

# GEN - Projet

## Rapport intermédiaire

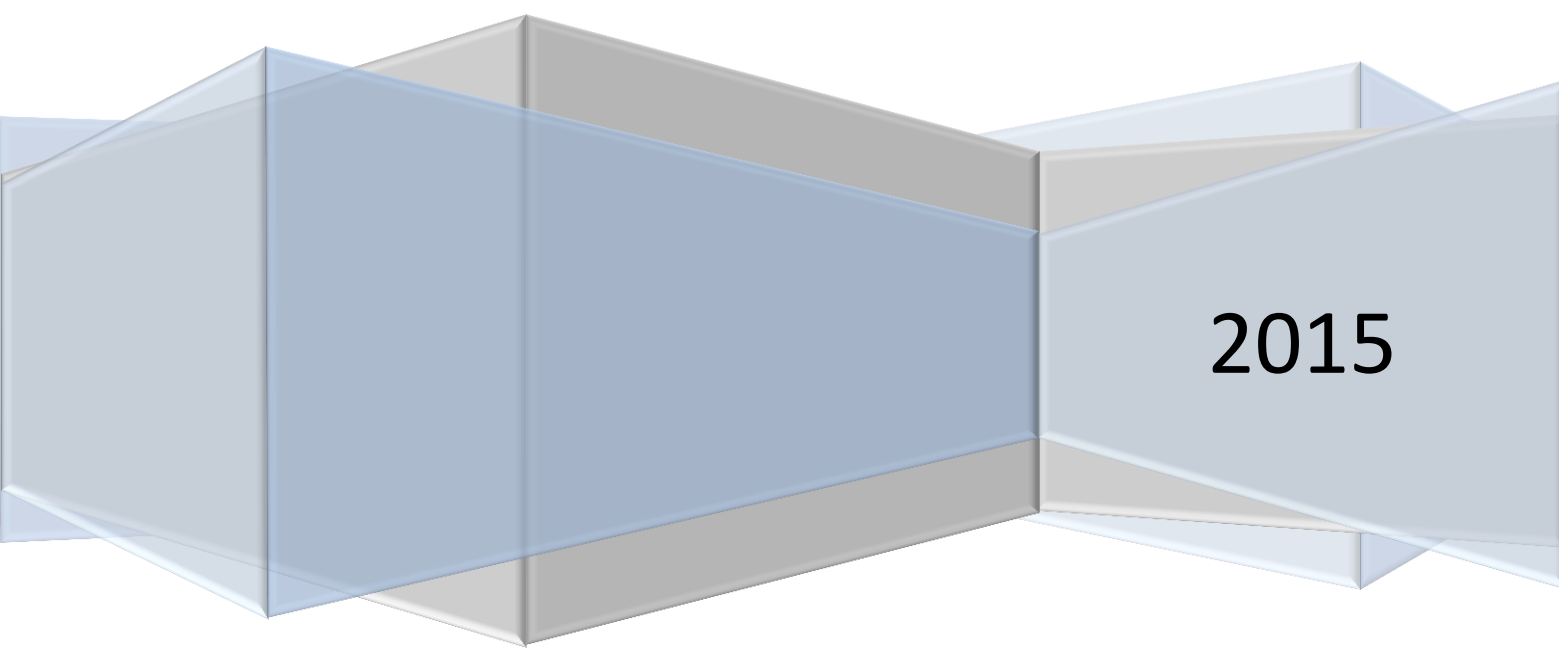
**Huck Mélanie**

**Moret Jérôme**

**Nolan James**

**Santamaria Miguel**

**Villa David**



2015

## Table des matières

Fonctionnement général .....	2
Mini-jeux.....	2
Pong.....	2
Challenger.....	2
LetterHero .....	2
Pac-Man.....	<b>Erreur ! Signet non défini.</b>
Snake .....	3
Responsabilités entre le serveur et le client – protocole d’échange .....	4
Cas d’utilisation .....	5
Ebauche du modèle de domaine.....	6
Ebauche des interfaces utilisateur .....	8
Participants et rôles.....	9
Plan d’itérations .....	9
Itération 1 (déjà accomplie) .....	9
Itération 2 (déjà accomplie) .....	10
Itération 3 .....	12
Itération 4 .....	13
Itération 5 .....	15
Itération 6 .....	15
Itération 7 .....	16

## Fonctionnement général

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) 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).

