

## SRD : Tower Defense

### *Meat war : Rise of the fallen vegan*

[Cours : INFO-F-209]

---

JACOBS Alexandre & INNOCENT Antoine & ANDRÉ Bob & MESTREZ  
Benjamin & VANGEEM Nicolas & ROOS Kim & PAQUET Michael & SINGH  
Sundeeep  
BA2 Informatique

---

Décembre 2016

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	But du projet . . . . .	2
1.2	Cadre . . . . .	2
1.3	Historique de document . . . . .	2
<b>2</b>	<b>Besoins de l'utilisateur</b>	<b>3</b>
2.1	Exigences fonctionnelles . . . . .	3
2.1.1	Use case : menu . . . . .	3
2.1.2	Use case diagramme : . . . . .	4
2.1.3	Diagramme d'activité : . . . . .	5
2.2	Use case : en jeu . . . . .	6
2.2.1	Use case diagramme : . . . . .	6
2.2.2	Diagramme d'activité : . . . . .	7
2.3	Exigences non fonctionnelles . . . . .	7
2.4	Exigence de domaine . . . . .	7
<b>3</b>	<b>Besoin du système</b>	<b>8</b>
3.1	Exigences fonctionnelles . . . . .	8
3.1.1	Use case : Connexion au jeu . . . . .	8
3.1.2	Use case diagramme : . . . . .	8
3.1.3	Use case : lancer une partie . . . . .	8
3.1.4	Use case diagramme : . . . . .	9
3.1.5	Use case diagramme : . . . . .	10
3.1.6	Use case : User dans le salon de discussion . . . . .	11
3.1.7	Use case diagramme : . . . . .	11
3.2	Exigences non fonctionnelles . . . . .	11
3.3	Design et fonctionnement du système . . . . .	12

# 1 Introduction

## 1.1 But du projet

Ce document joue le rôle du document de spécification des besoins ou SRD (System requirement Document) pour le projet INFO-F209 : "Meat war : Rise of the falling vegan". Ces exigences portent sur un travail de groupe (8 personnes) codé majoritairement en C++ avec une section C pour le serveur. Le produit final est donc un jeu de type Tower Defense multijoueur en réseau. Ce jeu n'ayant aucun but lucratif, nous considérons le client principal comme étant le joueur du jeu en question. Ainsi, ce document présente des spécifications propres à l'utilisateur et propre au système. Celui-ci va également mettre en avant la communication client/serveur afin de bien mettre en place l'aspect multijoueur réseau du produit.

## 1.2 Cadre

Ce document ne varie pas en fonction des différentes architectures ou systèmes d'exploitation.

## 1.3 Historique de document

Versions	Auteur	Date	Modifications
0.0	Alexandre & Antoine	10-12-2016	Création de la structure du document
0.1	Sundeep & Michael	10-12-2016	Ajout des use cases et diagrammes d'activité au niveau utilisateur.
0.2	Équipe	16-12-2016	Ajout des besoins d'utilisateur et besoins du système
0.3	Kim	16-12-2016	Ajout du glossaire et de l'index
1.0	Équipe	19-12-2016	<b>Deadline pour la phase 1</b>

## 2 Besoins de l'utilisateur

Les besoins de l'utilisateur seront expliqués au travers de plusieurs graphes/explications, en voici ici un résumé.

Le but de l'application est de permettre à l'utilisateur de pouvoir jouer à un jeu de type Tower Defense. Celui-ci sera disponible depuis un menu accessible après connexion et offrira au client plusieurs services. En effet, l'utilisateur devra se créer un compte avant d'avoir accès au menu.

Il pourra alors consulter son profil ainsi que celui de ses amis, et modifier le sien à souhait (pseudo, avatar, ...). Un classement sera également accessible et permettra de situer un joueur par rapport à un autre en fonction de leur score respectif.

Comme dit implicitement plus haut, l'utilisateur disposera d'une liste d'amis avec qui il lui sera possible de discuter via une fenêtre de conversation.

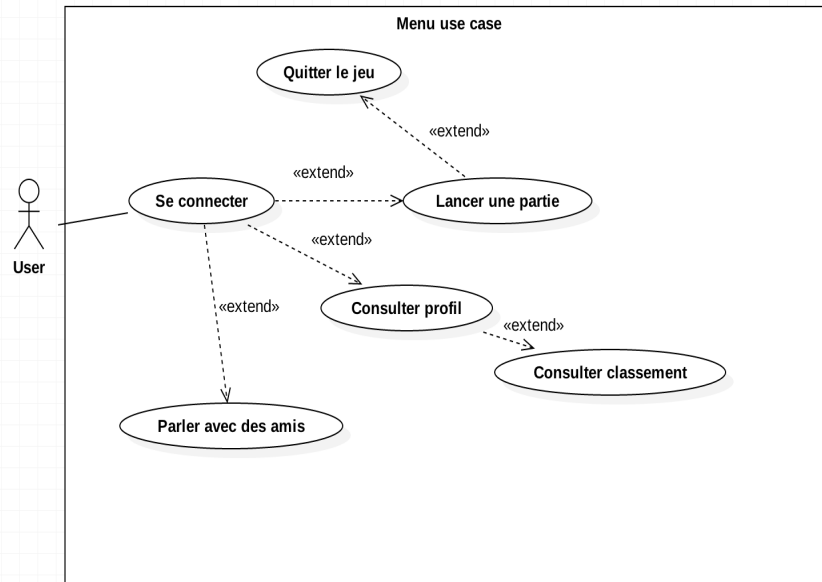
Le menu permettra bien évidemment enfin de créer une partie afin de jouer au jeu. Niveau de difficulté, mode, et nombre de joueurs (au maximum de 4) seront spécifiés par le client avant que celui-ci ne lance la partie. Une seule et unique carte sera cependant disponible.

