Rapport de projet :

Site pour serveur « Minecraft »

Réalisé par :

FAIVRE Loïc

Sous la direction d'Anne LAURENT et d'Arnaud CASTELLTORT

(Professeurs du département d'Informatique et Gestion à Polytech Montpellier)

Année universitaire 2014/2015

Table des matières

I Introduction

Le monde du jeu vidéo est aujourd'hui et depuis plus de 10 ans en pleine expansion. En effet, plus de 80% des français ont joué en 2014 au moins une fois à un jeu vidéo sur une plateforme quelconque. Cette évolution est principalement due à la notoriété que certaines grandes firmes se sont créée par le développement et la diffusion à grande échelle de MMORPG, prenons pour exemple « World Of Warcraft », « Minecraft ».

Enormément de petit groupe de personnes se sont par la suite formés, avec pour objectif de déployer des serveurs uniques, ayant leurs propres caractéristiques, sur lesquels les joueurs pourront se retrouver pour pouvoir s'amuser sur leur jeu favori.

Pour cela, de nombreuses interfaces comme les sites web sont apparus dans le monde du jeu vidéo, permettant de réunir un maximum d'utilisateur.

L'objectif de ce projet, était donc pour ma part de développer un site web pour un groupe de joueur ayant pour intention de créer un serveur « Minecraft », afin d'offrir de nombreuses fonctionnalités supplémentaires adaptés aux besoins des utilisateurs et au jeu, mais aussi pour promouvoir le serveur afin qu'il puisse pourquoi pas devenir un des plus grands serveurs français.

Après avoir fait une synthèse des besoins du groupe de joueurs et des potentiels utilisateurs de « SeriousCraft », nous détaillerons dans le rapport technique l'ensemble des fonctionnalités qui ont été développées, l'architecture adoptée, ainsi que le résultat final. Enfin, nous parlerons dans le rapport d'activité des méthodes utilisés, des outils de travail qui a permis d'amener à son terme ce projet.

II Cahier des charges

II.1) Présentation du sujet et analyse de l'existant

Ce site web pour un serveur « Minecraft », offrira aux utilisateurs (au sens joueur) un grand nombre de fonctionnalités qui seront directement en lien avec l'application (Boutique, Classement, etc…).

De multiples sites pour des serveurs « Minecraft » existent actuellement sur le web, on peut en citer quelques-uns comme exemple : funcraft.net, imagineyourcraft.fr, lifecraft.fr.

Il a donc été nécessaire de faire une analyse de l'existant permettant de révéler les différentes idées et inconvénients de ces sites web.

Inspirées pas les autres sites ou nécessitées :

- Page d'accueil de présentation
- Page expliquant comment se connecter sur l'application et de jouer rapidement et sans difficulté.
- Possibilité d'acheter des grades ou des objets uniques depuis le site, et utilisable en jeu en quelques simples clics.
- Page de classement suivant différents critères
- Page de vote pour le serveur

Inconvénients majeurs chez les sites concurrents :

- La plupart des autres sites sont surchargés et pas assez clair pour l'utilisateur
- Les utilisateurs ne sont en général que « visiteurs », si ce n'est la boutique qui ne donne pas un réel impact sur l'application en elle-même.

Il y a donc ici un défaut important que nous voulons corriger qui est : l'impact de l'utilisateur sur le serveur.

II.2) Analyse des besoins fonctionnelles

Nous avons donc par la suite établi les différentes fonctionnalités qu'un visiteur (non-inscrit) ainsi que celles d'un utilisateur (inscrit sur l'application), qui seront les deux seuls acteurs du système, les administrateurs n'interviennent pas directement dans la gestion de ce site web.

Un visiteur pourra :

- **Consulter les informations liées** à ce projet sur la page d'accueil
- **Savoir comment accéder** à l'application « Minecraft » **et se connecter** depuis la page Nous-rejoindre
- **Voir les évènements** qui se produisent sur le serveur
- **Voir le classement** des joueurs suivant différents critères
- **Des liens externes** comme la page Facebook, le compte Twitter, la chaîne YouTube de l'application.

L'utilisateur pourra quant à lui :

- **S'identifier**, pour avoir accès aux différentes fonctionnalités-utilisateur.
- **Acheter des objets ou des grades** depuis la page Boutique
- **Créer des évènements** sur la page Event
- **Voter pour le serveur** depuis la page de Vote
- **Avoir accès à toutes les informations de sa faction** sur la page Faction

II.3) Les diagrammes d'analyse

Comme on peut le voir dans l'annexe (1, 2 et 3), j'ai choisis de réaliser 3 diagrammes différents concernant les fonctionnalités, les actions des utilisateurs et des visiteurs, ainsi que les concepts qui doivent être implémentés dans ce projet (pour la base de données notamment).

