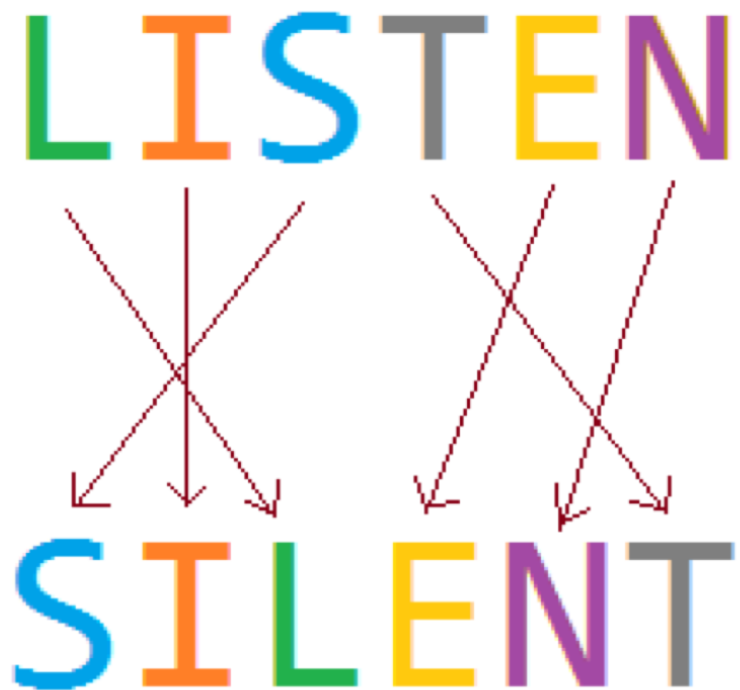


## Rapport de développement du projet AnagramQuest

Qassim-Tayab MANSOOR et Kacper KUPIS

Projet disponible aussi sur github : <https://github.com/Kax2/ProjetAnagramQuest>

### Anagrams



# TABLE DES MATIÈRES

<b>1. Architecture du code.....</b>	<b>2</b>
a. Serveur_UDP.....	3
Code.....	3
Game.java.....	3
Serveur_UDP.java.....	5
Resources.....	7
dictionaries.....	7
b. Serveur_web.....	7
Code.....	9
Game.java.....	9
MainVerticle.java.....	9
Resources.....	10
dictionaries.....	10
webroot.....	10
example.html.....	10
index.html.....	10
c. AnagramQuestApp.....	10
Code.....	10
MainActivity.java.....	11
Game.java.....	12
<b>2. Difficultés rencontrées et moyens mis en oeuvre pour les surmonter.....</b>	<b>12</b>
<b>3. Répartition de travail.....</b>	<b>13</b>

## 1.Architecture du code

## a. Serveur\_UDP

### Code

Game.java

#### Variables:

```
private final InetAddress userAddr;
```

Contient l'adresse IP de l'utilisateur (localhost dans notre cas)

```
private final int userPort;
```

Contient le port sur lequel la communication sera établie (port 888 dans notre cas)

```
private final ArrayList<String> anagramSequence;
```

Contient la liste des anagrammes

```
private ArrayList<String> dictionary;
```

Contient la liste des mots contenus dans le dictionnaire

```
private Map<String, Set<String>> anagramicClasses;
```

Liste de mots ordonnées et rangés par ordre alphabétique, reliée à une liste de mots que l'on peut créer avec ces lettres

```
private int currentAnagramToGuess;
```

Contient l'anagramme actuel à deviner

#### Fonctions:

```
public Game(InetAddress userAddr, int userPort, int finalLength, String dictPath)
```

Initialise les valeurs du port, de l'adresse IP de l'utilisateur, de la taille maximale du mot à deviner et du dictionnaire dans lequel se référer avant de commencer la partie

```
public String getCurrentAnagramToGuess()
```

Permet d'obtenir l'anagramme actuel à deviner, contenu dans la variable currentAnagramToGuess

```
private String sortWord(String word)
```

