|  |
| --- |
| Mini-Projet GEN |
| Rapport final |
| Huck Mélanie  Moret Jérôme  Nolan James  Santamaria Miguel  Villa David |

09/06/2105

Table des matières

[Introduction 2](#_Toc421612612)

[Analyse 2](#_Toc421612613)

[Règles du jeu 2](#_Toc421612614)

[Diagramme d’activité général 2](#_Toc421612615)

[Cas d’utilisation 2](#_Toc421612616)

[Modèle de domaine 2](#_Toc421612617)

[Base de données 2](#_Toc421612618)

[Conception du projet 2](#_Toc421612619)

[Protocole d’échange client-serveur 2](#_Toc421612620)

[Diagramme de classe 2](#_Toc421612621)

[Base de données 2](#_Toc421612622)

[Autres… 2](#_Toc421612623)

[Implémentation du projet 2](#_Toc421612624)

[Technologies utilisées 2](#_Toc421612625)

[Problèmes rencontrés et solutions apportées 2](#_Toc421612626)

[Gestion du projet 3](#_Toc421612627)

[Rôle des participants 3](#_Toc421612628)

[Plan d’itération initial 3](#_Toc421612629)

[Suivi du projet 3](#_Toc421612630)

[Stratégie de tests 3](#_Toc421612631)

[Stratégie d’intégration du code 3](#_Toc421612632)

[Etat des lieux 3](#_Toc421612633)

[Qu’est-ce qui fonctionne ? 3](#_Toc421612634)

[Qu’est-ce qu’il reste à développer ? 3](#_Toc421612635)

[Auto-critique 3](#_Toc421612636)

[Amélioration 3](#_Toc421612637)

[Conclusion 3](#_Toc421612638)

A l’intention de Monsieur Lefrançois et de Monsieur Greppin.

# Introduction

Ce document décrit les phases d’analyse, de conception, d’implémentation ainsi que de rendu du « mini-projet » développé dans le cadre du cours de GEN.

# Analyse

## Règles du jeu

Il s’agit d’un jeu de plateau, basé sur un système de mini-jeux.

Les joueurs commencent sur la première case du plateau (dite « départ ») ; le premier arrivé sur la dernière case (dite « arrivée ») gagne la partie. Les autres cases n’ont aucune action.

Le jeu se déroule par tour : à chacun d’entre eux, tous les joueurs lancent un dé (nombre compris entre 1 et 6) et avancent du nombre de cases indiqué. A la fin de chaque tour, un joueur (à tour de rôle) peut choisir un mini-jeu, qui sera disputé par tous les joueurs simultanément. Le gagnant pourra tirer deux fois le dé lors du tour suivant ; en cas d’égalité, personne ne peut tirer deux fois le dé.

Chaque mini-jeu est basé sur le protocole suivant : au début du jeu, le serveur envoie les données nécessaires (paramètres de la partie : la difficulté et un seed) aux clients ; durant le jeu, aucune communication n’est effectuée ; à la fin de la partie, chaque joueur envoie son score au serveur qui décidera du vainqueur de la manche.

## Mini-jeux

Conventions : chaque mini-jeu dure un certain temps (déterminé par l’administrateur via la difficulté).

### Challenger

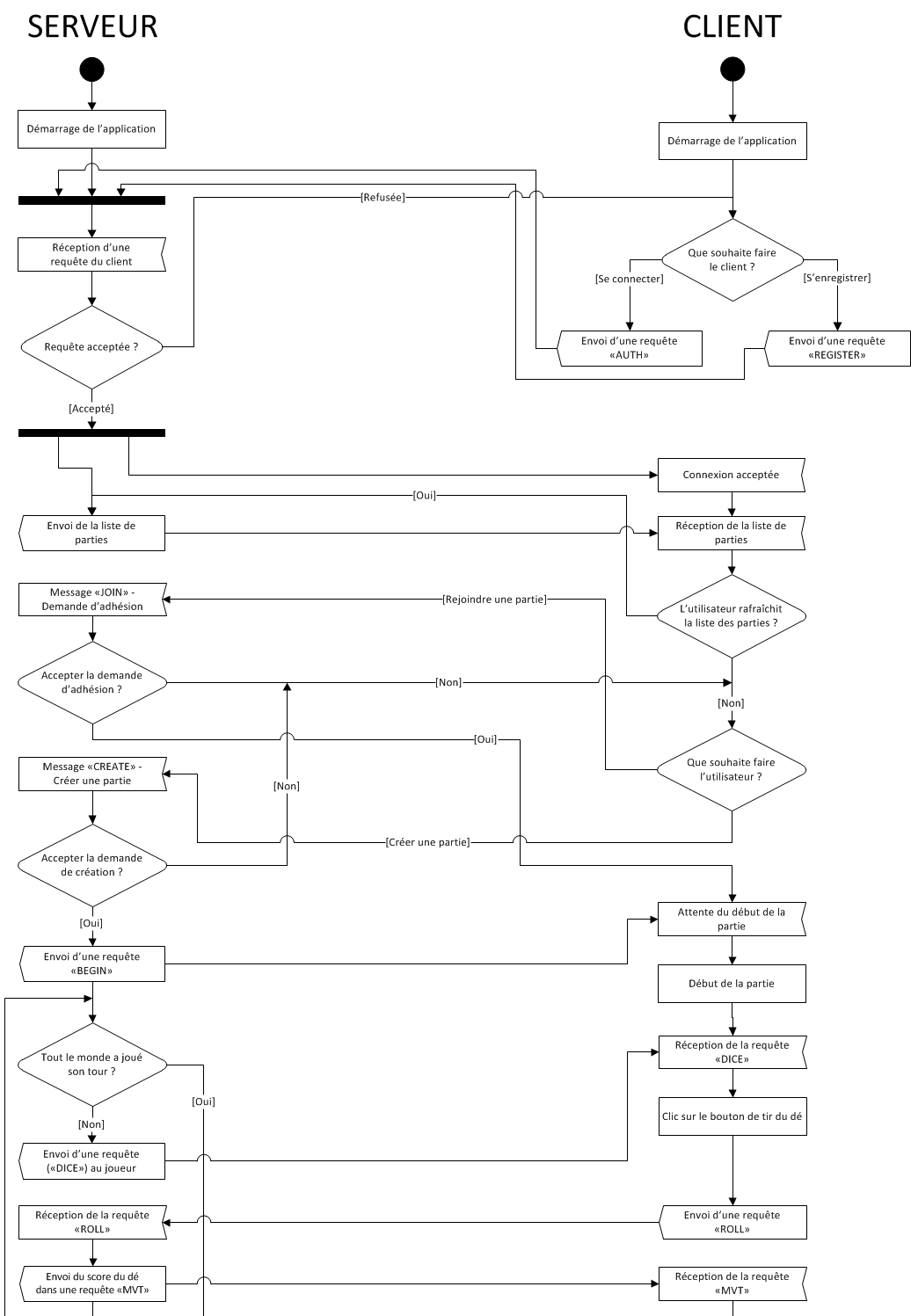
Une planète est située quelque part sur l’écran. Elle possède un nombre déterminé de lunes qui tournent en orbite à vitesse constante. Le joueur doit faire atterrir son curseur (sa « fusée ») sur la planète sans toucher les lunes. S’il y parvient, il gagne un point et une nouvelle planète avec des lunes plus nombreuses apparaît. Si au contraire il échoue (s’écrase sur une lune), il ne gagne aucun point et doit essayer à nouveau. Le jeu s’arrête après un temps déterminé, le but est donc d’atterrir sur un maximum de planètes durant cette période.

