Direction des Technologies de l'information et de la Communication, Université du Tours
Bat D, 60 rue du Plat d'Etain
37020 TOURS CEDEX 1, FRANCE
Tél. +33 (0)2 47 36 73 34
www.polytech.univ-tours.fr

DTIC - Pôle Infrastructure

Cahier de spécification système & plan de développement			
Projet :	**développement d'une plateforme d'automatisation de serveurs**		
Émetteur :	CHEN Jing	Coordonnées : DI5	
Date d'émission :	30 juin 2015		
Validation			
Nom	Date	Valide (O/N)	Commentaires

Historique des modifications		
Version	Date	Description de la modification

00 : 05/2015 ; Version initiale : synthèse de différents documents

Table des matières

Cahier de spécification système

1.1 Introduction

Ce document est un cahier de spécification fonctionnelle. Il a pour but de définir clairement toutes les demandes du client. Il permettra, lors de la validation finale, de vérifier que le livrable est conforme à ce qui avait été demandé.

Ce projet dans le cadre est un projet de fin de stage dans le cadre de mon cursus à l'école Polytech Tours. Ce projet est proposé par le Pôle Infrastructure de la Direction des Technologies de l'information et de la Communication. Il souhaite consolider l'administration de ses serveurs et développer une plate forme en début. Pour ce projet, mon encadrant de projet est Laurent Beunerche qui définit les grands axes de la réalisation de ce projet, ainsi que les tâches de développement des différentes fonctionnalités.

1.2 Contexte de la réalisation

1.2.1 Contexte

L'automatisation de l'intégralité de la chaîne de production informatique est l'une des techniques permettant aux administrateurs systèmes d'être plus efficaces, d'augmenter la qualité, la performance information. Quant à configuration une série de services et de serveurs, au lie de configurer les services et les machines un par un à main, on utilisera un outil de configuration automatique qui est plus puissant, plus efficace et moins de main d'œuvre. La supervision fait partie de la chaîne de production information et doit être aussi automatisée au maximum. Elle permet de se faciliter le travail et de s'assurer de la maintenance.

Une des solutions est d'automatiser la mise en supervision des équipements informatiques, préalablement déclarés dans un outil d'inventaire. Une autre solution est de se servir de l'outil de gestion de configuration pour installer automatiquement les agents de supervision puis de déclarer les nouveaux éléments dans l'outil de supervision. C'est ce que on va voir dans ce projet.

1.2.2 Objectifs

Le but de ce stage est de développer un système de déploiement de configuration à partir d'une base de données de cartographie application (à créer, appelé « **UbiQube** ») ainsi que des éléments du S.I. existants (ldap, etc.). Celui-ci les applications(appelée App à la suivant) sont un ensemble de services sous la forme d'un document description qui contient tous les les informations de déploiement (on va le montre dans les maquettes du site web, la partie 1.5.). Les livrables attendus à la fin du stage sont :

- la modélisation, le développement et l'intégration de la base de données représentant la cartographie.
- la conception et développement d'une interface d'administration (Ruby on rails).
- la conception et le développement des API de déploiement vers les serveurs (Ansible).
- la documentation associée à ces différents éléments.

Le périmètre des déploiements des configurations :

- la gestion des droits utilisateurs sur les serveurs.
- la configuration des firewalls.

1.2.3 L'état de l'art

Pour le moment, nous avons un framework simple du site web(**Ubiqube**) (l'adresse du site [1])et une machine Unix avec l'environnement installé(ruby, rails, unicorn, Nignx)(le source sur Github [2])