Ces diagrammes sont : un MCD, un diagramme de classe et un diagramme des fonctionnalités (Use-Case) que nous décrirons et dont nous expliquerons l'utilité.

a) MCD : (voir en annexe 1)

Le MCD est un diagramme important quand il s'agit de communication avec le client mais aussi d'envisager une base de données par la suite. En effet, c'est un diagramme qui est facilement compréhensible, et que j'ai utilisé pour présenter les concepts que je vais créer d'après les éléments que les clients m'ont présentés.

Les différents concepts (entités) présents sur le MCD sont : Joueur (concept principal), Faction, Serveur, Grade, Objet et Evènement. Comme on peut le voir certaines entités possèdent des id et d'autres non, ceci est dû aux différents plugins qui vont être utilisés sur l'application. Etant donné que la base de données entre le serveur et le site est partagée (schéma ci-après) et que certains nécessite des id, elles ont donc été rajouté dans le MCD.

b) Diagramme de classe : (voir en annexe 2)

Le diagramme de classe d'analyse, va dans notre cas nous permettre de mettre en évidence les différents acteurs de la base de données et les interactions qui vont leur correspondre. Ce diagramme de classe est plus utile d'après moi pour le développeur, pour permettre de faire le lien entre les différentes classes et faciliter la mise en place des fonctions.

c) Diagramme des fonctionnalités : (voir en annexe 3)

Enfin le diagramme des Use-Case permet de faire ressortir les différentes possibilités/fonctionnalités des utilisateurs, mais aussi de montrer ce qu'il est obligatoire de faire (« include ») par le système par exemple et ce qui n'est pas obligatoire mais qu'il est possible de faire (« extends »).

III Rapport technique

III.1) Conception / Programmation

Dans cette partie, nous allons parler des différents outils techniques que j'ai utilisés pour réaliser ce projet : les langages de programmation, un framework, et un schéma technique montrant les liens entre les acteurs (Site Web – Base de données – Serveur minecraft – Joueur).

a) Langages de programmation

PHP (Facilité de communication avec la base de données ainsi qu'un apprentissage facile).

HTML (Gestion de l'affichage des différentes pages du site web).

CSS (Création d'une feuille de style qui offre la possibilité de mettre en place ou de modifier des blocs par exemple)

Ce sont les 3 langages de programmations que j'ai utilisés car ils sont les plus adaptés concernant ce projet que ce soit sur leur simplicité d'utilisation, mais aussi leur portabilité.

b) Bootstrap – Description

Bootstrap est un framework HTML, CSS, JS, permettant d'obtenir de façon simple un design responsive (c'est-à-dire adapté à n'importe quel écran), ainsi que orienté mobile très rapidement.

Pour l'utiliser, il suffit d'intégrer sur chacune des pages (suivant son architecture) les liens vers le CSS récupérable depuis le site « getbootstrap.com » ou bien de télécharger le CSS et de l'intégrer dans ses propres dossiers (permettant par exemple d'éviter toute mise à jour imprévue sur un des objets Bootstrap que l'on utilise.

c) Bootstrap – Objets utilisés

Dans le projet, énormément de ces objets ont été utilisés :

- <nav class="navbar navbar-default" id="myNav"> (Barre de navigation)
- (Article avec un titre et un contenu)
- <form method="post" action=""> (Formulaire avec son contenu en responsive)

d) Schéma d'architecture (3 tiers)