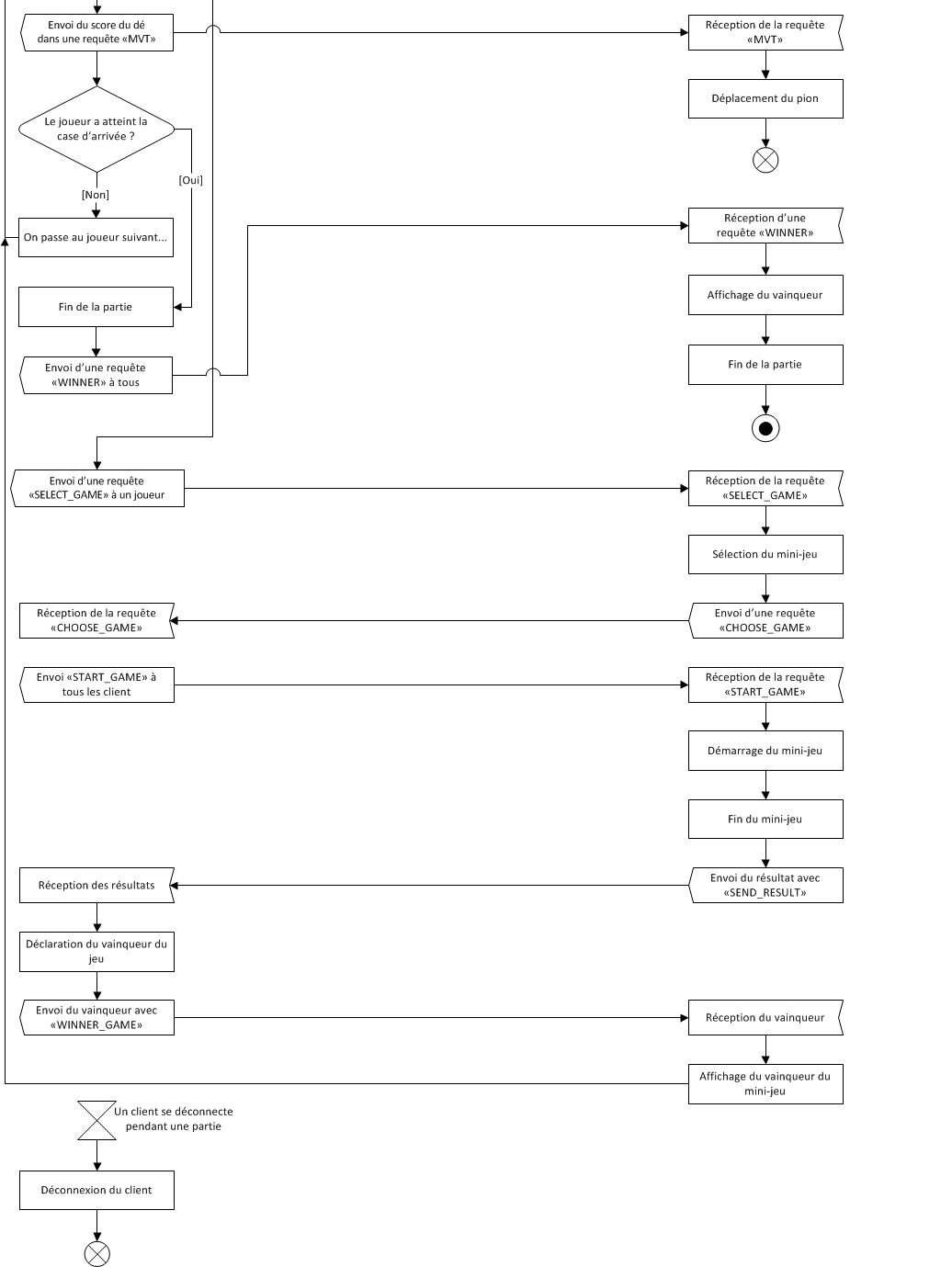
### LetterHero

Ce mini-jeu est une adaptation de « GuitarHero », avec les touches du clavier, mais sans musique : une série de boutons défilent à l’écran de haut en bas (dans l’une des trois zones verticales), et le joueur doit appuyer sur les bonnes touches de son clavier, au moment où le bouton arrive dans une zone propice (située en bas de l’écran). Plus il est synchronisé avec la position du bouton dans la zone, plus cela lui rapportera de points.

### Slurpeur

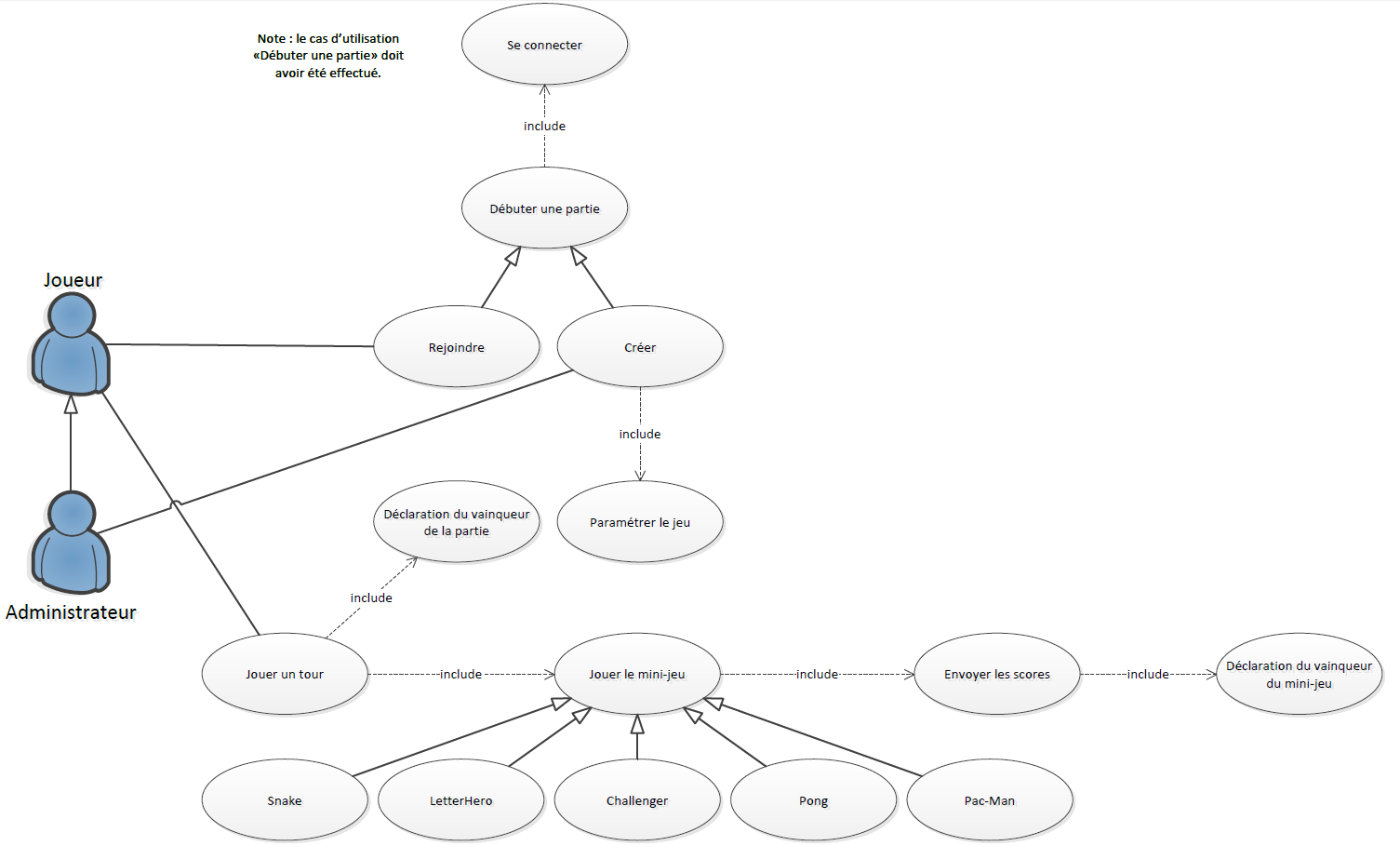
## Diagramme d’activité général





Joint en annexe.

## Cas d’utilisation



### Débuter une partie

Le joueur commence par se connecter **(Se connecter)**. Il a ensuite le choix de rejoindre ou de créer une partie. **(Rejoindre, Créer)**. S’il décide de créer une partie, il devient alors administrateur de celle-ci et peut dès lors paramétrer le jeu. **(Paramétrer le jeu)**. Sinon il rejoint simplement le salon d’une partie déjà crée.

#### Rejoindre

Le joueur entre dans le salon d’une partie. Il ne peut rien faire si ce n’est voir le nom des joueurs adverses et attendre que l’administrateur lance la partie.

#### Créer

Le joueur devenu administrateur configure le jeu.

##### Paramétrer le jeu

L’administrateur paramètre une partie (nombre de joueurs, nombre de case du plateau, difficulté, …).

### Jouer un tour

Chaque joueur tire le dé et est automatiquement déplacé.

### Jouer le mini-jeu

Chaque joueur joue, pendant 30 secondes, localement au jeu **(LetterHero, Challenger, Slurpeur)** choisi par un joueur.

#### LetterHero, Challenger, Slurpeur

Mini-jeux proposés.

