ECOLE PRATIQUE DES HAUTES ETUDES COMMERCIALES



Avenue du Ciseau, 15

1348 Louvain-la-Neuve

**Réalisation d’un site web d'administration de serveur de jeu et de gestion d'utilisateurs avec Spring boot et le Framework Angular**

Travail de fin d’études présenté en vue de l’obtention du diplôme de bachelier en Informatique et Systèmes orientation Technologie de l’Informatique

Goossens Jonathan



Rapporteur : VAN DORMAEL Louis

**Année Académique 2018 – 2019**

Table des matières

[**1.** **Introduction** 3](#_Toc17191925)

[**1.1.** **Présentation du client/projet** 3](#_Toc17191926)

[**1.2.** **Mise en situation** 3](#_Toc17191927)

[**1.3.** **Problématique** 3](#_Toc17191928)

[**1.4.** **Méthodologie** 4](#_Toc17191929)

[**2.** **Processus de candidature/recrutement** 5](#_Toc17191930)

[**2.1.** **Objectif** 5](#_Toc17191931)

[**2.2.** **Cahier des charges** 5](#_Toc17191932)

[**2.2.1.** **Etape 1 : s’inscrire** 5](#_Toc17191933)

[**2.2.2.** **Etape 2 : Déposer sa candidature** 6](#_Toc17191934)

[**2.2.3.** **Etape 3 : Valider une candidature** 6](#_Toc17191935)

[**2.2.4.** **Etape 4 : l’entretien oral** 6](#_Toc17191936)

[**2.3.** **Analyse fonctionnelle** 6](#_Toc17191937)

[**2.3.1.** **Inscription/Connexion** 6](#_Toc17191938)

[**2.3.2.** **Candidature Whitelist** 7](#_Toc17191939)

[**2.4.** **Solution implémentée** 8](#_Toc17191940)

[**2.4.1.** **Structure du code** 8](#_Toc17191941)

[**2.4.2.** **Design** 8](#_Toc17191942)

[**2.4.3.** **Etape 1** 8](#_Toc17191943)

[**2.4.4.** **Etape 2** 10](#_Toc17191944)

[**2.4.5.** **Etape 3** 11](#_Toc17191945)

[**2.5.** **Difficultés rencontrées** 12](#_Toc17191946)

[**2.6.** **Améliorations futures** 12](#_Toc17191947)

[**3.** **Communication et visibilité** 13](#_Toc17191948)

[**3.1.** **Objectifs** 13](#_Toc17191949)

[**3.2.** **Cahier des charges** 13](#_Toc17191950)

[**3.3.** **Analyse fonctionnelle** 13](#_Toc17191951)

[**3.4.** **Solution implémentée** 14](#_Toc17191952)

[**3.5.** **Difficultés rencontrées** 15](#_Toc17191953)

[**3.6.** **Améliorations futures** 15](#_Toc17191954)

[**4.** **Monitoring du serveur de jeu** 16](#_Toc17191955)

[**4.1.** **Objectifs** 16](#_Toc17191956)

[**4.2.** **Cahier des charges** 16](#_Toc17191957)

[**4.3.** **Analyse fonctionnelle** 16](#_Toc17191958)

[**4.3.1.** **Démarrage/Arrêt du serveur** 16](#_Toc17191959)

[**4.3.2.** **Commandes de manipulations en jeu** 16](#_Toc17191960)

[**4.3.3.** **Liste des utilisateurs et édition de grade** 17](#_Toc17191961)

[**4.4.** **Solution Implémentée** 17](#_Toc17191962)

[**4.4.1.** **Utilisation des Shell Script** 17](#_Toc17191963)

[**4.4.2.** **Démarrage/Arrêt du serveur** 18](#_Toc17191964)

[**4.4.3.** **Commandes de manipulations en jeu** 19](#_Toc17191965)

[**4.4.4.** **Liste des utilisateurs et édition de grade** 20](#_Toc17191966)

[**4.5.** **Difficultés rencontrées** 21](#_Toc17191967)

[**4.6.** **Améliorations futures** 21](#_Toc17191968)

[**5.** **Conclusion** 22](#_Toc17191969)

[**6.** **Bibliographie** 23](#_Toc17191970)

# **Introduction**

## **Présentation du client/projet**

Aldera-Life est une communauté qui alimente un serveur roleplay sur le jeu Grand Theft Auto 5. Ce serveur est basé sur la plateforme FiveM qui héberge le serveur et permet de s’y connecter. Grâce à cette plateforme, les différentes fonctionnalités du serveur peuvent être développées.

Le monde du roleplay sur GTA est très vaste et il est très difficile de se faire un nom au sein de cet univers… Il existe de nombreux serveurs et de nombreuses communautés qui dirigent ce monde… Il y a de nombreux moyens de se démarquer des autres et il est indispensable de les emprunter pour sortir du lot !

## **Mise en situation**

Chaque joueur qui décide de rejoindre la communauté d’Aldera-Life doit soumettre une candidature. Cette candidature doit contenir l’âge du joueur, son expérience dans le monde du roleplay (IRL, serveur, jeux, …), son background (histoire roleplay du personnage joué sur le serveur), l’avenir envisagé en RP, le nom et prénom du personnage roleplay et le Steam ID (essentiel car identifiant du personnage sur le serveur).

Toutes ces candidatures se concluent par un entretien oral avec des membres du staff. Une fois le candidat accepté par le staff, il est ajouté à la whitelist grâce à son Steam ID. Ce processus ne peut être réalisé que par un haut gradé du serveur grâce à une commande exécutable dans le jeu directement.

Différentes manipulations de monitoring du serveur sont réalisables uniquement grâce à un minimum de connaissances et de savoir-faire (redémarrage du serveur, arrêt du serveur, ajouter un joueur à un groupe, …).

## **Problématique**

Le premier problème vient du dépôt de candidatures des nouveaux joueurs voulant accéder au serveur… En effet, la procédure est telle qu’il faut contacter l’un des membres du staff responsable du recrutement pour présenter la candidature. C’est un processus ennuyeux et compliqué pour les membres du staff car il prend du temps, temps que chacun prend sur son temps libre.