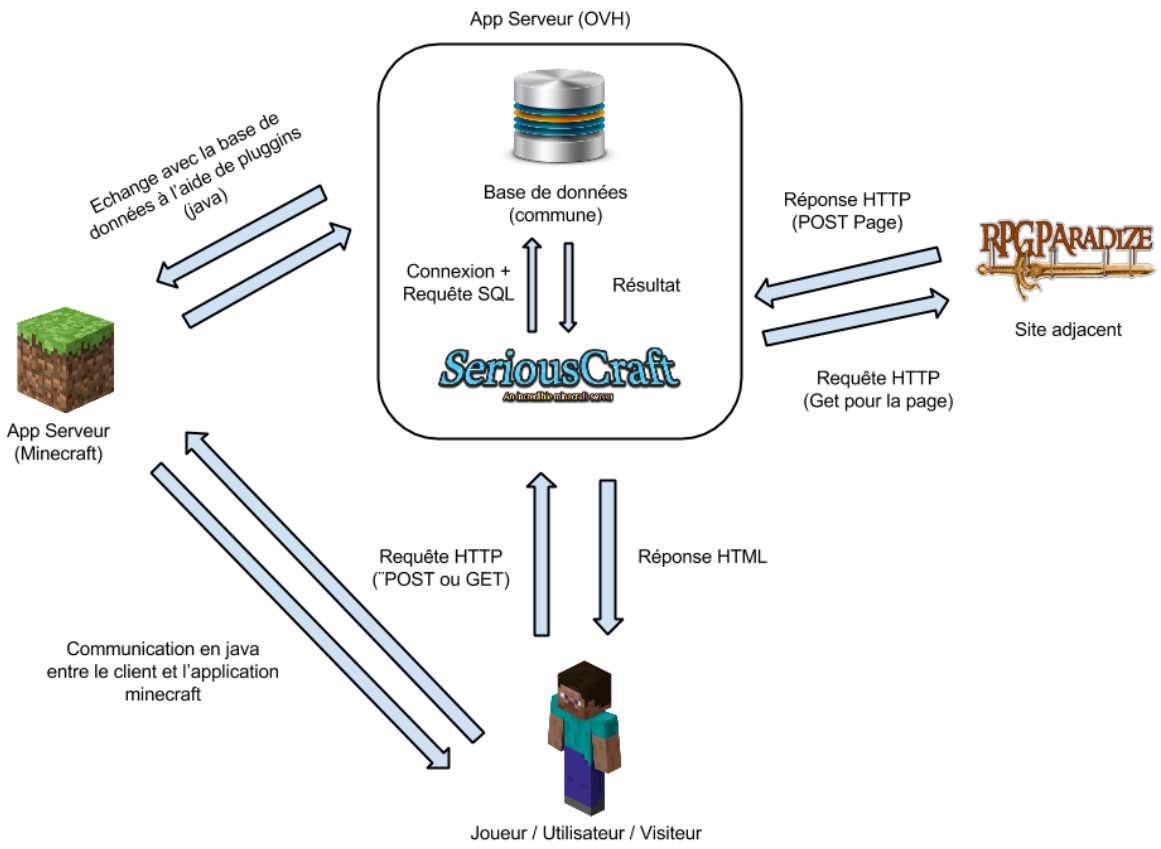

Illustration 1 : Schéma 3 tiers SeriousCraft

Comme le montre ce schéma, le joueur peut être représenté sous deux formes, la première étant la communication avec le serveur d'application Minecraft, qui va gérer automatiquement par l'intermédiaire de pluggins les échanges avec la base de données, ou avec notre site internet.

Pour communiquer avec SeriousCraft.fr, le joueur va à partir de la page html envoyé une requête http, qui va être traitée par l'intermédiaire du PHP (en faisant une connexion à la base de données, puis une requête), puis le serveur de base de données va renvoyer un résultat qui sera encore une fois traité en PHP pour afficher une page html retourné sur l'écran du joueur.

J'ai tenu à rajouter sur ce schéma un lien avec « les sites adjacents », car dans l'avenir je compte sécuriser les votes pour le serveur, et la seule solution qui s'offre à nous, est de récupérer la page du site de vote (site adjacent), l'afficher sur notre site web, récupérer la réponse du client, l'envoyer au site de vote, et enfin attendre la réponse pour voir si le client a voté correctement.

Dans les prochaines différentes parties, je vais vous décrire le code des points importants de mon site web avec à chaque fois un capture d'écran et un commentaire des parties du code.

e) Affichage des différents classements

```php
<div class="centerContent">
    <div class="panel panel-default">
      <div class="panel-heading">Classement Seriousflouz</div>
      <div class="panel-body">
            <table class="table table-striped">
                <th> Pseudo </th>
                <th> Faction </th>
                <th> Seriousflouz </th>

                <?php

                try {
                    $connexionBD = new PDO('mysql:host=localhost;dbname=bd_seriouscraft;charset=utf8','root', '');  // Point 1
                }
                catch (Exception $e) {
                    die('Erreur : ' . $e->getMessage());
                }

                $sql = 'Select J.pseudo_J, J.seriousflouz, F.nom_Faction
                        From Joueur J, Faction F
                        Where (J.nom_Faction = F.nom_Faction)
                        Order By J.seriousflouz DESC ;';                                    // Point 2
                $requete = $connexionBD->prepare($sql);
                $requete->execute();
                $resultats = $requete->fetchAll(PDO::FETCH_ASSOC); // Renvoie un tableau (array)

                foreach($resultats as $value){
                    echo ('<tr><td>'.' '.$value['pseudo_J'].'</td><td>'.' '.$value['nom_Faction'].'</td><td>'.' '.$value['seriousflouz'].'</td></tr>');   // Point 3
                }

                ?>
            </table>
      </div>
    </div>
```

```sql
Select J.pseudo_J, J.seriousflouz, F.nom_Faction
                        From Joueur J, Faction F
                        Where (J.nom_Faction = F.nom_Faction)
                        Order By J.seriousflouz DESC ;

Select J.pseudo_J, J.nb_Vote, F.nom_Faction
                        From Joueur J, Faction F
                        Where (J.nom_Faction = F.nom_Faction)
                        Order By J.nb_Vote DESC ;


Select J.pseudo_J, J.nb_PointBoutique, F.nom_Faction
                        From Joueur J, Faction F
                        Where (J.nom_Faction = F.nom_Faction)
                        Order By J.nb_PointBoutique DESC ;
```