### Envoyer les scores

Lorsqu’un utilisateur termine un mini-jeu le score qu’il a effectué est transmis au serveur.

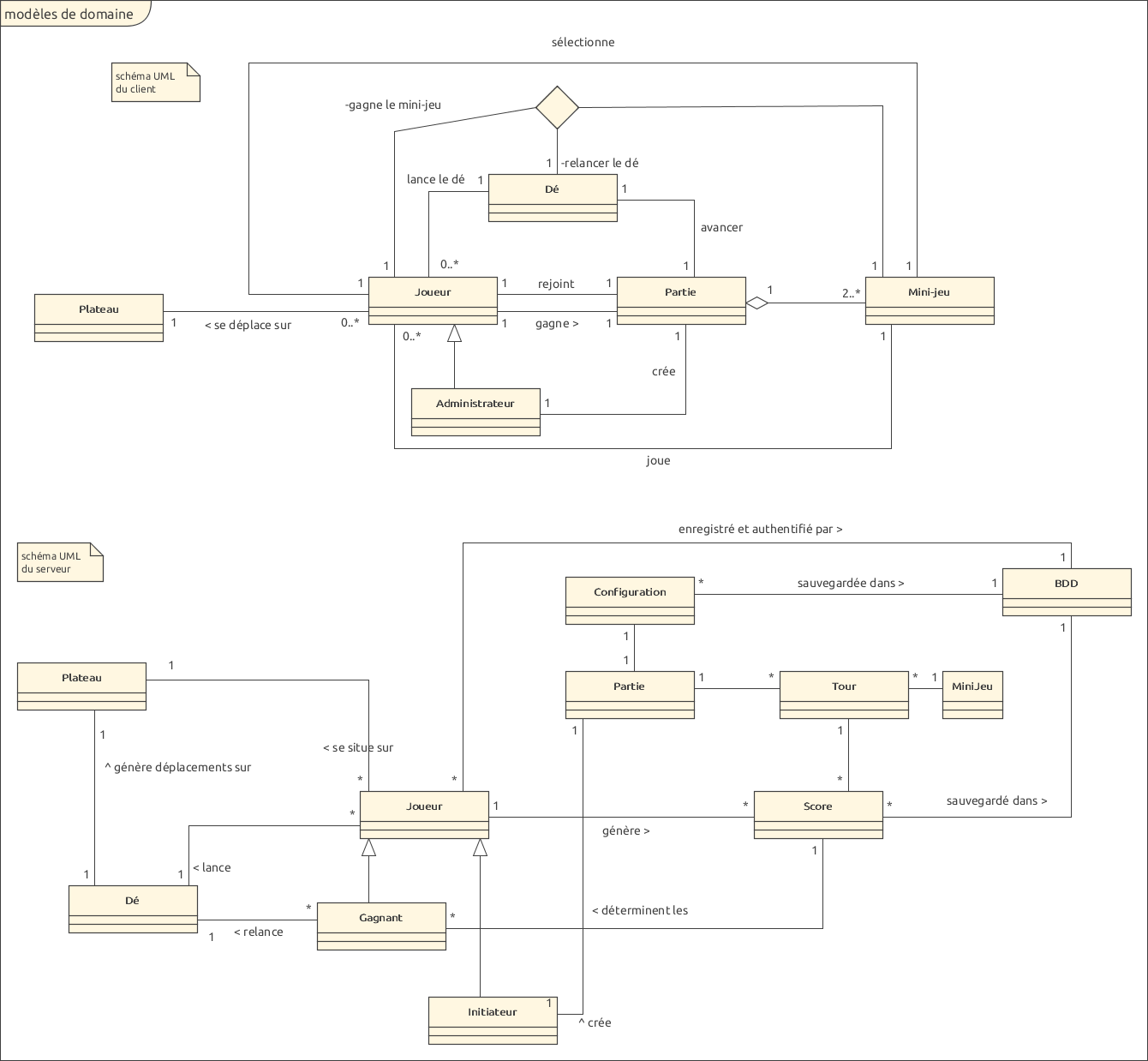
### Déclarer le vainqueur du mini-jeu

Le serveur détermine le vainqueur par rapport aux scores qu’il a reçu.

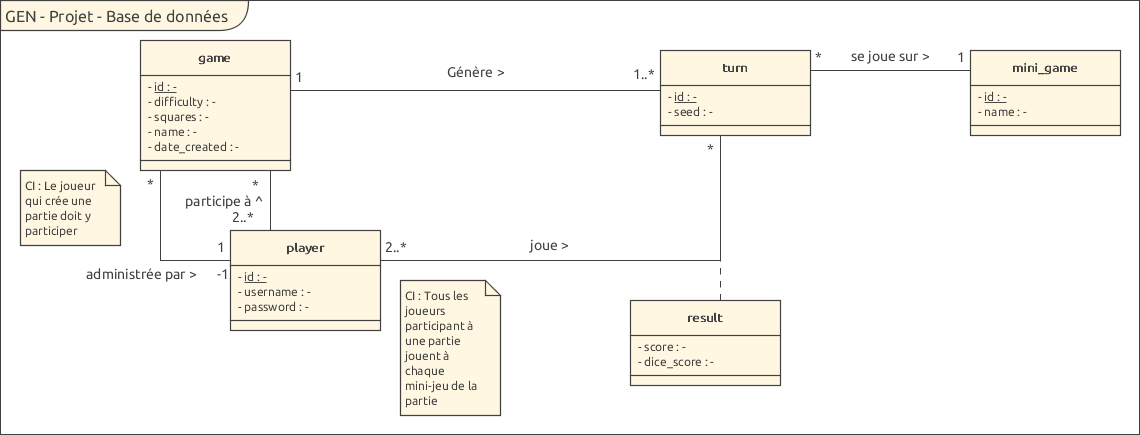
### Déclarer le vainqueur de la partie

Le serveur déclare le vainqueur lorsqu’il détecte un utilisateur arrivant sur la case « arrivée ».

## Modèle de domaine



## Base de données



# Conception du projet

## Protocole d’échange client-serveur

**Interaction entre client-serveur :**

**Connexion :**

* DEBUT 🡪 Connexion au serveur lors du lancement du jeu. Le client doit fournir l’adresse ip et port du serveur – (client <-> serveur) : TCP
* Le client peut s’identifier (pseudo/mdp) – (client -> serveur)
* Le client peut créer un compte (pseudo/mdp) – (client -> serveur)
* Le serveur valide/refuse l’identification – (serveur -> client)
* Le serveur créer/refuse le nouveau compte – (serveur -> client)

**Rejoindre une partie :**

* Serveur indique les parties existantes (1 max dans un premier temps) – (serveur -> client)
* Le joueur peut en créer une (et en devient l’admin) – (client -> serveur)
* Le joueur peut rejoindre une partie
* Le serveur peut annuler la création d’une partie (1 partie max dans un premier temps)
* L’administrateur peut lancer une partie