Le deuxième problème provient de la visibilité du serveur et de la communauté. En effet, les joueurs vont et viennent de manière aléatoire. Il est donc très difficile de recruter ou de garder des joueurs dans un monde si vaste. Pour satisfaire une telle quantité de personnes, il faut se démarquer de la concurrence et c’est là qu’est tout l’enjeu, le cœur du problème…

La troisième difficulté réside dans la capacité de l’administration du serveur à monitorer celui-ci. En effet, toutes les manipulations servant à administrer le serveur de jeu sont assez techniques. Très peu de personnes sont capables de le faire. Si l’une d’entre elles n’est pas disponible, le serveur peut être dans une position délicate et les joueurs mécontents.

Tous ces problèmes regroupés démontrent qu’il y a une faille dans le fonctionnement de la communauté qui risque de mettre en péril la durabilité du projet. Si le projet meurt, il entraînera dans sa chute la communauté. Celle-ci se divisera pour partir vers de nouveaux projets plus attrayants. Il faut donc trouver de quoi donner une seconde vie et une nouvelle ambition à Aldera-Life. C’est grâce à un site web que tout cela pourra se faire.

## **Méthodologie**

Pour pouvoir développer le système le plus adapté aux besoins de la communauté et du staff, j’ai travaillé en étroite collaboration avec des membres impliqués dans la vie de cette communauté. C’est grâce aux informations recueillies par ces personnes que j’ai pu évaluer le travail à effectuer et que j’ai pu le développer.

Au fur et à mesure du développement de la plateforme web, je présentais les fonctionnalités aux différents membres fondateurs de ce projet afin qu’ils valident, corrigent et donnent des pistes d’améliorations à ce qui était réalisé.

Je ne me suis pas attardé à gérer mon projet sous forme de sprint ou de manière agile. Je n’ai pas réussi à me faire à ce type de gestion de projet en solitaire. En effet, j’ai essayé au début de m’y fier mais ça devenait un fouillis monstre et incompréhensible même pour moi qui le gérait et qui écrivait les tâches.

# **Processus de candidature/recrutement**

## **Objectif**

Comme énoncé précédemment, il est nécessaire de faciliter le processus de recrutement pour les membres du staff, ceux-ci ayant déjà énormément de travail à accomplir. Ce processus doit aussi être facile d’accès pour les utilisateurs voulant rejoindre la communauté.

L’objectif final est d’avoir un système presque complètement autonome afin de réduire le temps dépensé à l’acceptation de la candidature. Cette partie doit être un gain de temps pour les utilisateurs comme pour les membres du staff.

## **Cahier des charges**

Le processus de recrutement doit être facilité en gardant des étapes précises et faciles d’accès :

1. Le joueur doit d’abord se créer un compte. Cette première étape d’inscription doit permettre de recontacter le joueur et de l’identifier.
2. Une fois le processus d’inscription terminé, le joueur doit pouvoir se connecter et accéder au formulaire de candidature contenant les informations nécessaires au bon déroulement du recrutement. Cette demande devra être complète et valide.
3. Les demandes de recrutement doivent être stockées pour être lues et validées par l’un des membres du staff responsable de la partie recrutement. Une fois la demande validée ou invalidée, un mail est envoyé au joueur pour le prévenir.
4. Dès que cette partie est terminée, le joueur sera convoqué pour un entretien oral.

Ce système vise à permettre aux membres du staff de valider plusieurs demandes d’un seul coup et de préparer un planning pour les entretiens oraux plus précis.

### **Etape 1 : s’inscrire**

La procédure d’inscription est une phase importante dans le processus car elle permet d’avoir les premiers contacts avec l’utilisateur. Les informations importantes demandées à l’inscription sont :

* Le pseudo
* L’adresse mail

Une vérification de l’adresse mail est nécessaire pour pouvoir accéder aux étapes suivantes du processus de candidature. Cette procédure permet d’une part d’être sûr que le mail existe et d’autre part que nos demandes de whitelist ne soient pas encombrées de fake comptes. L’adresse mail est utilisée dans notre cas pour garder un contact avec la personne, lui dire si sa demande a été acceptée ou non et lui donner des plages horaires pour l’entretien oral.

### **Etape 2 : Déposer sa candidature**

Une fois l’étape 1 correctement remplie, l’utilisateur peut se connecter au site web afin d’accéder aux différentes fonctionnalités. Il doit avoir la possibilité de créer une demande de whitelist en communicant les informations importantes à cette demande :

* L’âge de la personne
* L’expérience du joueur dans le monde du roleplay
* Le Steam ID
* Le nom discord (exemple : Goostry#8461)
* Le nom et prénom RP qui sera utilisé par le joueur
* Un petit background (histoire du personnage incarné)
* L’avenir de ce joueur au sein du serveur (orientation)
* La lecture du règlement

Chaque champ doit être scrupuleusement rempli par l’utilisateur afin que la demande soit la plus complète possible. En cas de remplissage erroné, l’utilisateur ne doit pas pouvoir valider sa demande et donc le processus ne passera pas à l’étape 3.

### **Etape 3 : Valider une candidature**

Cette étape est la plus importante du processus, elle sert à valider ou non la candidature d’un utilisateur souhaitant rejoindre la communauté. Ce sont les membres du staff qui gèrent cette partie, ils doivent avoir accès à toute les demandes effectuées pour les accepter ou les refuser.

Lorsque qu’une demande est acceptée ou refusée, la demandeur doit être prévenu du refus ou de l’acceptation. Si refus il y a, l’utilisateur doit avoir la possibilité de reformuler une et une seule demande, ce sera sa dernière chance… Lors de l’acceptation de la demande, les tranches horaires de l’étape 4 sont transmises afin qu’ils puissent prendre leurs dispositions pour passer l’étape 4.