Le fonctionnement de l'affichage du classement des joueurs est très simple. J'essaie tout d'abord de me connecter à la base de données (1). Si cela fonctionne, je créer une requête SQL que je prepare() (plus sécurisant quant à la connexion et la récupération d'information vers une base de données) puis que j'exécute en récupérant les résultats dans la variable $resultats (2). Enfin dans une troisième partie, j'affiche les différents points que je veux récupérer, notamment le pseudo, le nom de la faction, le nombre de seriousflouz que j'affiche sous forme de tableau.

Tout ceci étant stocké dans une div panel-body puis dans un tableau contenant le nom des attributs que je veux afficher (adapté à chacune des requêtes correspondant aux différents types de classement).

Enfin on peut aussi ajouter que la page faction, d'évènement fonctionne exactement de la même manière que celle-ci, sauf pour la création d'évènement que nous verrons un peu plus tard.

f) Connexion

```php
<?php
    if (!empty($_POST['pseudo_J_Input'])){                // Point 1
        $joueurExiste = False;
        $pseudoJoueurRecup = NULL;
        $mdpJoueurRecup = NULL;

        try {
            $connexionBD = new PDO('mysql:host=localhost;dbname=bd_seriouscraft;charset=utf8','root', '');
        }
        catch (Exception $e) {
            die('Erreur : ' . $e->getMessage());
        }

        $sql = '(Select pseudo_J, mdp_J
                    From joueur);';
        $requete = $connexionBD->prepare($sql);
        $requete->execute();
        $resultats = $requete->fetchAll(PDO::FETCH_ASSOC); // Renvoie un tableau (array)

        foreach($resultats as $value){                                                                          // Point 2
            if ($value['pseudo_J'] == $_POST['pseudo_J_Input'] && $value['mdp_J'] == $_POST['mdp_J_Input']){
                $pseudoJoueurRecup = $value['pseudo_J'];
                $mdpJoueurRecup = $value['mdp_J'];
                $joueurExiste =  true;
                break 1;
            }
        }
        if ($joueurExiste){
            setcookie('cookieSerious', sha1($pseudoJoueurRecup . '.' . $mdpJoueurRecup), time() + 3600);        // Point 3
            header("Refresh:0");
        }

        else {
            echo '<div class="alert alert-danger role="alert">Erreur : Pseudo et/ou Mot de passe incorrect</div>';          // Point 4
        }
    }
?>

<div class="panel panel-primary">
  <div class="panel-heading">
    <h3 class="panel-title">Connectez-vous !</h3>
  </div>
  <div class="panel-body">
    <form method="post" action="">
```

La connexion sur le site internet a été faite sans utilisation de session mais uniquement par l'intermédiaire de cookie. Cette méthode permet d'éviter la surcharge du côté du serveur s'il y a un flux très important de connexions simultanées, qui pourraient dans le pire des cas le faire crash.

Point 1 : On rentre dans cette condition dès que l'on reçoit une requête de type post avec comme contenu 'pseudo_J_Input' (soit la requête venue du formulaire de connexion).

Point 1bis : la connexion à la base de données et la requête expliquées plus haut.

Point 2 : On vérifie si le joueur existe dans la base de données avec son pseudo et son mot de passe récupérés par le POST, si on le trouve, on stocke le pseudo et le mot de passe dans 2 variables distinctes.

Point 3 : On créer un cookie dont le nom sera 'cookieSerious', avec une valeur cryptée, et une durée de vie d'une heure que l'on donne à l'utilisateur.

Point 4 : Si le joueur n'existe pas on affiche une petite erreur pour le signaler à l'utilisateur afin qu'il puisse retenter de se connecter.

g) Reconnaissance du cookie / de la connexion

```php
<div class="rightContent">
    <?php
        if (isset($_COOKIE['cookieSerious'])){                                          // Point 1
            $testCookie = false;

            $cookieSerious = $_COOKIE['cookieSerious'];

            try {
                $connexionBD = new PDO('mysql:host=localhost;dbname=bd_seriouscraft;charset=utf8','root', '');
            }
            catch (Exception $e) {
                die('Erreur : ' . $e->getMessage());
            }

            $sql = '(Select pseudo_J, mdp_J
                        From joueur);';
            $requete = $connexionBD->prepare($sql);
            $requete->execute();
            $resultats = $requete->fetchAll(PDO::FETCH_ASSOC);

            foreach($resultats as $value){                                              // Point 2
                $key = sha1($value['pseudo_J'] . '.' . $value['mdp_J']);
                if ($key == $cookieSerious){
                    $testCookie = true;
                    break 1;
                }
            }

            if ($testCookie){                                                           // Point 3
                include('deconnexion.php') ;
            }
        }

        else {
            include('connexion.php') ;                                                  // Point 4
        }
    ?>
```

Point 1 : Je vérifie s'il y a un cookie

Point 2 : Je teste si le cookie ainsi récupéré correspond à un cookie utilisateur en testant tous les cookies possibles.