**Déroulement d’une partie :**

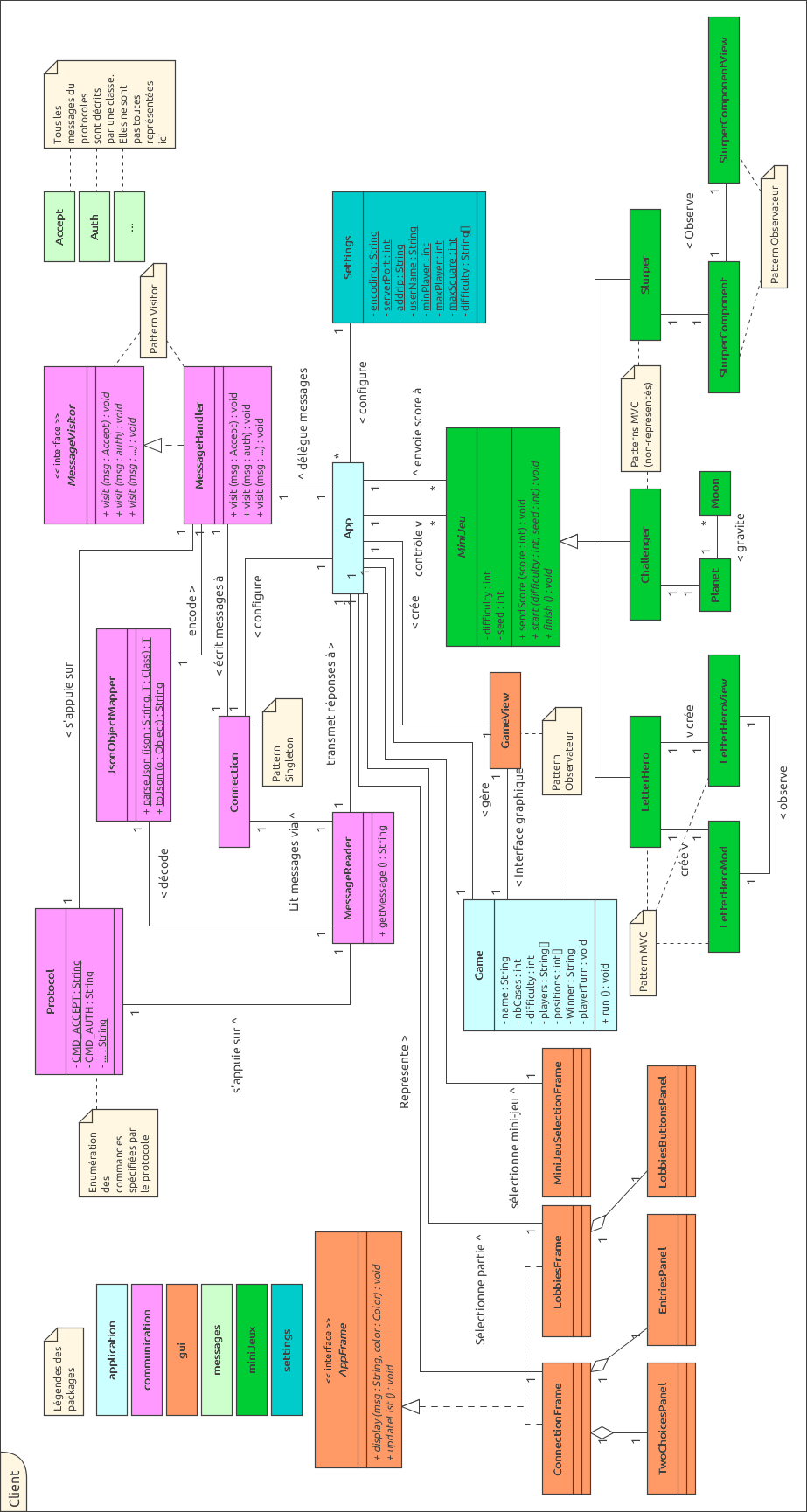
* Attendre validation des joueurs (avec timer) pour le lancer de dés – (client -> serveur)
* Indiquer aux joueurs le résultat du lancer de dés (pour les déplacements) – (serveur -> client)
* Demander au joueur concerné de choisir un jeu – (serveur-client)
* Attendre choix du joueur concernant le mini-jeu à lancer (avec timer) – (client -> serveur)
* Informer les joueurs du mini-jeu choisi ainsi que le seed à utiliser – (serveur -> client)
* Récupérer le résultat de chaque joueur – (client -> serveur)
* Informer le vainqueur d’un mini-jeu – (serveur -> client)
* (optionnel) L’administrateur kick un joueur (client -> serveur)
* Indiquer la déconnection d’un joueur – (serveur -> client)

**Fin de la partie :**

* Informer le vainqueur de la partie – (serveur -> client) 🡪 FIN

## Diagramme de classe

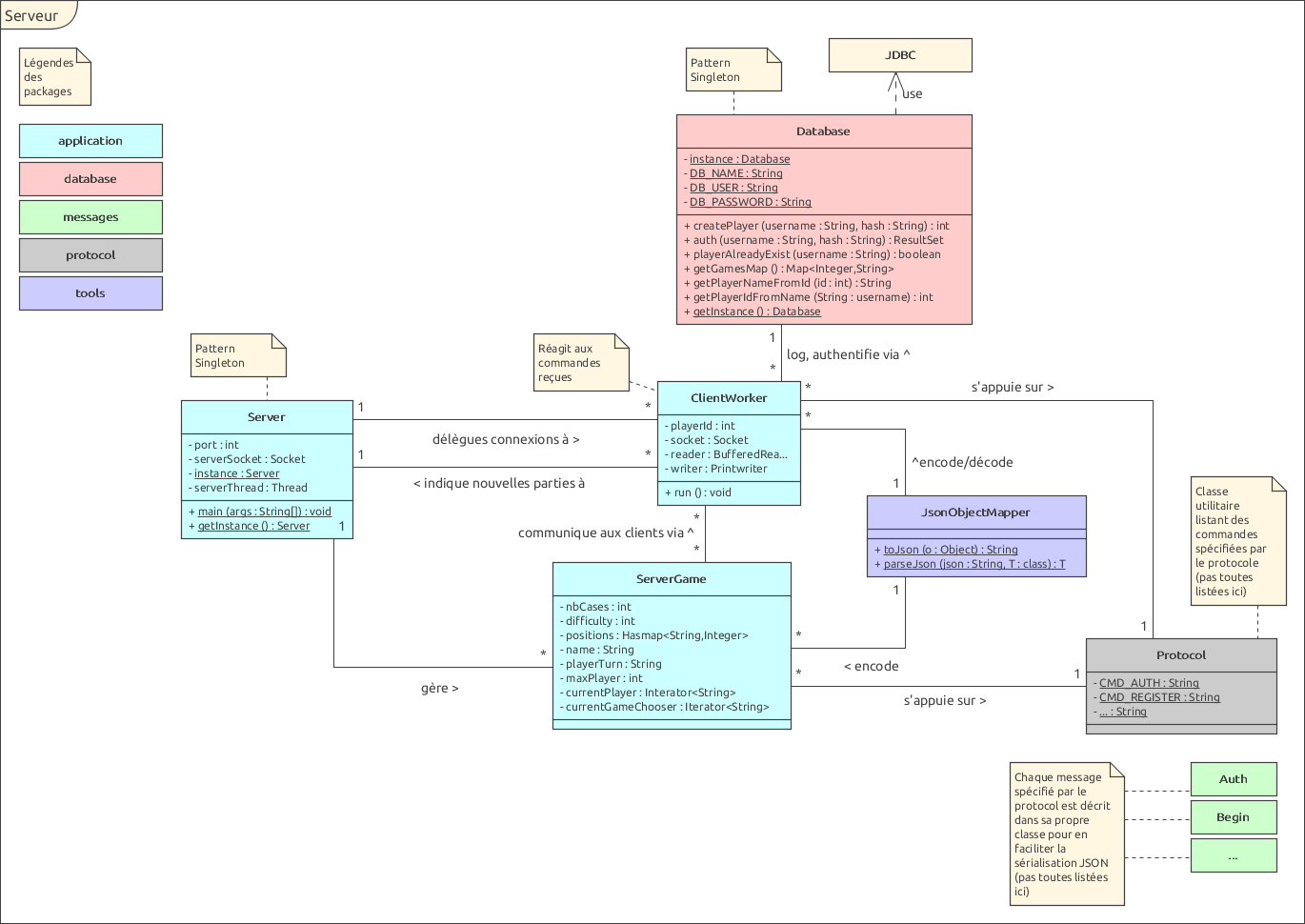
### Client



Le client est divisé en 6 packages :

* **application**, contient les classes permettant le fonctionnement de base du serveur. Une classe centrale *App* permet de transmettre les informations à tous les composants. Une classe *Game* représente une partie en cours. Cette classe contient toutes les informations propres à la partie.
* **communication**, contient toutes les classes utiles à la communication vers et depuis le serveur. Elle contient notamment la description du protocol (classe *Protocol*), un gestionnaire de connexion (classe *Connection*) qui rassemble tous les mécanismes de connexion vers le serveur, et des classes utilitaires permettant d’interpréter (*MessageReader),* d’encoder (*MessageHandler)* et des messages pour le serveur. Une classe utilitaire *JsonObjectMapper* aide ces dernières classes dans leur travail.
* **gui**, contient toutes les vues de l’application. Elle s’interface souvent sur la classe *App*, qui sert de modèle et de contrôleur.
* **main**, contient simplement *main* qui lance l’application (n’est pas représenté sur le diagramme)
* **messages**, contient la description sous forme de classe de tous les types de messages qui peuvent transiter sur le réseau, afin de les décoder / encoder en JSON.
* **miniJeux**, contient toutes les implémentations des mini-jeux. Ils dérivent chacun d’une classe abstraite proposant les méthodes *start(…)*  et *finish(…)* de base.
* **Settings,** contient les paramètres généraux du client.

