu que ce style intègre à la fenêtre un bouton permetant de quitter la fenêtre pop up au préalable. A cette fenêtre lui est octroyé comme propriétaire la fenêtre principal du portail de jeux pour organiser hiérarchiquement les fenêtres entre elles et permettent de fermer automatiquement le pop up lorsqu'on ferme la fenêtre principal. Pour son contenu, le pop a le message passé en parametre de la fonction à son appel qui devient un label, le label est stylisé (taille de la police a 18), puis il est placé dans une VBox pour le marger à 30px du bord gauche et lui donner un fond blanc. J'interdis également le changement de taille du pop up pour le garder toujours de taille initial. Enfin, je crée la scene de la fenêtre pour pouvoir afficher le contenu dans la fenêtre du pop up en lui ajoutant la VBox et en lui donnant une taille initial. On place la scene dans la fenêtre du pop up et on affiche la fenêtre.

3.1.3 Conception de la zone de jeu avec grille et clavier virtuel

La deuxieme partie de mon code est l'affichage des éléments du jeu Wordle.

L'interface du jeu Wordle contient la grille de jeu de 6 lignes et entre 2 et 14 colonnes et un clavier virtuel pour avoir le choix entre écrire les lettres dans la grille de jeu soit via le clavier physique de l'ordinateur, soit directement via un clavier virtuel intégré dans le jeu en cliquant sur les touches avec la souris.

J'ai voulu réaliser la grille dans un container (GridPane) pour organiser au mieux la grille notamment les espacements entre les cellules. J'ai aussi ajouté un "Listener" pour ajouter un côté responsive à la grille en doublant ou non la tailles des cellules et en reduisant ou aggrandissant l'espacement entre les cellules en fonction de la largeur de la scene.

Pour l'affichage du clavier virtuel, j'ai opté pour une matrice de String qui contient toutes les lettres de l'alphabet selon l'ordre d'un clavier " azerty " et en rajoutant les touches "valider" pour pouvoir valider la ligne de jeu et "supprimer" pour pouvoir supprimer un caractère dans la ligne de jeu. J'ai aussi opté pour un container GridPane pour gérer l'organisation du clavier.

L'interface du jeu Wordle couvre 3 des méthodes du code :

Méthode : showWordleGame

Cette méthode crée et affiche les éléments du jeu en appelant la méthode createGrid qui crée la grille de jeu avec comme exemple une valeur aléatoire entre 2 et 14 pour le nombre de colonnes de la grille, et createClavier qui crée le clavier virtuel. J'ai décidé de placer les éléments du jeu dans un container de type StackPane pour les organiser au centre de la scene (root.setCenter(gamePane) et d'y mettre un fond noir pour rester sobre et pour superposer ses éléments aux éléments du portail de jeu ce qui permet de ne pas avoir à changer de fenêtre ni de scène. Je décide de placer le clavier dans un container horizontal (HBox) pour le centrer au centre horizontalement, je fais créer un bouton qui permet de

quitter le jeu Wordle et revenir au Portail de jeux (le bouton a pour effet d'appeler la méthode createAndSetCenterContent pour recréer les éléments du portail de jeux par dessus le jeu). Puis j'ai décidé de recréer un container VBox qui prend la grille, le container du clavier, et le bouton pour quitter afin de les espacer de 20 pixels et de les placer au centre. Puis je décide de placer ce container dans le container crée au début pour les afficher par dessus le fond noir.

Méthode : createGrid

Cette méthode permet la création de la grille de jeu qui sera représenté sous la forme d'un container GridPane afin de gerer la position (au centre) et d'organiser les cellules plus facilement. Pour avoir un effet responsive de la grille de jeu, je met en place un "Listener" qui va vérifier la taille de la scene que l'utilisateur peut reduire ou agrandir via la fenêtre du logiciel. Pour cela, si la largeur de la scene est inférieur à 800, on met la variable doubleCellSize qui permet ou non de doubler la taille des cellules à false et l'espacement horizontal et vertical entre les cellule est à respectivement 0 et 5. Si la largeur est supérieur à 800, leurs valeurs est 5 et 10, et on met à true la variable doubleCellSize pour indiquer qu'il faut doubler la taille des cellules. C'est dans la partie en dessous du code précédent que le doublement de taille des cellules s'effectuent ou non, grâce à une boucle for qui parcours les cellules de la grille qui sont sous forme de noeuds et que nous devons convertir en cellules textuelles pour pouvoir les modifier (la modification s'effectue sur la taille min et max de la largeur et longueur de la cellule. En mettant la même valeur pour chaque, on oblige la cellule à garder la même taille. Enfin, vient la création de la grille en permettant directement de créer les cellules de tailles doubles ou non initialement. La création de la grille se fait grâce à 2 boucles for, la première pour les lignes (6 lignes) et la deuxième pour les colonnes (aléatoire entre 2 et 14) pour chaque itération, on crée une cellule textuelle, on modifie ça taille et on l'ajoute à la grille à sa bonne place.

Méthode : createClavier

Enfin, cette dernière méthode permet de créer un clavier sous forme d'un container GridPane dont on ajoute un espacement horizontal et vertical entre ses éléments de 3 pixels et qui sera contenu dans un container VBox dont on ajoute un espacement de 10 pixels entre les éléments qu'il contient. Puis vient la création d'une matrice qui contient toutes les lettres de l'alphabet du clavier et les touches "valider" et "supprimer", puis grâce à 2 boucles for qui vont parcourir la matrice, on crée à chaque itération un bouton qui contient la lettre situé à sa place dans la matrice, puis s'il s'agit des touches "supprimer" ou "valider", on réduit la taille du bouton (police réduit à 12 pixels, largeur à 50px et longueur à 30px) qui me paraissait trop grand comparé aux autres boutons. Et puis on ajoute ce bouton à sa place dans le container GridPane qui va constituer le clavier. Enfin j'ajoute le GridPane dans le container VBox du début.

3.2 Fonctionnalités du Menu et de l'Aide (Anthony)

Dans cette section, nous explorerons les différentes fonctionnalités du menu du jeu Wordle, y compris l'utilisation de *CustomDialog* pour afficher des instructions et des indices, ainsi que les options pour recommencer et quitter la partie.

3.2.1 Implementation des menus d'Aides et des Indices

Pour offrir des instructions et des indices aux joueurs, Wordle utilise une classe nommée *CustomDialog*. Cette classe a été conçue pour afficher des fenêtres de dialogue personnalisées.

Voici comment cela fonctionne :

Lorsque les joueurs cliquent sur les boutons "Aide" ou "Indice" dans le jeu, cela déclenche

l'affichage d'une fenêtre de dialogue contenant un titre ("Aide" ou "Indice") et un message d'accompagnement correspondant.

De manière similaire, la méthode `showIndice()` affiche un indice en utilisant une fenêtre de dialogue personnalisée.

3.2.2 Recommencer la Partie

Wordle propose également la fonctionnalité de recommencer la partie à tout moment. Lorsque les joueurs appuient sur le bouton "Recommencer la Partie", la partie en cours est réinitialisée, permettant aux joueurs de tenter à nouveau de deviner le mot Wordle. Voici comment cela fonctionne :

Le bouton "Recommencer la Partie" est associé à la méthode `showWordleGame()`, qui génère une nouvelle partie en appelant la méthode `showWordleGame()`.

3.2.3 Quitter la Partie

Enfin, Wordle permet aux joueurs de quitter la partie en cours et de revenir au menu principal du jeu. Lorsque le bouton "Quitter le jeu" est cliqué, la partie actuelle est fermée, et l'écran d'accueil est rétabli.

Ce bouton "Quitter le jeu" est lié à la méthode `createAndSetCenterContent()`, qui réinitialise l'écran pour afficher le menu principal du jeu.

Ces fonctionnalités du menu, y compris l'utilisation de *CustomDialog*, la possibilité de recommencer une partie et de quitter la partie, améliorent l'expérience de jeu en offrant une plus grande flexibilité aux joueurs.

3.2.4 Effets Visuels et Animations

Voici plusieurs fonctionnalités d'effets visuels et d'animations qui améliorent l'expérience utilisateur.

Désactivation d'une Ligne de la Grille

La méthode `disableRow` permet de désactiver une ligne spécifique de la grille, rendant les cellules correspondantes cachées et inutilisables.

Animation de Retournement de Cellule

La méthode `animateFlipTile` anime une cellule en effectuant un retournement visuel avec une couleur de feedback.

Secousse de la Grille

La méthode `shakeGrid` crée une animation de secousse pour la grille, ajoutant un effet dynamique lors de certains événements.

Animations de Victoire

La méthode `executeWinAnimations` gère les animations de victoire, notamment le fondu de la grille et l'agrandissement/rétrécissement. Elle prend également en charge l'affichage d'un message de succès.

Affichage de la Nouvelle Grille avec un Message

La méthode `showNewGridWithMessage` montre une nouvelle grille avec une animation de fondu entrant et affiche un message pendant un certain temps.

Ces fonctionnalités d'effets visuels et d'animations ajoutent une dimension interactive et visuellement attrayante au jeu Wordle.

3.3 Mécanismes de Jeu et Validation (Youssef)

3.3.1 Récupération et Stockage du Dictionnaire

Class : WiktionaryScraper
Cette classe effectue du web scraping sur le Wiktionnaire pour créer un dictionnaire de mots pour le jeu Wordle. Elle utilise la bibliothèque `Jsoup` pour analyser le HTML et `HttpClient` pour gérer les requêtes HTTP. Le processus commence par établir une connexion au Wiktionnaire, suivie de la récupération du contenu HTML. Le document HTML est ensuite analysé pour extraire les liens contenant des mots. Un filtre est appliqué pour éliminer les mots non désirés, comme les noms de pages spéciales ou des liens non pertinents. Cette étape est cruciale pour assurer la qualité et la pertinence des mots récupérés. Finalement, les mots sélectionnés sont ajoutés à un `JsonArray` et sauvegardés dans un fichier JSON.