Point 3 : Si c'est le cas, j'inclue le module déconnexion (graphiquement on aura donc un bouton de déconnexion, etc…)

Point 4 : Sinon j'inclue le module de connexion

Les includes sont aussi dans mon architecture un moyen de « Factoriser » mon code, afin de ne pas avoir à surcharger les pages, et copier-coller à chaque fois. Ceci évite aussi, s'il y a une modification à faire de devoir tout changer.

Pour continuer sur les cookies, on peut noter que la déconnexion possède un fonctionnement similaire à la connexion, on appelle la méthode setcookie(), en lui mettant cette fois-ci comme argument une date négative, ce qui a pour effet de le détruire.

h) Insertion d'évènement

```php
<?php
    if (!empty($_POST['nom_Evenement'])){                                                                   // Point 1
        $evenementNotExist = True;

        try {
            $connexionBD = new PDO('mysql:host=localhost;dbname=bd_seriouscraft;charset=utf8','root', '');
        }
        catch (Exception $e) {
            die('Erreur : ' . $e->getMessage());
        }

        $sql = '(Select * From evenement);';
        $requete = $connexionBD->prepare($sql);
        $requete->execute();
        $resultats = $requete->fetchAll(PDO::FETCH_ASSOC); // Renvoie un tableau (array)

        foreach($resultats as $value){                                                                      // Point 2
            if ($value['nom_Evenement'] == $_POST['nom_Evenement']){
                $evenementNotExist = False;
                break 1;
            }
        }

        if ($evenementNotExist){                                                                            // Point 3
            $nom_Evenement = $_POST['nom_Evenement'];
            $dateCreation = date("Y-m-d");
            $date_Debut = $_POST['date_Debut'];
            $date_Fin = $_POST['date_Fin'];
            $commentaire = $_POST['commentaire'];

            $sql = "INSERT INTO evenement (id_Evenement, nom_Evenement, dateCreation_Evenement, dateDeb_Evenement, dateFin_Evenement, description_Evenement, pseudo_J)
                    VALUES (NULL, '$nom_Evenement', '$dateCreation', '$date_Debut', '$date_Fin', '$commentaire', 'themonheal');";
            $requete = $connexionBD->prepare($sql);
            $requete->execute();
            echo '<div class="alert alert-success" role="alert">Félicitations, vous avez créé votre évènement avec succès !</div>';

        }
        else {                                                                                              // Point 4
            echo '<div class="alert alert-danger role="alert">Erreur, cet évènement existe déjà (changez le nom) !</div>';
        }
    }

```

Point 1 : Si je reçois un Post de type insertion évènement

Point 2 : Je vérifie que ce nom d'évènement n'existe pas déjà

Point 3 : S'il n'existe pas j'insère l'évènement dans la base de données

Point 4 : Sinon j'affiche un message d'erreur.

Le fonctionnement est similaire pour la page de vote, cependant, la page de vote n'est pas encore totalement opérationnelle, car on ne peut pas vérifier que le joueur a bien voté sur le site rpg-paradize (Voir perspective), la fonctionnalité de vote n'est cependant pas sécurisée, car l'utilisateur peut « spammé » le bouton vote, sans que l'on puisse vérifier que le vote a bien été fait sur le site de vote sur lequel il a été redirigé.

i) Réécriture d'URL

```
Options -Indexes +FollowSymLinks
Order allow,deny
Allow from all

php_flag output_buffering on

RewriteEngine on
RewriteRule Boutique Vue/Boutique.php
RewriteRule Classement Vue/Classement.php
RewriteRule Event Vue/Event.php
RewriteRule Faction Vue/Faction.php
RewriteRule Nous-rejoindre Vue/Nous-rejoindre.php
RewriteRule Social Vue/Social.php
RewriteRule Vote Vue/Vote.php
RewriteRule Accueil index.php
RewriteRule styles Vue/styles.css
```

```
<li><a href="Nous-rejoindre">Nous-rejoindre</a></li>
```

J'ai insérer dans le projet une réécriture d'url, marchant simplement avec les href, ici par exemple si on demande dans le href la page « Nous-rejoindre » on nous redirigera avec le htaccess sur la page Vue/Nous-rejoindre.php

j) Triggers

```
delimiter //
create trigger MultipleTriggersGrade before insert on grade
for each row
begin
 if(new.nom_Grade IN (Select nom_Grade From grade)) then
  SIGNAL SQLSTATE '45000' set MESSAGE_TEXT = 'Ce grade existe deja';
 end if;
 if(new.prix_Grade < 0) then
  SIGNAL SQLSTATE '45000' set MESSAGE_TEXT = 'Erreur, vous ne pouvez pas mettre un prix negatif';
 end if;
end;
```

Voici un exemple de trigger de type MYSQL étant d'après moi nécessaire (les autres triggers sont à récupérer dans le fichier envoyé par mail).