Transforme le mot contenu dans "word" en un tableau de caractères, puis trie les caractères du tableau dans l'ordre croissant.

```
public boolean setNextAnagramToGuess()
```

Définit le prochain anagram à deviner si l'utilisateur a réussi à deviner l'anagramme précédent

```
public boolean answerIsCorrect(String answer)
```

Permet de vérifier que l'utilisateur a réussi à trouver la réponse correcte.

```
private Map<String, Set<String>>  
createAnagramicClasses(ArrayList<String> dictionary)
```

Permet de générer une classe contenant une map contenant les lettres triées alphabétiquement avec les mots possibles de créer avec ces lettres.

```
public int getUserPort()
```

Permet d'obtenir la valeur du port de l'utilisateur contenue dans la variable userPort

```
public InetAddress getUserAddr()
```

Permet d'obtenir la valeur de l'adresse IP de l'utilisateur contenue dans la variable userAddr

```
public ArrayList<String> getAnagramSequence()
```

Permet d'obtenir la valeur contenue dans la variable anagramSequence

```
private ArrayList<String> loadDictionary(String path)
```

Charge le dictionnaire en mémoire, dont le nom est contenu dans la variable "path"

```
private ArrayList<String> findAnagramSequence(int maxLength)
```

Génère une liste d'anagrammes pour chaque mot allant d'une longueur de 1 à n

```
private ArrayList<String> findAnagrams(String initialWord)
```

Permet d'obtenir la liste des anagrammes possibles à partir d'un mot

```
public String getExampleWord()
```

Utilisée lorsque la partie est perdue. Permet de montrer un exemple de mot qu'il était possible de former avec les lettres actuelles.

Serveur\_UDP.java

**variables**

```
public static final String INTERFACE =  
"localhost";
```

Contient le nom de l'interface où se connecter

```
private static ArrayList<Game> gameInstances = new ArrayList<>();
```

Contient les instances du jeu

```
public static final int PORT =  
20000;
```

Contient le numéro du port UDP où se connecter

## Fonctions :

```
private static ArrayList<ArrayList<String>>  
datagramToCommandList(DatagramPacket packet)
```

Permet de traduire les paquets reçus dans la variable packet en commande pour le jeu

```
private static boolean isGameAlreadyStarted(InetAddress addr, int port)
```

Permet de vérifier que le jeu a commencé

```
private static void sendErrorPacket(DatagramSocket socket, InetAddress  
destAddr, int destPort, String errorMsg)
```

Permet d'envoyer un message d'erreur contenu dans la variable "errorMsg", sur le socket, l'adresse et le port mentionnée en paramètres

```
private static void printErrorMsg(String errorMsg, DatagramPacket  
receivedPacket)
```

Permet d'afficher un message d'erreur en fonction du paquet reçu

```
private static int getGameInstanceOfClient(DatagramPacket  
receivedPacket)
```

Permet d'obtenir le numéro de la partie actuelle

```
public static boolean parseStart(ArrayList<ArrayList<String>>  
commandList, DatagramPacket receivedPacket, DatagramSocket socket)
```

Permet de vérifier que l'utilisateur a entré des commandes valides pour commencer la partie

```
public static boolean parseAnswer(ArrayList<ArrayList<String>>  
commandList, DatagramPacket receivedPacket, DatagramSocket socket)
```

Permet de vérifier que le format de la réponse est correct

```
public static boolean parseAbandon(ArrayList<ArrayList<String>>  
commandList, DatagramPacket receivedPacket, DatagramSocket socket)
```

Permet de savoir si l'utilisateur souhaite abandonner sa partie

```
public static void main(String[] args) throws SocketException
```

Fonction main qui lève l'erreur SocketException dans le cas où la connexion n'a pas été établie correctement.