### **Etape 4 : l’entretien oral**

Cette étape n’a pas de rapport avec ce qui doit être développé et je ne vais donc pas l’expliquer ici. C’est juste un entretien sur un canal vocal.

## **Analyse fonctionnelle**

## **Inscription/Connexion**

Tout d’abord, le processus d’inscription est une démarche basique qui doit ressembler à de nombreux systèmes en place sur la plupart des sites Internet avec un processus similaire. Le pseudo de l’utilisateur sera la manière de l’identifier lors de la connexion, son adresse mail sera demandé ensuite pour permettre de confirmer l’identité de l’utilisateur via un mail de confirmation et le mot de passe qui doit être enregistré en base de données à l’aide d’une fonction de hachage.

Un email de confirmation doit être envoyé à l’utilisateur inscrit afin de pouvoir vérifier que l’adresse mail existe bien et qu’elle appartient bien à l’utilisateur qui vient de s’inscrire. Une fois l’email validé, l’utilisateur pourra accéder à toutes les fonctionnalités auxquelles son grade l’autorise à accéder. L’utilisateur pourra, une fois inscrit, se connecter à l’aide de son pseudo et de son mot de passe. Il peut aussi utiliser son adresse email pour remplacer le pseudo.

## **Candidature Whitelist**

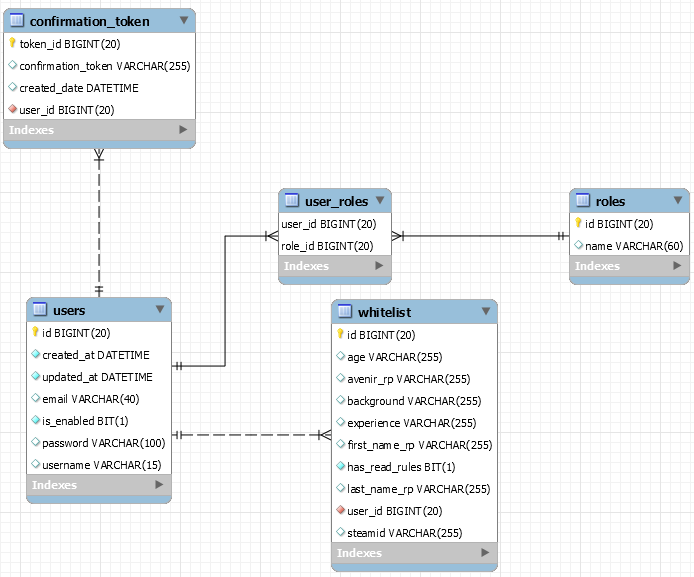
 Le processus de candidature doit être fait en 3 étapes importantes, la dernière n’étant pas utile dans cette analyse fonctionnelle. En effet, l’utilisateur doit tout d’abord avoir la possibilité de soumettre un formulaire afin de pouvoir espérer être accepté.

Figure 2.1 – Schéma de base de données 1

Le plus important dans cette procédure est la validation des candidatures par les membres du staff. Cette validation doit passer par une lecture minutieuse des informations données par le candidat ayant postulé. Une fois ces informations assimilées par le membre du staff, celui-ci doit avoir de quoi accepter ou refuser la demande… Un email doit être directement envoyé auprès du candidat afin qu’il soit informé de la réponse.

## **Solution implémentée**

### **Structure du code**

Tout le backend est entièrement programmé en Java à l’aide du Framework Spring Boot qui respecte le modèle de conception MVC (Modèle-Vue-Contrôleur). Chaque table de la base de données est représentée par un modèle dans le code. Chaque contrôleur représente un service REST qui permet de générer des données ou d’en créer.

### **Design**

Pour l’interface utilisateur, j’ai veillé le plus possible à répondre aux différents besoins de la communauté. La page de connexion/inscription est très simple et fonctionne sous forme de popup affichant tous les champs du formulaire. En ce qui concerne le formulaire de candidature, c’est un onglet qui reprendles différents types de champs de formulaire pour la demande de whitelist.

Le Framework Spring Boot donne la possibilité de travailler avec n’importe quel type d’interface graphique. J’ai donc choisi d’utiliser le Framework Angular pour sa vitesse et ses performances incomparables. Il permet de faciliter la génération de listes, de table, … Son fonctionnement en composants permet de facilement réutiliser ce qui a déjà été créé.

### **Etape 1**

La première étape fut celle qui a dirigé tout le reste du projet car elle englobait un grand nombre de fonctionnalités. Elle comprenait l’inscription et la connexion, la sécurité, la gestion de droits et l’envoi des emails…

Tout d’abord, il a fallu générer tout l’espace de travail en commençant par Maven et par les classes en rapport avec la gestion d’utilisateurs, les rôles et les utilisateurs. Cette partie permet de définir les premiers piliers de ce logiciel afin qu’il puisse être solide et bien réalisé.

Une fois les repository et les modèles finis, il a fallu s’attaquer à la sécurité du backend. Cette partie contient l’authentification grâce aux Json Web Token, la gestion de droits et la création des Controller qui permettent d’accéder aux différentes données d’inscription et de connexion.

Chaque requête REST est représentée dans le Controller et doit être autorisée pour n’importe quel utilisateur non encore connecté afin qu’il puisse s’inscrire et se connecter. A chaque connexion, un JWT est généré afin de garantir la connexion de l’utilisateur qui vient de se connecter, celui-ci est stocké dans le stockage local du navigateur. Il contient l’id de l’utilisateur, le rôle et la date de génération ainsi que celle d’expiration.