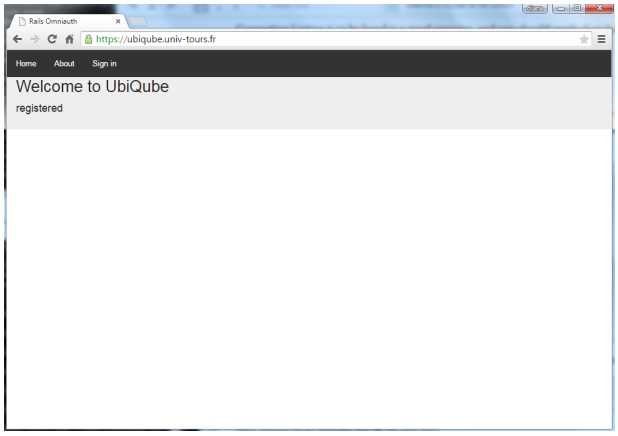

FIGURE 1.1 – Page Home

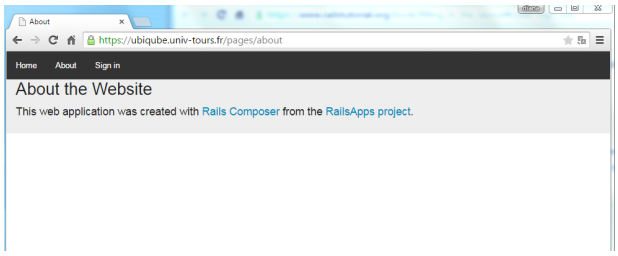

FIGURE 1.2 – Page About

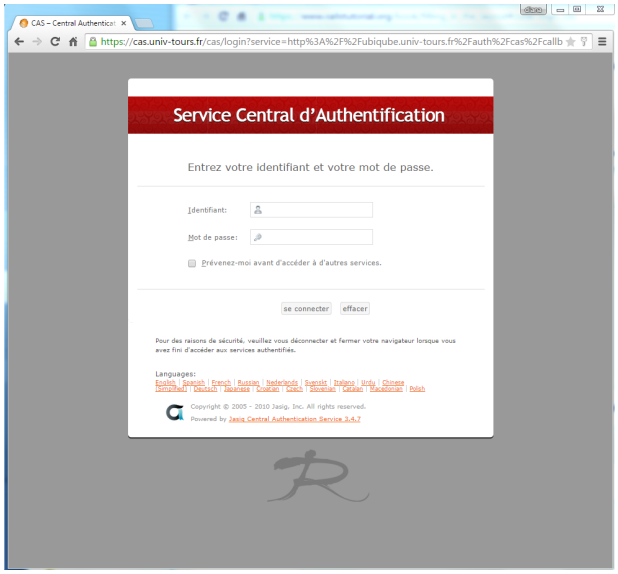

FIGURE 1.3 – CAS Login

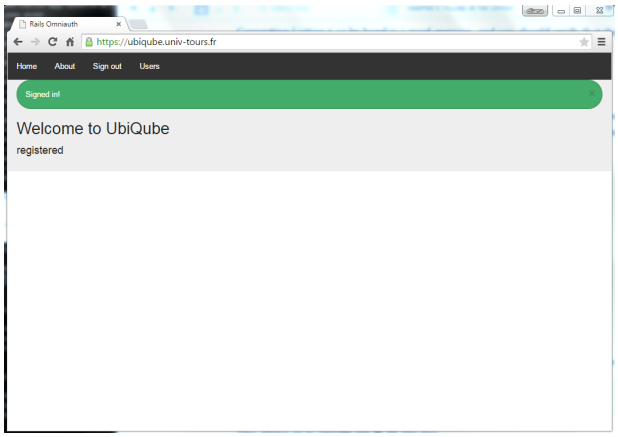

FIGURE 1.4 – Page succès Login

1.2.4 Hypothèse

Parmi toutes les créations à apporter à cette plate-forme, la principale est d'établir un cadre de gestion des App et de réaliser la supervision de déploiement. La partie inconnue réside dans la possibilité à affichage la cartographie dynamique de supervision de VMs. Si par manque de temps, on ne va pas faire celui-ci. Aussi ne pas améliorer l'apparence du site et ajouter pas trop de détailles(ou les effets dynamiques) pour améliorer l'expérience d'utilisateur. On réalisera que les tâches importantes (priorité forte). Mais à la fin le projet faut être une plate-forme fonctionnelle, utilisable et complète.