### Serveur



ServerSocket

* Crée un socket réceptionniste.
* Attend passivement que des clients se connectent.
* Lorsqu’un client se connecte, il crée un nouveau thread de type ClientManager dans lequel il passe le socket du client.

ClientWorker (thread)

* Récupère et stocke le socket du client.
* Lecture/Ecriture sur le flux réseau.

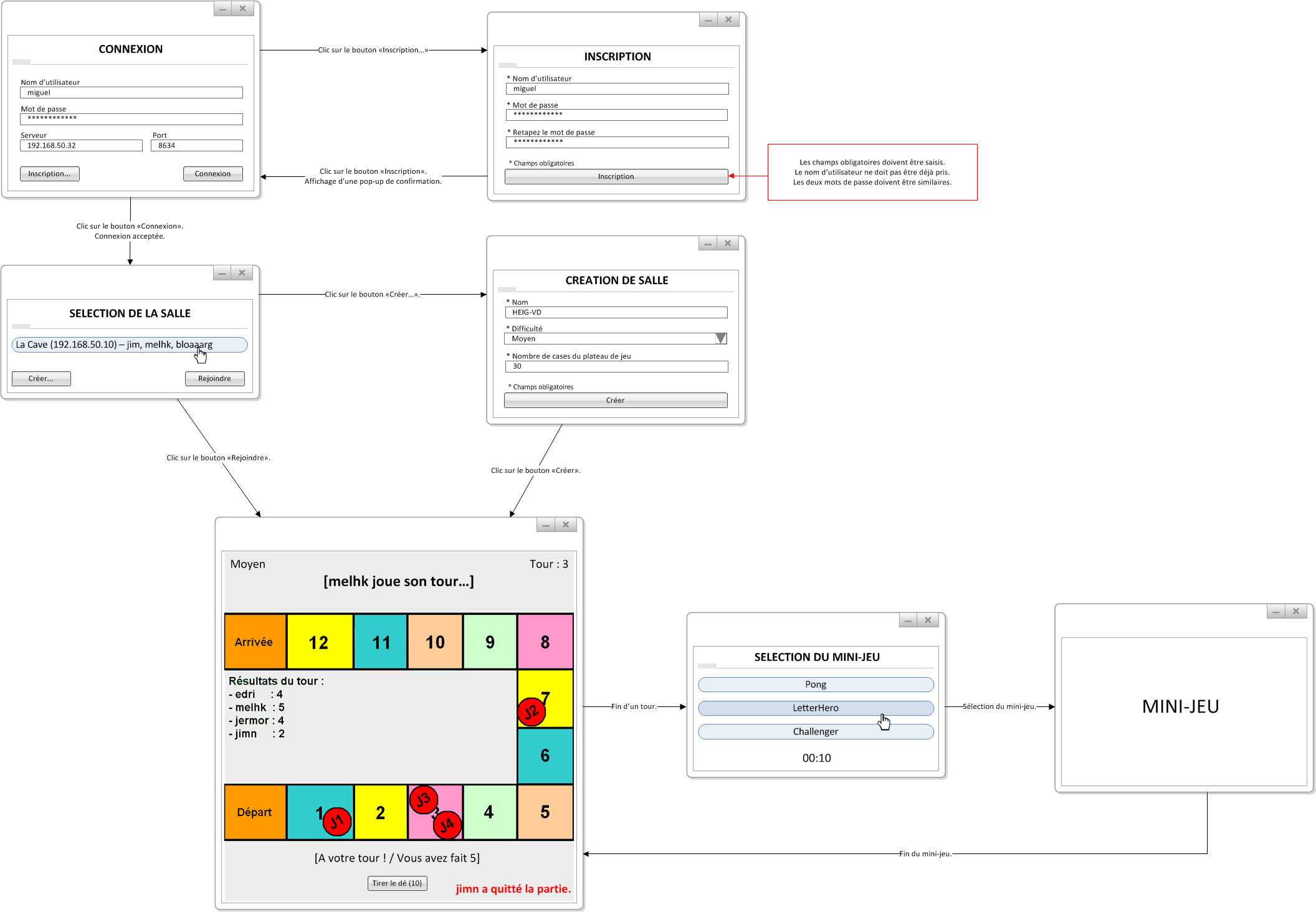
Extractor

* Contient un certain nombre de switch (sur la commande string reçue).
* Pour chaque cas, on utilise le JsonMapper pour créer un objet du type désiré.

MessageHandler (autant de méthodes process que de requêtes/réponses possibles)

* Reçoit l’objet désérialisé par le JsonMapper.
* Agit en conséquence (prépare la réponse, récupération des paramètres, réutilisation du JsonMapper, …).
* Appelle la méthode process désirée qui appelle la méthode write() de ClientManager.

## Ebauche d’une interface utilisateurs



# Implémentation du projet

## Technologies utilisées

Nous avons développé le client et le serveur avec Java, en faisant les interfaces graphiques sous Swing ; les tests ont été effectués sur Windows.

La base de données utilisée est du type MySQL.

La synchronisation du code a été effectuée avec GitHub (voir « Gestion de projet > Stratégie d’intégration du code »).

## Problèmes rencontrés et solutions apportées

Comme dans tout projet, plusieurs problèmes ont été rencontrés :

1. Nous nous sommes rendu compte qu’il était difficile de se synchroniser au cours d’un projet tel que celui-ci. En effet, étant 5 personnes dans le groupe, il a fallu s’organiser correctement.
2. Comme chacun s’est occupé d’une partie en particuliers, nous avons dû faire les formations et mises à niveaux nécessaires (serveur, mini-jeux, etc.), afin que chacun soit au courant du fonctionnement de l’application.
3. Le temps à disposition était restreint (avec le travail des autres cours à côté…), il a donc fallu là-aussi s’organiser correctement. Il a fallu de plus trouver la motivation nécessaire pour mener à bien ce projet, malgré l’énorme travail que nous avions dans les autres cours.
4. Nous avons développé le client et le serveur en même temps, mais avec chacun sur une partie ; lors de la « fusion » des deux, de nombreux problèmes de comptabilité et d’incompréhension sont apparus. Ils ont été résolus après quelques discussions mouvementées, devant un verre de blanc.
5. L’intégration des mini-jeux au client était quelque peu délicate… En effet, il a fallu faire en sorte que le client lance le bon mini-jeu au bon moment, à l’aide du polymorphisme notamment.
6. Certains fichiers ce sont effacés sur GitHub, nous ne savons toujours pas pourquoi à ce jour. Il a donc fallu rattraper le retard engendré par ce problème.

# Gestion du projet

## Rôle des participants

**« Client »**

Eric Lefrançois

**Représentant des utilisateurs**

Miguel Santamaria

**Chef de projet**

Miguel Santamaria

**Analystes**

David Villa : interactions client-serveur.

James Nolan : base de données.

Jérôme Moret & Miguel Santamaria : cas d’utilisations.

Mélanie Huck : modèles de domaines.

**Architecte, conceptrice en chef**

Mélanie Huck

**Programmeurs**

Tout le monde, pour les mini-jeux.

David Villa & Jérôme Moret => plateau du jeu (client).

James Nolan, Mélanie Huck & Miguel Santamaria => serveur.

Mélanie Huck => base de données.

**Responsable des tests**

Mélanie Huck & David Villa

**Responsable de la configuration**

Jérôme Moret

## Plan d’itération / Suivi du projet

## Itération 1 (déjà accomplie)

**Objectif général**

Modélisation métier et spécification des besoins du projet.

**Objectifs détaillés**

* **Gestion** (spécification) :
  + Analyse brute des demandes et besoins du client.
  + Estimation des technologies nécessaires.
  + Analyse des compétences du groupe.
  + Analyse de la faisabilité du projet.
  + Décision quant à l’acceptation du projet.

**Durée**

1 semaine

**Dates de début et de fin**

Du 18.03.2015 au 01.04.2015.

**Partage du travail entre les membres du groupe**

Analyse et discussion concernant le projet, lors d’une réunion de l’équipe. Choix des mini-jeux potentiels effectués personnellement, suivi d’une présentation aux groupes pour acceptation.

**Effort escompté**

Tous les membres de l’équipe étaient présents lors de la réunion d’analyse, ce qui comptabilise un total de 5x3 périodes (à savoir 11h15).

**Bilan de l’itération**

Après concertation avec le client, nous avons déduit les besoins du client ainsi que les technologies et infrastructure nécessaires. A savoir :

* un jeu de plateau multi-joueurs (-> client – serveur).
* style de jeu de tour par tour pour le plateau.
* l’exécution de mini-jeux intégrés au jeu de plateau.
* la possibilité de conserver un historique des parties jouées (-> base de données), ceci afin de pouvoir utiliser ces données pour une utilisation tierce (top100, etc…).