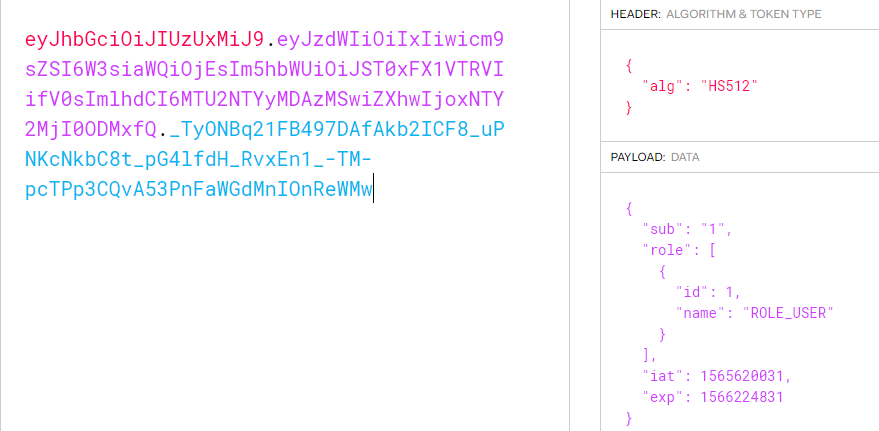


Figure 2.2 - Contenu d’un Json Web Token

Ensuite, il a fallu ajouter la vérification de l’adresse mail. Cette étape m’a demandé un peu de réflexion… Dois-je créer un service mail moi-même ? Cette question trouva vite sa réponse en réalisant des recherches sur l’utilisation des services mails avec Spring. La méthode la plus simple et pratique fut d’utiliser une dépendance existante qui utilise les services Gmail. Cela permet de créer uniquement un compte GMAIL et de l’utiliser comme mail de contact pour la vérification. A chaque création de compte, un token unique est généré afin de créer la page de vérification et donc valider le compte créé.

Une fois le backend terminé, la partie Angular a pu commencer… Il fallait avoir deux pages, une de connexion et une d’inscription interagissant avec le service REST de l’authentification. Cet affichage se devait d’être basique et facile d’utilisation… C’est pourquoi j’ai réutilisé du code que nous avions développé Dans le cadre du projet d’intégration. Ce code portait sur l’aspect esthétique du design. Les requêtes REST pouvaient ensuite être réalisées grâce au package http de Angular.

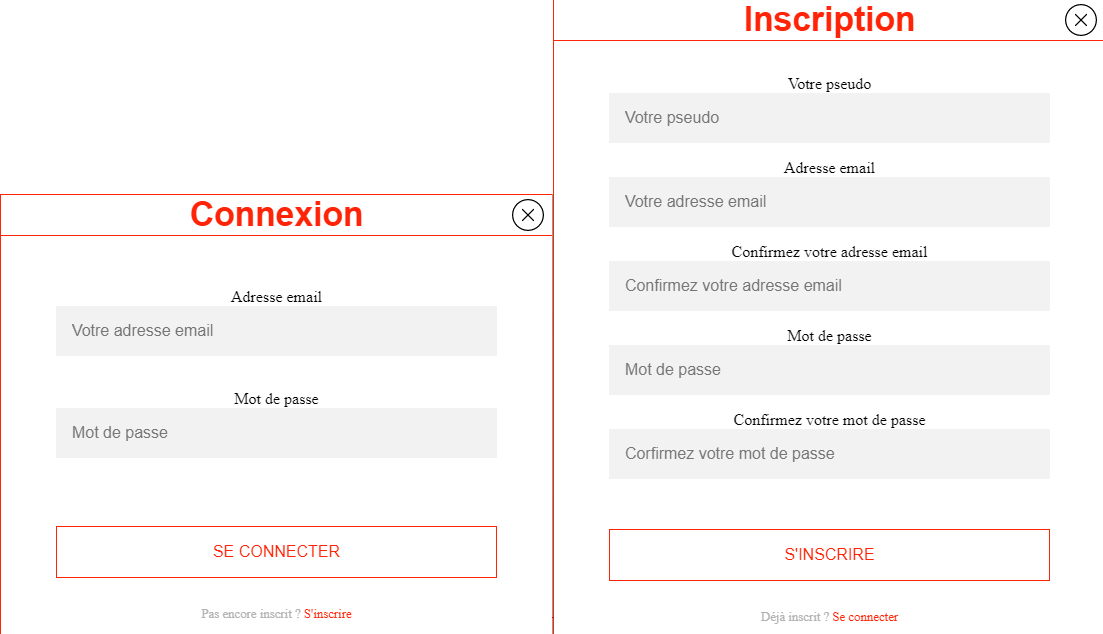


Figure 2.3 - Fenêtre de connexion et d’inscription

### **Etape 2**

La création de demande de whitelist est la seconde étape accessible aux utilisateurs du site. Elle doit être la plus compréhensible et donc la plus explicite possible afin que les utilisateurs puissent aisément faire leur demande.

Tout d’abord, il a fallu créer l’objet whitelist avec tous ses attributs qui seront mémorisés en base de données. Comme ils ont été choisis minutieusement précédemment et en accord avec les responsables du recrutement, ce fut assez simple. Une fois cette phase terminée, je me suis engagé dans la création du Controller et donc la création du service REST qui guidera les demandes de whitelist.

Dans ce Controller, il doit y avoir 3 appels REST : la publication des demandes de whitelist, l’affichage de la liste de toutes les demandes et le nombre de demandes faites par un joueur. Chaque demande doit avoir des permissions particulières en fonction du rôle de l’utilisateur. Seuls les ROLE\_USER peuvent publier une demande de whitelist, seuls les ROLE\_ADMIN et les ROLE\_MOD peuvent voir la liste des demandes. Pour le nombre de demandes enregistrées, il n’y a que l’utilisateur qui peut vérifier son nombre de demandes.

