

Projet Commun M1 Promotion 2016

Alexandre GINGEMBRE

Yohann HUGO

Jérôme MARINTHE

Arthur MOUREY

Mathieu MOUROT

**M1 MIAGE**

Les Colons de Catane

# 

Sommaire

2

Introduction 3

Choix en Amont 4

Outils : 4

Organisation : 4

Choix de Conception 5

Règles du jeu 5

Réseau : 5

Design Pattern : 5

Base de Données : 6

Sauvegarde : 6

Interface 7

Améliorations 9

Version 1.0 9

Version 2.0 9

Difficultés 10

Méthodes et Planning 10

Interface 10

Réseau 11

La sauvegarde et le chargement 11

Conclusion 12

# Introduction

Ce document est le rapport concernant le projet commun du second semestre de la promotion 2016 de Master M1 Miage de l’université de Lorraine. Les matières concernées sont les suivantes :

* Génie logiciel,
* Java Avancé,
* Réseaux
* Structuration de documents.

Notre groupe pour ce projet est composé de cinq membres :

* Alexandre GINGEMBRE,
* Mathieu MOUROT,
* Arthur MOUREY,
* Jérôme MARINTHE,
* Yohann HUGO.

Dans le cadre de ce projet nous devions choisir parmi une liste de quatre jeux de société, à savoir : les Colons de Catane, Santiago, Tigre et Euphrate et Acquire. Il fallait dans un premier temps choisir notre favori. Ce choix pour nous a été les Colons de Catane et nous avons eu la permission de travailler dessus.

Nous avons choisi ce premier car il était le plus intéressant selon les descriptions que nous trouvions sur internet. C’est un jeu de stratégie, gestion et commerce d’origine allemande crée en 1995, et dont il existe des championnats du monde. Il offre de plus un gameplay riche incluant un plateau, des cartes, différents types de pions et un moyen de gagner simple consistant à réunir dix points d’une manière quelconque selon différents moyens possibles.

Les impératifs suivants ont été mis en place par les différents professeurs :

* Restez le plus fidèle possible aux règles,
* Pouvoir jouer en réseau,
* Technologie Java,
* Connexion et interactions avec une base de données,
* Tests Unitaires,
* Sauvegardes en JSON,
* Développement basé agile avec SCRUM et Git.

Dans ce rapport nous aborderons donc en premier lieux les différents choix faits avant et pendant le projet notamment selon les différentes axes possibles, bases de données, pattern, etc… Dans une seconde partie nous détaillerons les différentes difficultés rencontrées et comment nous les avons gérées. Enfin, dans une dernière partie nous détaillerons les différentes améliorations possibles et envisagées pendant et après le projet. Nous ferons également une conclusion reprenant les points importants du rapport, servant de bilan et d’introspection sur notre travail et ce que nous pensons avoir gagné durant ce projet.

# Choix en Amont

## Outils :

Pour ce projet nous avons travaillé avec différents outils et technologies que nous allons vous présenter.

Le premier outil que nous avons choisi d’utiliser se nomme Trello. C’est un outil de gestion de projet lancé en ligne en 2011, possédant une version payante et gratuite. Cette dernière est amplement suffisante et ces fonctionnalités nous ont permis notamment d’organiser plus aisément notre projet.

L’outil utilise notamment deux composantes : les cartes et les colonnes. Les premières nous ont permis d’organiser nos « Users Stories » avec notamment un nom et une description. De plus, nous pouvons affecter une carte précise à une ou plusieurs personnes et y ajouter des contraintes, par exemple temporelles (date butoir). Enfin, les colonnes nous permettent d’assigner un état d’avancement à la tâche ce qui est très intéressant notamment dans le cas de l’utilisation d’une méthode agile comme SCRUM puisqu’il a pu nous servir de backlog.



Nous avons également utilisé des IDE (Integrated Development Environment), notamment Eclipse, et Intellij de la suite JetBrains, deux outils à notre sens performants.

En outre, nous avons utilisé Git pour créer un projet commun. La branche principale master étant notamment réservée pour les fonctionnalités finies.

## Organisation :

Pour ce projet la communication étant un point crucial, nous avons choisi d’effectuer de nombreuses réunions. Pour les choix importants, tous les membres de l’équipe assistaient à la réunion et pour les choix mineurs seulement les personnes affectées à ces tâches. En raison du planning des cours nous avons utilisé des applications comme Facebook, Skype et Mumble pour dialoguer lorsque nous ne pouvions être physiquement au même endroit.