Méthode : isUndesiredWord
La méthode `isUndesiredWord` joue un rôle clé dans le filtrage des mots. Elle examine chaque mot récupéré et détermine s'il s'agit d'un élément indésirable, en le comparant à une liste prédéfinie de mots non souhaités. Cette liste inclut des noms de pages, des titres de sections et d'autres termes non pertinents pour le jeu. Cette méthode garantit que le dictionnaire final contient uniquement des mots valides pour le jeu.

3.3.2 Sélection Aléatoire de Mots

Class : RandomWordSelector
La classe `RandomWordSelector` est responsable de choisir un mot au hasard à partir du fichier JSON. Elle lit d'abord le fichier pour charger l'ensemble des mots, puis utilise la classe `Random` pour sélectionner un mot de manière aléatoire. Cette sélection est essentielle pour assurer la variété et l'imprévisibilité des parties de Wordle.

Méthode : getRandomWord
La méthode `getRandomWord` de cette classe filtre d'abord les mots en fonction de leur longueur, conformément aux règles de Wordle, puis sélectionne aléatoirement l'un de ces mots. Cette approche garantit que le mot choisi correspond à la longueur requise pour chaque partie du jeu.

3.3.3 Validation des Mots en Français

Class : FrenchWordChecker
La classe `FrenchWordChecker` assure que les mots proposés par les joueurs sont valides en français. Elle utilise un fichier JSON contenant un dictionnaire général pour vérifier l'existence des mots.

Méthode : isWordInFrenchDictionary
Cette méthode parcourt le dictionnaire pour vérifier si un mot spécifié est présent. Elle effectue une comparaison insensible à la casse pour chaque mot du dictionnaire, garantissant ainsi que les mots sont validés indépendamment de leur casse.

3.3.4 Mécanismes Principaux du Jeu

Class : WordleGame
La classe `WordleGame` regroupe la logique du jeu, y compris la génération du mot cible et la vérification des propositions du joueur.

Méthode : compareWords
La méthode `compareWords` est centrale dans le gameplay. Elle compare le mot proposé par

le joueur au mot cible, en vérifiant lettre par lettre. Pour chaque lettre, elle détermine si elle est correcte et bien placée, correcte mais mal placée, ou incorrecte. Cette vérification est essentielle pour fournir le feedback coloré caractéristique de Wordle aux joueurs.

Méthode : startGame
La méthode `startGame` initialise ou réinitialise le mot cible pour une nouvelle partie. Elle appelle `getRandomWord` pour générer un nouveau mot cible, garantissant ainsi que chaque partie commence avec un défi nouveau et intéressant.

3.4 Interface Graphique et Interactivité (Youssef)

3.4.1 Construction du Clavier Virtuel

Méthode : createClavier
Sur la base du travail de mon camarade Dylan Gely, j'ai réorganisé les boutons du clavier virtuel et introduit de nouveaux éléments, comme des nouveaux symboles pour les boutons supprimer et valider, qui ajoutent des fonctionnalités essentielles pour améliorer l'interaction avec le jeu. Le style des boutons a été personnalisé pour s'aligner avec l'esthétique générale de l'application, renforçant ainsi l'unité visuelle. J'ai également ajusté la gestion du focus des boutons pour éviter qu'ils n'interfèrent avec la saisie dans la grille. Ces modifications visent à offrir une expérience utilisateur plus intuitive et agréable, en rendant le clavier virtuel plus facile à utiliser et en améliorant son intégration dans l'ensemble de l'interface.

Méthode : bindClavierToGrid
bindClavierToGrid connecte chaque bouton du clavier virtuel à la grille de jeu. Lorsqu'un bouton est pressé, la méthode identifie la cellule active dans la grille et y insère la lettre correspondante, ou effectue une action spéciale (comme effacer ou soumettre un mot). Cette méthode crée un pont direct entre le clavier et la grille, permettant une saisie rapide et intuitive, essentielle pour une expérience de jeu fluide.

3.4.2 Implémentation de la Grille de Jeu

Méthode : createGrid
Pour la fonction createGrid, j'ai apporté des changements substantiels à la conception et à la fonctionnalité de la grille de jeu faite par Dylan Gely. J'ai modifié la taille des cellules pour une meilleure visibilité et interaction, en adaptant leur style pour une cohérence esthétique avec l'ensemble de l'application. De plus, j'ai intégré des écouteurs pour gérer efficacement la saisie du texte et les interactions clavier, facilitant ainsi la saisie des utilisateurs et prévenant les erreurs communes. J'ai également implémenté une logique pour déplacer automatiquement le focus entre les cellules, ce qui fluidifie l'expérience utilisateur. Ces améliorations visent à rendre l'interface plus réactive et conviviale, tout en garantissant une interaction utilisateur sans faille.

Méthode : getTextFieldAt
getTextFieldAt permet d'accéder à une cellule spécifique de la grille en fonction de ses coordonnées (colonne et ligne). Cette méthode est essentielle pour lire et modifier le contenu des cellules lors de l'interaction avec le clavier virtuel ou pour valider les réponses. Elle offre une manière efficace de cibler des cellules spécifiques sans avoir besoin de parcourir toute la grille, optimisant ainsi les performances du jeu.

Méthode : updateGridCell
La fonction updateGridCell est appelée pour modifier le contenu d'une cellule de la grille. Elle est utilisée pour insérer ou supprimer des lettres dans les cellules, en réponse aux actions de l'utilisateur sur le clavier virtuel. Cette méthode garantit que les modifications apportées

à la grille sont cohérentes avec les entrées de l'utilisateur, contribuant à une expérience de jeu interactive et réactive.

Méthode : getSelectedCell
getSelectedCell identifie quelle cellule de la grille est actuellement active ou a le focus. Cette capacité est cruciale pour savoir où insérer une lettre lorsqu'un bouton du clavier virtuel est pressé, ou pour déterminer de quelle cellule supprimer une lettre. Elle permet une interaction contextuelle entre le clavier virtuel et la grille, rendant le jeu plus intuitif.

3.4.3 Fenêtre de Jeu et Contrôleurs

Méthode : showWordleGame
Dans la reprise du travail de Dylan Gely sur la fonction showWordleGame, j'ai introduit plusieurs modifications et ajouts significatifs. Premièrement, la récupération d'un mot cible aléatoire et l'ajustement de la taille de la grille en fonction de sa longueur améliorent directement la logique du jeu, le rendant plus dynamique et adaptable. J'ai également implémenté un système de gestion des erreurs via un errorLabel, ce qui augmente la convivialité en fournissant des retours directs sur les actions de l'utilisateur. Ces modifications visent à rendre l'expérience de jeu plus engageante, intuitive et esthétiquement plaisante.

Méthode : startGame
startGame est le point d'entrée pour lancer le jeu. Elle appelle showWordleGame pour construire l'interface utilisateur et préparer le terrain de jeu. Cette méthode assure que le jeu est correctement initialisé et prêt à être joué, fournissant une transition fluide de l'écran de démarrage au jeu lui-même.

Méthode : validateGuess
validateGuess est une fonction clé qui gère la logique de validation des propositions du joueur. Lorsqu'un mot est soumis, cette méthode vérifie si le mot est correct, met à jour la grille avec des indications visuelles (comme changer la couleur des cellules), et détermine si le joueur a gagné ou doit essayer à nouveau. Cette méthode est au cœur de la dynamique du jeu, rendant le processus de deviner les mots à la fois stimulant et gratifiant.

3.5 Système d'Indice et Intégration de Word2vec (Axel)

3.5.1 Intégration du système d'indices basé sur Word2vec

Script Python : word2vec.py
Ce script sert permet de charger le modèle dans un vecteur de mot et donne accès aux fonctions *similarity* et *most_similar* de la librarie gensim. Il propose des fonctions wrapper qui vérifient que les paramètres en entrée (les deux mots pour *similarity* ou le mot pour *most_similar* doivent faire partie du vecteur de mots, et le nombre de mots à recevoir pour *most_similar* doit être strictement supérieur à 0) afin d'afficher proprement un message d'erreur et d'éviter de planter le script.

Script Python : server.py
Ce script est un serveur http simple réalisé avec la librairie flask et permet d'accéder aux deux fonctions du script word2vec.py via des requêtes GET. J'avais initialement programmé une classe Java capable de lancer le script word2vec.py et d'en récupérer le résultat mais cela nécessitait de recharger le vecteur de mots du modèle à chaque appel ce qui prend environ 2 secondes et aurait rendu les interactions très lentes. À la place, ce script de serveur sera lancé et chargera le vecteur de mots une fois au début du programme et ne fera ensuite que répondre à des requêtes envoyé depuis la classe Java Server ce qui rend les interactions entre le code Java et le serveur très fluide.

3.5.2 Communication entre les modules Java et Python via un serveur HTTP

Class Server
Cette classe permet de lancer/ferme le serveur en lançant un processus qui contient le serveur Flask. Elle donne aussi des méthodes pour lancer des requêts et interpréter leurs résultats pour être ensuite utilisés dans les autres modules Java.

