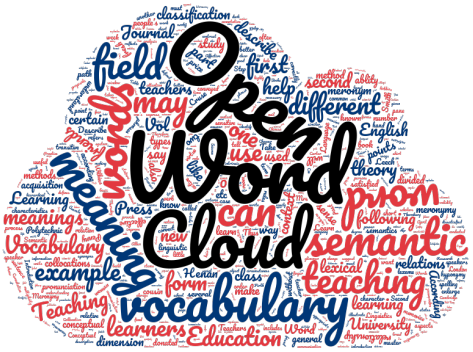
Project report and documentation

Project OpenWord Cloud

English project



DIAME Jean-François, REHIOUI Walid

Année 2018-2019

Table des matières

1 Preamble

OpenWord Cloud is a Java application that allows users to choose which words they want to learn and provides them with fun games to help them train and learn their English vocabulary and expressions. We made it so that people who don't like the know-by-heart approach could learn their vocabulary in an easy and interactive way !

In this report begins with a User Documentation written in English. It describes our application's features and gives guidelines to enhance the user's experience.
The second part of our report is some Technical Documentation (written in French) which describes all technical aspects of our application.
Finally, you will find at the end of this report a Sources section that lists the sites and tools we used to develop our application.

2 User documentation (English)

2.1 Home page

The Home page is quite simple, and we made sure it was as intuitive as possible while not being too restrictive to the user.

In the right part of the main interface, the user will find three buttons :

- "**Import**" : this button allows the user to load lists of words into the application, to be used in games.
- "**Create new list**" : this one allows a user to create a new OpenWordsCloud compatible list of words from a text file containing words (see **??**)
- "Clear Lists" : unloads loaded lists from the application.

The left part of the main interface is a simple list of games, which can be launched with a click. But in order to launch a game, the user must first select one or more list(s) of words by clicking it/them.

When launching the program you will see a greeting message on how to get started with the application. Once you've clicked on 'OK' the main interface will appear on the screen.

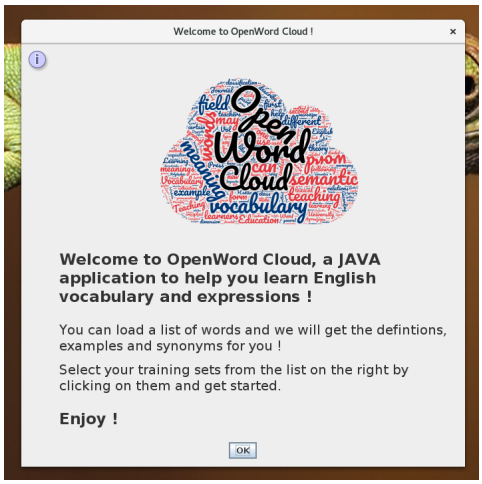


FIGURE 2.1 – Welcome message

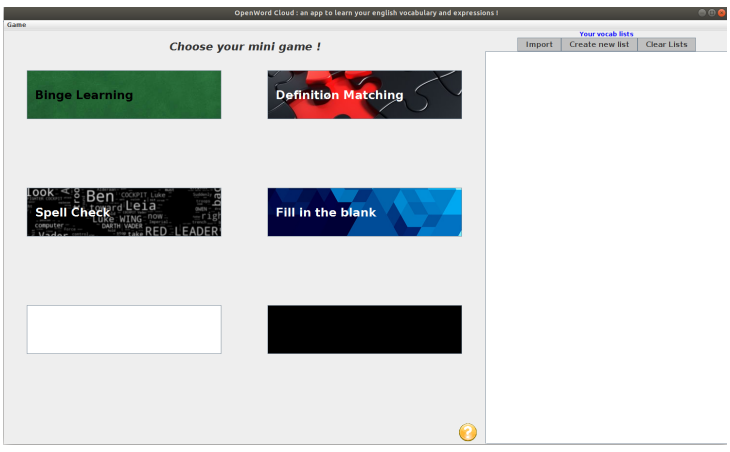


FIGURE 2.2 – The main interface

The main interface contains a space for vocabulary lists to the right and an environment reserved for the different games of the application.
The first thing a user would want to do is to load pre-existing vocabulary lists or to create new ones from it's own selection of words. To load pre-existing lists the user clicks on the 'Import'

button and select a .xml file.