## Resources

### dictionaries

Contient e un dictionnaire en francais et un dictionnaire en anglais : **english-scrowl-80.txt** et **french-debian.txt**

## b. Serveur\_web

Ce serveur permet de gérer la création de 4 pages web :

- Une page HTML de présentation du jeu avec ses règles

### Règles du jeu

**Le but du jeu est de trouver des anagrammes de plus en plus longues jusqu'à parvenir à deviner l'anagramme ultime !**

**Voir aussi :**

[Règles Exemples de jeu Dictionnaires supportés](#)

Une partie débute en communiquant une séquence de m lettres au joueur.  
Il s'agit d'un mot du dictionnaire dont les lettres ont été mélangées.  
Le joueur doit alors retrouver le mot ou une de ses anagrammes.

Si le joueur parvient à trouver le mot de m lettres, une nouvelle lettre est proposée au joueur qui doit alors trouver un mot constitué à la fois des lettres du mot précédemment trouvé et de la lettre ajoutée.  
Si le joueur réussit à trouver ce mot de m + 1 lettres, on lui propose une nouvelle lettre pour trouver une nouvelle anagramme de m + 2 lettres... et ainsi de suite jusqu'à parvenir au mot ultime de n lettres.  
S'il y parvient, il remporte le défi.

Voici quelques [exemples de jeu](#)

This project is Open Source under - [GNU General Public License v3.0](#)  
Build by Kacper KUPIS | Qassim-Tayab MANSOOR

- Une page HTML présentant quelques exemples de séquences d'anagrammes pour illustrer les règles du jeu

# Exemples d'anagrammes

## Voir aussi :

[Règles](#) [Exemples de jeu](#) [Dictionnaires supportés](#)

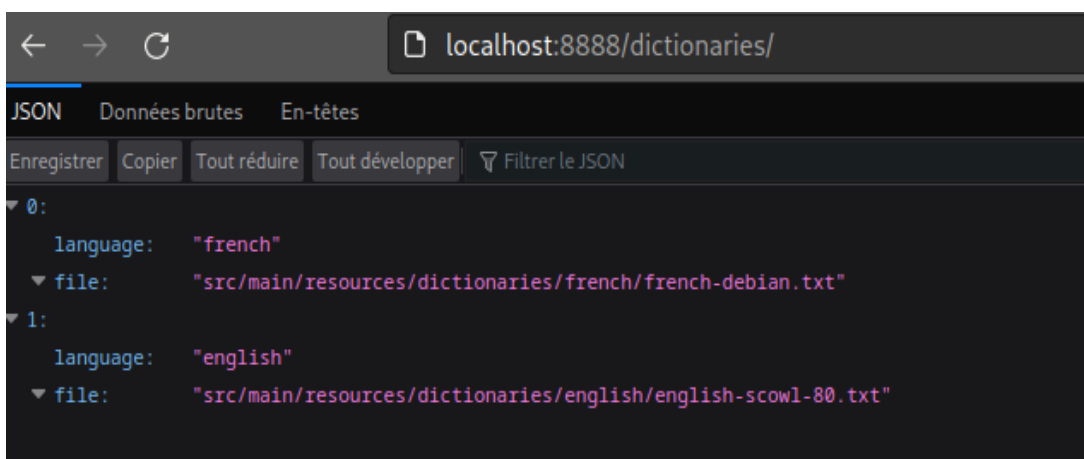
Pour illustrer le fonctionnement du jeu, déroulons un exemple :

- 1. Anagramme avec les lettres ET ? TE ou ET
- 2. Anagramme avec les lettres ET+S ? EST
- 3. Anagramme avec les lettres EST+R ? SERT
- 4. Anagramme avec les lettres SERT+E ? ESTER
- 5. Anagramme avec les lettres ESTER+R ? RESTER
- 6. Anagramme avec les lettres RESTER+A ? RESTERA
- 7. Anagramme avec les lettres RESTERA+P ? REPARTES
- 8. Anagramme avec les lettres REPARTES+I ? REPARTIES
- 9. Anagramme avec les lettres REPARTIES+N ? REPENTIRAS
- 10. Anagramme avec les lettres REPENTIRAS+E ? SERPETAIRE
- 11. Anagramme avec les lettres SERPETAIRE+O ? REPOSERAIENT
- 12. Anagramme avec les lettres REPOSERAIENT+T ? ENTREPOSERAIT
- 13. Anagramme avec les lettres ENTREPOSERAIT+T ? PROTESTERAIENT