Méthodes : getSimilarity et getMostSimilar
Ces fonctions permet de lancer des requête pour obtenir accéder aux résultats des fonctions *similarity* et *most_similar* de la librairie gensim et d'interpréter les résultats pour être utilisables en Java. La méthode getSimilarity convertit simplement le résultat en un nombre flottant tandis que la méthode getMostSimilar renvoie un ArrayList d'objets de type GensimPair, une classe qui représente une association entre un mot et sa similarité au mot utilisé dans la requête. Ces méthodes se chargent également d'interpréter les messages d'erreur du script Python word2vec.py pour throw des exceptions dérivées de la classe GensimException afin de représenter les différents types d'erreur qui peuvent être rencontré en communiquant avec le serveur (à savoir : *ServerClosedException* si le serveur est fermé, *KeyNotPresentException* si un mot utilisé en paramètre ne fait pas partie du vecteur de mots du modèle et *InvalidNumberOfWordsException* si le nombre de mots passé en paramètre de getMostSimilar n'est pas strictemet positif).

Méthode : getHint
Cette méthode sert est une fonction wrapper autour de getMostSimilar qui utilise des propriétés statiques de la classe pour faciliter l'obtention d'un indice : elle ne prend que le mot cible en paramètre, le sauvegarde et envoi un nouvel indice à chaque appel, en se chargeant d'envoyer une nouvelle requête vers le server Python si nécessaire. Elle réinitialise ce qui doit l'être quand elle reçoit un mot différent de l'ancien mot sauvegardé pour éviter d'avoir à le faire dans l'application principale.

Méthodes utilitaires
La classe Server propose également des méthodes utilisaires : start et close afin de gérer l'état du serveur, isClosed pour connaître l'état du serveur et getAdress pour obtenir l'adresse du serveur Python.

4 Développement Final

.

4.1 Améliorations du module d'indice (Axel)

Problèmes initaux

Bien que le module d'indice était fonctionnel au moment du rapport intermédiaire, des améliorations étaient encore possible. D'abord, pour lancer le serveur il fallait que la partie en Python charge le modèle de données ce qui est plutôt long (plusieurs secondes) et ralentit donc le chargement de l'application au lancement. De plus, le module était un wrapper autour du serveur Python et ne faisait essentiellement que donner accès aux fonctions sans plus ce qui nécessitait potentiellement plus de code au niveau du controller pour faire appel au module d'indice.

Liste d'améliorations

- *Parallèlisation* : Le module propose désormais une fonction permettant d'initier le lancement du serveur sans attendre la fin ce qui permet d'économiser plusieurs secondes sur le temps de chargement de l'application. Le champ représentant le statut du serveur a été changé en conséquence en rajoutant "STARTING" comme valeur possible.
- *Simplification de l'appel d'indice* : La fonction *getHint* a été rajouté pour réduire la quantité de code nécessaire dans le controller. Elle permet d'obtenir un indice à partir d'un mot, en donnant à chaque appel un nouveau mot. Elle se charge de garder en mémoire la dernière requête au serveur et d'en refaire lorsque c'est nécessaire pour avoir plus de mots et s'occupe aussi de réinitialiser les champs nécessaires lorsqu'elle reçoit un mot différent du dernier.
- *Attente du serveur* : La fonction *waitServerStarting* a été ajouté pour permettre de gérer le cas où l'application a besoin d'un indice alors que le serveur n'a pas fini de se lancer. Elle est appelée si nécessaire par la fonction*getHint*.
- *Limite d'indice* : Des fonctions utilitaires pour mettre, changer ou enlever une limite au nombre d'indices donnés par le module ont été rajoutés même si elles sont actuellement inutilisées.

4.2 Ajout des Scores (Anthony)

Dans notre application, la gestion des scores et des performances des joueurs est une fonctionnalité essentielle. Pour implémenter cette fonctionnalité, plusieurs classes et méthodes ont été conçues pour traiter différentes tâches liées au score.

4.2.1 Afficher le Temps Écoulé Depuis le Début de la Partie

Classe GameTimer : Cette classe gère le chronométrage de la partie. Elle utilise un objet Timeline de JavaFX pour mettre à jour régulièrement un label d'affichage du temps.

- **Fonction startTimer :** Lance un chronomètre au début de chaque partie. Le temps est affiché dans un format hh:mm:ss sur l'interface du jeu.
- **Fonction updateTimeDisplay :** Met à jour l'affichage du temps écoulé. Cette méthode est appelée à chaque seconde écoulée pour refléter le temps passé depuis le début de la partie.

4.2.2 Afficher la Série de Victoires Actuelle

Classe PlayerStats : Gère les statistiques du joueur, y compris les séries de victoires.

- **Fonction updateWinStreak :** Met à jour la série de victoires actuelle après chaque partie. Elle incrémente un compteur de victoires à chaque succès consécutif et le remet à zéro en cas de défaite.
- **Affichage dans l'UI :** Un label dédié dans l'interface utilisateur montre la série de victoires actuelle, aidant les joueurs à suivre leurs performances.

4.2.3 Bouton pour Afficher les Meilleurs Scores

Classe Scoreboard : Contient les logiques pour afficher et gérer le tableau des meilleurs scores.

- **Fonction displayTopScores :** Lorsqu'un utilisateur clique sur le bouton "Meilleurs Scores", cette méthode est appelée pour afficher un dialogue ou une nouvelle fenêtre avec les meilleurs scores enregistrés.
- **Interface Utilisateur :** Un bouton "Meilleurs Scores" est présent sur l'écran principal, donnant accès facilement aux scores les plus élevés.

4.2.4 Sauvegarder les Meilleurs Scores dans un Fichier

Classe ScoreManager : Gère la sauvegarde et la récupération des scores depuis un fichier.

- **Fonction saveScores :** Cette méthode est responsable de l'écriture des meilleurs scores dans un fichier de données persistant. Elle est appelée après chaque partie pour mettre à jour le fichier si nécessaire.
- **Fonction loadScores :** Au démarrage de l'application, cette méthode charge les meilleurs scores depuis le fichier et les met à disposition pour être affichés via Scoreboard.

4.2.5 Stratégies de Calcul des Scores

Le calcul des scores est basé sur le temps écoulé et le niveau de difficulté. Voici les détails des stratégies utilisées :

Niveaux de Difficulté :

- **Facile :** Moins de lettres ou mots plus simples. Le score est augmenté à un taux standard.
- **Moyen :** Plus de lettres et mots de difficulté moyenne. Le score est augmenté à un taux légèrement supérieur à celui du niveau facile.
- **Difficile :** Plus de lettres et mots complexes. Le score augmente à un taux nettement plus élevé pour refléter la difficulté accrue.

Temps Écoulé : Le score est influencé par le temps pris pour terminer la partie. Moins de temps signifie un score plus élevé. Un système de bonus temporel peut être mis en place, où les joueurs gagnent des points supplémentaires pour terminer rapidement.

Séries de Victoires : Les séries de victoires sont également prises en compte dans le calcul du score. Une série de victoires plus longue entraîne des multiplicateurs de score, récompensant ainsi la cohérence et l'habileté des joueurs.

Classe ScoreCalculator : Cette classe contient des méthodes pour calculer le score basé sur ces paramètres. Elle prend en compte le niveau de difficulté, le temps écoulé et la série de victoires pour générer un score final pour chaque partie.

Conclusion : En combinant ces éléments, l'application offre un système de scoring dynamique et motivant, incitant les joueurs à améliorer continuellement leurs compétences tout en tenant compte de la difficulté et de la rapidité de leur jeu.

4.3 Réorganisation et Refonte du package view (Youssef)

4.3.1 Problème initial avec le code de base

Le code de base souffrait principalement d'un problème de lisibilité et de maintenabilité en raison de sa longueur et de la concentration de multiples fonctionnalités dans un nombre limité de fichiers. Cette situation était particulièrement évidente dans le package `view`, où seulement deux fichiers - `CustomDialog.java` et `Main.java` - géraient toutes les interfaces et logiques de notre application Wordle. Cette concentration dense de code rendait difficile la compréhension, la modification et l'extension du programme, surtout avec l'ajout prévu d'autres jeux.

4.3.2 Modifications apportées

Pour résoudre ce problème, la structure du package `view` a été réorganisée et plusieurs nouveaux fichiers ont été introduits, afin de séparer les différentes responsabilités et de rendre le code plus clair et facile à maintenir. Les modifications principales sont :

- **Séparation des responsabilités** : Au lieu d'avoir tout le code dans `Main.java`, les différentes parties de l'interface utilisateur ont été séparées en fichiers distincts.
- **Introduction de l'interface `GameInterface`** : Cette interface définit une méthode `showGame(Stage stage)`, qui est implémentée par les différentes vues de jeu. Cela permet une plus grande flexibilité et une meilleure organisation du code relatif à chaque jeu.
- **Création de `GamePortal.java`** : Ce fichier sert de point central pour accéder aux différents jeux. Il gère l'affichage du menu principal et la navigation entre les différents jeux.
- **Développement de `WordleView.java`** : Ce fichier est spécifiquement conçu pour gérer l'interface et la logique du jeu Wordle. Il implémente l'interface `GameInterface` et contient toutes les méthodes nécessaires pour jouer à Wordle, y compris la création de la grille, la gestion des entrées de l'utilisateur, et l'affichage des animations et des aides.