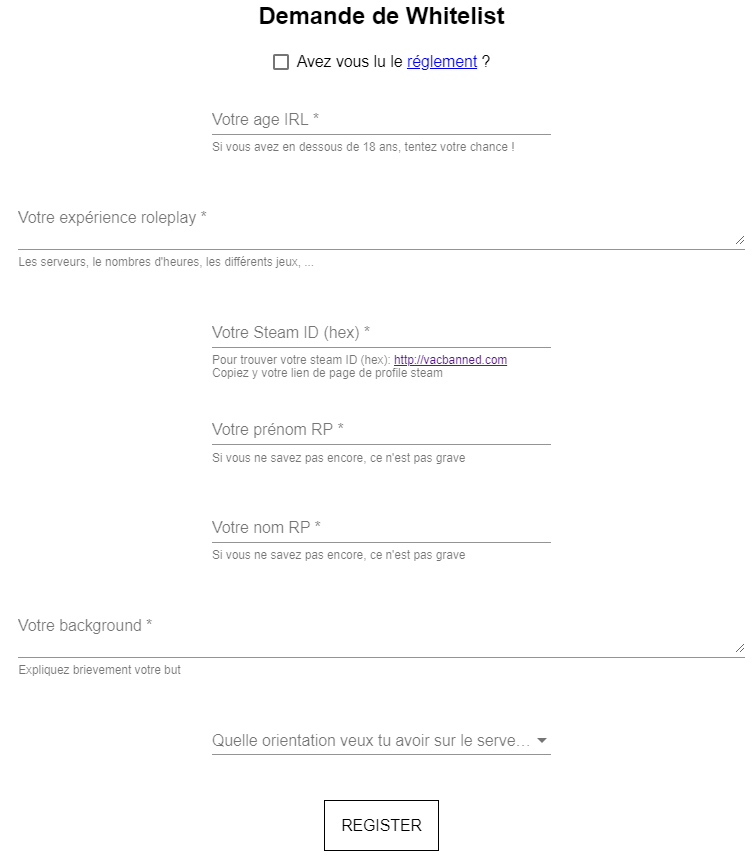


Figure 2.4 - Formulaire de demande de whitelist

Finalement, la partie la plus importante pour la compréhension de l’utilisateur fut le frontend. Cette partie a été réalisé à l’aide du package Material de Angular. Il permet d’ajouter des petits Tips et de donner un style plus sympathique aux différents champs du formulaire. Ensuite pour faire appel au service REST, il a suffi d’utiliser les appels du package http de Angular comme pour le système d’authentification. Une fois publié, le formulaire passe directement à l’étape 3.

### **Etape 3**

Cette étape est la dernière gérée par le site et donc du développement. Elle est cruciale dans le processus de candidature et déterminante pour les utilisateurs. Il est nécessaire que les membres du staff la prennent au sérieux et la réalisent au mieux.

Tout d’abord, il faut pouvoir récupérer la liste des demandes de whitelist en affichant toutes les informations de manière claire. Les informations importantes à récupérer sont toutes celles contenues dans la demande ainsi que le pseudo de l’utilisateur. Une fois cela récupéré, il faut pouvoir accepter ou refuser une demande afin que l’utilisateur puisse passer à l’étape 4.

Pour pouvoir faire une interface utilisateur simple, j’ai utilisé le package Material de Angular. Cela permet d’afficher les longs textes dans une banderole étendue grâce à ce package… En cliquant sur la ligne du tableau, la ligne s’étend et affiche les informations supplémentaires.

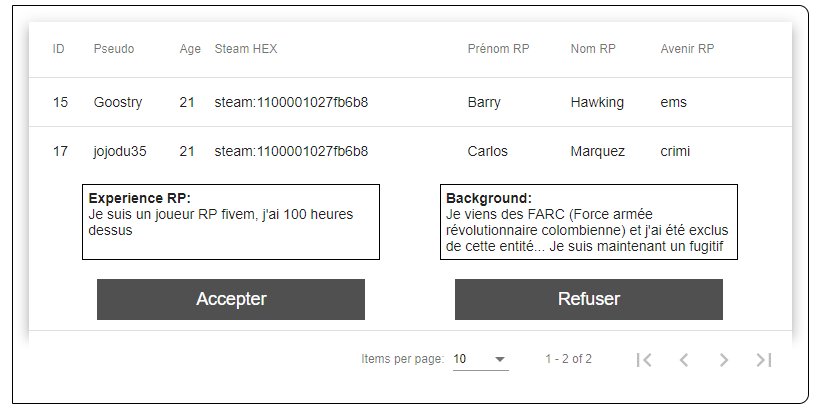


Figure 2.5 – Liste des demandes de Whitelist

## **Difficultés rencontrées**

Le premier gros challenge de cette partie du projet était la gestion de droits et donc la sécurité Spring Boot. Cette sécurité va suivre presque toutes les parties du projet, il était donc indispensable qu’elle soit bien réalisée. Il a fallu que j’explore de nombreuses sources pour comprendre de manière claire les différentes parties de la gestion de droit.

En effet, il y a de nombreuses manières d’établir les droits d’accès aux différents services REST. Que ce soit directement dans le Controller ou dans un fichier de configuration. J’ai donc choisi la méthode qui regroupe tous les droits, le fichier de configuration qui facilite la gestion des droits.

Ensuite, il y eut la partie sur les Json Web Token qui permet l’authentification de l’utilisateur pour accéder aux requêtes. Grâce à Spring, il est facile de générer les JWT mais j’ai eu besoin de me renseigner pour savoir comment les modifier et les manipuler de la meilleure manière possible. Ensuite, ce fut pour le stockage de ce JWT dans le navigateur que j’ai dû réfléchir… Je me suis renseigné sur la gestion de cookies avec Angular, rien n’était concluant. Je me suis alors tourné vers le stockage en local pour pouvoir faire les requêtes sécurisées.

Finalement, il y a eu l’affichage des formulaires et du tableau qui ont pris un peu plus de temps, car il a fallu adapter le style avec mon CSS… En effet, le package Material de Angular utilise Bootstrap pour ses styles tandis que j’utilise du CSS brut. Il a donc fallu que je m’adapte et que je cherche une solution pour la compatibilité.

## **Améliorations futures**

Pour ce qui est des améliorations futures, j’aimerais pouvoir instaurer un système de vote pour valider les candidatures. Cela signifie qu’il faudrait l’avis de deux ou trois membres du staff pour accepter ou refuser une candidature, ce qui permettrait d’être moins exclusif. Cela ajouterait énormément au processus de recrutement.