### Pong

### 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 quatre 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.

Au bout du temps déterminé par l'administrateur, le joueur avec le plus de points est déclaré vainqueur.

A savoir que chaque zone verticale possèdera son propre thread, pour des questions d'optimisation.

### Déroulement du jeu :

1. L'un des joueurs sélectionne *LetterHero* comme mini-jeu.
2. Le serveur envoie la même **séquence des lettres** qui défileront (ainsi que leur **position**) à tous les clients. Ainsi, chaque client sera traité également, et chacun aura ses chances.
3. Chaque joueur joue ensuite de son côté, durant le temps déterminé par l'administrateur.

4. A la fin du temps imparti, chaque client envoie son **score** au serveur, qui sélectionne le gagnant et envoie la réponse à tous les clients.

### Snake

Le jeu consiste à contrôler un serpent et à le guider afin que ce dernier se nourrisse et grandisse. Mais attention, toute nourriture n'est pas bonne à prendre !

#### Règle du jeu :

1. Une partie a une durée limitée et fixe pour tous les joueurs.
2. La taille ainsi que la vitesse du serpent sont influencées en fonction de ce que mange ce dernier.
3. Les bordures de la fenêtre de jeu n'engendrent pas de collision, le serpent réapparaît près de la bordure opposée.
4. Si le serpent entre en collision avec lui-même, il se dévore (quel affamé). Ce qui a pour conséquence de réduire sa taille à la portion du serpent jusqu'à la morsure (non-incluse).
5. Le serpent n'est pas considéré comme de la nourriture.
6. La nourriture apparaît à une intervalle (temps) fixe mais à des positions aléatoires et non-communes entre chaque joueur (bah oui, on n'est pas toujours juste avec les joueurs :p). Idem pour le type de nourriture.
7. La nourriture est bio et ne comporte donc pas de conservateur. Elle disparaît donc après un certain délai (fixe).

#### Condition de victoire :

Pour gagner la partie, il faut avoir le score le plus élevé parmi tous les joueurs à la fin du temps imparti au mini-jeu. Une égalité est équivalente à une défaite.

Le score du joueur correspond à la taille de son serpent à la fin de la partie.

#### Type de nourriture :

1. Sauterelle : Succulent mais pas très nourrissant. (+1 à la taille)
2. Grenouille : Cuisinée à la française ! (+2 à la taille, vitesse \*1.2)
3. Limace : Mal de ventre assuré... (-1 à la taille, vitesse / 1.2)
4. Scorpion : Mais vous êtes malade ?!? (-2 à la taille)

#### Client-serveur :

Le jeu s'exécute côté client. La seule information dont il a besoin est le nom du joueur (fournie par le serveur).

A la fin d'une partie, le client informe le serveur du résultat du joueur.

La récupération des résultats et le choix du vainqueur s'effectue côté serveur.