### 2.1 Exigences fonctionnelles

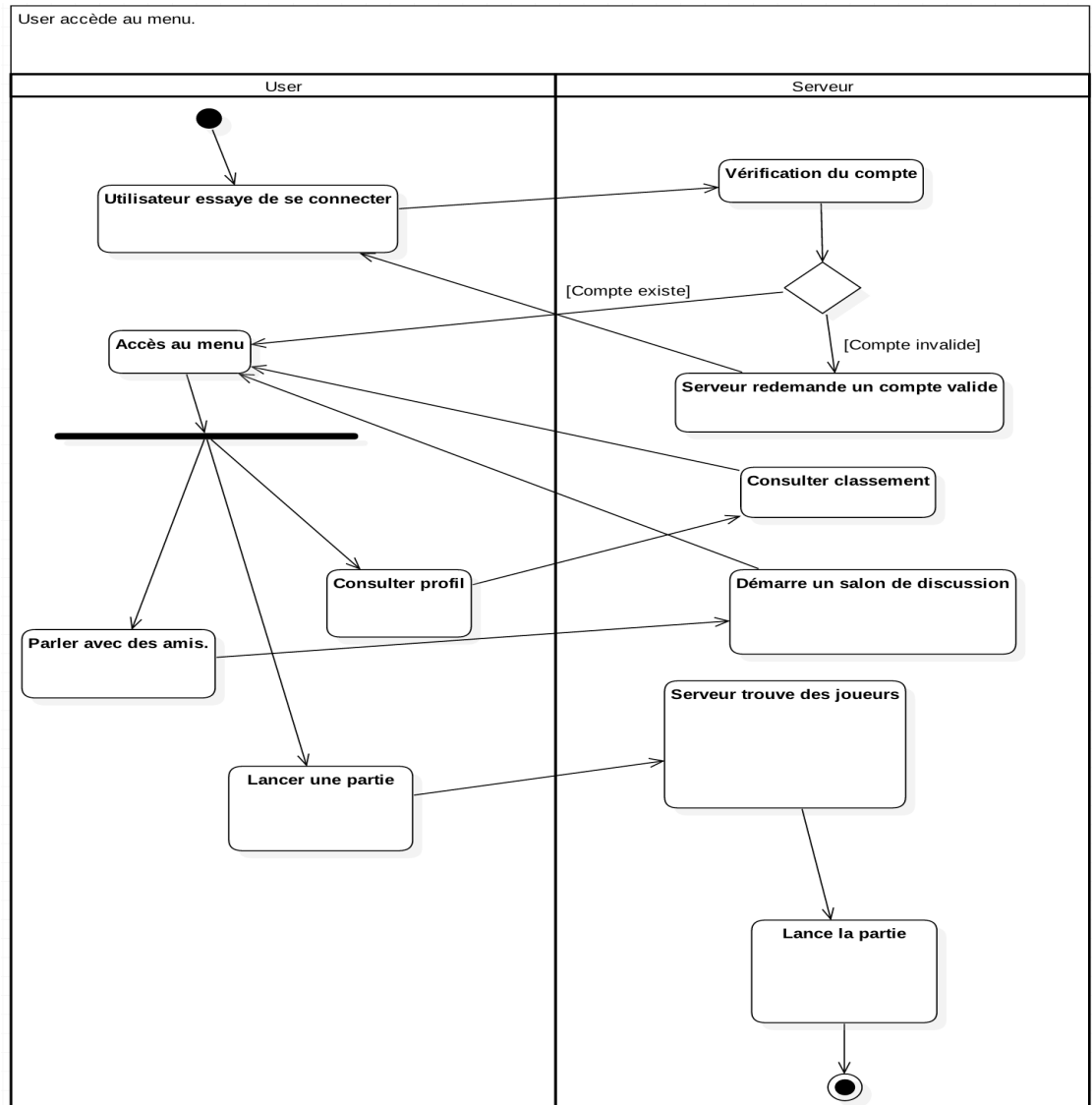
#### 2.1.1 Use case : menu

- **Acteurs** : pour ce premier use case, plusieurs acteurs entrent en jeu. Nous avons un acteur principal et un secondaire, qui sont respectivement le joueur et le serveur.
- **Préconditions** : pour ce qui est des préconditions il y en a qu'une seule, à savoir que le joueur se soit connecté.
- **Postconditions** : il y'a deux postconditions. Soit le menu est quitté et donc le joueur est déconnecté, soit le joueur a lancé une partie.
- **Flux d'exécution** : en ce qui concerne le flux d'exécution basique, le joueur va d'abord se connecter. Une fois connecté, il peut lancer une partie, consulter son profil et celui de ses amis. Finalement, avant de lancer une partie, il peut choisir le niveau de difficulté mais aussi le mode de jeu.