1.2.5 Bases méthodologiques

Le projet sera développé en framework Ruby on Rails sous Linux. Donc HTML et javascript du développement web et des commandes du linux pour administration système sont aussi utilisé dans le projet. Il est indispensable de respecter des règles de programmation :

— Faire des commentaires pour les classes, les vues et les fonctions.

— Rédiger le guide de développeur pour l'expliquer l'architecture globale de l'application, de l'installation et de l'utilisation certains outils.
— La version du code est géré par Git.

1.3 Description générale

1.3.1 Environnement du projet

La mise en œuvre d'une plateforme d'automatisation de configuration de serveurs est constitué d'une part d'un IHM web de gestion le fichier description d'APP sous Ruby on Rails et d'autre part d'un API qui configure les playbooks de déploiement pour Ansible.

L'environnement de ce projet tourne sur un serveur Linux avec Nignx, Ruby, Rails et MySQL installés. Ce serveur est accessible depuis le DTIC, à la fois sur le réseau filaire. Les matériaux supplémentaires comme le service d'identification LDAP de l'université Tours et l'outil d'automatisation Ansible.

Voici ci-dessous un diagramme de déploiement résumant les interactions entre ces éléments pour chacun de ces rôles :

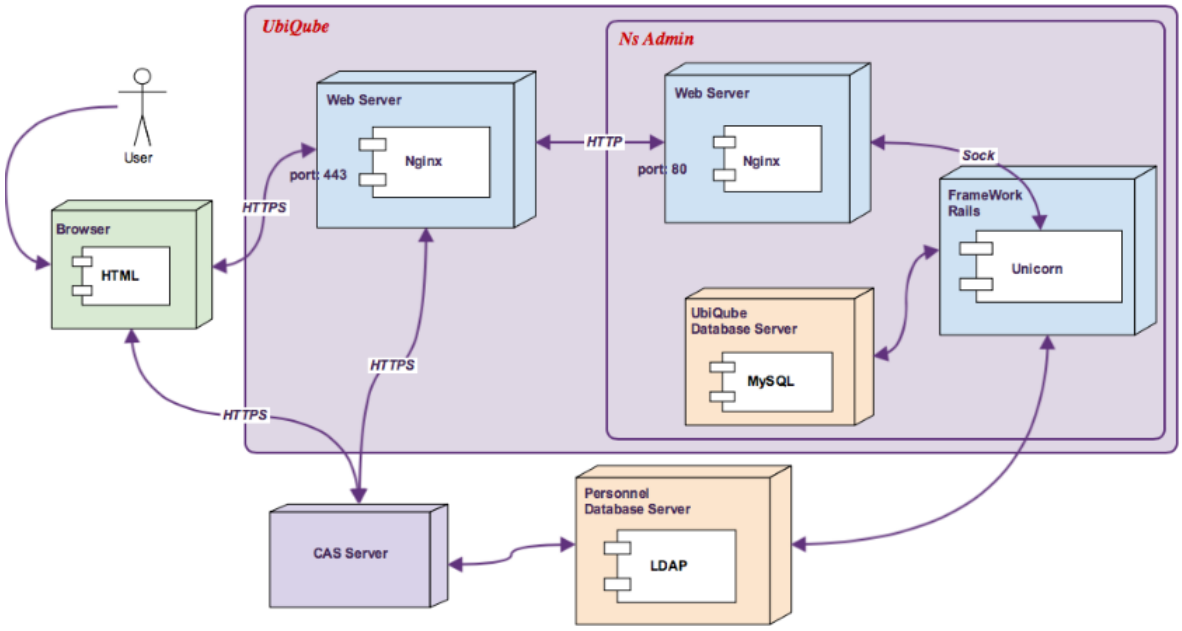

FIGURE 1.5 – Le diagramme déploiement

1.3.2 Caractéristiques des utilisateurs

Parce que ce système est utilisé dans l'entreprise intérieur donc actuellement les utilisateurs sont des personnels du DTIC. Mais il y a 4 groupes principaux. On détaille dans le tableau suivant tab.1.2