## Responsabilités entre le serveur et le client – protocole d'échange

### 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

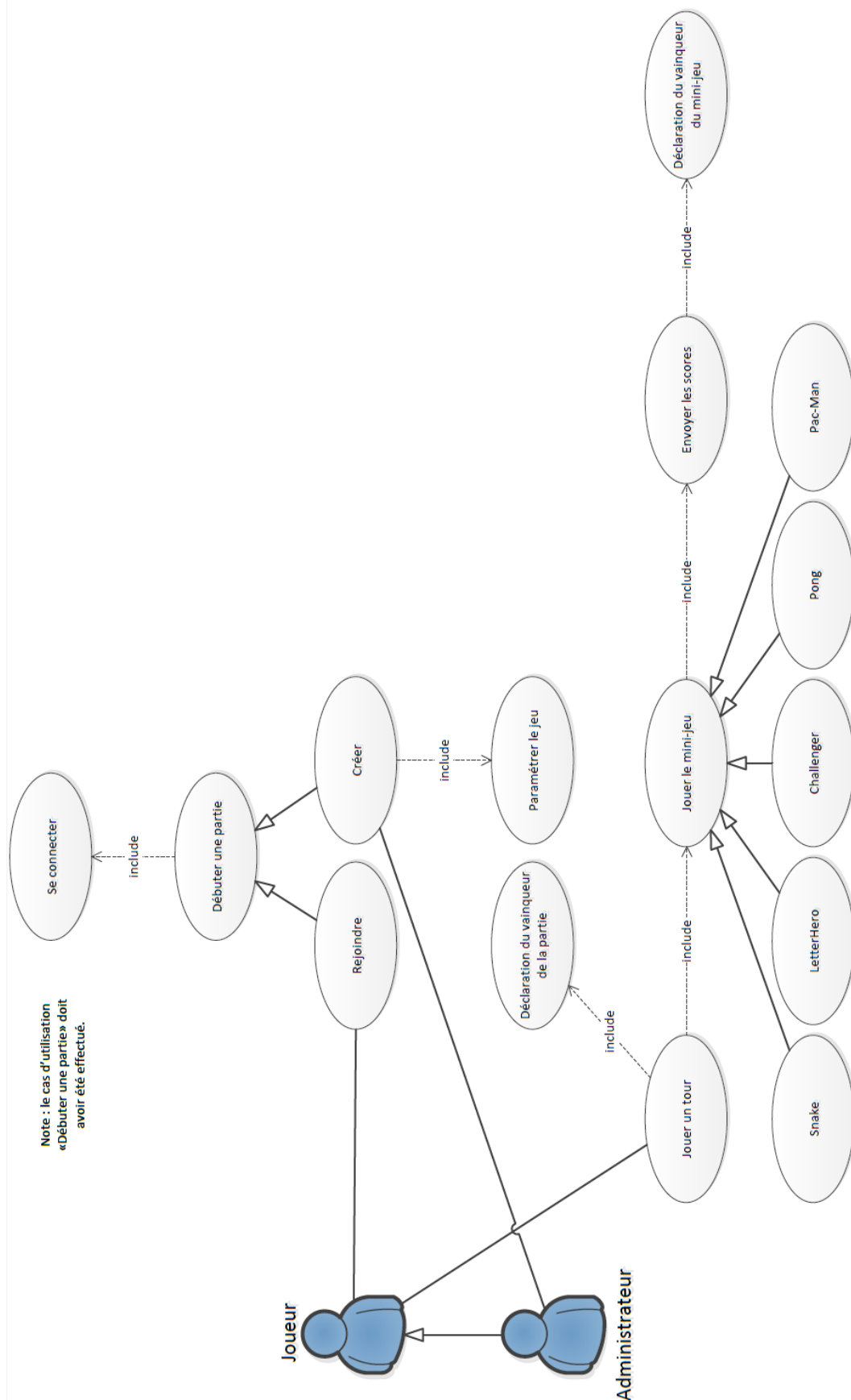
#### Déroulement d'une partie :

- 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

## Cas d'utilisation



Fourni en annexe.

## 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 créée.

### 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 1 minute, localement au jeu (**Snake, LetterHero, Challenger, Pong, Pac-Man**) choisit par un joueur.

### Snake, LetterHero, Challenger, Pong, Pac-Man

Eventuel jeu proposé.