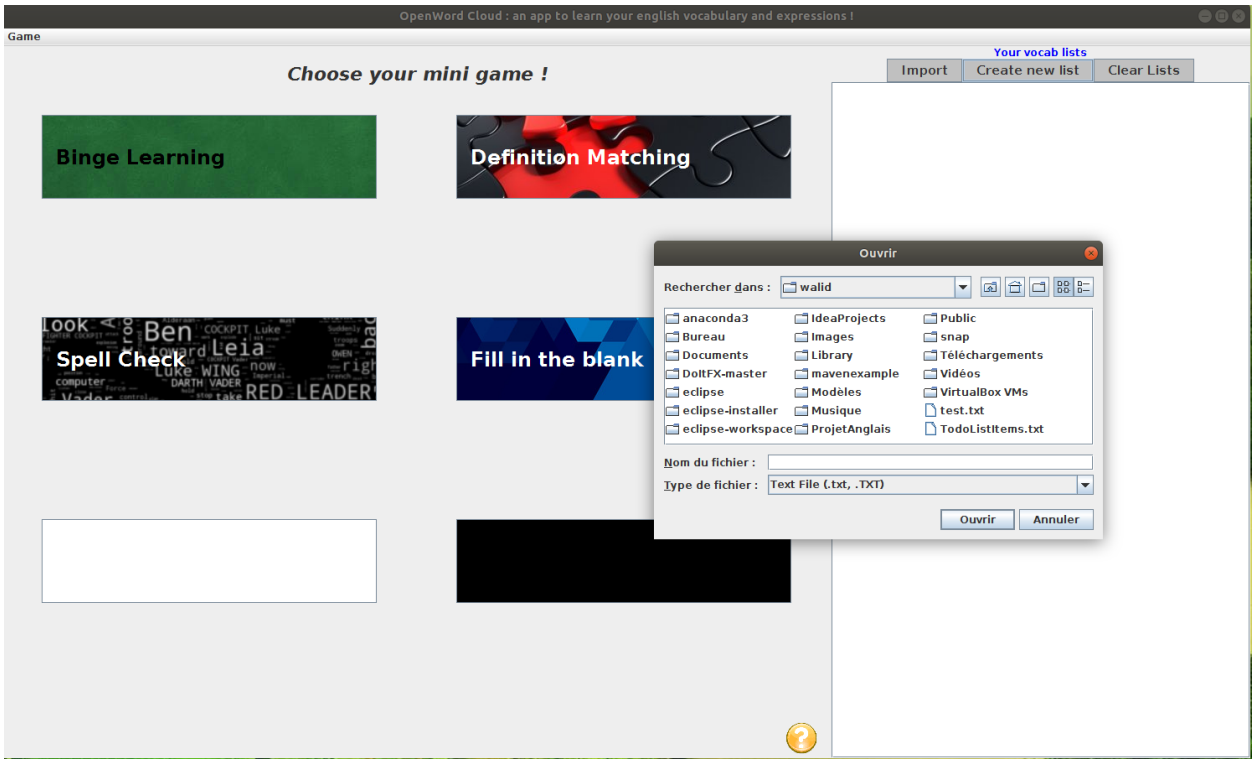


FIGURE 2.3 – Loading vocabulary lists

If the player wishs to create new vocabulary sets from it's own list of words in a .txt file, he clicks on the 'Create new list' button and selects the .txt file. Fetching for definitions, examples and synonyms on the Internet can take some time but once this process is done his new vocabulary list will be automatically separated in sections and displayed.

2.2 Create or load vocabulary lists

Here we will describe in more details the ins and outs of our vocabulary lists system.

The lists used by our application are stored as .XML files. Their structure is **OpenWordsCloud specific**, which means only an .xml files generated by an OpenWordsCloud application (or written by hand in the OpenWordCloud format) can be read by another OpenWordCloud app.

Because our application was designed to allow users to choose how to learn their English vocabulary and expressions, we created and integrated an OpenWordsCloud .xml file generator in the application.
This generator converts a user's personal list of words into an OpenWordsCloud compatible file, and then loads it into the App.
To use it, only two things are required of the user : **an internet connection** and that the words be stored **one word per line** in the user text file.
Disclaimer : when a word's definition or example is not found online, it is ignored.

To use this generator, the user can click on the "**Create new list**" button in the Home page.