J'ai créé des triggers adaptés à chacune des situations :

- Vérifier si on ne va pas insérer dans une des tables un tuple dont le nom existe déjà dans la table (c'est-à-dire insérer 2 fois un grade avec le même nom). Ceci semble trivial, mais le fonctionnement des pluggins que va utiliser le serveur Minecraft ne gèrent pas ceci.
- Ne pas insérer un prix, un nombre de vote, de point boutique négatif.
- Vérification sur les dates (date de création supérieure à la date de début d'un évènement, etc…)

III.2) Autocritique / Perspective de développement

Comme autocritique, je peux tout d'abord dire que j'aurais bien aimé même si je ne pense pas avoir eu assez de temps, travailler sur une architecture de type MVC, afin de faire un projet le plus propre possible, et offrant la possibilité d'être repris facilement par n'importe quel développeur Web.

Concernant le résultat obtenu concernant le projet, je peux dire que le cahier des charges que je me suis imposé pour le temps imparti (pour rappel : la mise en place de toutes les fonctionnalités sauf la boutique qui sera probablement gérée depuis l'application et la page de vote) a été correctement respecté. Le site n'est donc pas totalement abouti, et plusieurs mises à jour seront apportées avant le début des périodes scolaires pour déployer ce site en même temps que le serveur Minecraft, ces mises à jours seront donc :

- Redéveloppement de l'application avec une architecture MVC (mise à jour majeure)
- Rendre la page boutique fonctionnelle
- Améliorer la page de vote pour empêcher les utilisateurs de voter abusivement (sans aucune vérification) pour avoir leurs récompenses
- Permettre aux joueurs d'interagir entre faction (groupe auquel ils appartiennent) depuis la page faction

IV Rapport d'activité

IV.1) Outils utilisés

Pour mener à bien le déroulement de mon projet, j'ai utilisé certains outils qui m'ont servis dans différents domaine :

- Déploiement du site web : « Filezilla », soit une connexion FTP avec ma machine OVH
- Modélisation des diagrammes : « Modelio », « JMerise »
- Organisation du travail : « GanttProject »
- Test du site web en local : « Wamp/Lamp Serveur » (développement sur Windows ou Ubuntu)
- Création de base de données en ligne : Application PHPMYADMIN propre à OVH sur une base de données MySQL

IV.2) Planification

La planification a été la principale difficulté de ce projet, le respect de la Dead-Line est d'après moi un des points les plus importants à établir afin de prévoir ce qu'il est possible de faire dans le temps imparti.

Comme l'on peut le voir sur le diagramme de Gantt (figure 4 annexe), la première partie de mon projet s'oriente sur l'analyse de l'existant et bien entendu sur l'apprentissage des différents langages Web que je vais utiliser.

La seconde partie représente quant à elle la conception, c'est-à-dire la mise en place des différents diagrammes, de l'interface web, soit l'architecture du site web.

Par la suite vient la phase de développement des différentes fonctionnalités révélées dans le cahier des charges, que j'ai classé dans la partie Programmation sur le diagramme de Gantt.

Et enfin la rédaction du rapport qui quant à elle a été réalisée du début à la fin du projet afin d'éviter de perdre du temps.

V Conclusion

Pour conclure ce rapport sur le projet SeriousCraft, je peux affirmer que les objectifs que je m'étais fixés d'accomplir dans le cadre de Polytech ont en effet été réalisés. Toutes les fonctionnalités ont été implémentées, la possibilité de consulter le classement des joueurs suivant différents critères, les évènements qui vont se produire sur le serveur, une page de vote, une page faction et bien sûr la fonctionnalité de connexion sont toutes actuellement disponibles sur le site seriouscraft.fr.

Evidement comme dit précédemment, il y a encore beaucoup de travail avant de pouvoir proposer le site, il faudra par exemple terminer la page boutique qui n'est encore pas fonctionnelle.

Enfin je voudrais rajouter que ce projet a été très enrichissant autant sur le point technique, que sur le point personnel. Il m'a fallu gérer à la fois l'approfondissement d'un nouveau langage, la communication avec de réels clients, à qui je vais rendre ce projet quand je l'aurais totalement terminé, l'utilisation de nouveaux outils, et pour terminer, le point qui est d'après moi le plus important : le respect de la « Dead-Line ». Tous ces points m'ont donné une approche de ce que pourra être le travail dans une entreprise dans des périodes où la pression sera très importante.