## Envoyer les scores

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

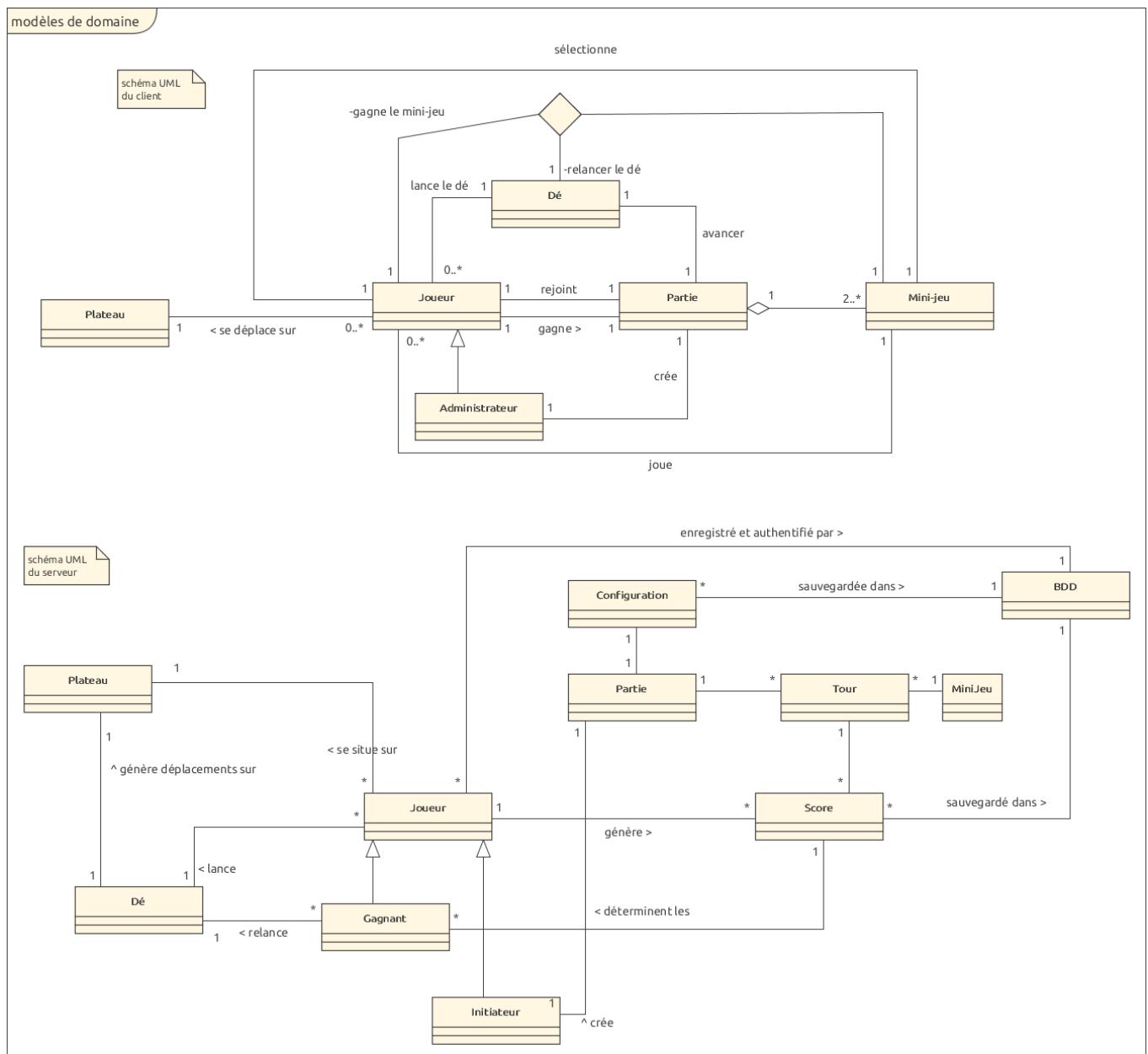
## Déclarer le vainqueur du mini-jeu

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 ».