# Choix de Conception

## Règles du jeu

Pour ce projet la première phase a été de comprendre les règles. Cela dit nous avons constaté en consultant sur internet que ce jeu n’avait pas toujours les mêmes règles et que différentes variantes pouvaient se présenter. Nous avons donc sélectionné un livret de règles nous paressant complet et l’avons adjoint à notre projet comme guide.

## Réseau :

Un des choix important qui s’est posé lorsque nous avons commencé à développer le tchat de l’application : comment allions-nous gérer la partie réseau de l’application.

En cours il nous a été présenté les sockets de Java et la librairie RMI. Nous devions donc trancher sur laquelle de ces deux approches nous utiliserions.

Les sockets nous semblaient plus complexes car ce serait à nous de gérer entièrement la partie réseau et les transmissions là ou l’usage de RMI serait plus simple car lui s’occuperait des échanges sans que nous aillions à programmer l’envoie et la réception des informations.

Après concertation nous avons décidé de choisir le RMI car nous pensions qu’une librairie serait du code plus « sûr », et que les seuls problèmes que nous aurions réellement proviendrait de notre utilisation et non pas de la libraire.

De plus au moment d’effectuer ce choix, l’ensemble du groupe avait travailler beaucoup plus avec RMI qu’avec des sockets, cela fut important et à notre sens orienta notre choix de façon assez notable car la connaissance d’une technologie reste un facteur crucial à prendre en compte pour un projet. Si nous devions refaire ce choix à l’heure d’aujourd’hui, il serait très probable que nous serions nettement plus indécis et que nous pourrions même choisir d’utiliser les sockets.

## Design Pattern :

## Nous avons décidé au commencement du projet d’utiliser une structure le plus possible Modèle-Vue-Contrôleur. Du côté client nous aurions les contrôleurs et les vues permettant les actions et consultations des informations et du côté serveur le modèle sur qui les actions seraient faites.

Cette structure nous est apparue assez simplement car nous avons déjà développé de cette manière sur différents projets. Celui-ci ne présentant aucune contre-indication disqualifiant ce pattern ou le rendant peu performant le suivre a donc été presque plus un réflexe qu’une réelle décision pour nous.

## Base de Données :

Nous avons décidé d’utiliser la technologie sqlite pour la base de données. En effet, cela nous permet d’avoir un JAR, donc sans installation sur les postes clients ce qui représente pour nous un avantage important.

Sqlite nous fournit également juste les outils dont nous avons besoin pour ce projet, ce qui rendra notre exécutable final le plus léger possible et nous évite l’utilisation d’un outil annexe comme PhpMyAdmin avec un serveur WAMP ou MAMP par exemple.

Pour exploiter cette base de données, nous avons utilisé JDBC avec des requêtes préparées, afin de garantir une sécurité minimum, et donc éviter les injections SQL. Pour rester dans cette optique de sécurité minimale, nous avons crypté les mots de passes avec l’algorithme SHA-ONE. Ce dernier n’est pas très sécurisé, mais il nous a permis d’effectuer nos tests et serait probablement remplacé dès que nous aurions mieux, ou besoin de mieux.

Le schéma de relation est joint en annexe.

## Sauvegarde :

La sauvegarde était une fonctionnalité non prévue à la base, mais qui a été rajouté courant du projet elle a donc été abordé un peu différemment du reste du projet car non débattue en amont.

Pour sauvegarder les données d’une partie nous avons donc décidé d’utiliser un format qui gagne en importance actuellement : le JSON.

La génération de la sauvegarde étant le point de départ nous avons voulu nous baser sur ce que nous avions vus en cours à savoir la librairie Java « Jackson ». Elle a pour avantage d’être rapide et légère. Lors de l’utilisation de celle-ci, pour garantir l’intégrité des données, nous avons décidé d’utiliser un principe vu en réseau : le checksum. Ce dernier est généré grâce à un SHA-ONE, qui est utilisé également sur git. Le risque de collision est très faible.

