Projet Commun M1 2016

Alexandre GINGEMBRE

Yohann HUGO

Jérôme MARINTHE

Arthur MOUREY

Mathieu MOUROT

## M1 MIAGE

Les Colons de Katane



# 

# Sommaire

Introduction 3

Conception 3

Réseau : 3

Base de Données : 3

Design Pattern : 3

Sauvegarde : 4

Interface Homme-Machine 4

Difficultés rencontrées et Améliorations 5

Conclusion 6

# 

# Introduction

Ce projet concerne les matières suivantes : Génie Logiciel, Java Avancé, Réseaux et Structuration de Documents.

Le projet a été réalisé en utilisant la méthode SCRUM, vu en cours de Génie Logiciel. Les backlogs de chaque sprint sont disponibles dans le dossier « Sprints » joint au dossier de rendu. Nous sommes partis sur des sprints d’une semaine, ce qui nous a permis de réaliser 6 sprints sur la durée totale du projet.

# Conception

## Réseau :

Nous avons décidé d’utiliser la librairie RMI de Java 8 pour réaliser la partie réseau. Nous avons effectué ce choix, car cette dernière nous permet d’abstraire notre code et de ne pas utiliser les couches basses du réseau, pour éviter le plus possible les anomalies réseaux. Nous n’avons pas sérialisé les objets avec une structure de type JSON ou XML, car RMI le fait pas défaut lors du transfert des objets par le réseau.

## Base de Données :

Nous avons décidé d’utiliser sqlite pour la base de données. En effet, cela nous permet d’avoir un JAR, donc sans installation sur les postes clients. Sqlite nous fournit également juste les outils dont nous avons besoin, ce qui rendra notre exécutable final le plus léger possible.

Pour exploiter cette Base de Données, nous avons utilisé JDBC avec des requêtes préparées, afin de garantir une sécurité minimum.

Les mots de passes sont cryptés avec l’algorithme SHA-ONE. Ce dernier n’est pas très sécurisé, mais il sera modifié dans l’avenir.

Le schéma de relation est joint en Annexe.

## Design Pattern :

Nous avons décidé d’utiliser le Design Pattern « MVC ».

## Sauvegarde :

Pour la sauvegarde des données, nous avons décidé d’utiliser le format JSON.  
Pour générer la sauvegarde, nous avons utilisé la librairie Java Jackson, vue en cours de Java Avancé. Cette librairie a pour avantage d’être rapide et légère.  
Lors de la sauvegarde, pour garantir l’intégrité des données, nous avons décidé d’utiliser un principe vu en réseau : le checksum. Ce dernier est générer grâce à un SHA-ONE, qui est utilisé également sur git. Le risque de collision est très faible.

Nous avons décidé de créer une classe par classe qui sera sauvegardé. En effet, lors des tests de sauvegarde des objets passant par le réseau, Jackson nous levé une exception de récursion infinie. Nous pensons que cela est dû au fait qu’il sauvegarde les états du réseau, qui comporte de très nombreuses informations, levant cette exception.

## Interface Homme-Machine

Nous avons investi dans le jeu afin de comprendre son fonctionnement lors de plusieurs parties et ainsi réaliser un jeu le plus proche de la réalité possible.

Cela nous a également permis de récupérer les images et les règles officielles.

Concernant la conception, nous avons utilisé du JavaFX et du FXML. La génération du plateau est générée avec JavaFX. Nous avons essayé de le générer avec du FXML, mais les premiers tests n’ont pas été concluant. Pour éviter de perdre du temps, nous avons donc décidé de générer les hexagones via du FXML. Le reste de l’interface est générée avec du FXML.

Ce procédé nous permet d’avoir un jeu responsive (mise à part le plateau).

# 

# Difficultés rencontrées et Améliorations

La première difficulté rencontrée a été la génération des Hexagones. Pour réussir cette génération, nous avons utilisé le cercle trigonométrique pour ainsi avoir les points d’un hexagone.

La seconde difficulté a été la réalisation du serveur. Lors de nos premiers tests entre deux machines, le projet ne fonctionnait pas. Le problème venait du fait que nous avions créé une interface Réseau pour toutes les classes. Ce procédé n’était pas viable pour fonctionner. Pour corriger ce problème, nous avons créé une Interface par Objet passant par le réseau.

La difficultés suivante a été la sauvegarde. La boucle infinie créer par les objets dépendant du réseau nous a forcer à créer une classe par objet qui est sauvegardé. Cette façon doit être refactorisée. Nous pensons que cette refactorisation doit être faite en ajoutant des annotations dans les classes qui doivent être sauvegardé. Malheureusement, le temps nous manquant nous n’avons pu réaliser cette dernière dans cette version 1.0. Si nous avions eu plus de temps, cette amélioration aurait été présente dans la version 1.1.

Nous avons eu également une autre difficulté sur le réseau : la lenteur de RMI. Nous aurions dû utilisé les sockets afin de corriger cette lenteur. Cette amélioration serait présente dans notre version 2.0.

Nous avons rencontré également des difficultés sur les tests. Notre projet possédant une partie majoritairement graphique, tester les éléments graphique a été assez compliqué. Avec plus de temps, nous aurions testé des outils tels que Automaton pour réaliser nos tests IHM.

# 

# Conclusion

Ce projet inter-matière nous a permis de nous rendre compte de la complexité de réaliser des projets concrets et conséquents. Il nous a permis également de nous initier à la méthode SCRUM, vu en cours de Génie Logiciel. Nous avons également pu mettre en applications les différents outils présentés en cours de Java Avancé. Ce projet nous a permis également de mettre en Réseau notre projet et d’utiliser la sérialisation en utilisant la structure JSON.  
 Nous avons pu également mettre en application l’outils Git et les difficultés liées au développement parallèle.

Ce projet a été positif pour tous les membres de notre groupe et de notre avis le meilleur projet de notre scolarité au vu de l’ampleur du projet et de son utilité.