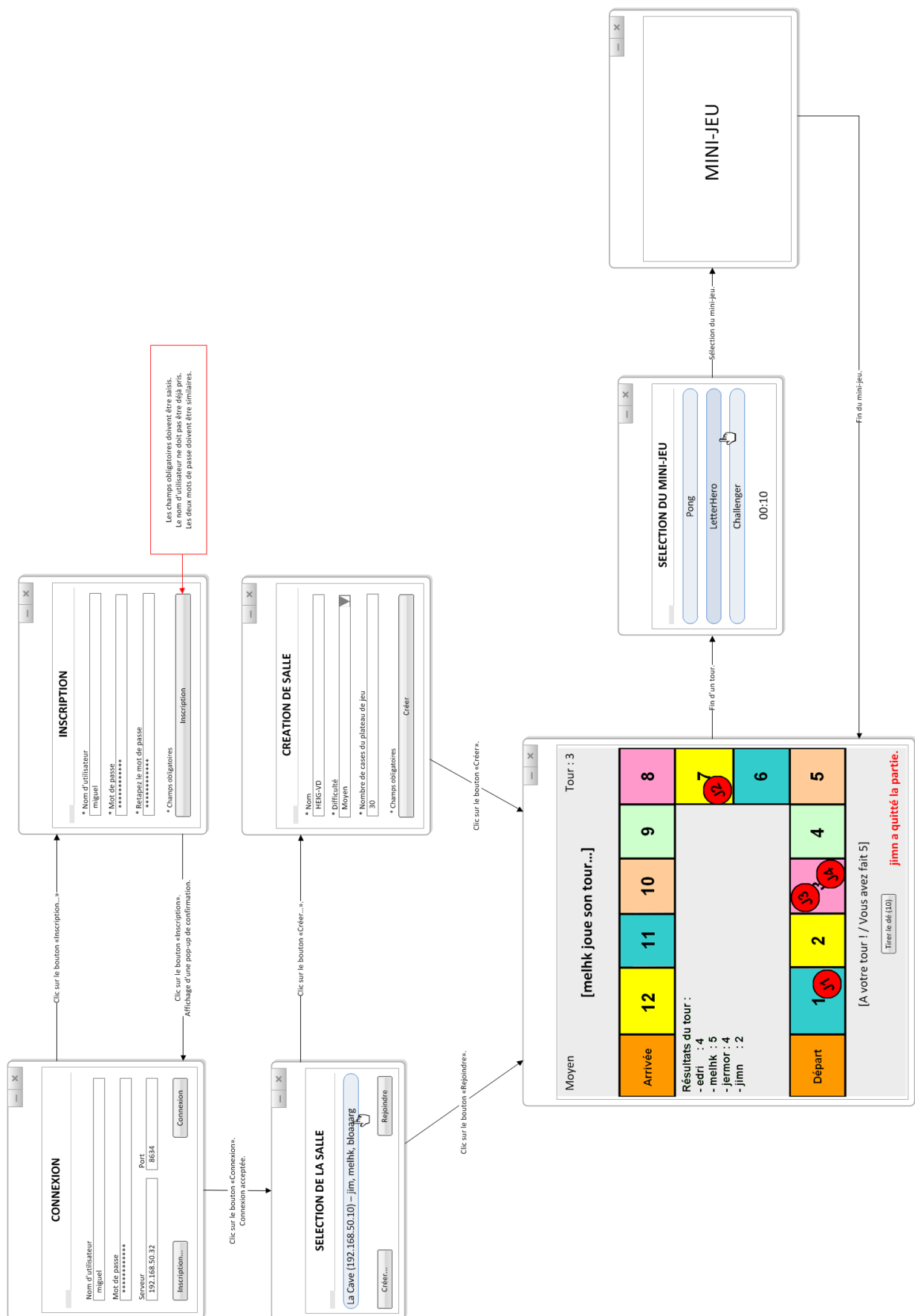
## Ebauche du modèle de domaine



Fourni en annexe.



## Ebauche des interfaces utilisateur



Fourni en annexe.

## Participants et rôles

### « 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'itérations

### 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) :
  - o Analyse plus détaillée des demandes et besoins du client.
  - o Conception du fonctionnement général de l'application.
  - o Analyse des interactions client-serveur.
  - o 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**

Du 01.04.2015 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 produire un programme mettant en œuvre les interactions client-serveur de base.