### 2.1.2 Use case diagramme :

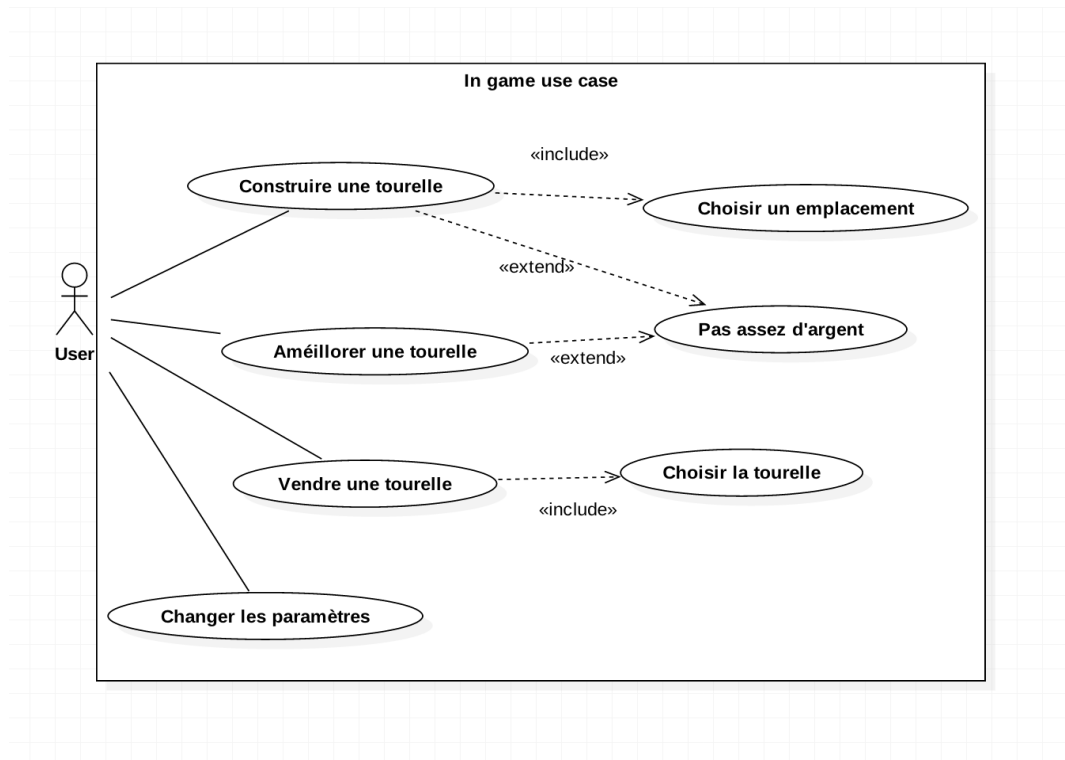


### 2.1.3 Diagramme d'activité :



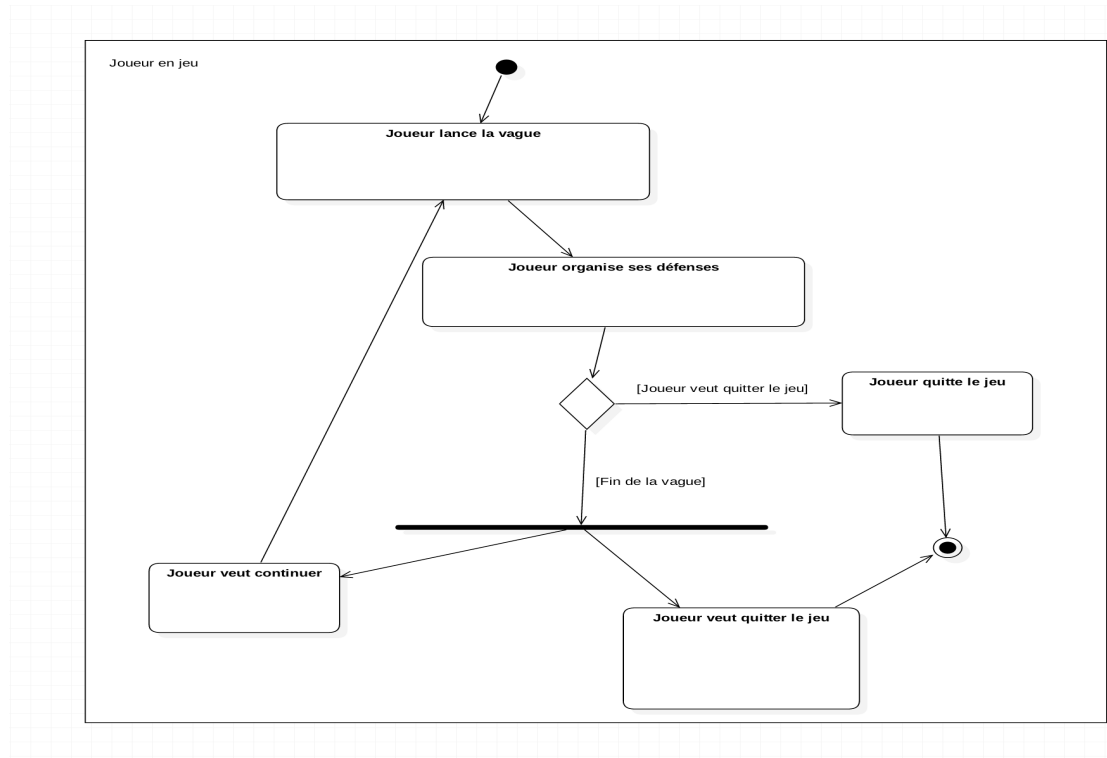
## 2.2 Use case : en jeu

### 2.2.1 Use case diagramme :



- **Acteurs** : ce deuxième use case a comme unique et principal acteur le joueur.
- **Préconditions** : pour ce qui est des préconditions le joueur doit être connecté et a dû lancer une partie via le serveur.
- **Postconditions** : la seule postcondition est qu'une fois la partie terminée, le joueur retourne au menu.
- **Flux d'exécution** : pour ce qui est du déroulement basique, une fois en jeu le joueur peut construire des tourelles, consulter la description de celles-ci, il peut également les améliorer, les vendre, mais aussi modifier les paramètres du jeu. Il existe également un flux d'exécution alternatif. Au lieu de jouer, l'utilisateur peut être tout simplement un spectateur de la partie.

### 2.2.2 Diagramme d'activité :



### 2.3 Exigences non fonctionnelles

- Menu et interface intuitifs.