This project is Open Source under - [GNU General Public License v3.0](#)

Build by Kacper KUPIS | Qassim-Tayab MANSOOR

- Une route /dictionaries retourne une sortie JSON indiquant la liste des dictionnaires supportés



```
localhost:8888/dictionaries/

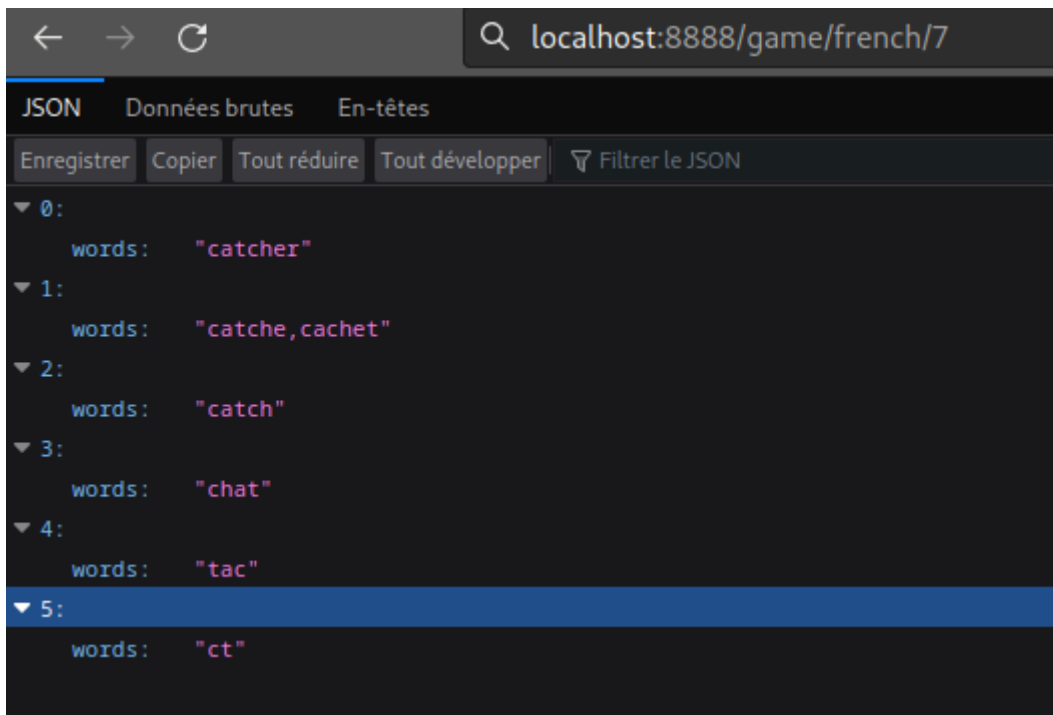
JSON  Données brutes  En-têtes

Enregistrer Copier Tout réduire Tout développer Filtre le JSON

0:
  language: "french"
  file: "src/main/resources/dictionaries/french/french-debian.txt"
1:
  language: "english"
  file: "src/main/resources/dictionaries/english/english-scowl-80.txt"
```

- Une route /game/:dictionary/:n permettant de tirer au sort une séquence d'anagrammes exprimée au format JSON (dont l'anagramme finale comprend n lettres, n étant paramétrable ainsi que le dictionnaire)





## Code

### Game.java

Cette classe est quasiment identique à la classe **Game.java** contenue dans le **Serveur\_UDP**, à la seule différence qu'elle contient des fonctions en moins et d'autres qui sont adaptées à des utilisation d'un serveur web VertX

### MainVerticle.java

Cette classe permet de créer les différentes pages web et de gérer leurs routes.