Utilisateur		Connaissance de l'informatique	Expérience de l'application	Droits d'accès utilisateurs.
User	user ordinaire	pas nécessaire	pas nécessaire	gestion son compte(s'inscrire, déconnecter, éditer l'info de contact), lecture le fichier de APP, créer une nouvelle APP, envoie la nouvelle APP à valider.
	Responsable Technique	bien connaît l'informatique	sait bien certain fonctionnalités	héritage tous les droits d'utilisateur ordinaire, définir les matériaux de APP(VMs, BDD, serveur etc.), supervision les VMs.
	Responsable Fonctionnel	connaît l'informatique et réseau	sait bien certain fonctionnalités	héritage tous les droits d'utilisateur ordinaire, assigner responsable technique.
Administrateur		connaissance approfondie de l'informatique et système	bien connaît et il peut paramétrer l'application	accès dans tous les fichiers de APP et peut tout modifier, gestion d'utilisateur, gestion LDAP(créer les filtres etc.), paramétrage **UbiQube**. Il a la plus haute autorité.
Direction des systèmes d'information (DSI)		pas nécessaire	sait certain fonctionnalités	valider, refuser ou archiver la nouvelle APP, gestion le groupe de personnel.

TABLE 1.1 – Caractéristique des utilisateur

Voici ci-dessous un diagramme de cas d'utilisation résumant les actions possibles pour chacun de ces rôles fig.1.6 :

