# PictioLAN

## Rapport final

Date

Avril – Juin 2014

Cours

GEN − Projet

Groupe

Magali Fröhlich

Calixte Melly

Simone Righitto

Frédéric Saam

Table des matières

[PictioLAN 1](#_Toc390768653)

[Rapport final 1](#_Toc390768654)

[1 Introduction 3](#_Toc390768655)

[1.1 Objectif du logiciel 3](#_Toc390768656)

[1.2 Fonctionnement générale de l’application 3](#_Toc390768657)

[1.3 Communication client-serveur 4](#_Toc390768658)

[1.4 Cas d’utilisation 4](#_Toc390768659)

[1.5 Scénarios principales de succès 5](#_Toc390768660)

[1.6 Protocole de communication entre le client-serveur 7](#_Toc390768661)

[1.7 Modèle de domaine 8](#_Toc390768662)

[1.8 Base de donnée et conception du dictionnaire 10](#_Toc390768663)

[1.9 Base de donnée pour les scores 11](#_Toc390768664)

[2 Conception du projet 11](#_Toc390768665)

[2.1 Base de données 11](#_Toc390768666)

[3 Implémentation du projet 12](#_Toc390768667)

[3.1 Technologies utilisées 12](#_Toc390768668)

[3.2 Problèmes rencontrés et solutions 12](#_Toc390768669)

[4 Gestion du projet 12](#_Toc390768670)

[4.1 Rôle principale/Groupe de développement 12](#_Toc390768671)

[4.2 Plan d’itérations initial 13](#_Toc390768672)

[4.3 Stratégie de tests 16](#_Toc390768673)

[4.4 Stratégie d’intégration du code de chaque participant 16](#_Toc390768674)

[5 Etat des lieux 16](#_Toc390768675)

[5.1 Ce qui fonctionne (résultats des tests) 16](#_Toc390768676)

[5.2 Ce qu’il resterait à développer (en proposant une planification) 16](#_Toc390768677)

[6 Autocritique 16](#_Toc390768678)

[7 Conclusion 16](#_Toc390768679)

[8 Annexe 16](#_Toc390768680)

[8.1 Manuel d’installation 16](#_Toc390768681)

[8.2 Manuel d’utilisation 16](#_Toc390768682)

# Introduction

## Objectif du logiciel

Notre but, au travers de ce logiciel, est d’offrir aux utilisateurs un mini-jeu qui reprend les concepts du jeu « Le Pictionnary ». Nous voulions montrer qu’il est possible de jouer à un jeu classique avec une pointe de modernité.

# Analyse

## Fonctionnement générale de l’application

Le but de l’application développée dans le cadre du projet GEN est de recréer le jeu Pictionary. Cette application sera élaborée en vue d’une utilisation sur un réseau local car c’est un jeu par définition multi-joueurs.

Pour rappel, le jeu traditionnel se joue par équipe. A chaque tour, un joueur tente de faire deviner un mot à l’aide d’un dessin si son équipe trouve le mot alors elle gagne un point. Le temps pour effectuer le dessin est limité. De plus, le joueur n’a pas le droit d’écrire le mot à faire deviner. Après un certain nombre de tour, la partie est finie et on compte les points pour désigner la meilleure équipe.

Notre objectif de base est de fournir une application qui doit pouvoir :

* Gérer des parties sur un serveur. Cela comprend une gestion des équipes et donc des joueurs connectés.
* Offrir une interface graphique pour dessiner un mot tiré au hasard.
* Minuter le temps pour le dessin.
* Afficher un chat simple pour afficher les propositions de mots.
* Stocker un dictionnaire qui sera utilisé pour sélectionner un mot.
* Gérer les scores.

Il y aura deux modes de jeux, soit par équipe soit tous contre tous. Commençons par expliquer le premier cas.

Un joueur pourra soit créer une partie, soit en rejoindre une. Le joueur créant une partie pourra spécifier le nombre de tours pour la partie, le nombre de joueurs et les catégories qu’il veut.

Les autres joueurs pourront rejoindre une partie créée en sélectionnant une partie disponible et en cliquant sur un bouton « rejoindre » Une fois tous les joueurs dans la partie ils pourront choisir leur équipe et avertir qu’ils sont prêts à l’aide d’un bouton. Ensuite une des équipes sera tirée au sort pour savoir qui va commencer à jouer. Au sein de l’équipe qui commence à jouer, il y a un autre tirage au sort pour désigner qui sera le dessinateur. Le dessinateur recevra un message pour lui avertir de son rôle et le mot qu’il doit dessiner. Le dessinateur aura accès au dessin et non au chat tandis que les « devineurs » pourront écrire dans le chat et voir aussi les propositions de l’équipe. Une fois que le dessinateur commence à dessiner, les autres joueurs de la partie (les deux équipes) vont voir le dessin. L’équipe qui commence pourra utiliser le chat et aura 1 minute pour trouver le mot si elle n’y arrive pas l’autre équipe, qui pendant cette minute à son chat désactivé, pourra elle aussi proposer des solutions, à l’aide du chat qui se sera réactivé pour trouver le mot. La partie se termine une fois que le mot a été trouvé ou que le temps général (pour les deux équipes) qui est de 2’30 est écoulé. Ensuite les équipes vont s’échanger de rôle. La partie s’arrête lorsque le nombre de tour a été atteint.

Dans le deuxième cas, tous les joueurs verront le dessin et tous les joueurs pourront utiliser le chat. Il y aura juste un tirage au sort pour savoir qui va être dessinateur.

Les utilisateurs pourront lors du lancement de l’application soit choisir de se connecter, si ils n’ont pas de compte ils pourront en créer un, soit d’être « anonyme ». Les joueurs connectés l’avantage d’une gestion de score à l’aide d’une base de donnée, tandis que les utilisateurs anonyme n’auront pas la fonctionnalité des scores.

Côté serveur, l’administrateur pourra gérer le dictionnaire (ajouter, modifier, supprimer des mots). Il pourra aussi voir les parties en cours.

L’application sera développée en vue d’une utilisation dans un réseau LAN. Ce qui implique que l’adresse IP du serveur sera connue par les utilisateurs qui se connecteront à l’application. On pourrait imaginer un mécanisme où l’application cliente se connecte sur un serveur web par une URL et qui en suite serait redirigée vers un serveur mais cela ne correspond pas à notre environnement de développement.

Certaines règles de jeu pourraient être facilement ignorées.

## Communication client-serveur

Le rôle du serveur est de gérer les connexions des joueurs et de gérer les parties en cours. Le rôle de l’application client est de fournir une interface pour que l’utilisateur puisse dessiner et utiliser le chat. Le serveur retransmet les informations aux applications clientes.

Lancement du jeu : Les joueurs se connectent au serveur. Une fois connectées, ils ont le choix de créer une partie ou d’en rejoindre une déjà créée.

Fin du jeu : Lorsque le nombre de tours est écoulé, la partie prend fin. L’utilisateur est ramené à l’écran d’accueil qui permet de créer une partie ou d’en rejoindre une.