4.3.3 Nouvelles Fonctions et Fichiers Ajoutés

- **GameInterface.java** :
  Fonction : Définit un contrat pour les vues de jeu, imposant la mise en œuvre de la méthode `showGame(Stage stage)`.
  But : Permet une uniformité dans la manière dont les jeux sont lancés et affichés, facilitant l'ajout de nouveaux jeux dans le futur.
- **GamePortal.java** :
  Fonction : Sert de hub pour lancer différents jeux. Contient des méthodes pour initialiser la scène principale et afficher les options de jeu disponibles.
  But : Centralise la navigation entre différents jeux, rendant l'expérience utilisateur plus fluide et organisée.
- **WordleView.java** :
  Fonction : Gère spécifiquement l'interface et la logique du jeu Wordle. Implémente toutes les fonctionnalités spécifiques à Wordle, telles que la création de la grille de jeu, la gestion des entrées de l'utilisateur, et l'affichage des aides et des indices.
  But : Isoler le code spécifique à Wordle pour une meilleure lisibilité et maintenabilité, et permettre des modifications ou des extensions spécifiques au jeu Wordle sans affecter le reste de l'application.

4.3.4 Analyse et Comparaison des Versions

Améliorations de la Structure du Code
Dans l'ancienne version, le package `view` était encombré et peu structuré, ce qui rendait difficile la navigation dans le code et l'ajout de nouvelles fonctionnalités. Avec la nouvelle version, chaque jeu et fonctionnalité a son propre fichier, clarifiant ainsi la structure globale et facilitant la maintenance et l'extension du code.

Expérience Utilisateur et Intégration de Nouveaux Jeux
Les nouvelles fonctions ajoutées améliorent non seulement la structure du code, mais offrent également une meilleure expérience utilisateur. Par exemple, avec l'introduction de `GamePortal.java`, l'application peut désormais accueillir plusieurs jeux de manière ordonnée, chaque jeu ayant sa propre classe de gestion comme `WordleView.java`. Cela permet d'ajouter de nouveaux jeux à l'application sans perturber l'architecture existante.

Modularité et Responsabilité Unique
`WordleView.java` apporte une amélioration significative en termes de modularité et de responsabilité unique. Auparavant, `Main.java` contenait une grande partie de la logique du jeu Wordle, ce qui le rendait volumineux et difficile à gérer. Maintenant, toute la logique spécifique au jeu Wordle est encapsulée dans `WordleView.java`, ce qui rend le code plus lisible, plus facile à déboguer et à maintenir.

Conclusion
En résumé, la nouvelle version du code offre une structure beaucoup plus claire et modulaire, avec une séparation distincte des responsabilités. Cela facilite la compréhension, le développement et la maintenance de l'application, en particulier avec l'ajout prévu de nouveaux jeux et fonctionnalités.

4.4 Autres Améliorations (Youssef)

Dans cette section du rapport, nous abordons les améliorations apportées au projet suite aux retours des enseignants lors de l'évaluation intermédiaire. Chaque problème identifié a été soigneusement analysé et des solutions spécifiques ont été mises en œuvre pour les résoudre.

4.4.1 Amélioration de la Lisibilité des Lettres dans les Cases

Lors de l'évaluation intermédiaire, les enseignants ont remarqué que les lettres à l'intérieur des cases étaient petites et mal alignées, ce qui rendait la lecture difficile, particulièrement pour les joueurs ayant des problèmes de vision. Pour améliorer la lisibilité, la taille des lettres a été augmentée et elles sont maintenant centrées dans chaque case. Cette amélioration a été réalisée en ajustant les propriétés de style CSS des TextField dans la classe `WordleView`.

4.4.2 Instructions du Jeu via le Bouton Aide

Les enseignants ont également noté que les instructions du jeu n'étaient pas immédiatement accessibles. Un bouton "Aide" a été ajouté, qui, lorsqu'il est cliqué, ouvre une fenêtre de dialogue contenant les règles et instructions du jeu Wordle. Cette fonctionnalité utilise un `CustomDialog` pour afficher le texte des instructions.

4.4.3 Intégration du Module d'Indice avec un Serveur

Il a été souligné que le jeu manquait d'une fonction d'indice dynamique. Le bouton "Indice" a été relié à un serveur backend pour obtenir un indice du mot à deviner. Cette intégration nécessitait une communication réseau dans la classe `WordleView`.

4.4.4 Fonctionnalité de Recommencement de Partie

Lors de l'évaluation intermédiaire, il a été remarqué que les joueurs ne pouvaient pas recommencer une partie en cours sans redémarrer l'application. Un bouton "Recommencer la partie" a été ajouté pour permettre aux joueurs de recommencer une partie à tout moment.

4.4.5 Centrage et Responsivité de la Grille

Les enseignants ont observé que la grille de jeu n'était pas bien centrée et que la fenêtre de l'application n'était pas responsive. La grille a été centrée correctement, et la fenêtre a été rendue responsive pour s'adapter à la taille de la grille.

4.4.6 Affichage du Score et du Temps

Enfin, il a été noté que l'affichage du score et du temps n'était pas optimal. Désormais, le score est affiché en haut à gauche de la fenêtre, a coté du temps écoulé. Cette disposition permet une lecture aisée et rapide des informations essentielles. Les méthodes `updateScoreDisplay` et `updateTimeDisplay` dans la classe `WordleView` ont été utilisées pour mettre à jour ces informations en temps réel pendant le jeu.

Chacune de ces améliorations a été mise en œuvre avec l'objectif d'optimiser l'expérience utilisateur et de répondre aux besoins identifiés par les enseignants. Ces changements ont non seulement amélioré l'aspect fonctionnel et esthétique du jeu, mais ont également contribué à rendre le jeu Wordle plus accessible, engageant et agréable pour tous les joueurs.

5 Tests Intermédiaires

Cette section détaille les stratégies et méthodologies de tests adoptées pour assurer la fiabilité et la robustesse de notre application Wordle. Les tests sont cruciaux pour identifier et résoudre les problèmes avant la mise en production.

5.1 Stratégies de tests unitaires et d'intégration

Dans cette sous-section, nous décrivons notre approche pour les tests unitaires et d'intégration. Les tests unitaires ont été conçus pour vérifier la fonctionnalité de chaque composant individuel du jeu, tels que le générateur de mots, le système de scoring et l'interface utilisateur. Nous avons utilisé des frameworks comme JUnit pour automatiser ces tests, permettant une validation rapide et efficace de chaque fonctionnalité. Les tests d'intégration, quant à eux, ont été mis en place pour s'assurer que tous les composants du jeu fonctionnent harmonieusement ensemble. Cette approche combinée garantit que chaque élément du jeu fonctionne correctement en isolation, ainsi qu'en conjonction avec les autres composants.

5.1.1 Classe FrenchWordChecker

- Test du Constructeur : Vérifier l'initialisation correcte du dictionnaire.
- Test de isWordInFrenchDictionary :
  - Confirmer la vérification correcte des mots existants.
  - Tester avec des mots non existants.
  - Vérifier la gestion des cas de chaînes vides et null.

5.1.2 Classe RandomWordSelector

- Test du Constructeur : Assurer le chargement adéquat des mots depuis le JSON.
- Test de getRandomWord :
  - Tester la sélection aléatoire des mots de différentes longueurs.
  - Vérifier la répartition uniforme des mots sélectionnés.

5.1.3 Classe WordleGame

- Test du Constructeur : Vérifier l'initialisation du jeu avec un mot cible.
- Test de compareWords :
  - Tester différents scénarios de comparaison de mots.
  - Vérifier la cohérence des retours de couleur.
- Test de startGame : Vérifier le démarrage d'une nouvelle partie avec un nouveau mot.

5.1.4 Classe WordList

- Test de la Structure de Données :
  - Vérifier la capacité de stockage.
  - Tester l'accès aux éléments.

5.1.5 Classe WiktionaryScraper

- Test de la méthode main :
  - Assurer la récupération correcte des mots.
  - Tester avec différents URL et situations d'erreur réseau.
- Test de isUndesiredWord :
  - Confirmer l'exclusion correcte des mots indésirables.
  - Tester avec une variété de mots indésirables.

5.2 Résultats des tests et gestion des anomalies

Dans cette dernière sous-section, nous présentons les résultats de nos efforts de test et comment nous avons géré les anomalies détectées pour chaque classe.

5.2.1 Classe FrenchWordChecker

- Les tests ont montré que le constructeur chargeait correctement le dictionnaire, mais échouait silencieusement sur des fichiers JSON mal formés. Une gestion d'erreurs plus robuste a été implémentée en conséquence.
- Les tests de 'isWordInFrenchDictionary' ont révélé des problèmes avec les mots comportant des caractères spéciaux, menant à l'ajout d'une normalisation des entrées.

5.2.2 Classe RandomWordSelector

- Le test du constructeur a confirmé la bonne lecture des mots depuis le fichier JSON, mais a soulevé des problèmes de performance avec de très gros fichiers.
- Les tests sur 'getRandomWord' ont initialement montré une distribution inégale des mots sélectionnés, nécessitant une révision de l'algorithme de sélection aléatoire.

5.2.3 Classe WordleGame

- Le test du constructeur a validé l'initialisation correcte, mais a révélé des lacunes dans la gestion des fichiers introuvables.
- Les tests sur 'compareWords' ont révélé des incohérences dans le feedback des couleurs pour certains scénarios de mots mal placés.

5.2.4 Classe WordList

- Les tests ont confirmé la bonne gestion des listes de mots, mais ont révélé des limites lors du traitement de listes extrêmement longues.

5.2.5 Classe WiktionaryScraper

- Les tests de la méthode 'main' ont validé le scraping des mots mais ont révélé des problèmes lors de la manipulation de structures HTML complexes.
- Les tests de 'isUndesiredWord' ont montré une bonne exclusion des mots indésirables, mais ont nécessité des ajustements pour couvrir certains cas limites non prévus.