### 2.4 Exigence de domaine

- Le jeu doit être multijoueur, les différents utilisateurs connectés sur un même serveur doivent pouvoir agir sur le terrain.
- L'abandon d'un des joueurs ne doit pas empêcher les autres de terminer leur partie.
- Un jeu de Tower Defense est un jeu de 2 à 4 joueurs qui ont chacun une partie de la carte.
- Chacun des joueurs a sa base dans chaque coin du jeu.
- Les ennemis apparaissent au milieu de la carte, leur vagues sont identiques pour chaque joueur.
- La partie se déroule sous forme de vagues successives d'ennemis jusqu'à ce que la partie se termine.
- Les joueurs peuvent améliorer ou acheter des tours à tout moment de la partie.
- La partie se termine si le temps est écoulé dans le cas d'une partie en mode chrono ou s'il ne reste qu'un joueur vivant dans le cas d'une partie classique.



## 3 Besoin du système

### 3.1 Exigences fonctionnelles

#### 3.1.1 Use case : Connexion au jeu

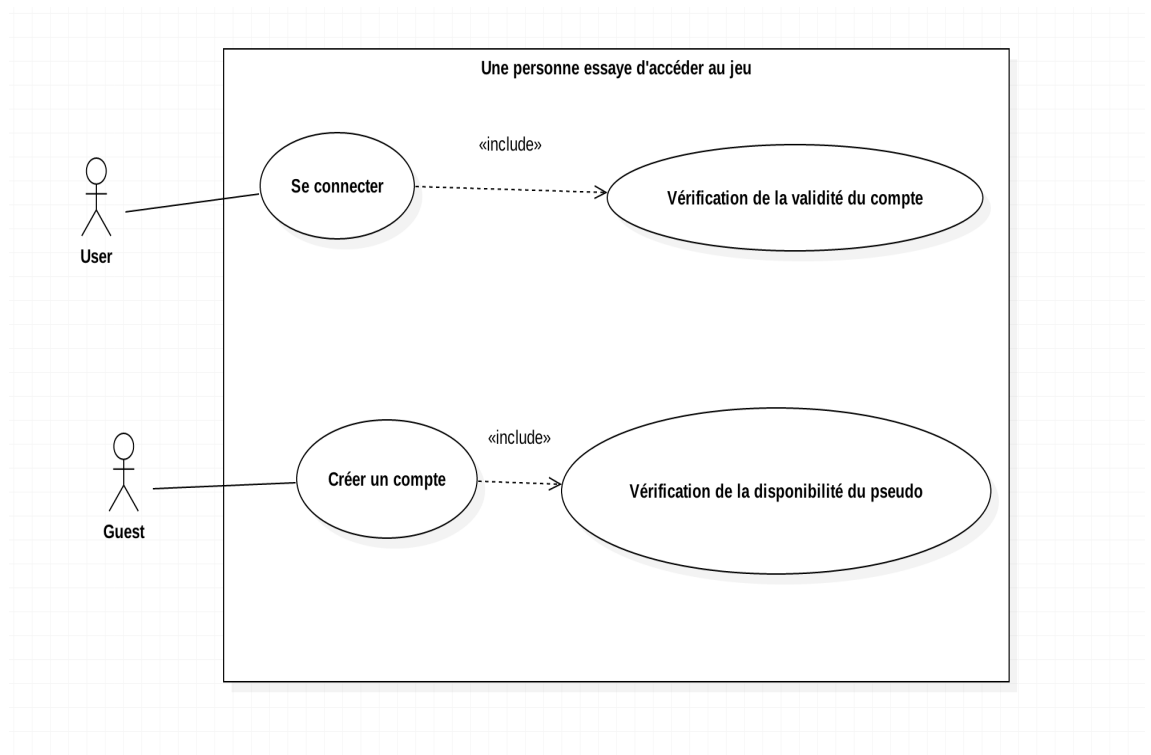
**cas général :** L'utilisateur doit pouvoir se connecter en utilisant son nom d'utilisateur et son mot de passe.