Après d’âpres discussions et débats, nous sommes parvenus à cerner les compétences de chacun des membres de l’équipe, et avons constaté que nous possédions les caractéristiques requises au développement de ce projet.

Nous avons donc donné une réponse favorable au client et accepté de développer ce projet.

Les objectifs de cette itération sont donc atteints.

Autocritique :

N’étant pas vraiment à l’aise avec la méthode UP, cette dernière n’a pas vraiment été appliquée à la lettre durant cette itération. Des débordements sur les itérations suivantes ont été effectués (principalement au niveau conceptuel).

Travail accompli :

Participation à la phase d’analyse avec l’équipe durant la réunion puis réflexion et proposition de mini-jeux qui seront sélectionnés ou non par la suite :

* **Jérôme** : Pong.
* **Miguel** : LetterHero.
* **James** : Challenger.
* **David** : Snake.
* **Mélanie** : Pac-Man.

## Itération 2 (déjà accomplie)

**Objectif général**

Conception.

**Objectifs détaillés**

* Gestion (spécification et conception) :
  + Analyse plus détaillée des demandes et besoins du client.
  + Conception du fonctionnement général de l’application.
  + Analyse des interactions client-serveur.
  + Analyse et définition des cas d’utilisations.
    - Diagramme des cas d’utilisations.
    - Description des acteurs.
    - Scénario principal par cas d’utilisation.
  + Ebauche modèle de domaine.
  + Analyse des objectifs de la base de données et conception d’un modèle.
  + Création du plan d’itération.
  + Ebauche des interfaces utilisateurs.

**Durée**

2 semaines

**Dates de début et de fin**

Du01.04.2015 au 22.04.2015.

**Partage du travail entre les membres du groupe**

Durant cette phase, chaque membre a principalement occupé un rôle d’analyste et de concepteur.

Dans un premier temps nous avons travaillé en commun afin de confronter les idées de fonctionnement de l’application. Une fois mis d’accord sur le fonctionnement globale, nous avons mis sur pied le plan d’itération et nous nous sommes partagés les travaux de conception.

**Effort escompté**

1. Effort total :

Tous les membres de l’équipe étaient présents pour la conception du fonctionnement général de l’application. Ce qui comptabilise un total de 5x2 périodes. A savoir 7h30.

Tous les membres de l’équipe étaient présents pour la création du plan d’itération. Ce qui comptabilise un total de 5x2 périodes. A savoir 7h30.

2. Effort personnel :

Miguel & Jérôme : analyse et définition des cas d’utilisations – 3 périodes chacun.

Miguel : ébauche des interfaces utilisateurs – 1 période.

James : analyse des objectifs de la base de données et conception d’un modèle – 2 périodes.

Mélanie : ébauche du modèle de domaine – 3 périodes.

David : analyse des interactions client-serveur – 1 période.

**Bilan de l’itération**

Après discussion au sein de l’équipe nous avons réussi à nous mettre d’accord sur le fonctionnement général de l’application. Ce qui nous a permis de facilement nous répartir les travaux d’analyse et de conception qui en découlaient.

Une fois cette phase de conception terminée, nous avons pu mettre à jour notre plan d’itérations en conséquence.

Les objectifs de cette itération sont donc atteints.

Autocritique :

Nous nous sommes demandés plusieurs fois à quel point les différentes tâches devraient ou ne devraient pas être faites en commun (par exemple à la définition du plan d’itération, la phase de conception du fonctionnement général, etc…), et comment sont répartis les rôles. Est-ce que les développeurs participent tous à des activités d’analyse ? Si ce n’est pas le cas, que font-ils durant ces activités ? Ceci dans le sens où ce n’est pas logique de commencer à programmer tandis que la conception n’est pas encore terminée.

## Itération 3

**Objectif principal**

L’objectif principal de cette itération est de décrire le protocole de communication entre les clients et le serveur, l’architecture des logiciels les implémentant, et de concevoir la base de données du serveur.

**Objectifs détaillés**

* Gestion (spécifications) :
  + Description du protocole de communication entre les joueurs et le serveur.
    - Commandes permettant à un joueur de communiquer avec le serveur, et vice versa.
    - Informations retournées par ces commandes par les entités destinataires aux entités émettrices.
    - Format du transfert des données.
  + Description des architectures des logiciels implémentant les clients et le serveur.
* Développement de l’infrastructure :
  + Implémenter la base de données selon le schéma produit à l’itération 2.

**Durée**

1 semaine

**Dates de début et de fin**

Du 22.04.2015 au 29.04.2015.

**Partage du travail entre les membres du groupe**

* **Miguel & James** : description des architectures des logiciels.
* **Mélanie** : implémentation de la base de données.
* **Jérôme & David** : Conception et documentation du protocole de communication entre les joueurs et le serveur.

**Effort escompté**

5 périodes par personne, pour un total de 25 périodes.

**Bilan de l’itération**

Tout OK, pas de replanification à opérer.

Présentation d'un diagramme de classes pour l'implémentation de l'aspect communication: OK.

Modèle conceptuel de la base de données, modèle qui a évolué par rapport à ce qui avait été prévu dans le cadre du rapport intermédiaire.

Implémentation de la base de données respectant le modèle conceptuel.

Le protocole a été réalisé, mais non présenté par manque de temps. Un feed-back sera donné à l'occasion de la prochaine itération

**Bilans personnels**

**Miguel**

Tout à fait, cette première itération de conception et d'implémentation à proprement parler n'était pas très touffue, mais nous a permis de partir sur de bonnes bases, notamment grâce à la conception d'un protocole de communication solide et réfléchi. L'architecture des logiciels, ainsi que l'implémentation de la base de données nous ont aussi permis d'avoir les idées plus claires, et de résoudre les quelques conflits apparus (tout le monde n'ayant pas forcément la même vision de l'application au début, comme dans tout projet). Les tâches ayant été bien réparties, nous avons pu accomplir l'itération dans les temps, sans problème particulier, ce qui en fait donc un succès.

**James**

Cette itération était la première qui nous demandait de réellement produire du code. Cela m'a personnellement fait prendre conscience de la difficulté de penser à tous les détails d'implémentation lors de la phase de conception. J’ai essayé de ne pas me laisser dérouter par ces inquiétudes et ai suivi le plan tel que nous l'avions défini. Cela m’a permis de me focaliser sur les besoins de l’itération. A priori, cette stratégie semble efficace car, à la fin de l’itération, nous avions un premier résultat que nous pouvions tester avec succès.

**Mélanie  
Cette itération était relativement aisée. Nous avons dû penser à l’architecture globale de notre application, ce qui ne nous a pas posé de gros problèmes d’implémentation. Cette partie était très intéressante, et je pense très importante pour le bon déroulement de la suite du projet.​**

**David**

Cette itération nous a permis, entre autre, de définir l'implémentation et les interactions entre nos clients et le serveur. Il est appréciable de noter que sans la méthode nous aurions surement déjà commencé l'implémentation de l'infrastructure réseau et de nombreux problèmes n'auraient été identifié que tardivement. Grâce à cette itération nous avons une idée clair et précise du comportement qu'adopterons nos clients/serveur et du protocole à utiliser, nous permettant ainsi de pouvoir travailler de concert, développeurs comme testeurs.

**Jérôme**Cette première itération d’implémentation m’a permis de plonger les mains dans le code. Tout à fait, en effet, le but a été d’implémenter un serveur qui permet d’attraper chaque connexion d’un client par socket et d’y déléguer le travail à un thread pour ne pas bloquer le serveur. A la fin de cette itération le serveur autorisait des connexions à outrance et gérait déjà la commande AUTHENTIFICATION.

## Itération 4

**Objectifs principaux**

Création de la base de données.

Implémentation et test des communications avec la base de données (depuis le serveur) ; à la fin de l’itération, le serveur est terminé (il peut toujours être sujet à quelques changements par la suite).

Implémentation et test du plateau de jeu (communications métiers avec le serveur).