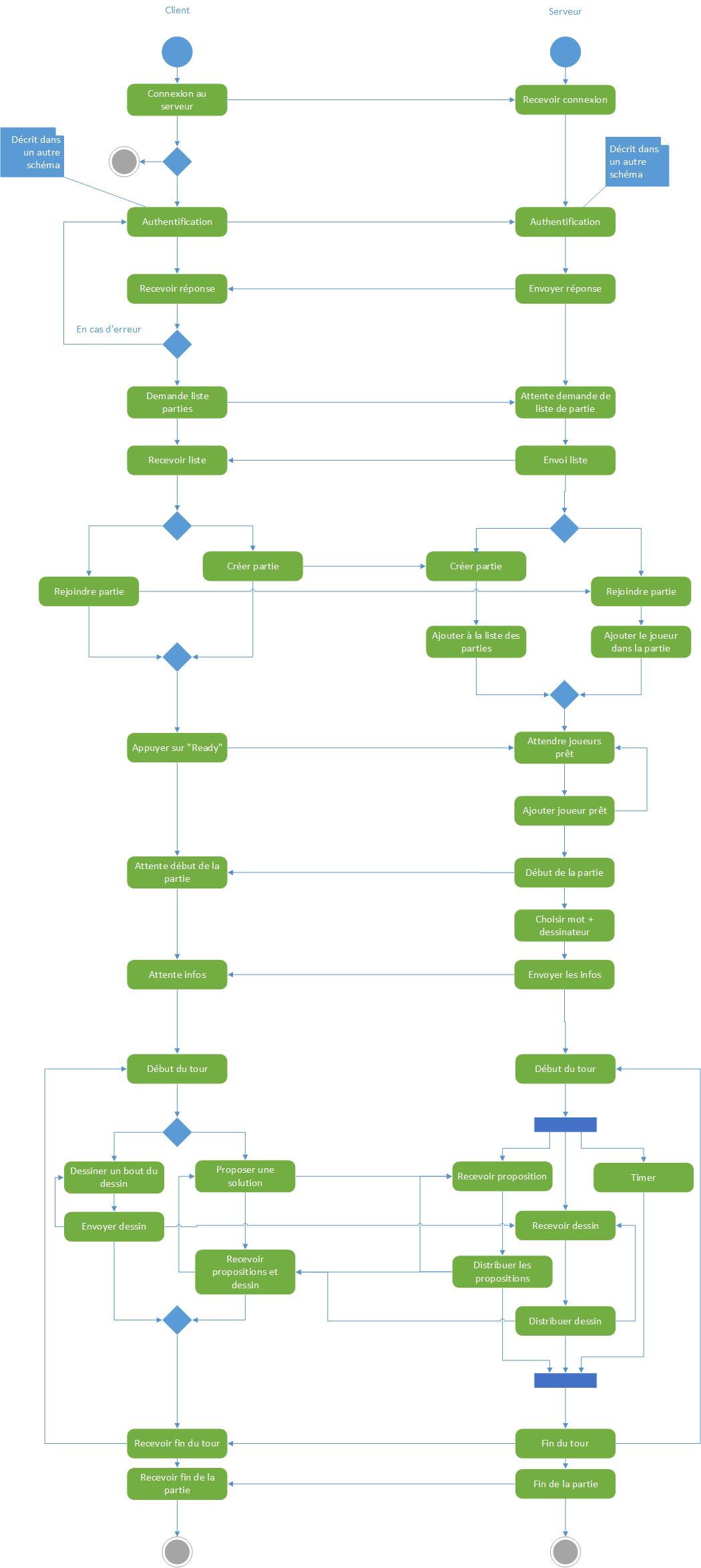
Le serveur enverra les signaux pour contrôler les joueurs (début et fin de partie). C’est également le serveur qui détecte le mot gagnant.

Le client envoie des requêtes auprès du serveur qui lance les signaux en suite pour gérer la partie.

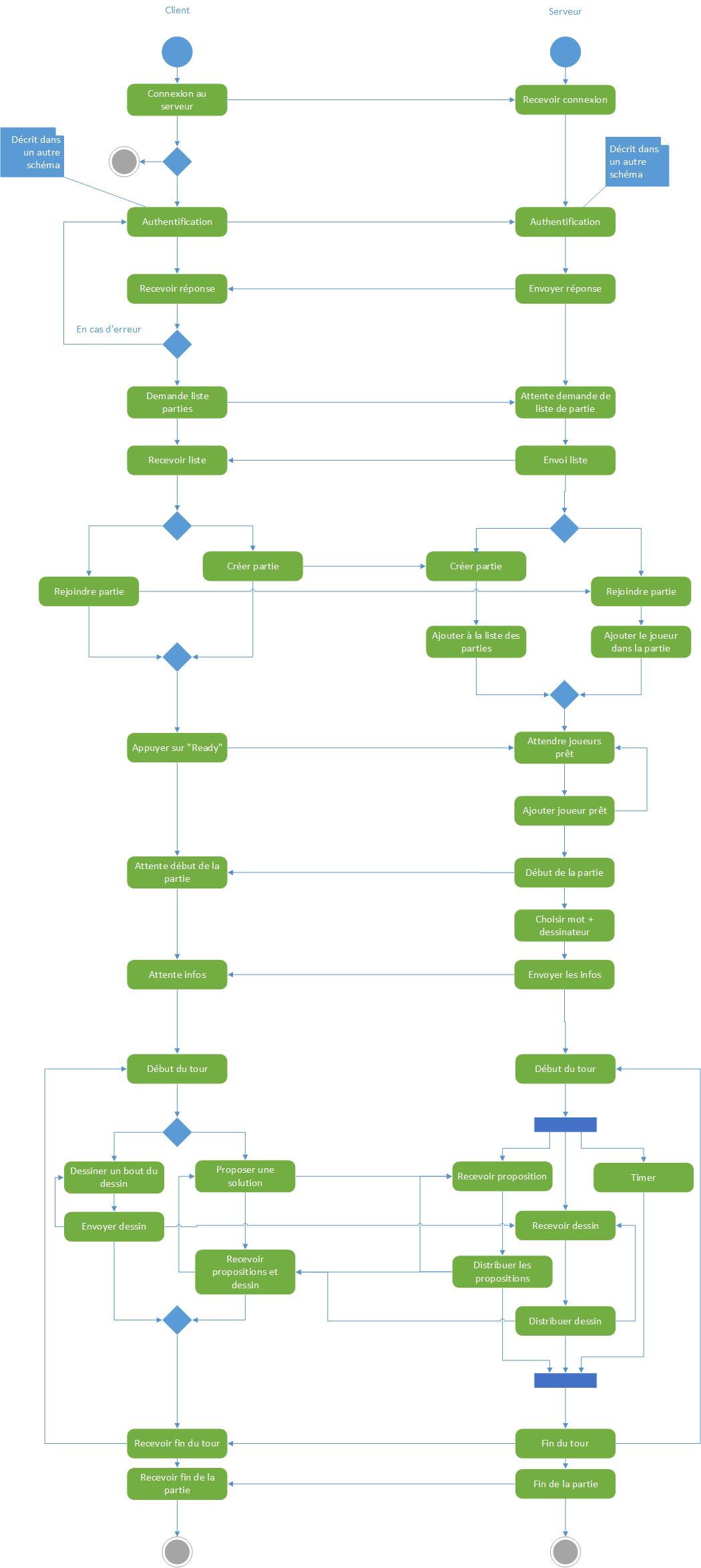
### Diagramme d’activité

Voici le fonctionnement général de l’application.