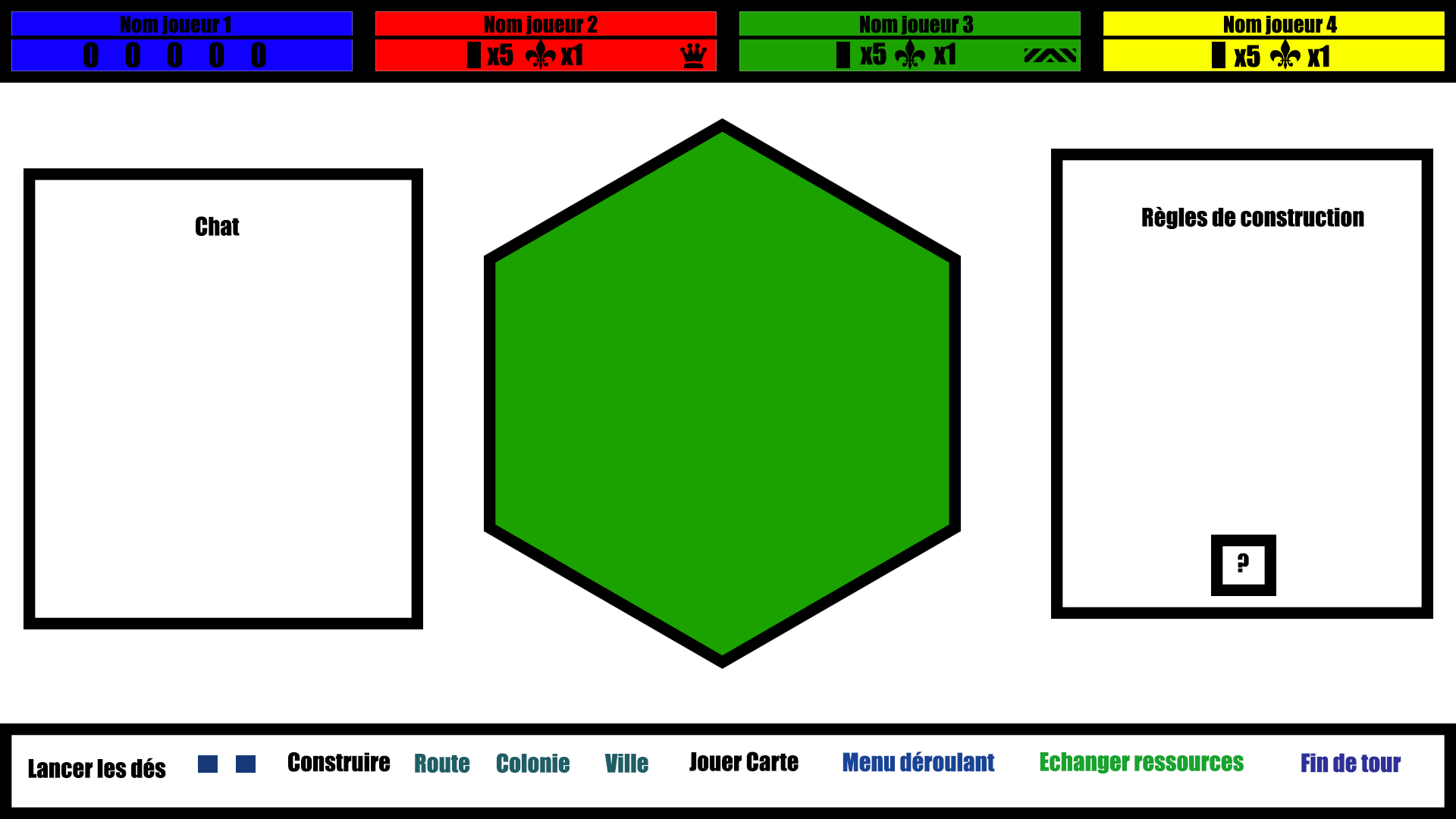
Nous avons choisi de créer une classe dédiée à la sauvegarde pour chaque type d’objet ayant besoin d’être sérialisé. En effet, lors des tests de sauvegarde des objets passant par le réseau, Jackson nous levait une exception indiquant une boucle infinie à cause d’une récursion. Nous pensons que cela est dû au fait que le JSON n’est pas conçu pour gérer ce genre de structure. Bien que cette manière ne soit pas optimale et crée un doublon de code puisque la classe de sauvegarde est très proche de celle de l’objet, cela nous a permis de corriger le problème et de générer une sauvegarde correcte que nous pourrons charger.

Nous avons également récemment trouvé un moyen possible à l’aide d’annotations JSON, mais le manque de temps nous a empêché de faire les modifications pour appliquer cette solution.

## Interface

Dans le cadre de ce projet nous avons tout d’abord commencé par jouer au Colons de Catane sur table, avec un jeu que nous avons acheté. Après quelques parties nous avons eu une bonne compréhension globale des règles. A partir de là nous avons pu commencer à réfléchir sur toute la partie concernant les interactions hommes-machine.