Dans chaque cas, les anomalies ont été documentées et classées selon leur sévérité. Des correctifs ont été développés, testés et intégrés dans le projet principal pour les problèmes critiques, tandis que les problèmes moins urgents ont été planifiés pour résolution dans les cycles de développement suivants. Cette approche méthodique a permis une amélioration continue de la stabilité et de la performance de l'application.

6 Tests Finaux

Cette section présente les stratégies et méthodologies de tests finaux adoptées pour notre application Wordle, incluant la gestion des anomalies détectées.

6.1 Classe CustomDialog

- **Tests Unitaires**
  - Test du Constructeur : Vérifier l'affichage correct du titre et du message, ainsi que la largeur et le style de la fenêtre.
  - Test des Interactions avec les Boutons : Simuler un clic sur le bouton "Fermer" et vérifier la fermeture de la fenêtre.

6.2 Classe GamePortal

- **Tests Unitaires**
  - Test du Constructeur : Assurer que le primaryStage et WordleGame sont correctement initialisés. Vérifier que initializeStage() est appelé.
  - Test de initializeStage : Confirmer le style, la taille et le titre de la fenêtre principale.
  - Test de initializeGamePortalContent : Vérifier la création correcte des éléments UI (labels, boutons, etc.) et leur bon positionnement dans le BorderPane.
- **Tests d'Intégration**
  - Interaction avec WordleGame : Vérifier que le lancement du jeu Wordle se fait correctement à partir du portail.
  - Test des Pop-Ups : Simuler les clics sur différents boutons et vérifier l'affichage correct des messages pop-up.

6.3 Classe Main

- **Tests Unitaires**
  - Test de la méthode start : Confirmer le bon chargement et l'affichage de la scène principale. Vérifier l'intégration correcte des composants du jeu dans la scène.
- **Tests d'Intégration**
  - Interaction Globale : Tester le flux de navigation complet depuis le lancement de l'application jusqu'à l'accès aux différents composants du jeu.

6.4 Stratégie de Test Global

Pour les nouvelles classes et fonctions ajoutées, il est essentiel de couvrir non seulement les aspects fonctionnels mais aussi les interactions utilisateur et l'intégration avec les autres composants de l'application.

Nous avons utilisé des outils et des frameworks de test tels que JUnit pour les tests unitaires, et des outils comme Selenium et Eclipse pour les tests d'interface utilisateur.

6.5 Résultats des Tests Finaux et Gestion des Anomalies

Dans cette sous-section, nous présentons les résultats des tests finaux et la gestion des anomalies détectées pour chaque classe.

6.5.1 Classe CustomDialog

- Les tests ont révélé une incohérence dans l'affichage des titres sous certaines configurations, résolue par une normalisation des paramètres de fenêtre.

- Les interactions avec les boutons étaient conformes aux attentes, sans anomalies détectées.

6.5.2 Classe GamePortal

- Des problèmes de performances ont été observés lors du chargement initial de la scène, conduisant à une optimisation du code de démarrage.
- Les tests d'interaction avec WordleGame ont révélé des problèmes de synchronisation, résolus par une révision de la séquence d'initialisation.

6.5.3 Classe Main

- Le test de la méthode start a révélé des incompatibilités avec certaines configurations de système, résolues par une mise à jour des paramètres de compatibilité.
- Aucune anomalie majeure n'a été détectée dans les tests d'intégration globale, confirmant une bonne navigation dans l'application.

7 Conclusion

7.1 Conclusion générale

La phase finale de notre projet de logiciel de jeux de vocabulaire en Java a été à la fois exigeante et gratifiante. Nous avons non seulement renforcé la fondation établie lors de la première partie du développement, mais avons aussi apporté des améliorations substantielles à notre jeu Wordle.

Notre équipe a démontré une capacité remarquable à s'adapter et à répondre aux défis, notamment en matière de conception d'interface utilisateur et de fonctionnalités avancées. Les améliorations apportées, comme la réadaptation du code du package View pour une meilleure maniabilité, l'optimisation de l'interface graphique et le développement approfondi du module d'indices, ont considérablement enrichi l'expérience utilisateur. La classe Score, spécialement développée par Antony, a introduit une dimension compétitive stimulante au jeu.

En dépit de l'inégale répartition des tâches en phase finale, notre équipe a fait preuve de résilience et de détermination pour atteindre les objectifs fixés. Les retours reçus lors de l'évaluation intermédiaire ont été intégrés avec succès, permettant une amélioration significative de la qualité du logiciel.

En conclusion, ce projet a été une aventure collective fructueuse, alliant apprentissage technique, travail d'équipe et innovation. Le produit final n'est pas seulement un jeu de vocabulaire fonctionnel, mais aussi un reflet de notre engagement, de notre passion et de notre capacité à surmonter les obstacles. Nous sommes confiants que ce logiciel offrira aux utilisateurs une expérience divertissante et éducative, et nous sommes impatients de poursuivre son développement et son amélioration dans le futur.

7.2 Conclusion Youssef Boudount

Alors que ce projet captivant touche à sa fin, je prends un moment pour contempler le parcours accompli et les enseignements tirés de cette expérience. La finalisation du jeu Wordle en Java représente non seulement l'achèvement d'un projet complexe mais aussi un jalon significatif dans mon développement personnel et professionnel.

Au cours de cette phase finale, j'ai eu l'opportunité d'approfondir mes connaissances en JavaFX, en me concentrant particulièrement sur l'optimisation de l'interface utilisateur. Les ajustements et les améliorations apportés, en réponse aux retours lors de l'évaluation intermédiaire, ont été des défis stimulants qui ont renforcé ma compréhension de la conception d'interfaces utilisateur intuitives et réactives.

La logique de jeu, que j'ai contribué à affiner, a continué à être une source d'apprentissage précieuse. Les améliorations apportées au système de validation des entrées et aux interactions entre les composants de l'interface ont consolidé mes compétences en programmation orientée objet et en gestion d'événements.

Ce projet a également été une occasion d'expérimenter le travail d'équipe dans un environnement agile, soulignant l'importance de la communication et de la collaboration pour le succès d'un projet logiciel. Malgré les défis et les inégalités dans la répartition des tâches, l'expérience globale a été extrêmement formatrice.

En conclusion, ce projet Wordle a été une aventure fascinante qui a significativement contribué à ma croissance en tant que développeur de logiciels. Les compétences acquises, les défis surmontés et les connaissances approfondies acquises au cours de ce projet sont des atouts que je chérirai et emporterai dans mes futures entreprises. Je suis reconnaissant pour les leçons apprises et suis impatient de mettre en pratique ces acquis dans de nouveaux projets et défis à venir.

7.3 Conclusion Dylan Dely

Pour conclure, développer l'interface graphique du portail de jeu et du jeu Worddle aura été une enrichissante expérience de programmation à commencer par l'apprentissage d'une bibliothèque et de ses composants, javafx et sans avoir aucune expérience en programmation d'interface graphique. Partir de 0 a été un défi à prendre mais voir l'évolution de cette interface, de partir de rien à avoir une fenêtre qui s'ouvre avec ce que l'on voulait voir qui s'affiche, a été tout de suite très excitant et motivant pour continuer jusqu'à la version de rendu. Cela n'aurait pas pu être possible sans le visionnage de multiples videos tuto Youtube, des pages de forums, des sites d'apprentissage javafx et java, etc.. qui m'ont permis de connaitre les différents composants que proposent java et sa bibliothèque javafx.

J'ai pensé a ce que l'esthétique du portail de jeux, de la grille de jeu et du clavier soit simple et intuitive pour l'utilisateur. Le côté responsive de l'interface à été un peu plus compliqué à mettre en place que le reste mais j'ai quand même essayé de rendre une version de rendu responsive à ma manière pour avoir un portail de jeu, une grille de jeu et un clavier soit adapté à la taille de la fenêtre.

Les contraintes du développement de l'interface ont bien été respecté, telles que le respect des noms des méthodes, variables et classes du code source, des commentaires anglais pour chaque méthodes, et enfin l'utilisation d'Eclipse comme espace de travail. De plus, notre projet s'est tenu sur un gitlab commun et son utilisation,du clonage de ma branch, des commits et des push, m'ont permis de bien me préparer pour les UE utilisant gitlab/github comme génie logiciel ou design patern et je suis sûr que cette apprentissage me sera utile à l'avenir, professionnellement ou non.

Pour terminer, je pense avoir mené à bien jusque là mon défi et ma tâche qu'est de développer une interface utilisateur d'un portail de jeu et du jeu wordle pour une première fois, avec javafx, et dont j'ai pu enrichir mes connaissances en terme de programmation en language java et maintenant avec javafx, et en terme de compréhension du développement d'une interface graphique.

7.4 Conclusion Anthony Genti

À l'achèvement de ce projet de développement sur les fonctionnalités du jeu Wordle, je tire une conclusion empreinte de satisfaction et d'apprentissage. La réalisation des différentes caractéristiques, depuis le menu et l'aide jusqu'aux effets visuels et animations, a été à la fois stimulante et enrichissante.

La gestion des fenetres d'instructions et des indices a constitué un aspect crucial, nécessitant une compréhension approfondie de la manipulation des fenêtres de dialogue personnalisées. Les fonctionnalités de recommencement et de sortie du jeu ont renforcé mes compétences dans la conception d'interfaces utilisateur intuitives, ainsi qu'une réflexion approfondie sur l'architecture globale de l'application.

Entre la version intermédiaire du projet et sa version finale, j'ai apporté des améliorations significatives à l'affichage. Des ajustements tels que le curseur, la taille de la police, et d'autres détails visuels ont été intégrés pour garantir une expérience utilisateur plus fluide et esthétiquement plaisante.