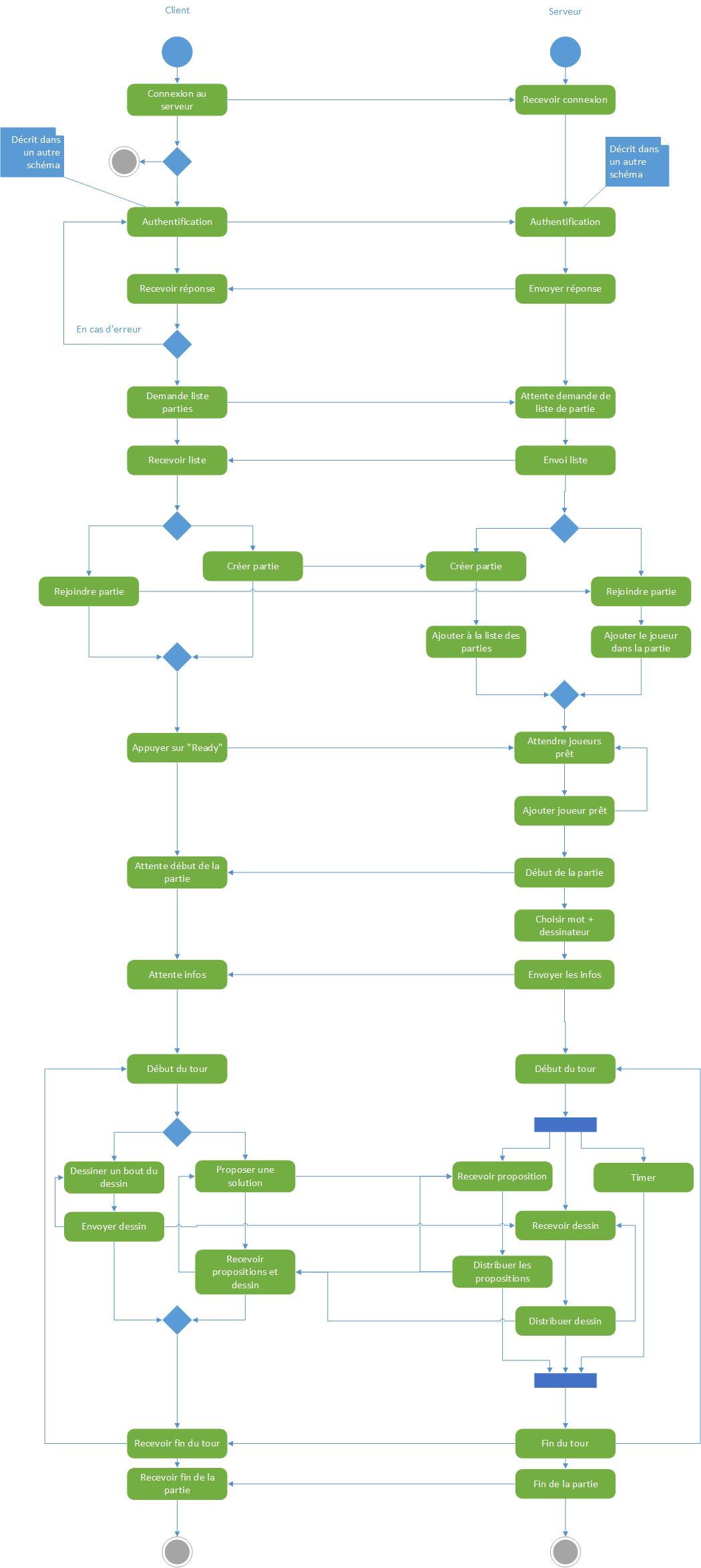
Voici le lancement de l’application et la connexion au près du serveur.



A présent, voici la création (ou l’opération de rejoindre une partie). Ainsi que le lancement de la partie.



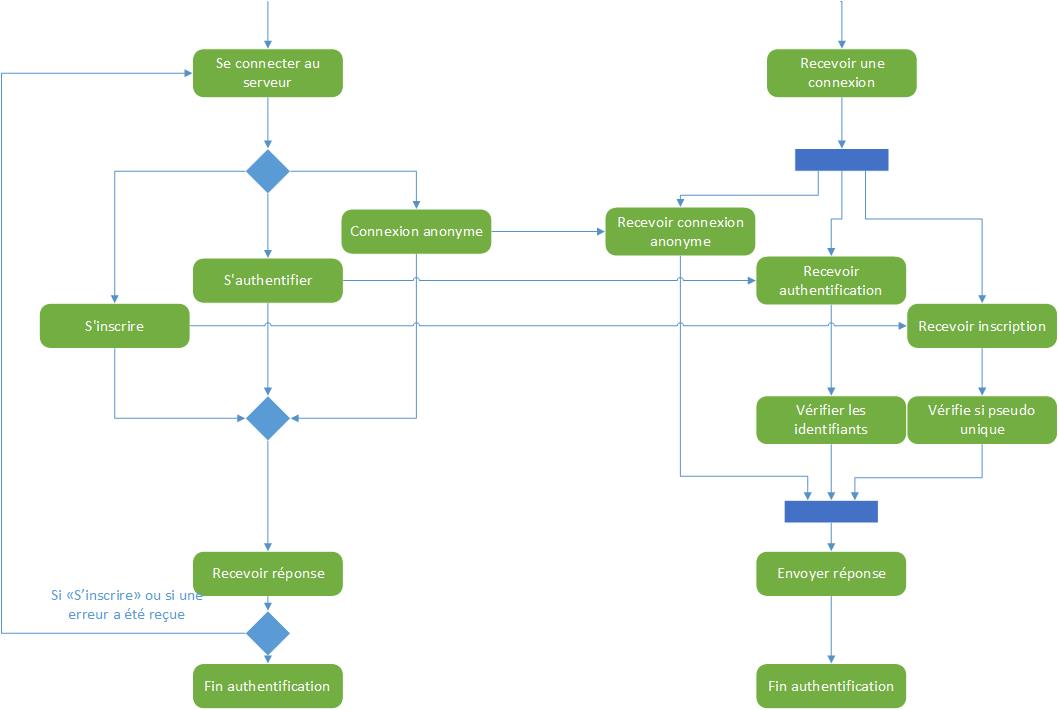
La suite du schéma décrit le déroulement des tours de la partie ainsi que la fin de la partie.



|  |  |
| --- | --- |
| Le serveur a donc le rôle suivant :   * Valider l’authentification * Valider une inscription * Lancer la partie * Sélectionner un mot * Sélectionner un dessinateur * Détecter les gagnants * Fin des tours | Le client a donc le rôle suivant :   * Faire une requête d’authentification * Faire une requête d’inscription * Créer/Rejoindre un partie * Dessiner * Utiliser le chat |