A l’avenir, il serait peut-être plus judicieux d’arriver à stocker les informations utiles (JWT) à l’intérieur de cookies plutôt que dans le stockage local du navigateur. Cela permettrait de vérifier plus facilement l’expiration du JWT qui correspondrait à la même expiration que le cookie, ainsi l’utilisateur serait directement déconnecté.

# **Communication et visibilité**

## **Objectifs**

Dans cette partie, l’essentiel va être de présenter le serveur de manière à attirer du monde et à ce que les utilisateurs comprennent les objectifs et la direction que la communauté va prendre.

Il faudra donc présenter tous les faits importants de la communauté, sa manière de fonctionner, les règles qui la régissent et les nouveautés qui arrivent. C’est grâce à ces éléments réunis que le projet pourra prospérer.

## **Cahier des charges**

Ce nouveau système à implémenter devra permettre aux utilisateurs d’en apprendre plus sur la communauté et son fonctionnement.

L’utilisateur doit pouvoir explorer les différentes fonctionnalités du serveur. Cette exploration doit lui donner envie de rejoindre la communauté et de s’investir dedans. La présentation doit donc être attractive et compréhensible. L’utilisateur doit pouvoir se projeter dans l’avenir en s’imaginant déjà où il sera lorsqu’il rejoindra le serveur et la communauté.

Il est important que les utilisateurs aient également la possibilité de naviguer à travers le règlement du serveur. Ce règlement doit être facile et accessible tant à la navigation qu’à la compréhension. Il faut pouvoir accéder à une partie en un seul clic.

Finalement, les dernières nouveautés et mises à jour du serveur doivent être publiées afin de garder les utilisateurs informés. Cela permet de les tenir en haleine afin qu’ils continuent de s’impliquer dans la communauté d’Aldera-Life.

## **Analyse fonctionnelle**

La présentation de la communauté doit se faire via un site vitrine afin de pouvoir démontrer toutes les fonctionnalités du serveur. Il faut y décrire les métiers qui s’exercent sur le serveur, les différentes activités disponibles pour se détendre, les activités criminelles et l’esprit du serveur en tant que tel.

Ensuite, un onglet permettant d’accéder aux différentes règles du serveur doit apparaître. Dans cette partie, un menu latéral permettrait d’arriver directement à la section demandée (comme une sorte de table des matières). Chaque règle du serveur doit être expliquée clairement et être accessible facilement.

Finalement, un dernier onglet permet d’avoir en avant-première les nouveautés du serveur et de la communauté. Les différents patchnotes y sont listés ainsi que les différentes annonces importantes faites par l’administration du serveur. Cela doit être affiché sous forme de publication et pouvoir être mis à jour par n’importe quel membre du staff.

## **Solution implémentée**

Comme expliqué plus haut, la page de présentation du serveur n’est qu’un simple site vitrine avec des informations statiques permettant de comprendre l’esprit du serveur ainsi que sa manière de fonctionner. Les informations présentées dans cette partie ont été collectées auprès des différents membres du staff. Il n’y a que de l’HTML et du CSS intégré dans Angular. Il y a le popup de connexion et d’inscription qui s’affiche au-dessus de la page de présentation.



Figure 3.1 – Page de présentation du serveur

Ensuite, comme le règlement était déjà exprimé dans le channel d’une application appelée discord, il m’a suffi de mettre en forme et de transposer le texte en HTML pour en faire une page en tant que telle et permettre une navigation simple grâce au Typescript de Angular pour naviguer de section en section par simple clic sur la table des matières. Tout cela a ensuite été mis en forme à l’aide de CSS pour le rendre plus agréable à lire et à comprendre.



Figure 3.2 - Page de règles du serveur

## **Difficultés rencontrées**

La seule difficulté rencontrées dans la réalisation de cette partie fut la mise en forme de la page de présentation pour qu’elle soit « responsive » et que les différentes images soient rendues dans les bonnes « grid » et positions. En effet, le CSS donne beaucoup de fil à retordre quand il s’agit de placer les éléments sur la page et de représenter les images comme il faut.

## **Améliorations futures**

Au niveau des améliorations à y faire, la dernière partie du cahier des charges n’a pas pu être réalisée… Je l’ai quand même décrite dans le cahier des charges afin d’expliquer de manière claire quel pouvait être le but de cette fonctionnalité.

Pour ce qui est de l’amélioration du site vitrine, il pourrait évoluer au fur et à mesure en ajoutant des informations et des fonctionnalités supplémentaires en fonction des patchnotes et des nouveautés du serveur. Un serveur RP étant en constante évolution, il y aura toujours des choses à y rajouter et à faire évoluer pour qu’il soit toujours plus attrayant.

# **Monitoring du serveur de jeu**

## **Objectifs**

Le monitoring et la gestion du serveur de jeu ont besoin d’un upgrade pour être prise en charge par un plus grand nombre de personne. De ce fait, il est nécessaire de créer des tâches permettant à l’administration de s’occuper du serveur de jeu.

Différentes tâches doivent être réalisables telles que faire des annonces en jeu, vérifier le démarrage du serveur, le démarrer ou l’arrêter ou encore le redémarrer, gérer les utilisateurs inscrits sur le site, …

## **Cahier des charges**

Ce cahier des charges va donc lister les différents éléments qui doivent être développés afin d’aider l’administration à gérer son serveur. Il faut pouvoir :

1. Démarrer et éteindre le serveur de manière simple
2. Effectuer des annonces en jeu
3. Afficher une liste des joueurs en ligne sur le serveur
4. Avoir la liste des personnes inscrites sur le site avec possibilité de modifications
5. Ajouter un joueur à la whitelist et recharger celle-ci en jeu directement
6. Ajouter un groupe à un utilisateur en jeu

Grâce à cela, le serveur pourra avoir un meilleur uptime et le management de celui-ci sera optimisé.