**Pré condition :** L'utilisateur n'est pas encore connecté.

**Post condition :** L'utilisateur s'est connecté.

**Cas exceptionnel :** L'authentification échouera et l'utilisateur sera invité à recommencer si son mot de passe ou son num d'utilisateur est mauvais.

#### 3.1.2 Use case diagramme :



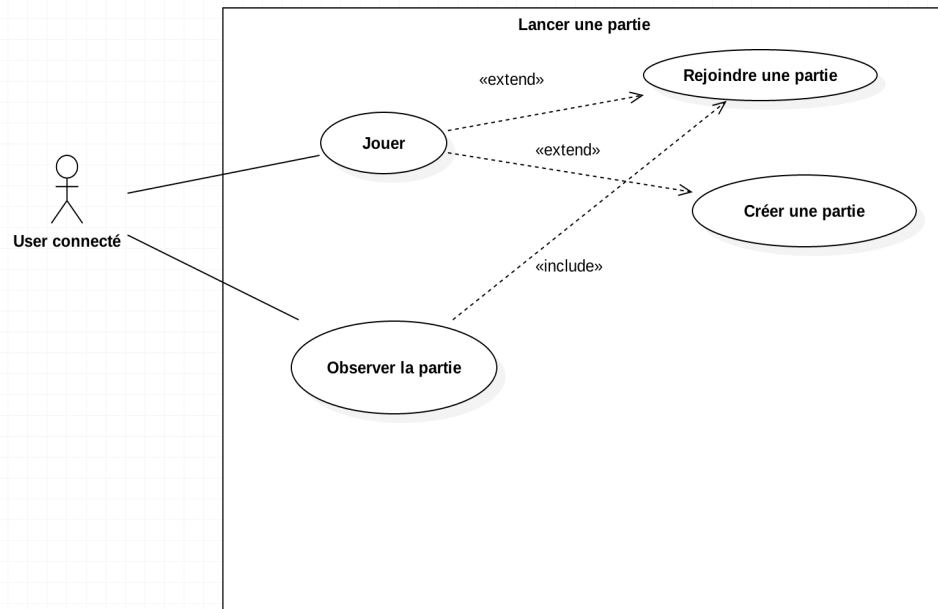
#### 3.1.3 Use case : lancer une partie

— **Acteurs :** pour ce use case, le seul acteur est l'utilisateur ayant un compte.

— **Préconditions :** en ce qui concerne les préconditions, il y en a qu'une seule à savoir que le système ai bien trouvé le compte de l'utilisateur et l'ai fait accéder au jeu.