Voici le détail du fonctionnement de l’authentification :

client serveur

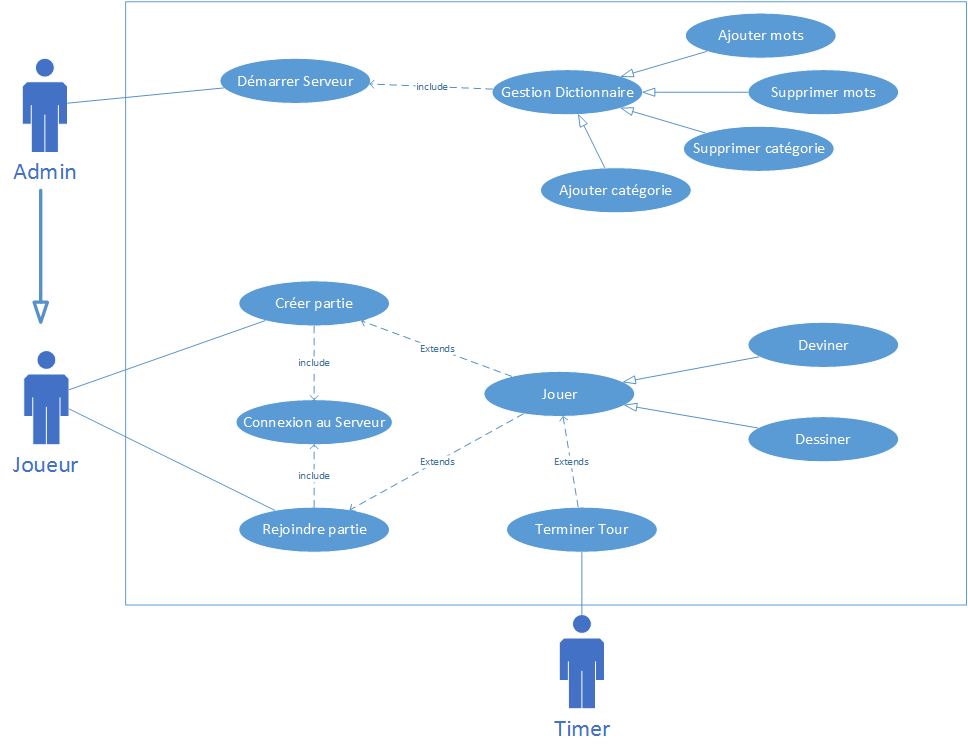


Le client peut effectuer 3 types d’authentification au près du serveur :

* La connexion qui requiert un login et un mot de passe.
* La connexion anonyme qui ne requiert aucune information
* L’inscription qui permet d’ajouter un utilisateur à la base de données.

L’authentification anonyme est pratique pour les tests mais elle ne permet pas à l’utilisateur de sauvegarder ces scores dans la base de données.

### Diagramme de contexte



### Description des acteurs

**Acteur principal : Joueur**

* Le joueur de l’application peut démarrer une partie ou rejoindre une partie. Pour cela, il doit obligatoirement être connecté au serveur.
* Le jouer peut avoir deux rôles : Soit dessiner un mot tiré du dictionnaire soit deviné le mot tiré du dictionnaire et faire des propositions sur le chat.

**Acteur secondaire : Timer, Administrateur**

* Le Timer met fin au tour d’une partie. Une partie est composée de plusieurs tours.
* L’Administrateur peut gérer les mots proposés au joueur et les classer par catégorie.

## Scénarios principales de succès

### Démarrer Serveur

1. Le serveur démarre
2. Le serveur attend les connexions

### Gestion Dictionnaire

1. L’administrateur choisi une opération à faire (Ajouter mots, Supprimer mots, Ajouter catégories, Supprimer catégories)

### Ajouter mots

1. L’administrateur saisie la catégorie du mot
2. L’administrateur saisie le mot à ajouter
3. Le mot est ajouté dans la base de données

### Supprimer mots

1. L’administrateur saisie la catégorie du mot
2. L’administrateur saisie le mot à effacer
3. Le mot est éliminé de la base de données

### Ajouter catégorie

1. L’administrateur saisie le nom de la catégorie
2. La catégorie est ajoutée dans la base de données

### Supprimer catégorie

1. L’administrateur saisie le nom de la catégorie
2. La catégorie (avec tous les mots contenue) est supprimée de la base de données

### Connexion au Serveur

1. Soit l’utilisateur se connecte en anonyme
2. Soit l’utilisateur saisit son user et mot de passe pour se connecter avec son compte
3. Soit l’utilisateur doit créer un nouveau compte

### Créer partie

1. Le joueur se connecte au serveur
2. Le joueur saisie les informations sur la partie à créer
3. La partie est crée

### Rejoindre partie

1. Le joueur se connecte au serveur
2. Le joueur se connecte à une partie existante sur le serveur

### Jouer

1. Le joueur se voit attribuer un rôle (dessiner / deviner)
2. Le joueur joue son rôle tant que la partie n’est pas terminée

### Dessiner

1. Le joueur se voit attribuer un mot à dessiner
2. Le joueur dessine tant que le système n’indique pas le contraire
3. Fin du tour

### Deviner

1. Le joueur voit le dessin dans la zone spécifique
2. Le joueur envoie une réponse
3. Les étapes 1 et 2 se répète jusqu’à la fin du tour.

### Terminer Tour

1. Le timer fige le dessin lorsqu’il arrive à 1 min
2. Le timer met fin au tour lorsqu’il arrive à 2min 30

## Protocole de communication entre le client-serveur

Ce chapitre détaille la liste des messages échangés entre l’application cliente et l’application serveur. Le serveur et le client doivent communiquer entre eux pour les cas suivants :

### Le joueur se connecte au serveur

1. Le joueur envoie un message au serveur pour lui annoncer une nouvelle connexion. (Les messages échangés ici seront les messages liés au protocole TCP).
2. Le joueur envoie un message pour indiquer s’il veut s’enregistrer auprès du serveur ou s’il veut se connecter en tant qu’utilisateur déjà inscrit.
3. Le joueur envoie en suite un message texte contentant son pseudo et son mot de passe.  
   Le joueur à la possibilité d’envoyer un message qui indique qu’il souhaite se connecté en tant qu’utilisateur anonyme, il devra néanmoins indiquer un pseudo.

### Ecran d’accueil du serveur pour créer une partie

1. Le serveur envoie à l’application cliente la liste des parties créées sur le serveur. Le serveur envoie plusieurs messages contenant le nom de la partie et le nom du joueur qui l’a créé.
2. Le joueur envoie un message au serveur pour indiquer s’il veut créer une partie ou s’il veut rejoindre une partie.
3. Le joueur envoie un message texte au serveur indiquant les informations liées à la partie sélectionnée.

### Création d’une partie

1. Le joueur envoie le nom de la partie au serveur.
2. Le serveur envoie au joueur le nom des joueurs inscrits.
3. Le joueur envoie un message pour lancer la partie.