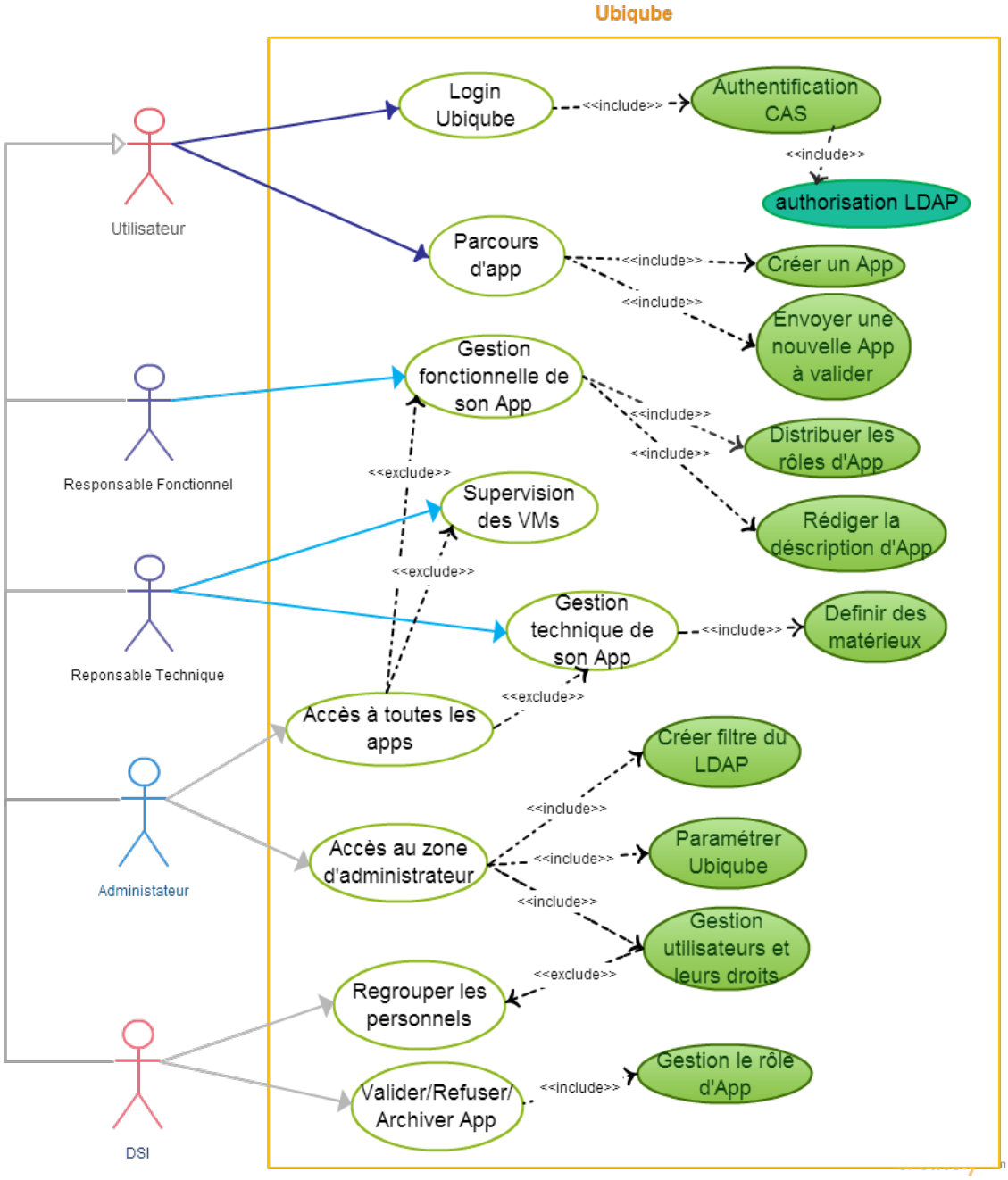

FIGURE 1.6 – Diagramme d'utilisation

1.3.3 Fonctionnalités et structure générale du système

Le framework Ruby on Rails utilise une architecture MVC (Modèle-Vue-Contrôleur) pour le web :

- **Modèle** : Il contient un ensemble de classes Ruby comme User, Session, Application etc. Il fournit aussi une interface de sauvegarder et interroger la base de données.
- **Vue** : View est la page HTML qui est remplie des données traversant contrôleur. Les zones dynamiques sont effectués par Javascript.
- **Contrôleur** : le Controlller reçoit le résultat que le Routage analyse, détermine laquelle contrôleur va être appliquée et mettre le contrôleur à exécuter certain action. C'est-à-dire, un « Http Request » est envoi par l'utilisateur(Browser) à « Routing ». Un fichier « route.rb »(Routes) assigne certain

« Action Controller ». Après l'action du contrôleur va être appelée. En le même temps, le contrôleur échange des données avec Model et View.