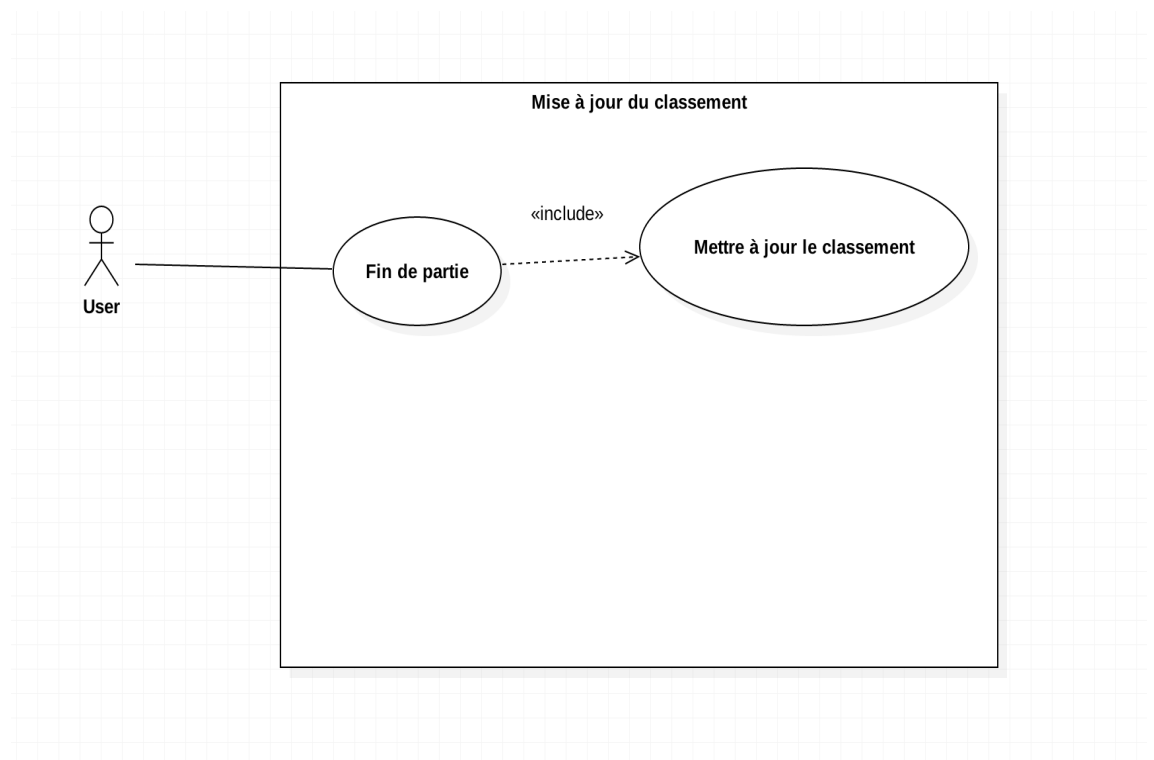
- **Postconditions** : pour ce qui est des postconditions, l'utilisateur doit avoir rejoint un salon de jeu ou avoir créé un salon ou encore rejoint une partie en tant que spectateur.
- **Flux d'exécution** : pour ce qui est du flux d'exécution basique, Le joueur va soit rejoindre une partie grâce au système qui lui trouve un salon libre, soit créer une partie. Il peut finalement rejoindre une partie en tant que spectateur grâce au système qui lui trouve une partie déjà lancée.

#### 3.1.4 Use case diagramme :



- **Acteurs** : pour ce use case, il n'y a pas vraiment d'acteurs qui entrent en jeu. Le système se chargera de tout.
- **Préconditions** : pour ce qui est des préconditions, la seule et unique précondition est que l'utilisateur était dans une partie en tant que joueur et non en tant que support.
- **Postconditions** : pour les postconditions, il faut que la partie se soit bien totalement terminée.
- **Flux d'exécution** : en ce qui concerne du flux d'exécution basique, Le joueur va jouer sa partie la partie se termine et le système va mettre à jour le classement si besoin.

### 3.1.5 Use case diagramme :



### 3.1.6 Use case : User dans le salon de discussion

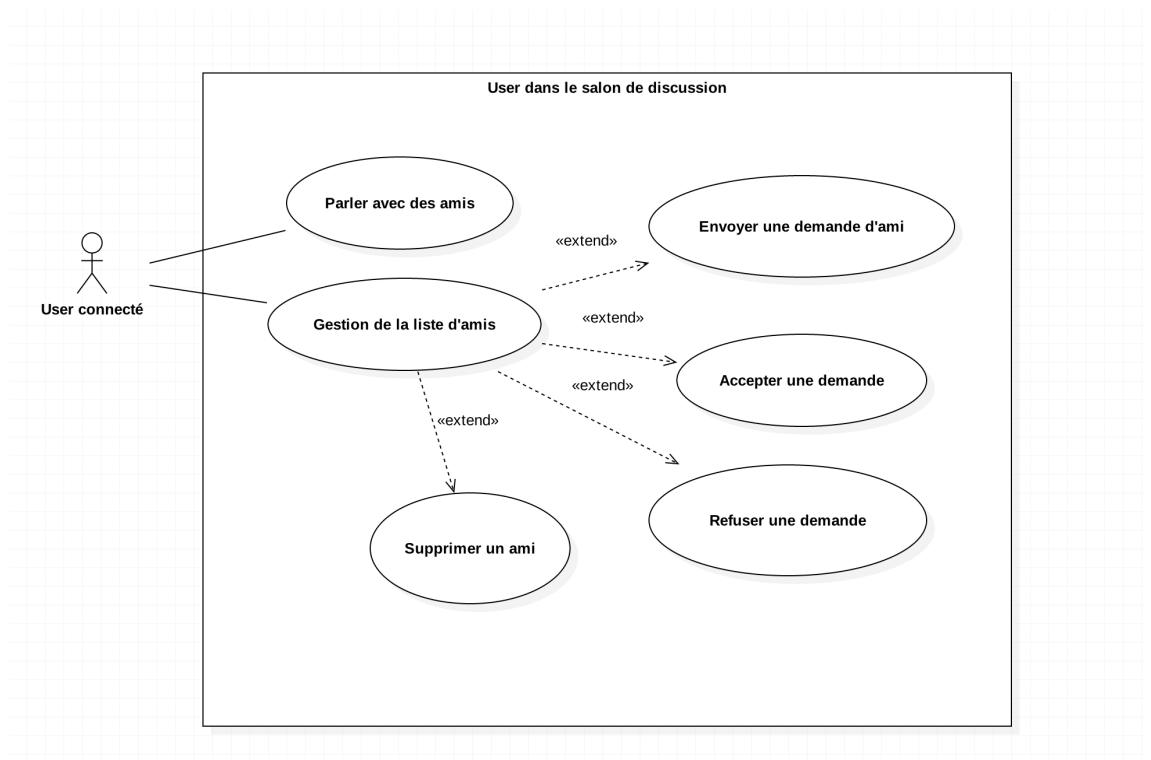
Ce dernier use case ne comporte qu'un seul acteur, il s'agit de l'utilisateur.