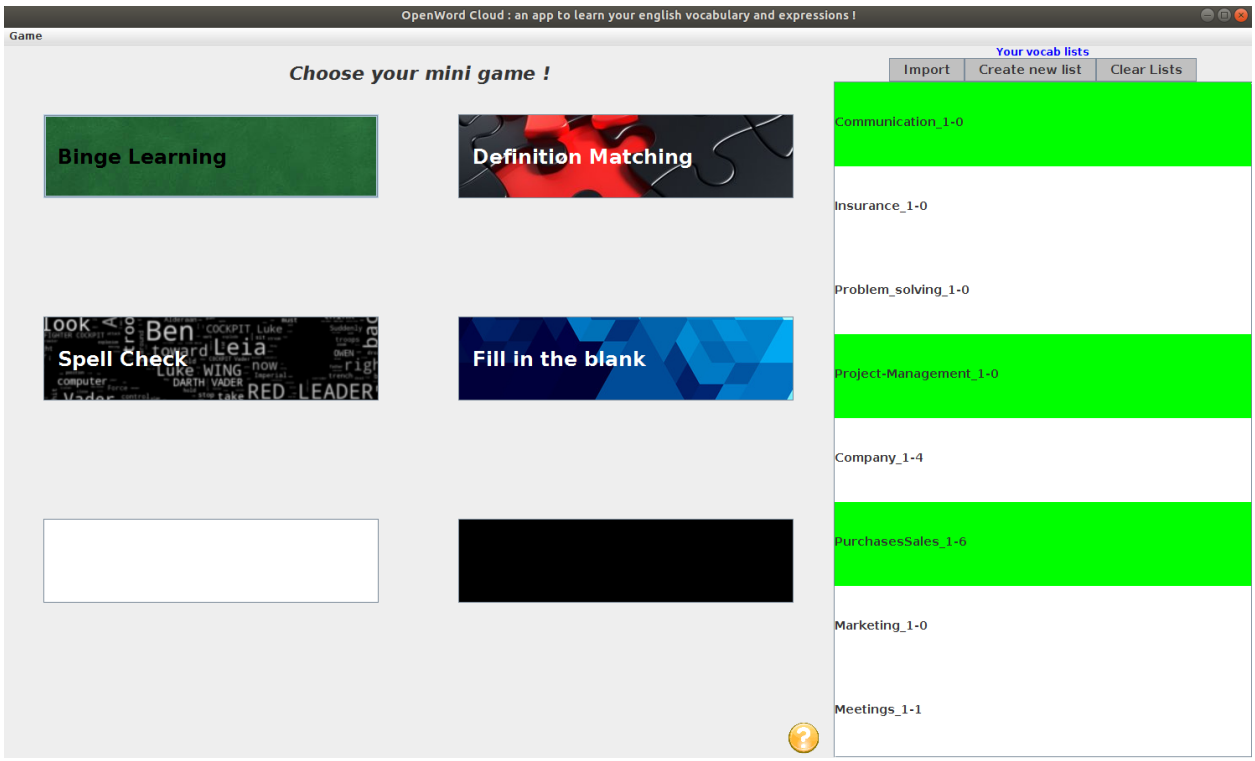


FIGURE 2.4 – Loading vocabulary lists

The user must then **select his text file**, click on "**Open**" and **wait** for the OpenWordsCloud file to be generated and loaded.

To simply **load an existing OpenWordsCloud .xml file**, the user can click on "**Import**", and then navigate in the new window to the right folder to select the file. The user must finalise the procedure by clicking on "**Open**".

2.3 Binge Learning

The Binge Leaning area is the place where to study before playing.
In this area, the definition for a word in the selected list(s) is displayed. To see a usage example or synonyms, the user can simply click on the board once. To swich back to the definition, the user can simply click again.
When the user wants to study another word, he can click on one of the side arrows to change the displayed word.

2.4 Mini-Games

2.4.1 Definition Matching

The objective is simple : the player must match a definition to the word displayed on the left of the screen.
Every wrong answer makes the water level rise.

Once the player has selected the right definition, he will be able to move on to the next word.
If the player makes it to the end of the list, he has won !

2.4.2 Spell Check

The objective is simple : the player must find a potential mistake in the displayed sentence.
Every wrong answer makes the water level rise.
If there is a mistake, the player must click on it, else he must click on "**Nothing was misspelled here**".
Once the player has selected the right answer, he will be able to move on to the next word.
If the player makes it to the end of the list, he has won !

2.4.3 Fill in the Blank

FIGURE 2.5 – Example of exercise

The objective is simple : the player must find the missing word in the displayed sentence.
Every wrong answer makes the water level rise.
Once the player has selected the right answer, he will be able to move on to the next word.
If the player makes it to the end of the list, he has won !