## **Analyse fonctionnelle**

### **Démarrage/Arrêt du serveur**

Le serveur ne pouvait être redémarré et managé que par un petit nombre de personnes qui avaient les connaissances et les compétences techniques nécessaires. Il est donc nécessaire de faciliter ce travail grâce au site web qui va automatiser la méthode juste en cliquant sur un bouton.

Pour pouvoir démarrer le serveur, l’arrêter ou le redémarrer, il faut effectuer des commandes sur une console linux au travers un screen linux. Pour pouvoir effectuer ces commandes via le site web, il faut créer des shell script qui permettent de faire les manipulations nécessaires.

### **Commandes de manipulations en jeu**

Pour ce qui est des commandes, il faut qu’elles puissent être faites même si le membre de l’administration n’a pas la capacité de se connecter en jeu. Les interactions avec les joueurs en seront facilitées de même que la gestion de celui-ci.

Ces commandes sont principalement réalisées à partir de la console. Les annonces faites aux joueurs demandent de se munir à nouveau d’un shell script prenant en argument le message à envoyer.

En jeu, il y a des rôles permettant de gérer les droits des utilisateurs. Il existe une commande qui permet de donner ce rôle. Il faut donc un shell script qui permet de choisir quel utilisateur doit avoir le grade et quel grade lui donner.

Un système de whitelist autorise seulement certains joueurs à se connecter. Pour ajouter un joueur, il faut son Steam ID. Celui-ci doit être sous forme hexadécimale et doit ressembler à ceci : « **steam:11000010b78bce8 ».** A nouveau au travers d’un shell script qui prendra comme argument le Steam ID.

Pour finir, afin que la whitelist prenne bien en compte les derniers utilisateurs qui ont été ajoutés, il faut recharger celle-ci à l’aide d’une commande… Comme pour les autres, un shell script sera créé afin d’exécuter cette commande dans la console du serveur.

### **Liste des utilisateurs et édition de grade**

Pour pouvoir ajouter un rôle à l’un des utilisateurs du site, il faut pouvoir afficher la liste de ceux-ci. Une fois cette liste affichée, il faut avoir la possibilité de changer le rôle de l’utilisateur afin qu’il puisse accéder aux parties du sites.

Les personnes qui peuvent accéder à cette partie ne peuvent être que les membres de l’administration et donc ces requêtes ne peuvent être faites que par ces personnes-là.

## **Solution Implémentée**

### **Utilisation des Shell Script**

Pour toutes les manipulations qui se font sur le serveur de jeu directement, il faut se munir de Shell script et de commandes directement dans une console Linux. Pour cela, il y a des packages Spring qui permettent d’exécuter ce genre de commande sur le VPS qui host le Backend.

Pour chaque Shell script créé, il faudra créer un appel REST qui permettra d’y accéder depuis le site WEB et donc depuis le code Angular juste par un simple Controller. Cela permet de bien séparer la vue du model et des Controller.

### **Démarrage/Arrêt du serveur**

Pour commencer, il a fallu regarder et imaginer de quelle manière allait être réalisés les différents Shell script et les appels REST par après. Pour arrêter le serveur, c’est très simple, il suffit de faire un CTRL-C ou de fermer le screen qui héberge le serveur. Il ne faut donc pas de Shell script à proprement parlé pour cela vu qu’il n’y a qu’une seule commande. Elle peut donc être simplement exécutée dans un appel REST.

Pour ce qui est du démarrage du serveur, il y a une série de commandes à exécuter afin de créer un nouveau screen et de démarrer la console du serveur dans celui-ci. Ce script existait déjà à l’origine, je lui ai juste rajouté la vérification qu’il n’y a pas déjà un screen ouvert à ce nom-là. Et ensuite, j’ai créé l’appel REST qui permet de démarrer le serveur.

Une fois la partie backend réalisée, à nouveau il a fallu concevoir une page sobre mais complète qui permet de faire appels à ces Controller. Cette page ne peut être accessible qu’aux membres possédant le grade ROLE\_ADMIN. Les appels REST ne sont faisables que par ces derniers.

Ensuite, il a fallu créer un bouton permettant de démarrer le serveur, un autre pour l’arrêter et une liste déroulante affichant les serveurs à démarrer ou arrêter. Une petite puce verte ou rouge doit être allumée à côté du nom du serveur pour afficher s’il est ouvert ou non.

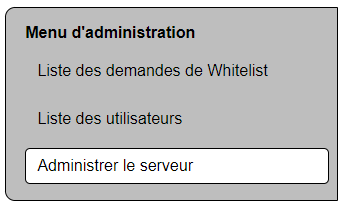


Figure 4.1 – Menu de navigation au travers l’administration

### **Commandes de manipulations en jeu**

Pour chacune des commandes effectuées en jeu, il y a un ou plusieurs arguments à placer. Pour ce qui est de la commande qui permet de faire une annonce en jeu, il y a juste une commande à effectuer sur la console du serveur avec le texte que l’on veut afficher.

Pour cela, il a suffi de créer un service REST qui demande uniquement un string en input pour annoncer le nécessaire et qui effectue la commande dans la console du screen du serveur de jeu.

Pour ce qui est de l’affichage sur le site web, un simple input permet de mettre le texte nécessaire à l’annonce sur le serveur et un bouton lance l’appel REST grâce au package http de Angular. Une fois l’annonce envoyée, elle est affichée dans le jeu.

Pour les autres manipulations/commandes qui nécessitent un ou plusieurs arguments, il suffi de faire pareil et de mettre un ou plusieurs inputs en fonction de la commande à réaliser. Par exemple, pour la commande qui permet d’ajouter un groupe à un joueur, il faut lui passer en argument l’id en jeu du joueur et le groupe dans lequel on veut l’ajouter.

****

Figure 4.2 – Page de monitoring du serveur

Pour chaque requête, il y a un bouton qui permet d’envoyer l’appel REST et d’effectuer la requête nécessaire. Si tout s’est bien déroulé, l’appel REST doit renvoyer un code 201 avec un texte qui dit qu’elle a bien été effectuée.