Nous voulions faire une représentation correcte et fidèle du jeu et nous avons donc réfléchi en amont à commencer représenter tout cela. Après quelques débats et plusieurs maquettes nous avions décidé de cette disposition finale des éléments :



Maquette finale de la fenêtre principale du Projet M1-Miage Colon de Catane

Les éléments en haut représentes les joueurs. Le plus à gauche (bleu) est le joueur à qui appartient la fenêtre il peut voir ainsi ses ressources, ses points et s’il possède des cartes spéciales. Les 3 autres éléments du haut sont les adversaires. Dans le cas où la partie n’est que pour 3 joueurs le dernier élément (jaune) n’est pas visible. Le joueur peut ainsi voir combien de cartes de ressources possède chacun de ses adversaires, s’ils possèdent les cartes spéciales et bien évidemment leurs points.

Nous avons choisi d’afficher un nombre de carte ressources plutôt que le détail car nous ne sommes pas supposés connaître les cartes ressources de l’adversaire. Nous voulions rester fidèle au jeu autant que possible.

La partie centrale la plus importante en volume contient à gauche un tchat, important pour négocier un échange, ou relire les actions faites comme le résultat du lancer de dés. Le tchat possède en outre 3 onglets, un diffusant l’ensemble des messages, un pour les messages d’action et enfin un pour juste les messages des joueurs.

Le plateau est le losange central et prend un assez grand espace car il y a beaucoup d’éléments à afficher :

* les tuiles ou cases qui sont les hexagones,
* les espaces constructibles de colonies (points noirs),
* les routes possibles (intersection des hexagones),
* les colonies et les villes respectivement en cercle et triangle,
* les ports qui sont des points d’intersections bleus et donc des espaces constructibles spéciaux,
* le voleur qui est représenté par un effet grisé de la case,
* les jetons qui montrent quel hexagone produit des ressources lors d’un lancé de dé.
* les ressources produites (fond de l’hexagone différent pour chacune des ressources)

L’élément au milieu à droite est une image pour rappeler aux joueurs les différents coûts d’achats dans le jeu, il contient aussi un accès aux règles et la possibilité de sauvegarder.

Enfin le dernier élément en bas, est la barre d’actions récapitulant ce qu’un joueur peut faire

Pour réaliser cette maquette nous avons utilisé du FXML qui permet d’associer un contrôleur à une fenêtre. Cela nous a permis de décomposer notre fenêtre en plusieurs éléments appelés dans un seul et unique fichier et ainsi de créer une structure réelle.

# Améliorations

Nous estimons qu’il est actuellement possible de jouer avec notre jeu. Toutefois certains ajustements seraient nécessaires avant de faire une release d’une version exploitable.

## Version 1.0

La version 1.0 est la version que nous visions pour le rendu du projet, elle contient toutes les demandes du corps professoral.

Nous pensons avoir rempli une grande partie des objectifs fixées. Nous avons une application inspirée d’un jeu de plateau qui fonctionne en réseau, incorpore une base de données pour identifier le joueur et permet une sauvegarde.

Toutefois par manque de temps nous admettons aussi que cette version n’est pas aussi bonne que nous l’aurions voulu, il y ainsi toujours des problèmes, des « bugs ».

Elle n’est pas non plus très performante car nous avons manqué du temps nécessaire pour faire des tests dans un environnement différent du simple local. Ces tests nous ont notamment révélé une certaine lenteur que nous supposons dû à RMI, ou tout du moins à l’utilisation que l’on en fait.

Enfin nous n’avons pas eu le temps de faire beaucoup de refactorisation pour améliorer notre code et notamment le rendre plus modulaire ou clair.

## Version 2.0

Cette version est celle que nous aurions aimé pouvoir rendre. Elle reprend globalement l’ensemble des fonctionnalités de la version 1.0, mais mieux testée, totalement débuguée et bien évidemment refactorisée.

Elle devait nous permettre si possible d’inclure des nouvelles fonctionnalités telles que la création de plusieurs parties simultanément.

Nous avions aussi envisagé une application adaptive pour différents écrans d’ordinateurs. Cela a bel et bien était réalisé excepté pour le plateau dont il fallait revoir la génération et qui n’était pas dans les fonctions les plus pressantes.

# Difficultés

## Méthodes et Planning

La première difficulté pour nous a été d’employer la méthode de SCRUM. Cette méthode vue en cours nous semble rempli d’avantage mais nous n’avons pu réellement la mettre en place.

Tout d’abord le Scrum master, nous ne pouvions choisir une personne dans notre groupe pour remplir ce rôle car pour rendre ce projet dans les meilleurs conditions possibles les cinq membres devaient faire partie de l’équipe de développement.

Ensuite nous n’avions pas de contact avec le « client » ce qui fais que tout ce que nous faisions en nouvelle fonctionnalité était automatiquement validé.