Pour ce qui est des préconditions, l'utilisateur a dû être capable de se connecter et d'avoir lancé le salon de discussion. Ensuite, pour ce qui est des postconditions, aucune n'entre en jeu.

Le flux d'exécution basique dans ce dernier use se déroule de la manière suivante : Le joueur/utilisateur va soit vouloir parler avec un ami, au quel cas le système va ouvrir un salon de discussion entre le joueur et son ami. Par ailleurs, il peut également vouloir organiser sa liste d'amis, ce qui mène à plusieurs cas dont le système se chargera. Ces cas sont :

- Le joueur reçoit une demande d'amis et l'accepte ou la refuse.
- Le joueur envoie une demande d'amis.
- Le joueur supprime un ami.

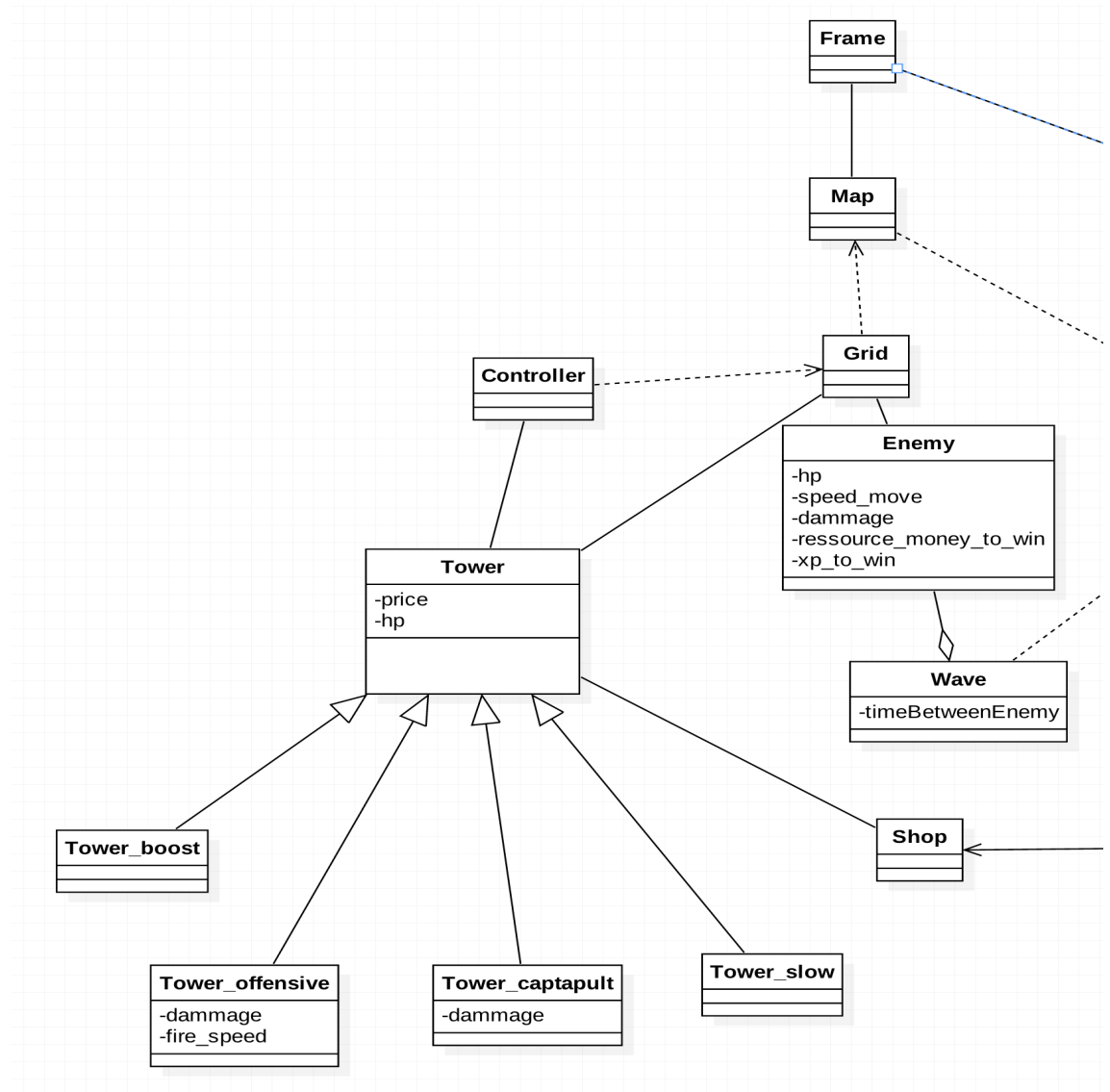
### 3.1.7 Use case diagramme :