### Rejoindre une partie

1. Le joueur envoie le nom de la partie sélectionnée au serveur.
2. Le serveur envoie le pseudo du joueur à l’application cliente.

### Dessiner durant un tour

1. Le serveur envoie un mot tiré au hasard à un joueur au hasard.
2. Le joueur qui a reçu le mot envoie un message au serveur pour commencer le tour.
3. Le serveur démarre le timer chez les autres joueurs.
4. Le joueur commence son dessin qui est retransmis au serveur.
5. Le serveur envoie le dessin aux autres joueurs.

### Deviner le mot d’un tour

1. Le joueur reçoit le départ du timer.
2. Le joueur fait ses propositions sur le chat, le joueur envoie un message texte au serveur.
3. Le serveur retransmet le message aux autres joueurs de la partie.
4. Si une réponse est correcte le serveur envoie un message aux autres joueurs pour annoncer la fin du tour.

### Fin du tour du timer

1. Lorsque le timer du serveur arrive à 1 minute le serveur envoie un message d’arrête au joueur qui dessine.
2. Lorsque le timer du serveur arrive à 2 minutes 30 alors le serveur envoie un message aux autres joueurs pour mettre fin au tour.

## Modèle de domaine

Ce modèle de domaine décrit le fonctionnement de l’application serveur.



Tirer un   
mot  
aléatoirement

Dessiner

Ecrire

Jouer

Jouer

0…\*

Appartenir

Connecter

Partie

Chat

Dessin

Tour

Equipe

Joueur

Connexion

1

1

1

1

1…\*

1

1

1

2…\*

0…1

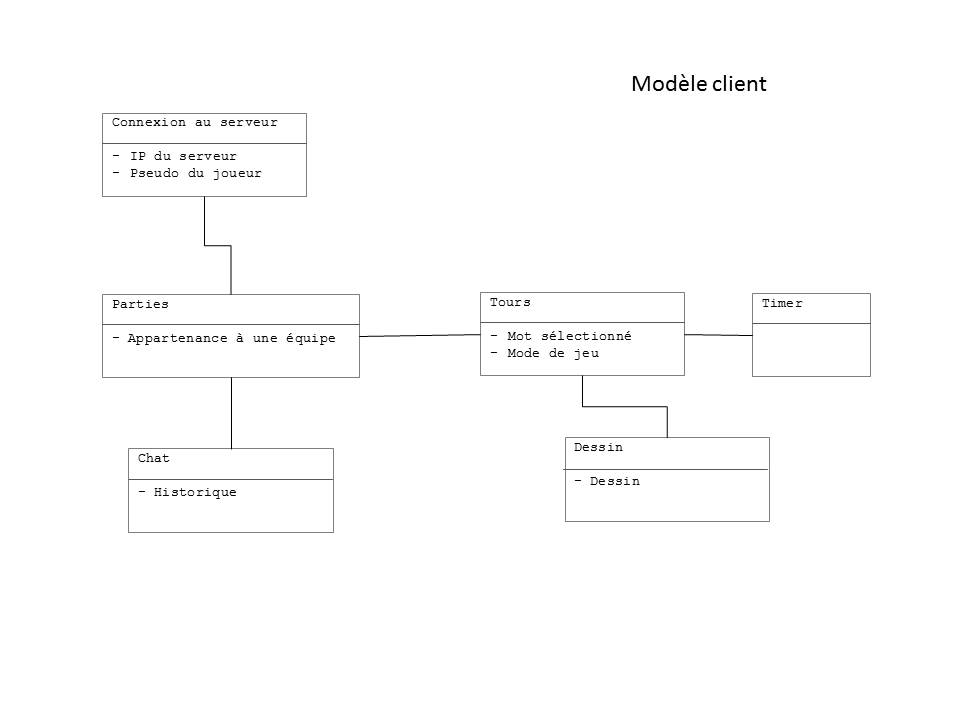
0…\*

1

1

* Connexions : Gère les connexions des joueurs.
* Equipes : Contient le nom de l’équipe et la liste des joueurs appartenant à l’équipe.  
  L’équipe n’est valable que pour une partie.
* Joueurs : Contient le pseudo du joueur.
* Parties : Contient les scores, le nombre de tours qu’ils restent avant la fin de la partie, ainsi que les informations pour retrouver l’administrateur du jeu.
* Tours : Ainsi qu’un mot sélectionné par tour. Ici, le mode de jeu sélectionné représente le joueur qui effectue le dessin.
* Dictionnaire : Base de données contenant les mots ainsi que les catégories à disposition.
* Timer : Contient le nombre de minutes restantes, ainsi que les secondes s’y rapportant.
* Chat : Zone de discussion, contient l’historique des messages.
* Dessin : Gère la transmission du dessin.

Ce modèle de domaine décrit le fonctionnement de l’application cliente.



Gérer

Dessiner

Jouer

Ecrire

Connecter

1

1

1

1

1

1

1..\*

1

0..\*

1

* Connexion au serveur : Contient l’IP du serveur ainsi que le pseudo du joueur.
* Parties : Contient l’équipe du joueur.
* Chat : Zone de discussion, contient l’historique des messages.
* Tours : Contient le mode de jeu qui indique si c’est au tour du joueur de dessiner.   
  Dans ce cas, il y a également un mot sélectionné par tour.
* Dessin : Zone de dessin.

## Base de donnée et conception du dictionnaire

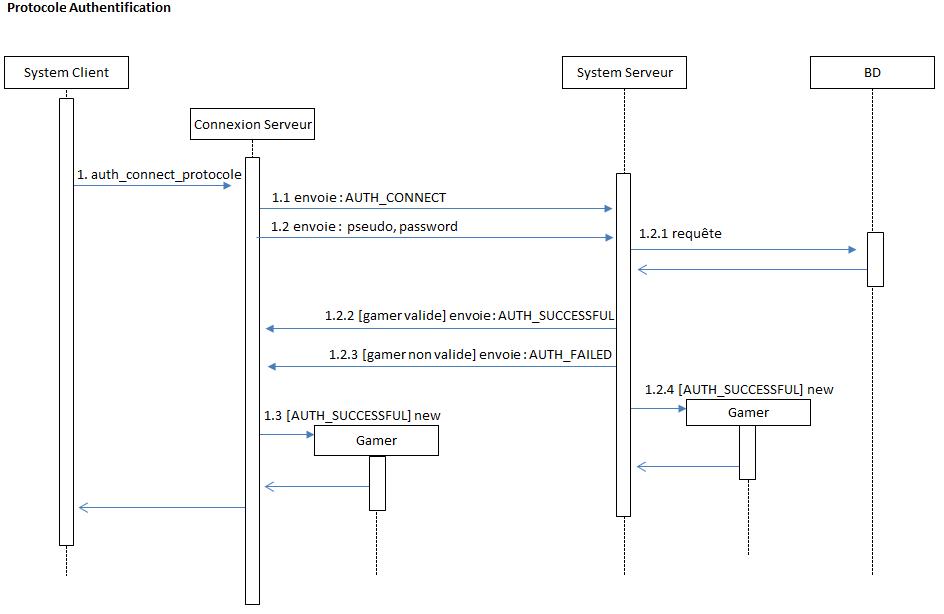
Pour le fonctionnement de l’application, il faut tirer un mot aléatoirement issu du dictionnaire. Les mots tirés au hasard sont classés par catégorie et le joueur peut choisir la catégorie pour son tour.