### **Liste des utilisateurs et édition de grade**

Afin de pouvoir ajouter les grades et voir qui est inscrit sur le site internet, il faut pouvoir avoir accès à la liste des personnes inscrites et avoir les informations sur ces personnes (pseudo, email et grade). Un service REST permet d’accéder à cette liste. Ensuite, il faut un autre service REST permettant de modifier le rôle des différents utilisateurs. Cette fonctionnalité ne doit être faisable que par les ROLE\_ADMIN.

La liste d’utilisateurs est affichée sur le site WEB dans un tableau du package Material de Angular. Pour pouvoir modifier le grade des utilisateurs, il faut d’abord sélectionner l’utilisateur en cliquant sur la ligne qui le concerne. Une fois la sélection faite, une liste déroulante apparait permettant de choisir le grade à appliquer sur l’utilisateur. Pour finir l’action de modification, un bouton en bas de la page permet de faire appel au service REST de modification de grade.

Cette partie du site n’est accessible qu’aux administrateurs via le menu latéral de navigation au sein de l’onglet d’administration du site web et du serveur de jeu. Cela permet une navigation aisée et rapide entre les différentes fonctionnalités.

****

Figure 4.3 – Page de modification des rôles

## **Difficultés rencontrées**

La plus grande difficulté rencontrée durant la réalisation de cette partie fut la partie script devant interagir avec une entité séparée du logiciel d’origine. Ces scripts doivent interagir avec un screen distant du backend et ont donc posé des problèmes. Il faut pouvoir faire appel à ces scripts et les exécuter à l’endroit adéquat pour que la fonction s’exécute correctement. Heureusement, j’ai pu trouver les commandes java qui permettent d’interagir avec le Runtime.

## **Améliorations futures**

A l’avenir, il pourrait être intéressant de créer des fonctionnalités supplémentaires permettant d’interagir avec le serveur de jeu. Elles pourraient venir au fur et à mesure des développements du jeu, vu que les besoins du serveur augmenteraient petit à petit. Il pourrait y avoir, par exemple, une fonction de ban qui permettrait d’exclure quelqu’un du serveur.

# **Conclusion**

Je rappelle que l’objectif principal de ce travail était de trouver une solution qui facilite l’administration d’un serveur de jeu et qui permette un recrutement plus aisé. J’ai donc travaillé main dans la main avec les différentes personnes responsables de ce serveur pour répondre au mieux à leurs besoins. Toutes les solutions que j’ai développées ont été expliquées dans ce document.

Le site que j’ai développé a été présenté à l’administration du serveur et ils en sont satisfaits, ils sont même très enthousiastes à l’idée de la mettre en place et de la partager à leur communauté. Je continuerai à collaborer avec eux en vue de continuer à améliorer ce système. Il est disponible à la page : <https://aldera-life.fr/>

Malgré les nombreuses améliorations possibles, je pense qu’au vu du retour du client et des solutions développées, on peut dire que ma mission a été accomplie et que le problème est résolu.

Je pense avoir acquis de nombreuses compétences techniques grâce à ce projet, elles me suivront tout au long de mon parcours professionnel. Cette évolution s’est faite tout au long de mon travail de fin d’étude et elle me permettra de ne plus reproduire les mêmes erreurs ainsi que de respecter les timings donner à l’origine.

Le code source est disponible sur la page GitHub : <https://github.com/JonGoo/AlderaLife>. Il sera mis à jour au fur et à mesure des améliorations faites.

# **Bibliographie**

Guest Contributor. Spring Security: Email Verification Registration. [online] stackabuse.

Available at: <https://stackabuse.com/spring-security-email-verification-registration/>

Le Tutoriel de Spring Email. [online] 07planning.

Available at: <https://o7planning.org/fr/11145/tutoriel-spring-email>

Guide to Spring Email. [online] Baeldung.

Available at: <https://www.baeldung.com/spring-email>

Créer une application de connexion avec Spring Boot, Spring Security, JPA. [online] 07planning. Available at: <https://o7planning.org/fr/11705/creer-une-application-de-connexion-avec-spring-boot-spring-security-jpa>

Spring Boot + Spring Security + JWT + MySQL + React Full Stack Polling app. [online] CalliCoder. Available at: <https://www.callicoder.com/spring-boot-spring-security-jwt-mysql-react-app-part-1/>

Angular Security - Authentication With JWT: The Complete Guide. [online] angular university. Available at: <https://blog.angular-university.io/angular-jwt-authentication/>

Austin (2017). Authentication in Angular & JWT. [online] Medium.

Available at: <https://medium.com/@amcdnl/authentication-in-angular-jwt-c1067495c5e0>

Angular. [online] angular.io.

Available at: <https://angular.io/>

JSON Web Tokens. [online] jwt.io.

Available at: <https://jwt.io/>

Spring Security - Roles and Privileges. [online] Baeldung.

Available at: <https://www.baeldung.com/role-and-privilege-for-spring-security-registration>

JPA / Hibernate One to Many Mapping Example with Spring Boot. [online] CalliCoder.

Available at: <https://www.callicoder.com/hibernate-spring-boot-jpa-one-to-many-mapping-example/>

Angular Material. [online] angular.io.

Available at: <https://material.angular.io>

How to Run a Shell Command in Java. [online] Baeldung.

Available at: <https://www.baeldung.com/run-shell-command-in-java>

Introduction to Spring Method Security. [online] Baeldung.

Available at: <https://www.baeldung.com/spring-security-method-security>

Using Spring @Value with Defaults. [online] Baeldung.

Available at: <https://www.baeldung.com/spring-value-defaults>

How to execute shell command from Java. [online] Mkyong.com.

Available at: <https://www.mkyong.com/java/how-to-execute-shell-command-from-java/>

How to use gmail from terminal (Linux). [online] EmbedJournal.

Available at: <https://embedjournal.com/how-to-use-gmail-from-terminal-linux/>