En ce qui concerne les effets visuels et animations, la mise en œuvre de la désactivation sélective des lignes de la grille, de l'animation de retournement de cellule et de la secousse de la grille a représenté une exploration approfondie des capacités graphiques de JavaFX. Cela a considérablement élargi mes connaissances dans le domaine des animations d'interface utilisateur.

Les animations de victoire et l'affichage de la nouvelle grille avec un message ont ajouté une dimension ludique au jeu. Cependant, intégrer ces éléments de manière fluide a constitué un défi technique, me poussant à approfondir mes compétences en gestion des transitions

et des événements dans l'environnement JavaFX.

Dans l'ensemble, ce projet a été une expérience formatrice, consolidant mes compétences en programmation orientée objet, en conception d'interfaces utilisateur et en gestion d'animations. Les défis rencontrés ont été autant d'opportunités d'apprentissage, et je suis ravi de pouvoir appliquer ces connaissances dans les phases futures du développement de Wordle.Je suis confiant dans la qualité et la robustesse du produit final.

7.5 Conclusion Axel Bertrand

Pour conclure, le développement de ce projet a été intéressant pour moi et j'ai apprécié travailler à permettre la communication entre deux langages différents. L'implémentation de la capacité à faire appel à du code dans un autre langage m'a fait traverser plusieurs solutions pour l'optimiser puisque mon premier essai s'est finalement révélé peu performant et m'a poussé à chercher d'autres solutions comme mettre en place de la parallèlisation.

Cela m'a aussi poussé à me poser des questions pour la gestion d'erreurs lors de la communication entre différents langages pour correctement recevoir et prendre en compte les erreurs pour rendre le code final plus robuste.

J'ai aussi appris en étant confronté à des sitations qui ne m'arrivent pas lorsque je travaille seul sur un projet comme la gestion de versions et de dépendances qui m'a été un obstacle à surmonter ou encore la gestion d'un IDE dans le cadre d'un projet de plus grande envergure que ce à quoi j'étais habitué.

Ce projet m'a donc beaucoup appris dans la réalisation d'interfaces, la robustesse du code et dans le travail en groupe dans un projet.

7.6 Synthèse des travaux réalisés : Intermediaire

Au cours de cette phase du projet, nous avons réalisé d'importantes avancées dans le développement de notre jeu Wordle. Nos efforts se sont concentrés sur plusieurs aspects clés :

- **Affichage et Interface Utilisateur :** Nous avons développé une interface utilisateur intuitive, comprenant un menu principal permettant de choisir le jeu à lancer, une zone de jeu affichant clairement la grille et un clavier virtuel. Des boutons pour l'aide, recommencer la partie, obtenir un indice, et retourner au menu principal ont été intégrés pour faciliter la navigation.
- **Mécanismes de Jeu :** Nous avons implémenté la logique de base du jeu, incluant la récupération d'un dictionnaire de mots et la sélection aléatoire d'un mot cible. Des mécanismes pour empêcher la saisie de mots inexistants en français et pour indiquer par des couleurs la précision des réponses (lettres correctes, mal placées ou incorrectes) ont été intégrés.
- **Interaction Utilisateur :** La possibilité de taper des lettres au clavier ou à la souris a été ajoutée pour améliorer l'interaction avec le jeu.
- **Responsive Design :** Un effort particulier a été fait pour assurer que notre application est responsive et accessible sur divers appareils, augmentant ainsi son accessibilité et sa convivialité.

Ces réalisations marquent des étapes importantes dans le développement de notre jeu Wordle, offrant aux utilisateurs une expérience riche et engageante. Nous continuerons à améliorer et à peaufiner ces fonctionnalités dans les prochaines phases du projet.

7.7 Perspectives et améliorations futures

À l'avenir, notre objectif est d'enrichir le jeu Wordle avec un système d'indices innovant en exploitant le potentiel de Word2vec. Cette fonctionnalité permettra non seulement d'ajouter

une dimension éducative au jeu, mais aussi d'offrir une aide contextuelle aux joueurs. Par ailleurs, l'amélioration de l'interface utilisateur, avec des animations visuelles et un design plus attrayant, est cruciale pour augmenter l'engagement des utilisateurs. L'intégration d'un serveur HTTP pour la communication entre les modules Java et Python est également une étape essentielle pour la mise en œuvre complète du projet. Enfin, nous envisageons d'étendre notre logiciel en ajoutant de nouveaux jeux de vocabulaire, augmentant ainsi la diversité et l'intérêt de notre application.

7.8 Synthèse des travaux réalisés : Finale

La phase finale de notre projet a vu l'accomplissement de toutes les tâches prévues, aboutissant à un jeu Wordle riche en fonctionnalités et offrant une expérience utilisateur complète et engageante. Voici les principaux développements et améliorations réalisés :

- **Affichage et Interface Utilisateur :** Nous avons peaufiné l'interface utilisateur, en rendant le menu principal plus interactif pour choisir le jeu à lancer. La zone de jeu a été optimisée pour afficher clairement la grille et un clavier virtuel, accompagnés de boutons utiles tels que l'aide, la réinitialisation du jeu, l'obtention d'indices, et le retour au menu principal.
- **Mécanismes de Jeu :** La logique du jeu a été affinée avec la sélection aléatoire d'un mot cible à partir d'un dictionnaire enrichi et des mécanismes améliorés pour la validation des entrées en français. Les indicateurs de précision des réponses ont également été optimisés.
- **Gestion des Scores :** Nous avons introduit un système de score comprenant l'affichage du temps écoulé, la série de victoires actuelle, et un tableau des meilleurs scores. Les scores sont désormais sauvegardés dans un fichier pour une consultation ultérieure.
- **Interaction Utilisateur :** L'interaction avec le jeu a été améliorée, permettant aux utilisateurs de taper des lettres au clavier ou à la souris de manière plus intuitive et réactive.
- **Responsive Design et Contrôles de Fenêtre :** Un effort supplémentaire a été fait pour assurer que l'application reste responsive sur divers appareils. Les contrôles standards de fenêtre ont été intégrés pour une expérience utilisateur améliorée.
- **Intégration de Word2vec :** Le système d'indices basé sur Word2vec a été intégré avec succès, offrant aux joueurs des mots similaires et enrichissant l'aspect éducatif du jeu. Un serveur HTTP a été mis en place pour la communication entre les modules Java et Python.
- **Améliorations Visuelles :** Des animations visuelles et un design plus attrayant ont été ajoutés pour augmenter l'engagement des utilisateurs et améliorer l'esthétique globale du jeu.

Ces améliorations représentent l'aboutissement de nos efforts collectifs pour créer un jeu Wordle complet et attrayant. Chaque aspect du jeu a été minutieusement travaillé pour offrir une expérience utilisateur riche et satisfaisante. Cette phase finale a non seulement consolidé les bases établies précédemment, mais a également introduit des innovations significatives qui définissent notre jeu comme une référence dans le domaine des jeux de vocabulaire.

8 Annexes

Cette section des annexes comprend des informations supplémentaires et des ressources qui complètent le corps principal du rapport. Les annexes offrent un aperçu détaillé des aspects techniques du projet, illustrant concrètement les méthodes et les résultats obtenus.

8.1 Code source commenté et documentation technique

Le code source commenté de notre projet est fourni pour offrir une compréhension approfondie de la structure et de la logique de notre application Wordle. Chaque segment du code est soigneusement commenté, expliquant non seulement la fonctionnalité implémentée, mais aussi les motivations derrière certaines décisions de conception. Cette approche vise à rendre le code accessible et compréhensible, facilitant ainsi la maintenance et les éventuelles extensions futures. Pour un accès complet au code source et à sa documentation technique, veuillez consulter notre dépôt GitLab : `https://gitlab.com/ceri-projet-programmation-2023/groupe-3/-/tree/main/project?ref_type=heads`.

8.2 Copies d'écran de l'application

Les captures d'écran de l'application illustrent son interface utilisateur et son design. Ces images démontrent le résultat de notre travail sur l'interface graphique et offrent une vue tangible de l'expérience utilisateur. Elles incluent des vues de la fenêtre principale du jeu, du clavier virtuel, des écrans de score, ainsi que des exemples d'interactions utilisateur typiques. Ces captures d'écran sont essentielles pour visualiser l'aspect et la convivialité de l'application, offrant une perspective concrète sur son fonctionnement et son design.

8.2.1 Écran d'Accueil

Description : Cette capture montre l'écran d'accueil de notre application Wordle. C'est la première interface que l'utilisateur voit après le lancement de l'application. Elle présente un design accueillant et donne accès aux différentes fonctionnalités du jeu, notamment le démarrage d'une nouvelle partie. (Figure 6)

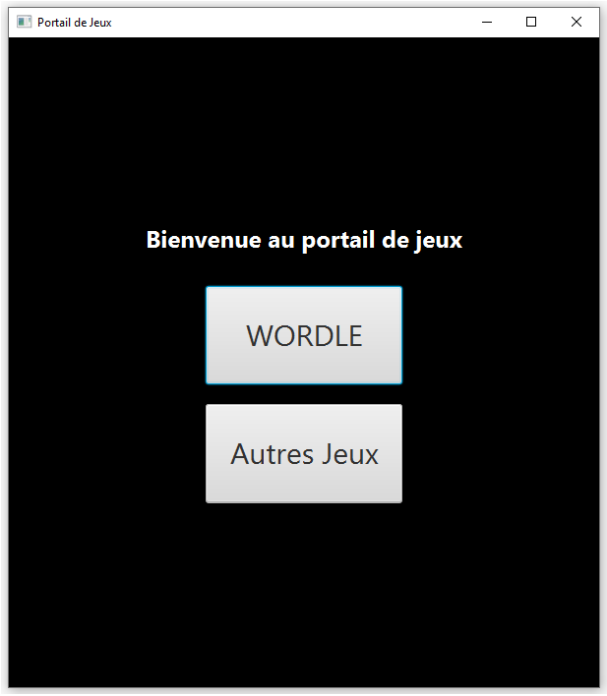

Figure 6. Écran d'Accueil

8.2.2 Fenêtre Choix de Niveau

Description : Cette section présente la fenêtre de sélection du niveau de difficulté dans notre jeu Wordle. L'utilisateur peut choisir entre trois niveaux de difficulté : facile, intermédiaire et difficile. Cette fonctionnalité permet d'adapter le jeu à différents types de joueurs, offrant une expérience personnalisée et adaptée à leur niveau de compétence. L'interface est conçue pour être simple et intuitive, avec des boutons clairement étiquetés pour chaque niveau.