Enfin Scrum est une méthode qui demande des réunions régulières, des sprints d’une durée définie et une grande communication au sein de l’équipe avec des mises au point une fois par jour. Associer cette méthode a nos cours la rend un peu plus complexe à mettre en pratique même dans de bonnes conditions.

## Interface

Une difficulté à laquelle nous n’avions pas pensé lorsque nous avons choisi ce sujet parmi les autres : comment faire le plateau. En effet la génération d’un plateau composé d’hexagones est plus difficile qu’un plateau composé de carré comme celui d’un jeu de dame par exemple.

Pour résoudre ce problème nous avons dû faire une étape de réflexion et recherche et avons fini par trouver un moyen de générer grâce au cercle trigonométrique pour avoir les différents point d’un hexagone qui seraient contenus dans ce cercle.

## Réseau

Deux difficultés assez importantes avec le réseau, notamment liées au choix de l’utilisation du RMI pour ce projet.

La première difficulté que nous avons rencontrée a été lors de l’échange d’objets entre le client et le serveur via RMI. Pour accéder à la référence d’un objet distant, l’objet doit être une instance d’une classe qui :

* implémente une interface qui extends java.rmi.Remote
* doit extends UnicastRemoteObject
* est correctement exportée pour permettre les appels RMI

Nous avons mis du temps avant de nous rendre compte de cela et avons donc perdu beaucoup de temps à essayer de corriger des bugs d’objets dupliqué. Ces objets dupliqués provenaient du fait que nous accédions mal à la référence des objets distants que nous manipulions.

La seconde difficulté provient d’une lenteur de la part de RMI dès que nous passons d’un test local sur machine à un test en Local Area Network entre deux machines.

C’est notamment cette difficulté qui nous a fait douter du bien fondé de notre choix concernant la technologie employé pour le réseau.

## La sauvegarde et le chargement

Une des plus grosses difficultés en termes d’impact a été la sauvegarde. Elle n’était pas prévue au départ et nous avons donc choisi de la mettre petit à petit dans le projet. Toutefois la boucle infinie créée par les objets dépendant du réseau nous a forcé à créer une classe par objet qui est sauvegardé. Cette approche nous a permis d’avoir une sauvegarde valable mais qui doit être refactorisée.

Nous pensons comme dis ci-avant que cette refactorisation doit être faite en ajoutant des annotations dans les classes qui doivent être sauvegardées. Malheureusement, le temps nous manquant nous n’avons pu réaliser cette dernière dans cette version 1.0.

Le chargement en lui-même pour des raisons analogues est actuellement en partie non fonctionnel. Il y a un problème entre la sauvegarde contenant bien les éléments et la reprise de la partie que nous pensons associer à notre usage de RMI.

# Conclusion

Pour conclure, ce projet inter-matière a été perçu de manière différente par rapport aux autres projets que nous avons réalisé jusqu’ici. En effet, nous n’avions pas plusieurs projets de moindre envergure comme nous étions habitués, mais un seul plus conséquent.

A l’instar des projets classiques il nous a permis de pratiquer et nous initier à un ensemble de concepts vus en cours tel que la méthode SCRUM et la refactorisation pour le cours de Génie Logiciel. Pour le cours de Java Avancé, nous avons fait l’usage d’outils tel que le RMI et Jackson et de technologies comme le FXML et le format JSON présenté en cours de structuration de documents.

Nous avons pu mettre en application l’outils Git et les difficultés liées au développement en parallèle notamment lorsque nous ne pouvons rester en contact aisément comme lors de la période des vacances, ou que nous ne pouvons faire assez de mise au point.

Il a été intéressant de réaliser l’importance et la criticité que le réseau à dans une application assez conséquente, notamment pour un jeu qui se veut en temps réel. En effet c’est un aspect qu’il ne faut pas négliger car si le programme manque de performances sur ce point alors il ne sera pas utilisé.

Nous estimons globalement avoir tous progressé durant ce projet, et avons trouvé de nombreux points sur lesquels nous pourrons travailler pour nous améliorer et produire un travail de meilleure qualité.

Nous sommes aussi plus critiques vis-à-vis des outils que nous utiliserons. Les outils nous servent souvent à simplifier mais cela prend parfois trop de temps pour les maitriser, et il serait ainsi plus simple de faire la chose nous-même. Par exemple, pour la partie réseau, l’utilisation des sockets auraient été très probablement un meilleur choix.

Ce projet a donc été positif pour tous les membres de notre groupe et de notre avis le meilleur projet et le plus concret de notre scolarité au vu de son ampleur.