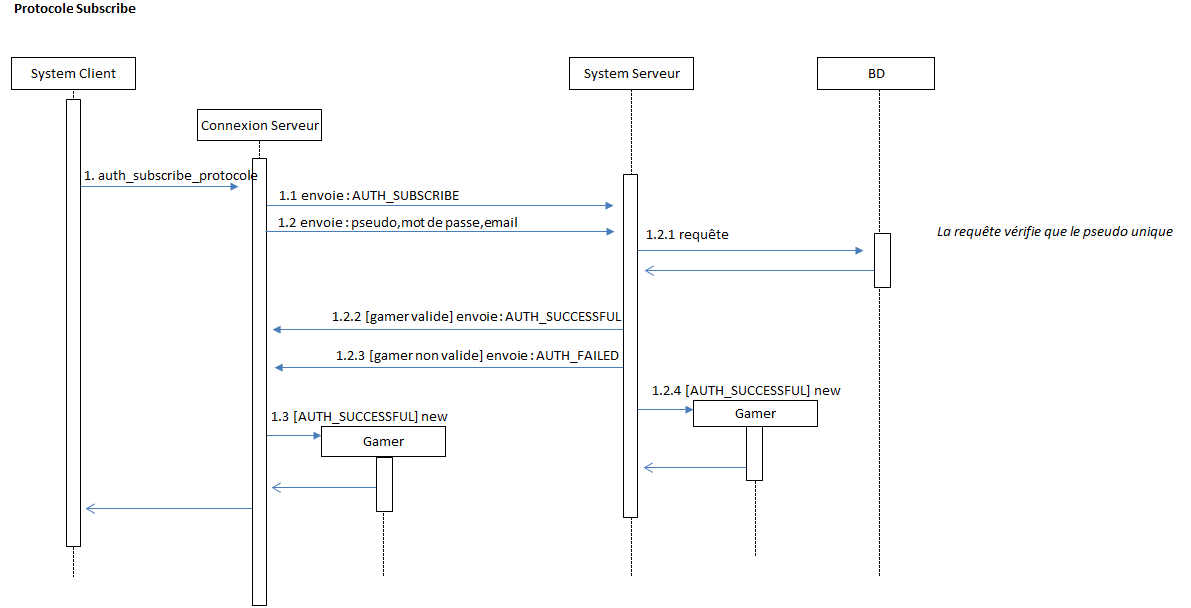
La structure de donnée à mettre en place n’est pas particulièrement complexe. Au début, les données étaient stockées dans un fichier XML mais le cahier des charges proposaient également de gérer des statistiques c’est pourquoi nous avons finalement opté pour une base de données.

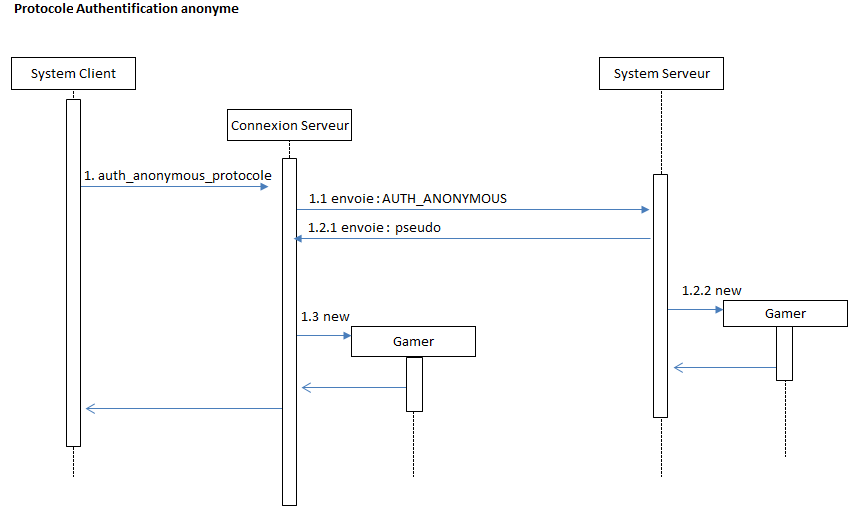
Les joueurs inscrits sur le serveur seront enregistrés sur dans une base de donnée. Les joueurs enregistrés pourront publier leurs scores et les consulter.