### Variables:

```
private JSONArray dictionariesJSON
```

contient le dictionnaire au format JSON

### Fonctions:

```
public void start(Promise<Void> startPromise) throws Exception
```

Fonction qui permet la mise en place des pages HTML et des routes pour le serveur VertX

```
private String isDictionaryInDictionaries(String dictionary)
```

Permet de vérifier que le dictionnaire demandé par l'utilisateur dans l'URL existe bien dans les dictionnaires disponibles en entrée.

```
private JSONArray getInfoOnDictionaries()
```

Permet d'obtenir la liste des dictionnaires disponibles dans le dossier indiqué  
Crée ensuite une liste JSON avec en clé "langage" le langage du dictionnaire, et en valeur "file" le fichier texte qui se situe dans le répertoire indiqué.

## Resources

### dictionaries

Contient e un dictionnaire en français et un dictionnaire en anglais : **english-scrowl-80.txt** et **french-debian.txt**

### webroot

#### example.html

Ce fichier permet de mettre en forme la page demandée à l'URL suivante :

<http://localhost:8888/rules/>

#### index.html

Ce fichier permet de mettre en forme la page demandée à l'URL suivante :

<http://localhost:8888/examples/>

## c. AnagramQuestApp

## Code

### MainActivity.java

## Variables :

```
private String selectedDictionary;
```

Variable contenant le dictionnaire sélectionné

```
private Game game;
```

Variable contenant une instance du jeu à créer

```
private ArrayList<String> availableDictionaries = new ArrayList<>();
```

Variable contenant la liste des dictionnaires disponibles qu'il est possible d'utiliser

```
/* On emulator use 10.0.2.2 ip address to access host machine  
address ip */  
private String URLtoDictionaries= "http://192.168.1.76:8888/dictionaries";
```

Variable de type String contenant l'URL sur laquelle doit se connecter l'application Android.

```
private String URLtoGenerateAnagramSequence =  
"https://localhost:8888/game/:dictionary/:n";
```

URL à laquelle il faut accéder pour récupérer la nouvelle séquence d'anagrammes

## Fonctions:

```
protected void onCreate(Bundle savedInstanceState)
```

Fonction principale de l'application Android

```
public void onItemClick(AdapterView<?> adapterView, View view, int i,  
long l)
```

Lorsqu'un dictionnaire est sélectionné on affiche en log le dictionnaire sélectionné, et on enregistre dans selectedDictionary, le contenu du dictionnaire sélectionné

```
public void onNothingSelected(AdapterView<?> adapterView)
```

Lorsqu'aucun dictionnaire n'est choisi, on ne fait rien

```
private void showAnagramToGuess()
```

Montre l'anagramme à deviner

Game.java

Cette classe est quasiment identique à la classe **Game.java** contenue dans le **Serveur\_UDP**, à la seule différence qu'elle contient des fonctions en moins

## 2. Difficultés rencontrées et moyens mis en oeuvre pour les surmonter

1. Compréhension de l'algorithme de génération de séquences d'anagrammes. Pour résoudre ce problème nous avons pris comme inspiration l'algorithme en pseudo code fourni en le modifiant. Cependant l'algorithme implémenté n'est pas le plus efficace car il peut mettre jusqu'à quelques dizaines de secondes pour renvoyer un résultat.
2. Génération trop long de la séquence d'anagrammes, se problème nous a posé particulièrement problème car sur Vertx ainsi que sur android avec la bibliothèque Volley le temps de timeout est assez bas pour améliorer l'expérience utilisateur ce qui a fait qu'on pouvait ne pas avoir de réponse au requête, solution : augmentation des temps de timeout pour les deux.

## 3. Répartition de travail

FICHIER	NOM
<b>Serveur_UDP</b>	
Game.java	Kacper KUPIS
Serveur_UDP.java	Kacper KUPIS
<b>Serveur_Web</b>	
Game.java	

MainVerticle.java	Qassim-Tayab
example.html	Qassim-Tayab
index.html	Qassim-Tayab
<b>AnagramQuestApp</b>	
MainActivity.java	Kacper KUPIS
Game.java	Kacper KUPIS