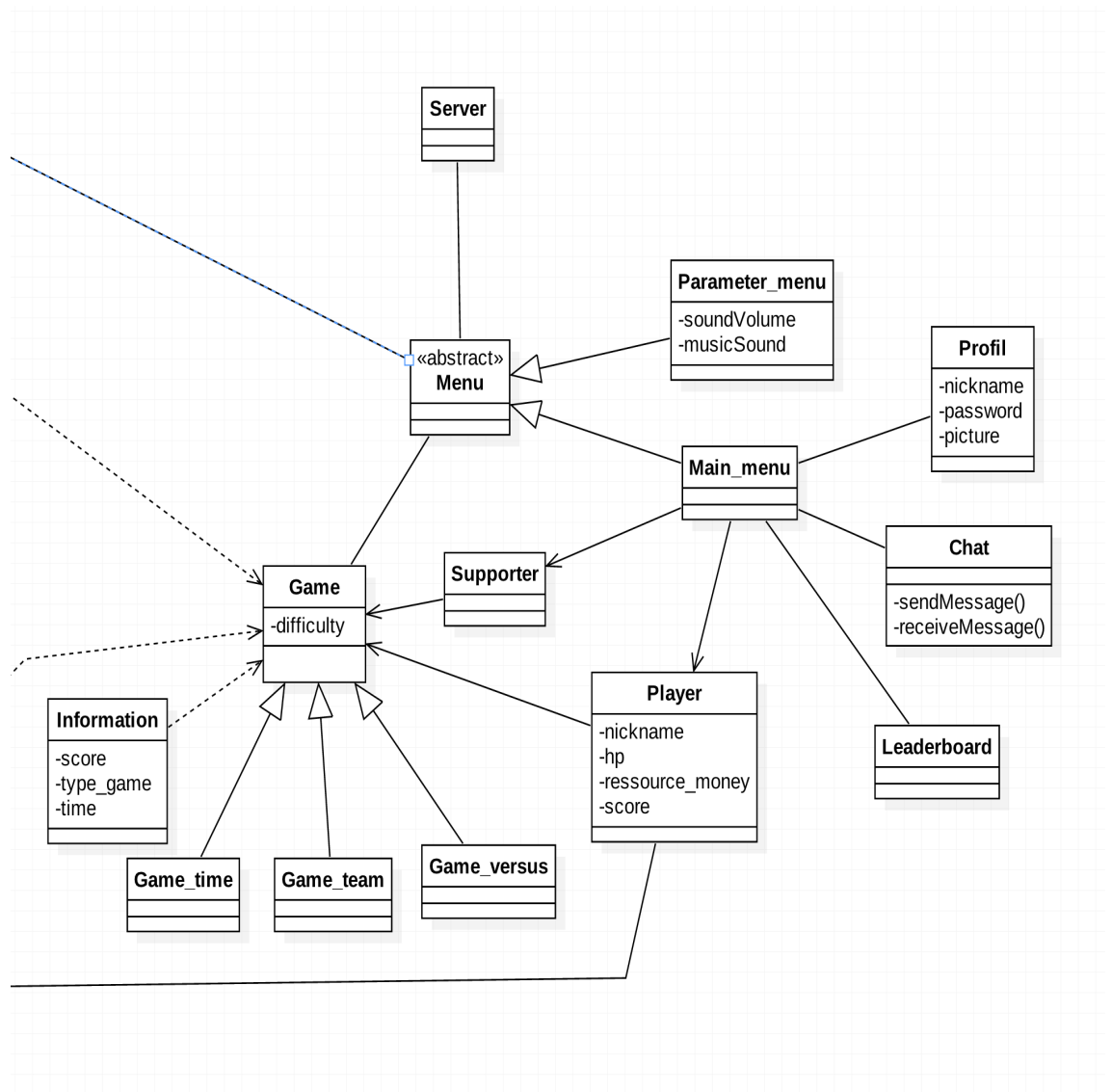


## 3.2 Exigences non fonctionnelles

- Portabilité.

### 3.3 Design et fonctionnement du système





## Index

abandon, 7

base, 7

classement, 3, 10

consulter son profil, 3

ennemis, 7

liste d'amis, 3, 11

menu, 3, 6, 7, 13

mode chrono, 7

partie, 3, 6–10

partie classique, 7