### Objectifs détaillés

En particulier, il s'agira de :

- **Conception de l'infrastructure** - Définir les classes et les méthodes nécessaires permettant de faire communiquer un serveur avec plusieurs clients.
- **Développement de fonctionnalité** – Permettre à un client de se connecter et de se déconnecter du serveur. Cette fonctionnalité sera requise dans tous les cas d'utilisation faisant interagir les joueurs et le serveur. Elle ne répondra que partiellement à ces cas d'utilisation car cette fonctionnalité est générique et ne se focalise pour l'instant pas sur les besoins métier. Il ne sera possible que d'établir une connexion persistante avec le serveur, et permettre à ce dernier d'être notifié si un utilisateur s'est déconnecté.
- **Développement de fonctionnalité** – Permettre à un joueur d'envoyer des informations au serveur. En pratique, le joueur se servira de cette fonctionnalité pour indiquer son score à la fin d'une partie mais dans cette itération, on se contentera d'implémenter l'envoi de messages quelconques. On attend à la fin de l'itération qu'un joueur puisse envoyer un message au serveur sans spécifier ce que le serveur en fera.
- **Développement de fonctionnalité** – Permettre au serveur d'envoyer des informations au serveur. En pratique, le serveur se servira de cette fonctionnalité pour demander à un joueur de choisir un jeu mais dans cette itération, on se contentera d'implémenter l'envoi de messages quelconques. On attend à la fin de l'itération que le serveur puisse envoyer un message à un client donné sans spécifier ce que ce dernier en fera.
- **Développement de fonctionnalité** – Permettre au serveur de lister tous les joueurs connecté à un instant donné. Cette fonctionnalité permettra au serveur d'envoyer des messages broadcastés à tous les joueurs, typiquement pour leur signaler le début d'un mini-jeu. A cette itération, on ne souhaite que donner la possibilité au serveur d'obtenir une liste de ces joueurs.

### 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

- **Mélanie** : conception de l'infrastructure UML + tests.
- **James & Miguel** : développement des bases du serveur.
- **Jérôme** : développement des communications client->serveur + check de la configuration.
- **David** : responsable de l'analyse => contrôle que les spécifications soient suivies + tests.

### 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

## **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 l'infrastructure** – créer les tables et leurs colonnes dans la base de données selon le diagramme UML produit lors de la phase de conception.
- **Développement de fonctionnalité** – 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** – 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** – 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** – 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** – 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 un, 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** – 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** – 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.
- **Développement de fonctionnalité** - Développement du modèle du plateau de jeu :
  - Récupération des paramètres de jeu.
    - Nombre de case
    - Difficulté
    - ...
  - Récupération du nombre de joueurs.
  - Implémentation de la structure de données représentant la grille de jeu.
    - 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.

### **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**

-

## **Itération 5**

### **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** : Développement de la vue du plateau de jeu :
  - Création de l'interface graphique.
    - Algorithme pour le dessin de la grille de jeu.
      - Doit d'adapter aux nombres de cases. (Dans la limite du minimum et maximum).
    - Gestion des superpositions des joueurs (2 joueurs sur la même case).
    - Fenêtre de sélection du mini-jeu
    - Etc...

-

### **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**

Tous les membres seront assignés au développement et aux tests.

### **Effort escompté**

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

### **Bilan de l'itération**

-

## **Itération 6**

### **Objectif général**

Implémentation et test des mini-jeux, et intégration au plateau de jeu.

### **Objectifs détaillés**

- **Développement des fonctionnalités** : 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 27.05.2015 au 03.06.2015.

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

Tous les membres seront assignés au développement et aux tests des mini-jeux.

**Effort escompté**

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

**Bilan de l'itération**

-

**Itération 7****Objectif général**

Finalisation.

**Objectifs détaillés**

- **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 03.06.2015 au 10.06.2015.

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

- Miguel : validation + vérification.
- Mélanie : contrôle de l'architecture et de la conception en générale.
- Jérôme : contrôle de la configuration de l'application.
- David & James : tests.

**Effort escompté**

- Miguel : 6 périodes.
- Mélanie : 10 périodes.
- Jérôme : 5 périodes.
- David & James : 12 périodes chacun.

Total : 45 périodes.

**Bilan de l'itération**

-