**Objectifs détaillés**

En particulier, il s’agira de :

* **Développement de fonctionnalité** (cas d’utilisation « Se connecter », partiel) – Enregistrer un utilisateur dans la base de données à partir de l’application du serveur. Cette fonctionnalité répond au cas d’utilisation qui demande au joueur de créer un compte. Cette fonctionnalité remplit partiellement le cas d’utilisation car il s’agit de l’implémentation côté serveur du cas d’utilisation. Le client devra implémenter la récolte des identifiants de façon ergonomique.
* **Développement de fonctionnalité** (cas d’utilisation « Se connecter », partiel) – Valider l’authentification d’un compte à partir d’un pseudonyme et d’un mot de passe (ou d’un hash ?) en interrogeant la base de données. Cette fonctionnalité permettra au joueur de se connecter. Elle remplit partiellement le cas d’utilisation car il s’agit de l’implémentation côté serveur du cas d’utilisation. Le client devra implémenter la récolte des identifiants de façon ergonomique.
* **Développement de fonctionnalité** (cas d’utilisation « Créer ») – Créer une partie et l’enregistrer dans la base de données. Cette fonctionnalité répond au cas d’utilisation indiquant que le premier joueur puisse créer et paramétrer la partie. Elle remplit partiellement le cas d’utilisation car il s’agit de l’implémentation côté serveur du cas d’utilisation. Le client devra implémenter la récolte des identifiants auprès de l’utilisateur de façon ergonomique.
* **Développement de fonctionnalité** (cas d’utilisation « Rejoindre ») – Rejoindre une partie. Cette fonctionnalité répond au cas d’utilisation permettant à un joueur de rejoindre une partie déjà existante. En principe, elle répond plus ou moins à la totalité de ce cas d’utilisation.
* **Développement de fonctionnalité** (cas d’utilisation « Jouer un tour ») – Faire lancer le dé une fois à tous les joueurs. Le joueur qui a obtenu le plus haut score à l’éventuel tour précédent devra le lancer deux fois. Les joueurs devront répondre dans un temps déterminé et le serveur lancera lui-même le dé si l’un ou plusieurs des joueurs n’a pas réagi à temps. A chaque fois qu’un joueur lance un dé, le résultat est communiqué aux autres joueurs. Si, grâce à son lancer, le joueur gagne le jeu, ce dernier est interrompu. Cette fonctionnalité devrait résoudre la totalité des cas d’utilisation de la gestion des dés et de la déclaration du vainqueur du point de vue du serveur, mais le client devra implémenter la représentation graphique de cette procédure.
* **Développement de fonctionnalité** (cas d’utilisation « Jouer le mini-jeu », partiel) – Demander à un joueur de choisir le prochain mini-jeu. Le joueur doit alors pouvoir communiquer son choix. S’il n’a pas répondu à temps, le serveur choisit pour lui. Lorsque le choix est effectué, ce choix est communiqué à tous les joueurs.
* **Développement de fonctionnalité** (cas d’utilisation « Envoyer les scores » et « Déclaration du vainqueur du mini-jeu ») – Le joueur doit pouvoir indiquer le score obtenu pour une partie. Le serveur enregistre cette information dans la base de données et attend que tous les joueurs aient envoyé leur score, à chaque tour. Lorsque tous les scores sont reçus, le vainqueur est déterminé et la procédure reprend à la gestion du dé. Cette fonctionnalité répond au cas d’utilisation de la gestion du score d’un point de vue serveur. Bien entendu, il s’agira d’implémenter les mini-jeux qui généreront ce score en interaction avec le joueur.

**Durée**

3 semaines.

**Dates de début et de fin**

Du 29.04.2015 au 20.05.2015.

**Partage du travail entre les membres du groupe**

* Mélanie : développement de l’infrastructure de la base de données + intégration + tests.
* James & Miguel : développement des fonctionnalités du serveur.
* Jérôme & David : développement des fonctionnalités du plateau de jeu.
* David : validation des interactions client->serveur, et vice-versa.

**Effort escompté**

* Mélanie : 12 périodes.
* James & Miguel : 18 périodes chacun.
* Jérôme & David : 17 périodes chacun.

Total : 82 périodes.

**Bilan de l’itération**

Fonctionnel en gros, conformément à ce qui était prévu, si ce n'est qu'il manque:

* La base de données
* Le mini-jeu

Notons que tous les messages sont prêts.

**Bilans personnels**

**Miguel**

Tout à fait, je pense que prévoir une itération sur 3 semaines était une erreur, parce que nous avons repoussé la tâche jusqu’au dernier moment (à cause du travail que nous avions dans les autres cours), ce qui nous a forcé à travailler à outrance durant les derniers jours. Nous sommes tout de même arrivés à quelque chose de correct dans le temps imparti, mais cela nous a demandé un grand effort. En tant que chef de groupe, il était aussi difficile de gérer l’équipe.  
Le projet poursuit son cours, nous devrions arriver à le terminer dans les temps.

**James**

Cette itération était effectivement un peu laborieuse. J'ai été trop facilement tenté de mettre les délais "lointains" de côté pour répondre aux obligations d'autres cours. C'est un peu tard que je me suis aperçu qu'il restait encore beaucoup à faire. En plus de ça, je n'ai pas assez tenu compte des imprévus dans mon timing. Cela a posé quelques malentendus fâcheux au sein du groupe, mais heureusement, au final, nous étions tous soulagés et satisfait du travail que nous avons fourni.

**David**

Cette itération était plus imposante que je ne le pensais. Ceci combiné à la charge de travail des autres cours a fait que nous n'avons pu réussir à atteindre la globalité des objectifs annoncés. Mais ceci reste une erreur d'organisation et, heureusement, nous avions prévu du temps pour modifier les itérations suivantes sans perturber la date final du rendu.

A côté de ce problème, je suis satisfait du travail que j'ai produit ainsi que l'implémentation de mon modele de visitor. Il s'agissait de la première fois que j'en mettais un en pratique et je suis content du résultat.

**Mélanie**

Cette itération m’a posée également passablement de soucis d’organisation. La charge de travail par rapport au reste des cours à fait que je n’ai pas pu travailler autant que je le voulais sur ce projet. Néanmoins, je suis satisfaite du résultat final, d’autant plus que nous sommes arrivés au bout de quelque chose de fonctionnel à la fin de cette itération.

**Jerôme**

À la fin de cette itération j'ai pu prendre conscience de la difficulté à répartir les tâches dans une équipe. Tout à fait, en effet, je me suis rendu compte qu'il était plus efficace de placer des collaborateurs sur des tâches complètement indépendantes des miennes car le peu de temps qu'on travaille ensemble fait que l'on part dans des directions opposées qui créent pas mal de conflit. Aussi, git a posé pas mal de soucis avec des suppressions non attendues. Enfin, j'espère pouvoir récupérer le retard que j'ai pris pour remettre l'église au centre du village.

## Itération 5

**Objectif principal**

Il s’agira ici principalement de rattraper le retard engendré lors de la dernière itération, en implémentant en parallèle les mini-jeux.

**Objectifs détaillés**

* **Développement de l’infrastructure** : réalisation de la base de données et connexion avec le serveur, afin de travailler sur de véritables données.
* **Développement de fonctionnalité** (cas d’utilisation « Jouer le mini-jeu »): développement des mini-jeux, chaque mini-jeu est développé individuellement mais reçoit les mêmes informations de base de la part du serveur. Chaque mini-jeu renvoie également les mêmes informations lorsqu’il se termine.
* Premiers tests des mini-jeux, au fur et à mesure de leur développement.
* Intégration des mini-jeux au plateau de jeu lorsque ceux-ci sont suffisamment développés, et premiers tests de l’intégration.
* Au minimum 2 mini-jeux seront développés parmi ceux proposés plus haut.

**Durée**

1 semaine

**Dates de début et de fin**

Du 20.05.2015 au 27.05.2015.

**Partage du travail entre les membres du groupe**

* David & Jérôme : finalisation du serveur.
* James, Mélanie & Miguel : développement des mini-jeux.

**Effort escompté**

6 périodes par personne, pour un total de 30 périodes.

**Bilan de l’itération**

La base de données est créée et les données sont tirées directement pour les logins et les mini-jeux (niveau de difficulté). Le résultat aux mini-jeux n’est pas encore enregistré (jeux indépendants et pas encore intégrés au plateau).

Il existe 3 mini-jeux (dont un pas totalement terminé à 2-3 bugs graphique et manque juste timer de fin). L’intégration des mini-jeux au plateau n’est pas encore faite. Tous les jeux se terminent à la fin d’un timer et donne un score (pas encore envoyer au serveur).