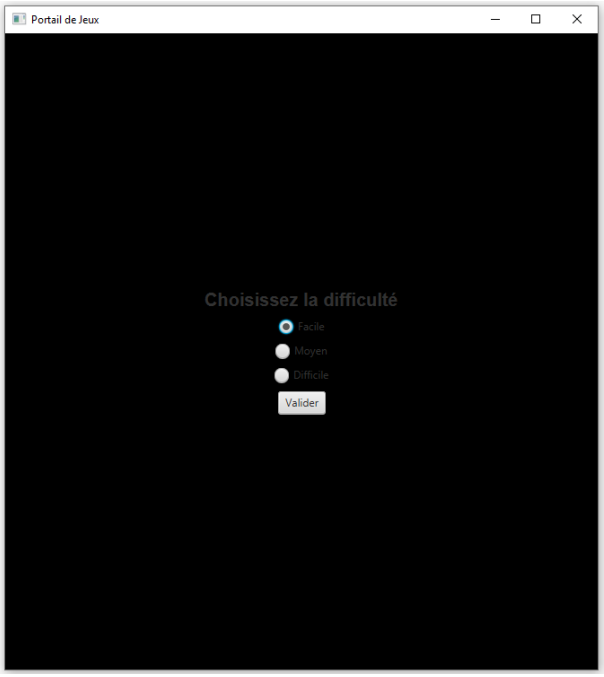

Figure 7. Fenêtre Choix de Niveau

8.2.3 Interface de Jeu

Description : Ici, nous montrons l'interface principale de jeu de Wordle. Cette capture d'écran illustre la grille de jeu avec les cellules pour entrer les mots, ainsi que le clavier virtuel en dessous. Elle met en évidence la clarté de l'interface et comment les utilisateurs interagissent avec le jeu pour deviner les mots. (Figure 7)

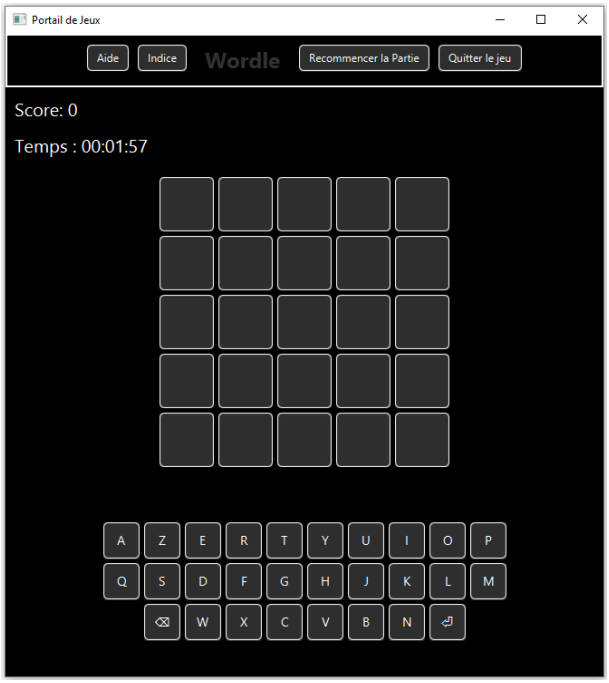

Figure 8. Interface de Jeu

8.2.4 Exemple de Partie en Cours

Description : Cette capture d'écran montre une partie en cours. Elle met en évidence les fonctionnalités interactives de l'application, comme les cellules qui changent de couleur pour indiquer les lettres correctes, mal placées ou incorrectes. Elle donne un aperçu de la dynamique du jeu et de la façon dont les indices visuels aident les joueurs. (Figure 8)

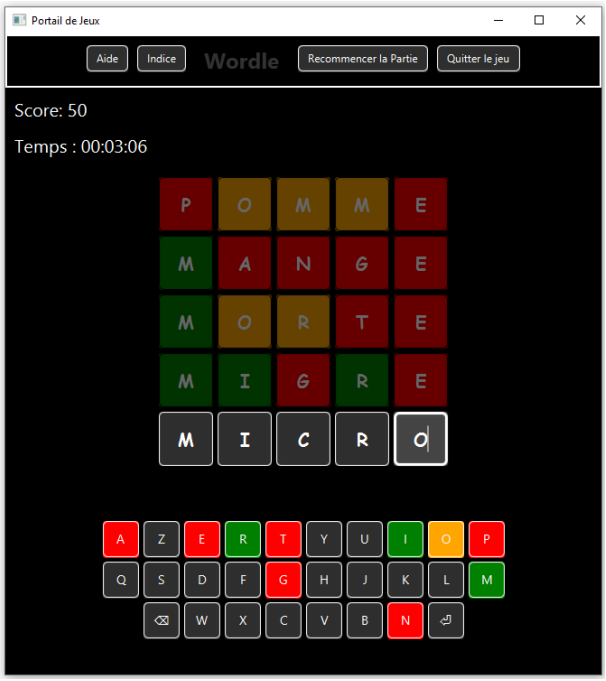

Figure 9. Exemple de Partie en Cours

8.2.5 Écran d'Aide et Tutoriel

Description : Cette image présente l'écran d'aide et le tutoriel du jeu. Elle fournit des informations sur les règles du jeu et des conseils pour les nouveaux utilisateurs. Cet écran est essentiel pour rendre le jeu accessible aux débutants. (Figure 9)

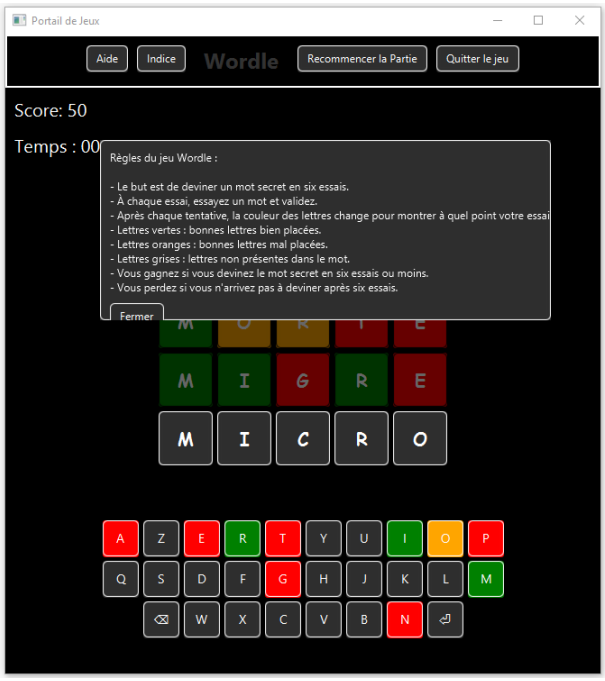

Figure 10. Écran d'Aide et Tutoriel

8.2.6 Fenêtre Indice

Description : Cette section illustre la fenêtre Indice de notre jeu Wordle. Cette fonctionnalité affiche un mot indice, déterminé à l'aide du module Python et du serveur, pour aider le joueur dans la recherche du mot cible. Le mot indice est sélectionné en fonction de sa pertinence et de sa proximité avec le mot cible, en utilisant des techniques avancées de traitement du langage naturel. Cette fenêtre offre un support utile aux joueurs, enrichissant l'expérience de jeu tout en conservant un défi intéressant.