2.4.4 Scoring system

Every time a player gets something right, his score increases. At anytime of a game, the player's score is displayed at the top of the screen

FIGURE 2.6 – Example of finding the right answer

The score is calculated so that if the player makes very few mistakes, his score will increase and so that a player playing with a longer list can reach higher score.

2.5 Game Help

in every environment of the application we provide a helping window, that can be opened by clicking on the help sign. Each helping window explains the essential aspects of the corresponding environment.

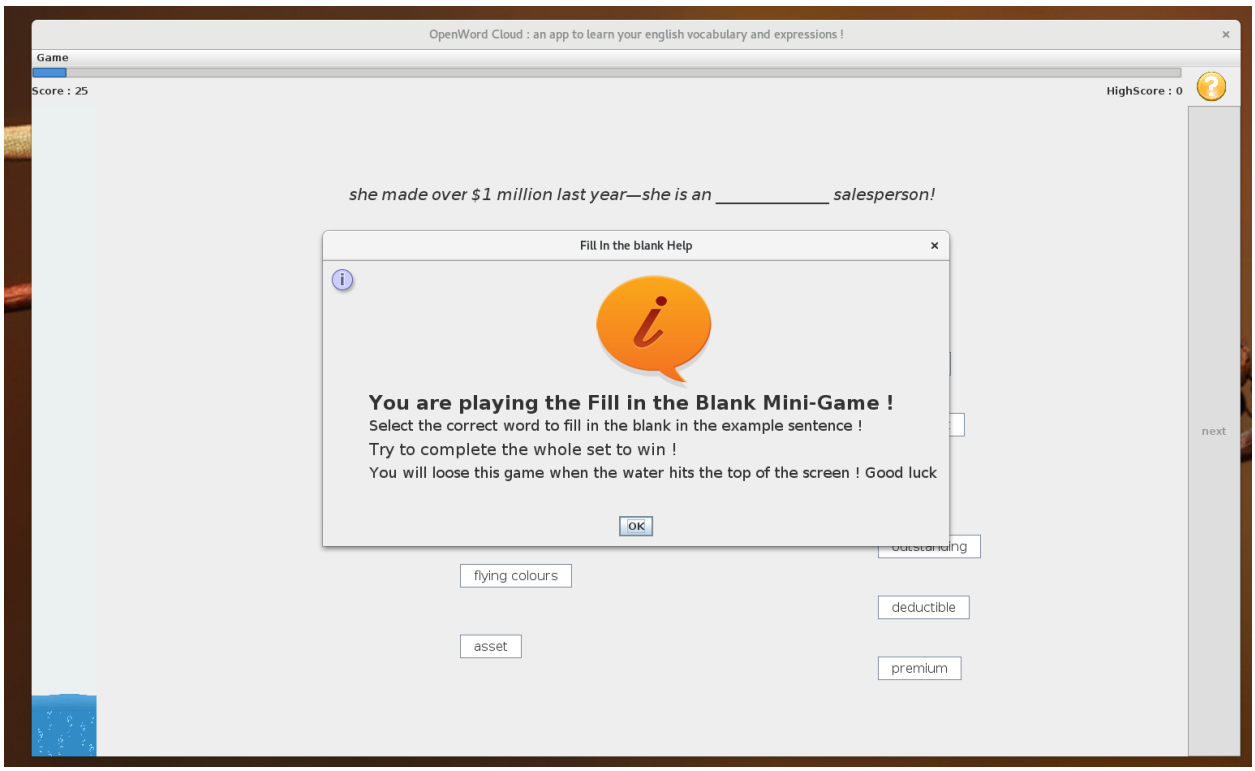


FIGURE 2.7 – Example of a help window

3 Documentation technique (Français)

3.1 Patron de conception

Dans le cadre de ce projet, nous avons essayé de mettre en oeuvre le maximum de bonnes pratiques apprises durant nos (presque) deux ans d'école.
Ainsi, nous avons utilisé un patron de conception (Design Pattern) qui correspond au Modèle-Vue-Contrôleur (MVC).

Les classes Java qui correspondent au Modèle contiennent le coeur du jeu : les listes des listes de vocabulaire disponibles et sélectionnées, ainsi que les fonctions qu'on utilise pour rendre les jeux aléatoires et récuperer les informations nécessaires à la construction des jeux. Les Vues s'occupent de tout ce qui est affichage, ce sont les classes qui assurent l'interaction entre l'utilisateur et le fond de l'application.
Enfin, les classes qui correspondent aux Contrôleurs ont pour fonction de s'occuper de l'interaction entre les deux. Lorsque le joueur clique sur une liste ou un button du jeu, il déclenche le contrôleur qui lui est associé et va eventuellement modifier le modèle, qui met alors à jour ses données, et prévient la vue qu'il y a eu du changement.