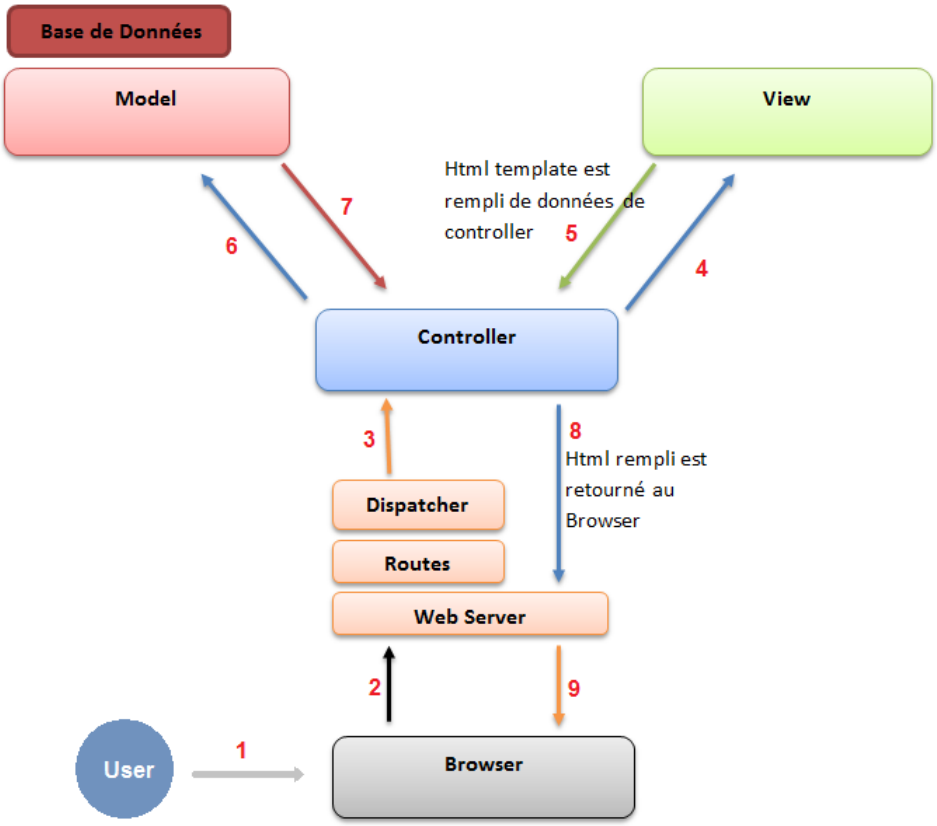

FIGURE 1.7 – L'architecture MVC du Ruby on Rails

Sur le schéma fig.1.7, il y a un cycle request/response d'utilisateur qui tracer deux flux via application. La web application basée sur RoR permet de soumettre et sauvegarder des données dans Database et de parcourir et modifier les données. Il y a deux tours dans ce cycle.(le numéro dans la liste suivent présente le numéro de flèche sur le schéma en dessus. Le processus du cycle écrit dans l'ordre.)

Le premier cycle est l'affichage la page pour l'utilisateur.

1. L'utilisateur ouvert son navigateur, enter l'url et appuie Entrée.
2. Quand l'utilisateur appuie Entrée, le navigateur envoie un GET request pour l'url.
3. Le GET request arrive Rails routage (config/routes.rb). Le routage mappe l'url au "Controller Action".
4. L'action reçoit GET request et le passe à View.
5. Le view render le page comme HTML.
8.9. Le contrôleur envoie le HTML au navigateur. Le page charge et l'utilisateur voit le page avec la forme.

Le deuxième cycle est que l'utilisateur rend une forme et sauvegarde dans le Database.

1. L'utilisateur remplit dans la forme et le rend.
2. Quand l'utilisateur rend la forme, le navigateur envoie un POST request à Rails app.

3. Le POST request arrive Rails routage (config/routes.rb). Le routage mappe le POST request au "Controller Action".
6. L'action reçoit POST request et le contrôleur action extrait des données de forme et utilise le modèle à sauvegarde des données dans Database.
7. Le modèle complète son task.
4. Le contrôleur action passe le request au View.
5. Le View rend la page comme HTML.
8.9. Le contrôleur envoie le HTML au navigateur. Le page charge et l'utilisateur voit le page.

Les principaux objets du système est le pendant de la partie 1.3.1 (externe/interne).

RQ : ces points seront détaillés par la suite (cf. parties 1.6 et 1.7)

1.3.4 Contraintes de développement, d'exploitation et de maintenance

Contraintes de développement

Préciser les contraintes liées aux :

— matériels : CAS(authentification login) de université tours, LDAP de l'université Tours ;
— langages de programmation : Ruby 2.1.3 , HTML/CSS, Javascript, Python, AJAX ;
— logiciels de base à utiliser pour le développement : Atom(éditeur sous Mac), le terminale sous Linux/Unix ;
— environnements nécessaires : système : Unix/Linux, le framework : Rails 4.1.6, base de données : MySQL, le serveur : Nignx ;
— bibliothèques de programmes imposées : jQuerty ;
— protocoles de communication imposés : les protocoles sont imposés dans la fonctionnalité du déploiement.
— logiciel dessine des maquettes du site : Balsamiq Mockups.

Contraintes d'exploitation

Ce projet étant susceptible d'être mis en production, il faudra veiller à corriger le plus de failles de sécurité possible. Les mécanismes internes à Rails déjà nombreuses test pour vérifications.

Maintenance et évolution du système

L'application web est module : une gestion des APPs simple et générique sera proposée par défaut. En utilisant les systèmes de bundles et de services intégrés à Ruby on Rails.

1.4 Description des interfaces externes du logiciel

1.4.1 Interfaces matériel/logiciel

Un serveur est nécessaire pour faire tourner l'application web Nignx. Celle-ci a besoin d'une connexion à une base de données MySQL, qui sera installée sur la même machine. Le framework Rails intègre appelé pour faire le lien avec la base de données. En changeant juste quelques réglages, on peut donc utiliser d'autres types de base de données.