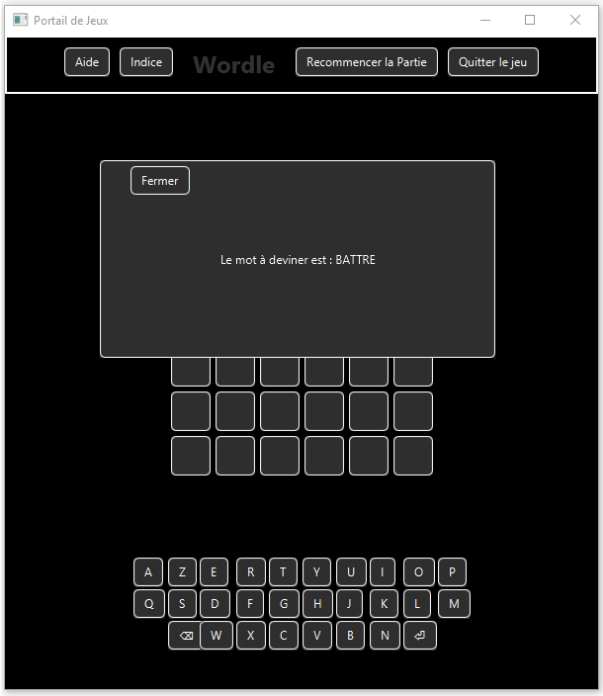

Figure 11. Fenêtre Indice

8.3 README du Projet

```
Projet Wordle en Java :

Description
Ce projet est une implémentation en Java dujeu de vocabulaire Wordle, réalisé dans le cadre

Fonctionnalités
Portail général au lancement de l'application pour sélectionner le jeu.
Affichage des éléments du jeu : grille de jeu, clavier virtuel.
Jouabilité selon les règles classiques de Wordle.
Sauvegarde des informations relatives à chaque partie.
Système d'indices basé sur la similarité entre mots avec Word2vec.
Installation et Exécution
Assurez-vous d'avoir Java et Python installés sur votre machine.

Clonez le dépôt Git :
git clone https://gitlab.com/ceri-projet-programmation-2023/groupe-3.git
Ouvrez le projet dans Eclipse ou un autre IDE compatible avec les projets Eclipse.
Lancez le fichier Main.java situé dans le dossier src/view pour démarrer l'application.
Structure du Projet

Le projet est organisé comme suit :
src: Contient le code source Java.
controller: Logique de jeu et gestion des indices.
model: Modèles de données, incluant le traitement des mots.
view: Interface utilisateur du jeu.
resources: Scripts Python et modèle Word2vec.
data: Dictionnaire de mots et scripts de traitement.
bin: Fichiers compilés Java.

Dépendances
JavaFX pour l'interface utilisateur.
Gensim pour Word2vec en Python.
Apache HttpClient pour les requêtes HTTP.
Jsoup pour le scraping Web.
Plusieurs autres bibliothèques Java, voir pom.xml ou équivalent pour plus de détails.

Auteurs
Anthony Genti
Youssef Boudount
Axel Bertrand
Dylan Gely

Licence
Ce projet est publié sous la licence GPL.
```

9 Références

Cette section compile l'ensemble des ressources, bibliothèques et documents de référence utilisés tout au long du développement du projet. Ces références ont joué un rôle crucial dans la conception et l'implémentation des divers aspects du jeu Wordle, en nous fournissant les outils et les connaissances nécessaires pour surmonter les défis techniques rencontrés.

9.1 Bibliothèques et outils utilisés

Dans le cadre de ce projet, nous avons utilisé une gamme de bibliothèques et d'outils pour faciliter le développement et améliorer les fonctionnalités de notre application :

- **JavaFX** : Pour la construction de l'interface utilisateur, nous avons utilisé JavaFX, une bibliothèque puissante qui nous a permis de créer une interface graphique riche et interactive.
    - **javafx.application.Application** : bibliothèque de base pour les applications JavaFX.
    - **javafx.geometry.Pos** : Utilisé pour gérer l'alignement au centre des éléments du portail et du jeu dans le container
    - **javafx.scene.Scene** : Bibliothèque qui gère les scènes de fenêtres
    - **javafx.scene.control.Button** : Utilisé pour créer des boutons (clavier virtuel)
    - **javafx.scene.control.Label** : Utilisé pour créer des labels (Titre du Portail)
    - **javafx.scene.control.TextField** : Utilisé pour créer une zone de texte (cellule de la grille de jeu)
    - **javafx.scene.layout.*** : Bibliothèque qui permet d'utiliser tout les container (borderPane, StackPane, VBox, Hbox)
    - **javafx.stage.Stage** : Bibliothèque qui gère les fenêtres (Stage)
    - **javafx.stage.StageStyle** : Utilisé pour définir le style de la fenêtre (Stage) (en UTILITY)
    - **javafx.scene.Node** : Utilisé pour obtenir les cellules de la grille de jeu qui sont sous forme de noeuds
- **Apache HttpClient** : Utilisée pour les requêtes HTTP, notamment pour l'intégration avec les API externes et les opérations de scraping web. Les modules spécifiques utilisés incluent :
    - **httpclient5** : Pour les requêtes HTTP de base.
    - **httpclient5-cache** : Pour la mise en cache des requêtes HTTP.
    - **httpclient5-fluent** : Pour une interface de programmation plus fluide.
    - **httpclient5-testing** : Pour tester les requêtes HTTP.
    - **httpclient5-win** : Spécifique aux systèmes Windows.
    - **httpcore5** : Fondement des requêtes HTTP.
    - **httpcore5-h2** : Support du protocole HTTP/2.
    - **httpcore5-reactive** : Pour le traitement asynchrone.
    - **httpcore5-testing** : Pour tester les fonctionnalités HTTP de base.
- **Jansi** : Pour améliorer la sortie dans la console, permettant des affichages colorés et formatés.
- **JNA (Java Native Access)** : Bibliothèque qui permet aux programmes Java d'appeler des fonctions de bibliothèques partagées natives.
    - **jna** : Fonctionnalités de base de JNA.
    - **jna-platform** : Extensions spécifiques à la plateforme.
- **Reactive Streams** : Pour le développement d'applications asynchrones non bloquantes.

    - **reactive-streams** : API standard pour la programmation asynchrone avec des flux de données.
- **Apache Commons CLI** : Utilisé pour l'analyse des lignes de commande dans les applications.
    - **commons-cli** : Fournit des outils pour interpréter les arguments de ligne de commande.
- **Apache Commons Lang** : Bibliothèque utilisée pour les opérations courantes sur les chaînes de caractères, les collections et les dates. Les composants spécifiques comprennent :
    - **commons-lang3** : Pour les opérations sur les chaînes de caractères et autres utilitaires courants.
    - **javadoc**, **sources**, **test-sources**, **tests** : Modules supplémentaires pour la documentation, le code source, les tests, etc.
- **Gson** : Pour la manipulation des données JSON, notamment dans le traitement des scores et la gestion des configurations.
- **Eclipse** : L'environnement de développement intégré (IDE) Eclipse a été notre principal outil de développement, offrant un support robuste pour Java et une intégration facile avec diverses bibliothèques et plugins.
- **Jsoup** : Pour le scraping des données du Wiktionnaire, Jsoup a été essentiel pour analyser et extraire les informations nécessaires depuis les pages web.
- **Python**
    - **gensim** : Sert à charger et utiliser le modèle word2vec pour le module d'indices.
    - **flask** : Sert à créer le serveur HTTP pour faire des requêtes depuis le module d'indices.
- **Autres**.
    - **java.util.Random** : Utilisé pour générer une taille de grille aléatoire dans le jeu Wordle (entre 2 et 14 colonnes)

9.2 Sources documentaires et tutoriels suivis

Pour compléter notre compréhension théorique et pratique des technologies et techniques employées, nous avons consulté diverses sources documentaires et suivi plusieurs tutoriels :

- **Documentation officielle de JavaFX** : Pour maîtriser les aspects de l'interface utilisateur JavaFX, la documentation officielle a été une source d'information inestimable.
  Openjfx - install javaFX
  Documentation BorderPane
  Documentation VBox
  Documentation TextField

- **Tutoriels JavaFX sur YouTube** : Plusieurs tutoriels sur YouTube ont été suivis pour obtenir des conseils pratiques et des démonstrations sur la mise en place d'interfaces utilisateur complexes.
  Tuto install javafx

- **Documentation de Jsoup** : La documentation officielle de Jsoup a guidé notre utilisation de cette bibliothèque pour le scraping web efficace.

- **Stack Overflow** : Pour les problèmes de codage spécifiques et les questions techniques, Stack Overflow a été une ressource fréquemment consultée.

- **Blogs de développement Java et forums** : Des articles techniques et des discussions

de forums ont été consultés pour des conseils sur les meilleures pratiques de développement Java et la résolution de problèmes complexes.
GeeksforGeeks - Tutoriel JavaFX
Gestion des fenêtres
Layout JavaFX
HBox et VBox
Style des fenetres
Documentation GridPane
Documentation GridPane
Responsive

- **Tutoriels et guides en ligne sur le traitement JSON avec Gson** : Ces ressources ont été essentielles pour comprendre l'utilisation efficace de Gson dans le contexte de notre application.

Ces références ont été essentielles pour nous guider à travers les différentes phases du projet, en nous offrant les connaissances et les outils nécessaires pour développer une application robuste et fonctionnelle.

10 Remerciements

Nous tenons à exprimer notre sincère gratitude à toutes les personnes qui ont contribué au succès de ce projet. Leur soutien, leurs conseils et leur expertise ont été inestimables tout au long de cette aventure.

Nous adressons tout d'abord nos plus chaleureux remerciements à nos professeurs et superviseurs de projet, M. Noe CECILLON et M. Jarod DURET. Leurs précieux conseils, leur encadrement rigoureux et leur patience ont été des piliers dans la réalisation de ce projet. Leurs orientations éclairées et leurs critiques constructives ont grandement contribué à la direction et à la qualité de notre travail.

Nous exprimons également notre reconnaissance envers l'ensemble du corps enseignant du CERI pour leur enseignement rigoureux et leur engagement à fournir une éducation de qualité, nous préparant ainsi à relever les défis de ce projet.

Un remerciement spécial à nos camarades de classe, pour l'environnement stimulant qu'ils ont créé, et pour toutes les discussions enrichissantes et les séances de brainstorming partagées. Cette atmosphère collaborative a été un élément clé de notre motivation et de notre progression.

Enfin, nous tenons à remercier toutes les personnes qui ont indirectement contribué à ce projet, en particulier les développeurs et les contributeurs des bibliothèques et outils que nous avons utilisés. Leur travail a été essentiel pour réaliser nos objectifs techniques.

Cette expérience a été incroyablement formatrice, et nous sommes fiers de ce que nous avons accompli ensemble. Merci à tous pour avoir rendu cela possible.

L'équipe du projet Wordle - Groupe 3