1)      Mini jeu de clique sur un bonhomme qui se déplace aléatoirement (chaque clique réussi donne 10 pts).  Jeu à terminer

2)      Challenger. Une planète, plusieurs satellites (qui augmentent à chaque tour voir double en cas d’échec). Le but est de toucher la planète sans se prendre un satellite avec la souris

3)      LetterHero : Sorte de guitare Héro avec lettres qui descendent à l’écran (3 lettres en même temps). Il faut taper la lettre au clavier quand la lettre passe dans une zone définie. Les lettres changent tout au long du jeu.

**Bilans personnels**

**Miguel**

Tout à fait, cette partie était beaucoup plus « fun » en ce qui me concerne, vu qu’il s’agissait du développement du mini-jeu LetterHero. Celui-ci m’a beaucoup appris, notamment en ce qui concerne Swing, avec lequel je n’étais pas forcément très à l’aise. Nous étions partis sur l’optique de faire des mini-jeux simplistes, afin d’avoir le temps nécessaire pour les faire, mais après m’être dit « Peu m’en chaut, il s’agit de quelque chose d’amusant qui m’apportera une bonne expérience ! », j’ai décidé de miser plus de moyens là-dedans.  
Cette itération s’est déroulée sans incident particulier, et nous avons cette fois-ci pris nos précautions quant à la répartition du travail (éviter de tous faire durant les derniers jours).

**James**

Comme mon rôle dans cette itération était de développer un Mini-jeu, je n'ai eu que très peu besoin d'interagir avec les collègues. Ceci rend évidemment les choses plus faciles, mais ne sont pas très intéressantes du point de vue de la gestion du projet. Je dois toutefois admettre que je ne suis pas un grand fan de la production d'interfaces graphiques, mais l'avantage est que cela m'a forcé à plonger dans Swing et je pense avoir beaucoup appris.

**Jérôme**

Dans cette itération, mon rôle fût d’implémenter les interactions avec la base de données. J’ai dû rester de marbre et affronter JDBC et son API. API finalement assez simple, j’avais tout de même, au début, oublié d’installer le driver MySQL Connector, pièce maitresse dans la procédure de connexion à la base de données. Quelques détails ont été corrigés dans la base de données, ils sembleraient, en effet, tout à fait que James ait voulu être plus catholique que le pape lors de la conception de la base de données. Pendant tout ce temps la vérification du bon fonctionnement de ma partie a dû être effectuée à l’aide de la procédure de test console et pas graphique il a fallu mettre de l’eau dans son vin pour attendre ce moment tant attendu qui sera prêt à la prochaine itération : « L’interface graphique client ! ».

**Mélanie**

Cette itération a été la plus amusante pour moi, étant donné que j'étais chargée de la création de l'un des mini jeux. J'ai passé quelques heures intenses à faire slurper mon slurpeur, et swinguer swing. Je n'ai donc pas eu beaucoup d'interactions à avoir avec mes collègues, à l'exception du fait de se mettre d'accord sur ce que le mini jeu reçoit du / revoit au serveur.

**David**

Je n'ai pas grand-chose à dire concernant cette itération. Mon travail était le commencement de la GUI du logiciel, je travaille tranquillement dans mon coin en me débâtant contre les layouts de swing et leur auto-ajustement des tailles de panels.

## Itération 6

**Objectif principal**

Implémentation et test du plateau de jeu (GUI, gestion des joueurs, actions, …) ; à la fin de l’itération, le plateau de jeu est terminé et communique avec le serveur.

**Objectifs détaillés**

* **Développement de fonctionnalité** (cas d’utilisation « Débuter une partie » « Paramétrer le jeu » et « Jouer un tour ») - Développement du modèle du plateau de jeu :

         Récupération des paramètres de jeu.

o   Nombre de case

o   Difficulté

o   …

         Récupération du nombre de joueurs.

         Implémentation de la structure de données représentant la grille de jeu.

o   Gestion des déplacements

         Implémentation d’un gestionnaire de tours.

         Communications client-serveur pour les tirs de dés.

         Implémentation et communications client-serveur pour le choix du mini-jeu.

         Gestion des comptes à rebours pour le tir et du dé et le choix d’un mini-jeu.

         Gérer les mises à jours de chaque client par rapport aux informations du serveur.

* **Développement de l’infrastructure** : Développement de la vue du plateau de jeu :

         Création de l’interface graphique.

o   Algorithme pour le dessin de la grille de jeu.

  Doit d’adapter aux nombres de cases. (Dans la limite du minimum et maximum).

o   Gestion des superpositions des joueurs (2 joueurs sur la même case).

o   Fenêtre de sélection du mini-jeu

o   Etc…

* **Développement des fonctionnalités** : tests de l’ensemble du programme afin de corriger les derniers problèmes.
* Rédaction : Finalisation de la documentation.
* **Développement des fonctionnalités** : Déploiement du projet.

**Durée**

2 semaines

**Dates de début et de fin**

Du 27.05.2015 au 10.06.2015.

**Partage du travail entre les membres du groupe**

Tous les membres seront assignés au développement et aux tests.

**Effort escompté**

12 périodes par personne, pour un total de 60 périodes.

**Bilan de l’itération**

Applicatif quasi fonctionnel, il reste la partie client à finaliser, ce qui ne devrait pas causer de problème.

**Bilans personnels**

**Miguel**

Tout à fait, comme mentionné dans le bilan de l’itération, cette dernière s’est déroulée sans encombre particulière. Etant donné qu’elle durait 2 semaines, nous avons évité de refaire les erreurs que nous avions commises lors de l’itération 4), à savoir de s’y prendre au dernier moment. Nous devrions réussir à finaliser le projet ainsi que la documentation sans trop de problèmes d’ici mercredi prochain.

## Stratégie de tests

### Tests fonctionnels

Les tests dits fonctionnels ont été effectués au fur et à mesure des itérations. De ce fait, chaque nouvelle fonctionnalité a été testée à la fin de son développement, ceci pour éviter que des bugs s’enchaînent au fil de l’avancement du projet.

Ces tests ont été effectués particulièrement par David et Jérôme.

### Tests finaux

D’une fois que notre application a été terminée, nous l’avons retestée en profondeur, afin de détecter les éventuels bugs encore présents, ou ceux qui seraient apparus en imbriquant les différentes itérations. Il est en effet possible qu’un nouveau bug apparaisse à la toute fin du développement, entraînant un enchaînement en cascade de nouveaux bugs.

Ces tests ont été effectués par Mélanie.

## Stratégie d’intégration du code

Pour synchroniser notre code et l’intégrer au projet, nous avons utilisé la technologie git, avec l’outil GitHub. Nous avons en effet un répertoire GitHub sur lequel chaque membre du projet est collaborateur. Nous pouvions ainsi synchroniser facilement nos travaux, et gagner un temps précieux.

Lorsqu’un membre a implémenté une fonction et l’a testée, il l’ajoute au répertoire git puis en informe les autres membres, qui récupèrent la nouvelle version et s’assure qu’elle fonctionne avec leurs ajouts/modifications.

# Etat des lieux

## Qu’est-ce qui fonctionne ?

## Qu’est-ce qu’il reste à développer ?

# Auto-critique

Plutôt que de faire une auto-critique générale du projet, nous avons décidé d’en faire une par personne, afin de découvrir ce qui s’est fait ressentir tout au long du projet, ainsi que le ressenti final de chaque membre.

**Miguel**

L’ auto-critique qui suit est axée plutôt sur la gestion de projet, étant donné que j’ai été volontaire dans le rôle de chef de projet.  
Tout à fait, en règle générale je pense que ce projet a pu être mené à bien tout au long du semestre. Nous avons en effet fourni un travail plutôt bon, en ayant réussi à nous partager les tâches de manière équitable et homogène, et en implémentant la quasi-totalité de ce que nous avions planifié. Chaque membre de ce groupe de projet a joué le (mini-)jeu, en faisant ce qui lui était demandé selon le planning imparti, sans prendre de réel retard ; cela s’avère finalement être un véritable exploit, étant donné la charge de travail surhumaine que ce semestre nous a « offert ».  
Nous avons bien-sûr rencontré quelques problèmes organisationnels et techniques, comme dans tout projet digne de ce nom, mais nous avons toujours su rester authentiques en les gérant correctement en interne, sans que cela se ressente sur le reste.  
Finalement, un projet tel que celui-ci nous apporte une bonne expérience, notamment pour notre futur parcours professionnel ; j’ai également apprécié mon rôle de chef de projet, car il a su me responsabiliser et m’a appris à gérer correctement une équipe.

## Amélioration

# Conclusion

Nous évaluons notre travail comme étant plutôt bon, bien que forcément pas parfait.