Table des figures

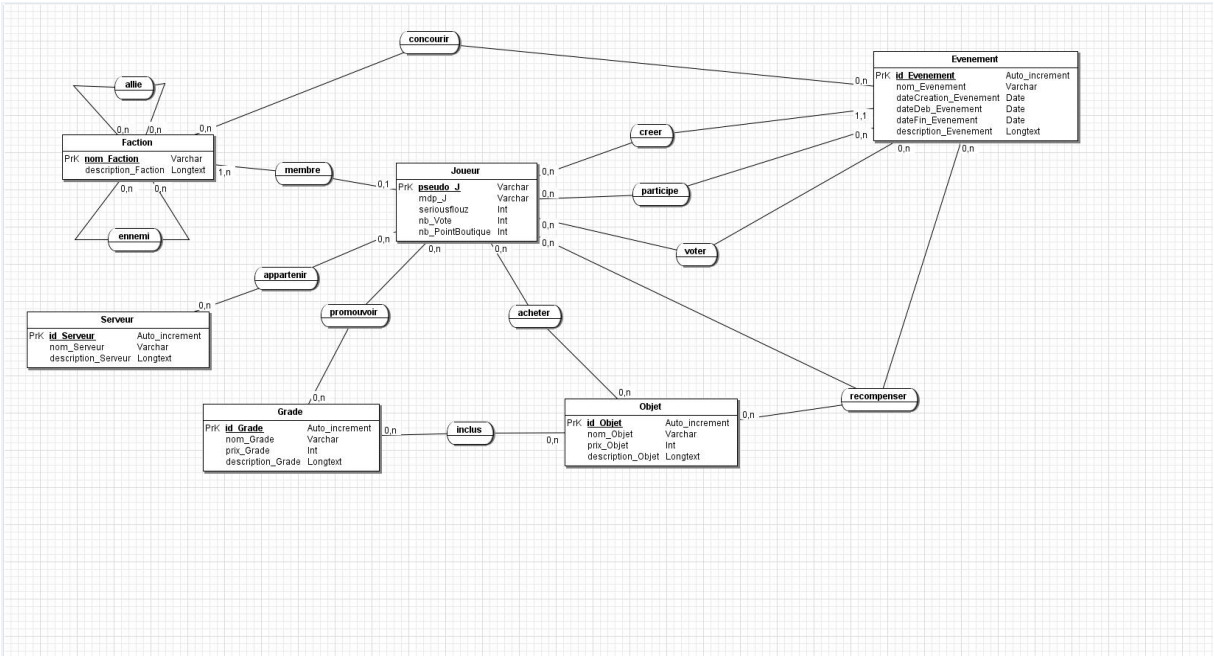

Figure 1: Modèle conceptuel de données

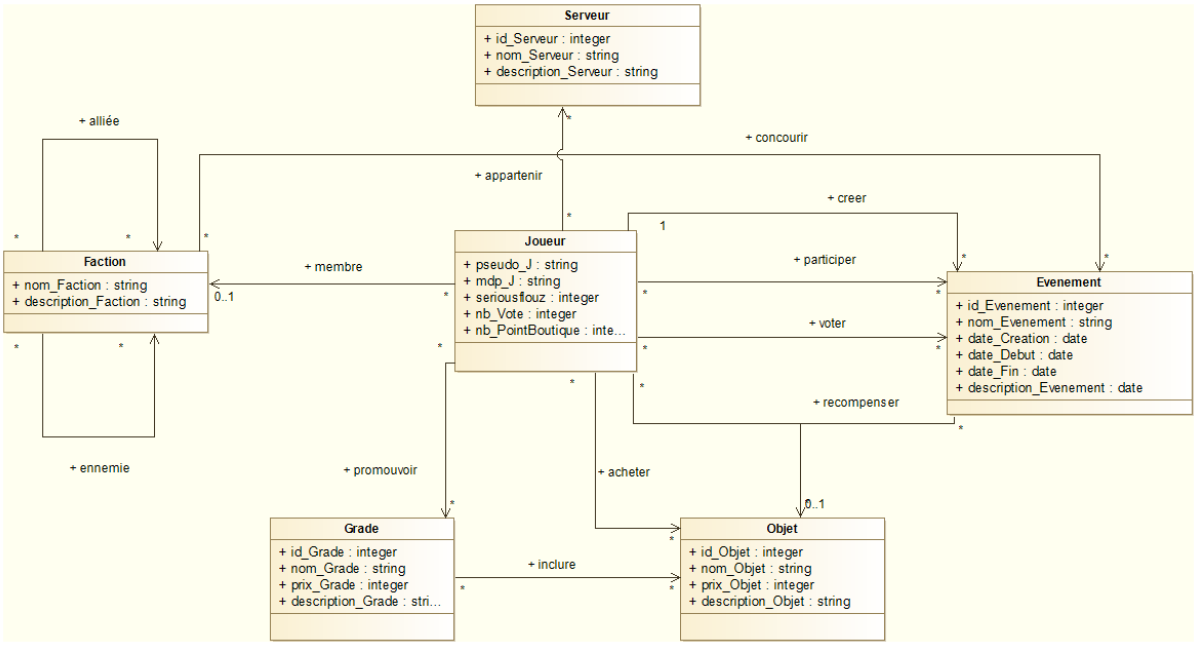