1.4.2 Interfaces homme/machine

L'application sera sous la forme d'un site web, utilisable dans un navigateur.

En ce qui concerne l'ergonomie du système, une barre de menu apparaîtra en haut de chaque fenêtre et sera identique partout pour naviguer dans le site, on utilisera les popup natives en Javascript pour les messages d'erreurs, sauf dans le cas où un formulaire est invalide : dans ce cas, le message d'erreur sera affiché juste à côté du champ concerné du formulaire.

On pourra ré-agencer les Apps et les listes de Apps par glisser, et éditer un champ en cliquant dessous. Vous trouverez à la section1.5 les mockups pour **UbiQube**.

1.4.3 Interfaces logiciel/logiciel

Accès à la base de données : RoR gère lui-même l'accès à la base de données, il faut d'indiquer le type de base de données et il saura utiliser le bon driver.

Quant à Login (CAS d'université tours), le service d'authentification utilise un annuaire LDAP. Pour s'y connecter, il est nécessaire d'avoir un compte d'université tours pour accéder. Au niveau de déploiement, il suffit d'écrire un playbook pour Ansible.

1.5 Maquette des IHMs de plateforme

1.5.1 Univers pages

Les pages suivent(fig.1.8, fig.1.9, fig.1.10, fig1.11) sont les pages pour tous les utilisateurs : la page d'accueil du site, la page d'index après login, parcourt l'information d'App et créer une App.