# Conception du projet

## Protocole d’échange entre le client/serveur

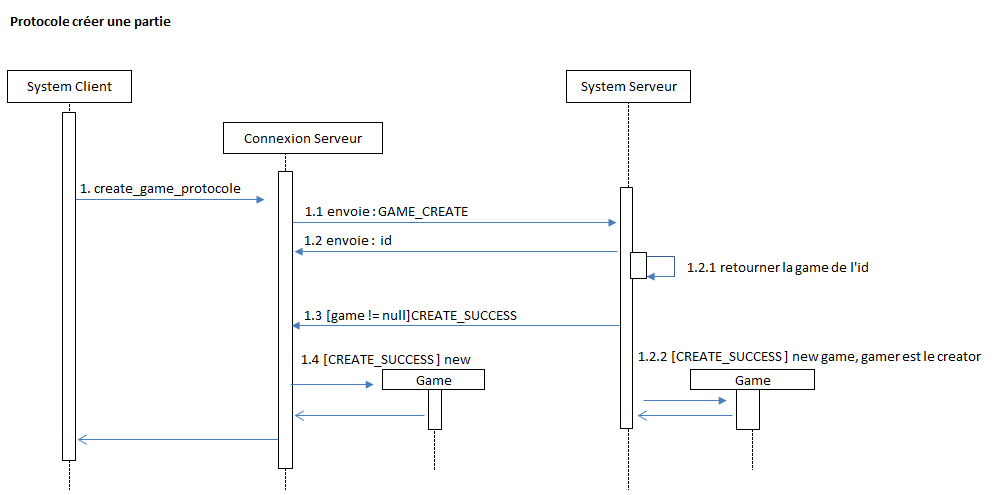


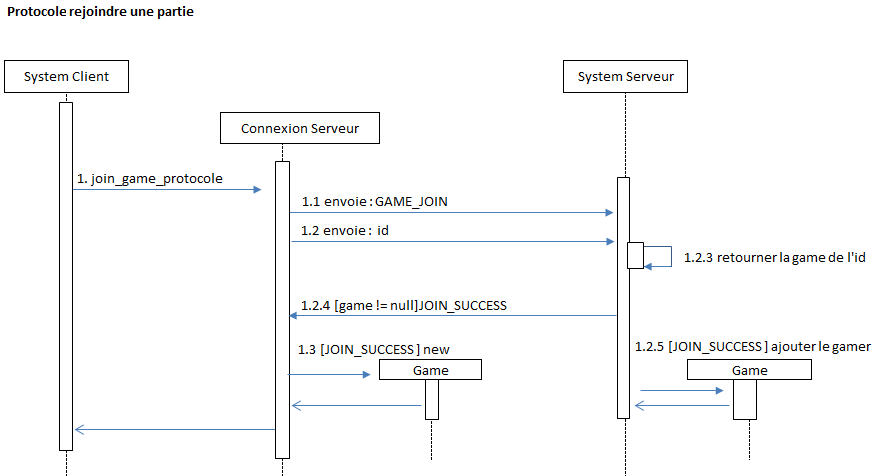




1.1.1

1.2 -

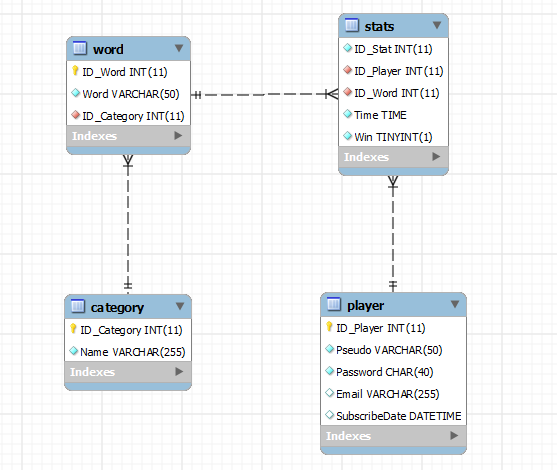




## Base de données

### Schéma conception et relations





### Commentaires

Nous avons mis uniquement le schéma relationnel car, vu qu’il n’y a pas de table d’association nécessaire pour cette base, le schéma conceptuel et le schéma relationnel sont les mêmes.

#### Tables

Une base de données est nécessaire pour notre projet, elle nous permet de stocker les pseudos des joueurs ainsi que les mots de passe (hashé en SHA-1) afin de pouvoir créer un système de connexion.

Il est également nécessaire de stocker les listes de mots ainsi que les catégories pour pouvoir offrir un choix au créateur de la partie.

Pour finir nous avons décidé de relier les joueurs et les mots par une table statistique qui nous permet du coup de stocker des statistiques pour les joueurs mais aussi pour les mots.

#### Liaisons

Avec ce schéma il nous est possible de :

* Récupérer les mots par catégorie
* Voir les statistiques par mot
* Voir les statistiques par joueur

En revanche, il n’est pas possible de voir les statistiques d’un joueur par partie.

# Implémentation du projet

## Technologies utilisées

* MySQL
* SqlLite
* Java Swing
* GitHub

La technologie SQLlite permet d’intégrer la base de données au projet. Grâce à cette technologie la base de données est portable et donc très pratique pour un déploiement ou un test.

Bien sûr, il serait facile de modifier la base de données pour l’adapter à un autre type de base de donnée de type MySQL.

## Problèmes rencontrés et solutions

# Gestion du projet

## Rôle principale/Groupe de développement

Il y a 4 personnes qui composent le groupe de développement.

|  |  |  |
| --- | --- | --- |
| **Rôle standard** | **Responsabilités** | **Responsable** |
| Représentants des utilisateurs | * Collecte des besoins pour le projet. * Spécifications des tests de fonctionnalités * Explication des aspects métiers | Melly Calixte |
| Chef de projet | * Planification * Coordinations avec les utilisateurs | Fröhlich Magali |
| Analyste | * Spécifications * Collecte des demandes de changement | Righitto Simone |
| Architecte,  concepteur en chef | * Conception de l’architecture du produit | [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New),  Fröhlich Magali |
| Programmeur | * Participe à la conception du produit * Ecrit les tests unitaires * Codage | Righitto Simone,  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New),  Fröhlich Magali,  Melly Calixte |
| Responsable des tests | * Participe à l’intégration continue des composants * Ecrit les tests fonctionnels * Met en place l’architecture permettant de lancer régulièrement les tests fonctionnels | [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone |
| Responsable de la configuration | * Gestion de la base des artefacts du projet * Gestion des releases * Allocation des droits * Responsable de la configuration (logicielle & matérielle) * Intégration des changements | [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) |

## Plan d’itérations initial

|  |  |  |
| --- | --- | --- |
| **Semaine** | **Objectif et déroulement** |  |
| 28 avril2014 | Rendu du rapport intermédiaire | 1 |
| 28 avril 2014  05 mai 2014 | Objectif : Connexion au serveur, Dictionnaire  Le but de cette itération est de permettre à l’administrateur de démarrer le serveur et de le configurer. En parallèle, les fichiers XML qui constituent le dictionnaire seront développés.  Le serveur pourra gérer en suite le dictionnaire.  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone : Analyse et création du fichier XML pour la gestion des mots qui composent le dictionnaire.  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) : Conception du serveur et de l’application cliente.  Fröhlich Magali : programmation des classes pour la gestion du dictionnaire sur le serveur.  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : Programmation des classes de base pour l’application cliente.  Fonctionnalités produites à la fin de l’itération :  Démarrer le serveur et gérer le dictionnaire. Gérer le dictionnaire correspond aux actions : ajouter, modifier, supprimer un mot ou une catégorie. | 2 |
| 05 mai 2014  12 mai 2014 | Objectif : Gestions des joueurs  Le but de cette itération est de pouvoir gérer l’enregistrement des joueurs sur le serveur ainsi que leur authentification à travers la base de données.  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone : Programmation des classes pour la gestion des joueurs sur le serveur.  Saam Frédéric : Conception du modèle observable-observé pour préparer les interfaces graphiques. Programmation des interfaces du côté serveur.  Fröhlich Magali : Programmation des classes pour gérer les connexions des joueurs au serveur.  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : Programmation de la base de données et des interfaces graphiques côtés client.  Fonctionnalités produites à la fin de l’itération :  Gérer les joueurs correspond à la possibilité de supprimer un joueur et d’afficher la liste des joueurs enregistrés sur la base de données. | 3 |
| 12 mai 2014  19 mai 2014 | Objectif : Création d’une partie, Rejoindre une partie  Le but de cette itération est de pouvoir gérer la création des parties par un joueur sur le serveur.  Les joueurs doivent pouvoir se connecter au serveur pour rejoindre une partie créée.  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone : Programmation des classes pour la gestion des parties :  Création des parties et Rejoindre une partie.  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) : Conception des classes pour la gestion des parties.  Fröhlich Magali : Gestion des connexions et programmation des messages échangés pour la création de la partie. Cela comprend également la possibilité pour le joueur de rejoindre une partie.  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : Programmation des classes pour la gestion des parties : Rejoindre une partie.  Fonctionnalités produites à la fin de l’itération :  Les joueurs peuvent se connecter au serveur et gérer des parties. Cela comprend la partie authentification, la création d’une partie et la possibilité de rejoindre une partie.  Dans un premier temps, on ne considérera que le mode sans équipe. | 4 |
| 19 mai 2014  26 mai 2014 | Objectif : Dessiner, Chat  Le but de cette itération est de pouvoir voir l’interface finale de l’application cliente. Il doit être possible d’effectuer un dessin sur l’application graphique.  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone : Programmation de l’interface graphique générale.  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) : Conception des classes pour le dessin et le chat.  Fröhlich Magali : Programmation de l’interface graphique pour dessiner.  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : Programmation de l’interface graphique pour écrire dans le chat.  A la fin de cette itération, il sera possible dessiner et d’écrire un message dans le chat.  On ne considérera que le mode sans équipe. Ce cas d’utilisation ne sera donc pas gérer complétement à cette itération. | 5 |
| 26 mai 2014  02 juin 2014 | Objectif : Dessiner, Chat  Le but de cette itération est de pouvoir utiliser l’interface graphique de l’application cliente pour effectuer un dessin retransmis sur les autres applications clientes. Il doit être également possible d’utiliser le chat entre les différentes applications.  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone : Programmation de l’affichage des scores sur le serveur.  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) : Conception et programmation du Timer et de la gestion des scores.  Fröhlich Magali : Programmation des échanges pour afficher le dessin  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : Programmation des échanges pour afficher le chat  A la fin de cette itération, il sera possible dessiner et d’écrire un message dans le chat. Ces données seront retransmises aux autres applications clientes. Il sera également possible d’afficher les scores sur le serveur.  On ne considérera que le mode sans équipe. | 6 |
| 02 juin 2014  09 juin 2014 | Objectif : Gérer les tours  Le but de cette itération est de pouvoir utiliser l’application cliente/serveur pour jouer une partie. Il doit être possible de tirer un mot du dictionnaire aléatoirement pour qu’un utilisateur le dessine.  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone : Test générale de l’application.  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) : Programmation des messages échangés entre le timer et les applications.  Fröhlich Magali : Programmation des messages échangés pour la gestion des équipes.  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : Test générale de l’application.  A la fin de cette itération, il sera possible de jouer une partie en équipe ou en mode tous contre tous. | 7 |
| 09 juin 2014  16 juin 2014 | Objectif : Conception final de l’application.  A présent, le serveur doit pouvoir gérer plusieurs connexions pour gérer plusieurs parties en même temps.  [Righitto](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Simone : Découverte réseau.  [Saam Frédéric](https://webmail.heig-vd.ch/owa/?ae=Item&t=IPM.Note&a=New) : Gestion de plusieurs parties lancées en même temps sur le serveur.  Fröhlich Magali : Documentation.  [Melly](https://webmail.heig-vd.ch/owa/?ae=PreFormAction&t=IPM.Note&a=ViewRcpt&sT=AD.RecipientType.User) Calixte : Test final de l’application.  A la fin de cette itération, le développement de l’application sera terminé. | 8 |
| 16 juin 2014  23 juin 2014 | Présentation et rendu final | 9 |