Bien que parfois difficile à suivre et à organiser, cette façon de programmer nous a permis d'écrire efficacement notre projet, de façon claire et par intégrations.

De plus, nous avons conçu et implémenté une structure de données adaptée à la gestion des mots, leurs définitions, phrases d'exemple et synonymes. Il s'agit d'un objet appelé **Dictionnary**, qui contient des objets **Entry** et qui contient des HashMap pour l'indexation des mots. Cette structure permet d'accéder aux informations nécessaires très rapidement, ce qui rend l'éxpérience utilisateur plus agréable.

3.2 Parsing de pages HTML de définitions

La librairie JSoup 1.10.2 nous a permis de réaliser des requêtes sur internet et de parser les pages retournées.

Nous l'avons utilisée :
- pour interroger le site wordreference.com afin d'obtenir une définition pour mot.
- pour interroger le site thesaurus.com afin d'obtenir des synonymes pour un mot.
- pour interroger le site sentence.yourdictionary.com afin d'obtenir un example pour un mot.

À partir de ces information, nous pouvons ajouter une entrée à un fichier XML de listes de vocabulaire. Cela semble assez simple en principe, mais il faut noter que afin de récupérer les différentes informations dans les différentes pages web, il a fallu étudier leurs structures pour déterminer précisément où trouver la donnée recherchée. Ensuite il a fallu créer un parseur adapté à chaque site et à chaque donnée à y récupérer. Nous avons rencontré deux autres difficultées :

— La réalisation de 3 requêtes internet par mot peut s'avérer ralentie par une mauvaise connection internet.
— Pendant l'implémentation des parseurs, les protection des sites contre les attaques internet se sont déclenchées en réponse à notre batterie de test qui réalisait trop de requêtes dans un laps de temps réduit.

3.3 Parsing des fichiers XML de listes de vocabulaire

La librairie **JDOM 2.0.6** pour le parsage de fichiers .xml nous a permis de charger très facilement tous les éléments nécessaires au bon apprentissage dun mot de vocabulaire dans notre application. Cette libraire nous a donc permis d'avoir une 'base de données' très simple, facilement générable et très facile à parser sous la forme de fichiers XML. Pour que l'application marche, il est absolument nécessaire de garder le fichier **'dictionnaire.dtd'** qui définit la structure de tous les fichiers .xml de listes de vocabulaire à parser

3.4 Esthétisme et ergonomie

Puisqu'il s'agit d'un produit qui doit être accessible à un public qui n'est pas informaticien, nous avons mis un effort particulier sur l'interface graphique, en s'attardant sur l'aspect esthétique.

Nous avons fait une interface pas trop extravagante, épurée et qui assure la lisibiité des exercices et des mots. Nous avons utilisé JavaFx de Java à cet effet. Nous avons de l'expérience en programmation java ansi que la programmation en javaFX, de par sa simplicité et les nombreuses possibilités qu'elle offre

Les images que nous avons utilisées sont des images libres ou des images que nous avons créés nous mêmes (par exemple, les .gif de défaite et victoire ainsi que le logo de l'application).

4 Conclusion

Ce projet nous a permis d'explorer d'autres façons de concevoir un logiciel, tout en nous interrogeant sur les possibilités pédagogiques qu'un tel jeu pourrait offrir. En tant que concepteurs et développeurs de logiciels c'était intéressant d'avoir la liberté de création et d'implémentation d'un logiciel dès le début et déterminer nous mêmes ce qu'on voulait faire.

Notre objectif était de créer une application qui puisse être utilisable peu importe le niveau d'anglais de l'utilisateur et où la base de données de mots ou examples pour les exercices ne soit pas statique mais qu'elle puisse evoluer selon la volonté de l'utilisateur. Nous pensons avoir atteint cet objectif

5 Sources

Nous avons utiliser la platforme github pour avoir des cas d'applications pour les différentes applications utilisés
https://github.com/

5.0.1 Sites Web

Nous avons utilisé des sites Web d'aide comme

StackOverflow pour débugger notre application

5.0.2 Outils

Nous avons utilisé deux librairies Java dans notre l'application

- **JDOM 2.0.6** - Une librairie pour le parsage des fichiers .xml contenant les mots
- **JSOUP 1.10.2** - Une librairie pour le parsage des résultats des requêtes HTML contenant les definitions et examples des mots choisis