Figure 2 : Diagramme de classe

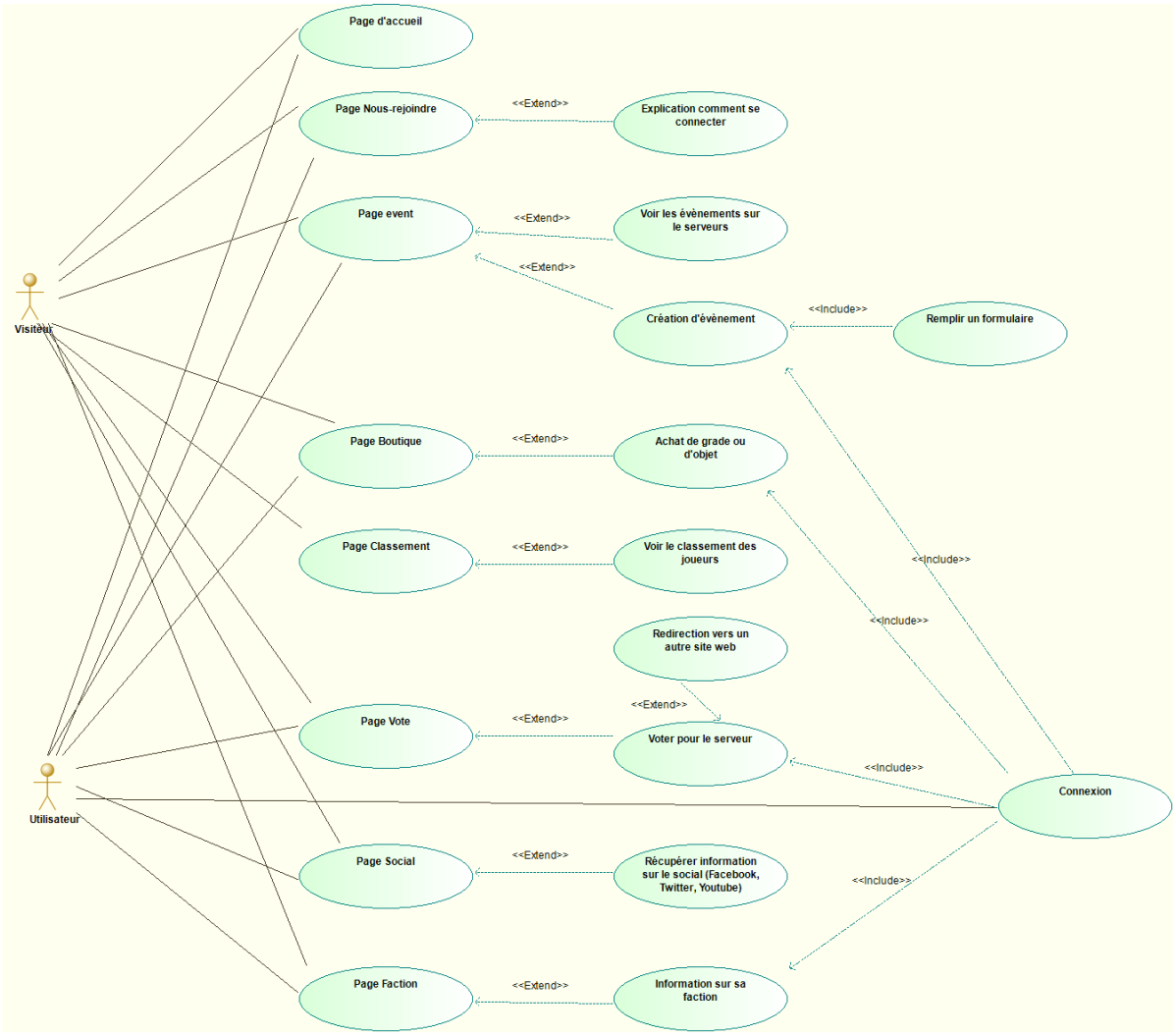

Figure 3 : Diagramme des fonctionnalités (USE - CASE)

Figure 4 : Diagramme de gantt