## Stratégie de tests

Il y a deux types de tests qui ont permis de tester les applications. Les tests unitaires qui ont permis de tester les protocoles de tests. Nous avons testé les protocoles en affichant leurs résultats dans la console.

Chaque méthode « protocole » effectue une action, elles peuvent être appelées sans interface graphique.

Par exemple, voici un test effectué dans la méthode main de la classe PictioLAN. Ce teste permet de tester la vue JClient hors connexion.

**public** **static** **void** main (String[] args) {

*modele\_gamer* = **new** Gamer();

*modele\_gamer*.setGame(**new** Game(0, "", 2, **false**, "", 2));

*modele\_gamer*.getGame().setIsDrawer(**true**);

JClient client = **new** JClient("test");

}

Voici un autre exemple de test qui permet de créer une partie avec l’interface graphique principale :

**public** **static** **void** main (String[] args) {

*modele\_gamer* = **new** Gamer();

*modele\_gamer*.setGame(**new** Game(0, "", 2, **false**, "", 2));

*modele\_gamer*.getGame().setIsDrawer(**true**);

JClient client = **new** JClient("test");

*modele\_gamer* = **new** Gamer();

*modele\_gamer*.launchConnexion("127.0.0.1");

*modele\_gamer*.getConnection().auth\_anonymous\_protocole();

*modele\_gamer*.getConnection().create\_game\_protocole(2, **false**, 4, "Animaux");

//modele\_gamer.getConnection().join\_game\_protocole(new Game(0,"Animaux",4,false,"Anonyme\_0",2), 0);

*modele\_gamer*.getGame().setClient(**new** JClient("test"));

}

Les éléments graphiques de l’interface ont été testés à la main. Il est possible d’automatiser également ces tests mais dans le cas de notre application ça n’a pas été nécessaire.

## Stratégie d’intégration du code de chaque participant

Les méthodes protocoles peuvent-être modifiées indépendamment des autres tant que l’état de la partie reste cohérent par rapport au diagramme d’activité. Ainsi, tout le monde peut modifier ces méthodes tant qu’il garantit l’état de la partie.

En pratique, cela est difficile à mettre en place. Nous avons découpé le travail en deux groupes :

1 groupe qui s’est concentré sur les interfaces graphiques et un groupe qui s’est concentrés sur le protocole de communication.

# Etat des lieux

## Ce qui fonctionne (résultats des tests)

## Ce qu’il resterait à développer (en proposant une planification)

# Autocritique

**Frédéric Saam**

Ce projet nous a permis de travailler en groupe de plus de deux personnes, ce qui arrivait rarement avant le début de ce semestre. Il s’agit d’un projet intéressant car il permet de choisir librement une application à créer sans avoir trop de contraintes.

En revanche, le projet n’aurait jamais dû prendre place en même temps que le cours de PRO, car cela a créé une surcharge de travail sur toute la durée du semestre.

Pour finir, le fait de nous faire planifier notre projet selon un plan d’itération est une bonne idée, mais c’en est une mauvaise de faire un plan par semaine (itération trop courtes) et de vérifier que ça soit fait chaque semaine, ce serait mieux toutes les 2 ou 3 semaines afin d’éviter les surcharges pendant les périodes où il y a beaucoup de test et de laboratoires à rendre.

**Calixte Melly**

Ce projet m'a permis de découvrir une méthode de développement intéressant et de plus c'est la première fois que j'ai l'occasion d'utiliser une autre méthode que celle du "GANTT".

Qui plus est le logiciel a réalisé était intéressant, le travail de groupe a toujours bien marché et la gestion du projet a été menée à bien grâce à notre chef.

C'est donc une expérience positive que je retiens pour ce projet.

**Magali Frohlich**

La gestion du temps a été un défi constant durant tout le projet. La planification des itérations est vrai casse-tête. Il est difficile de produire des résultats concluants pour un client toutes les semaines. Il faut développer à chaque fois le code qui permet de faire fonctionner le background de l’application ainsi qu’une interface graphique. Le développement du projet PRO en parallèle n’était pas évident à gérer non plus.

La gestion du dessin était un point ambitieux, il a posé beaucoup de problème dont certains points ne sont pas encore résolus.

Je n’avais jamais travaillé avec mes collègues mais tout c’est bien passé je garde un bon souvenir de notre équipe.

**Righitto Simone**

# Conclusion

# Annexe

## Manuel d’installation

## Manuel d’utilisation