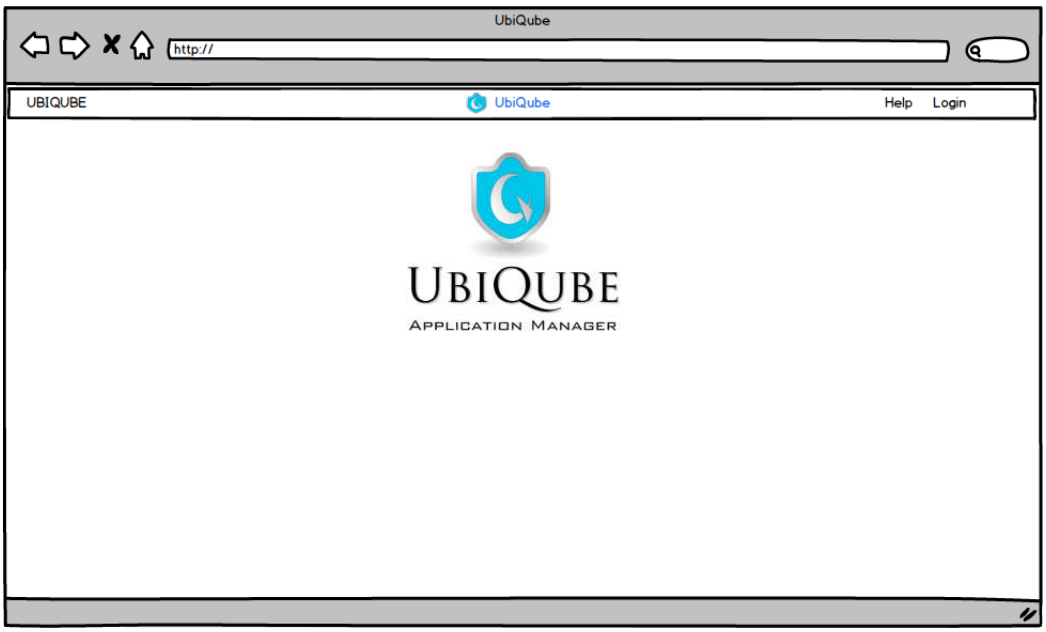

FIGURE 1.8 – L'Accueil du site web

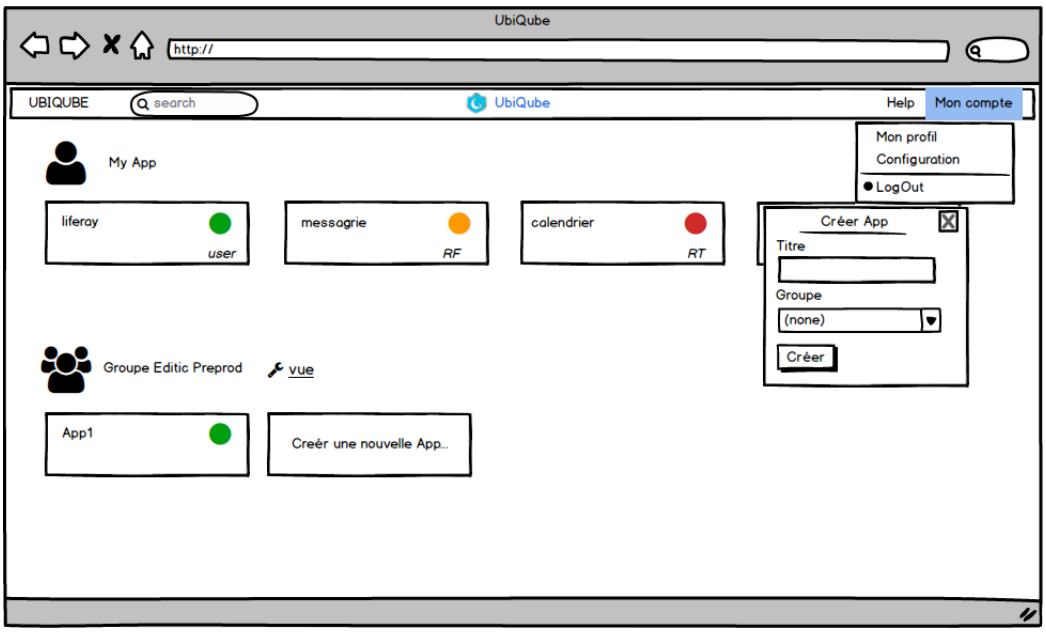

FIGURE 1.9 – Home page

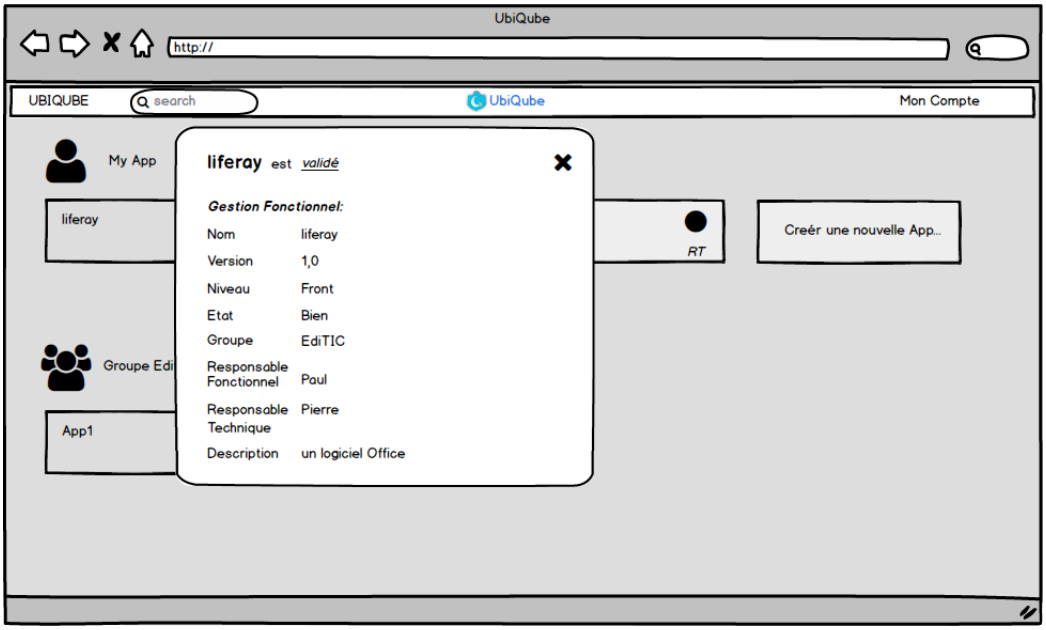

FIGURE 1.10 – Read-only App

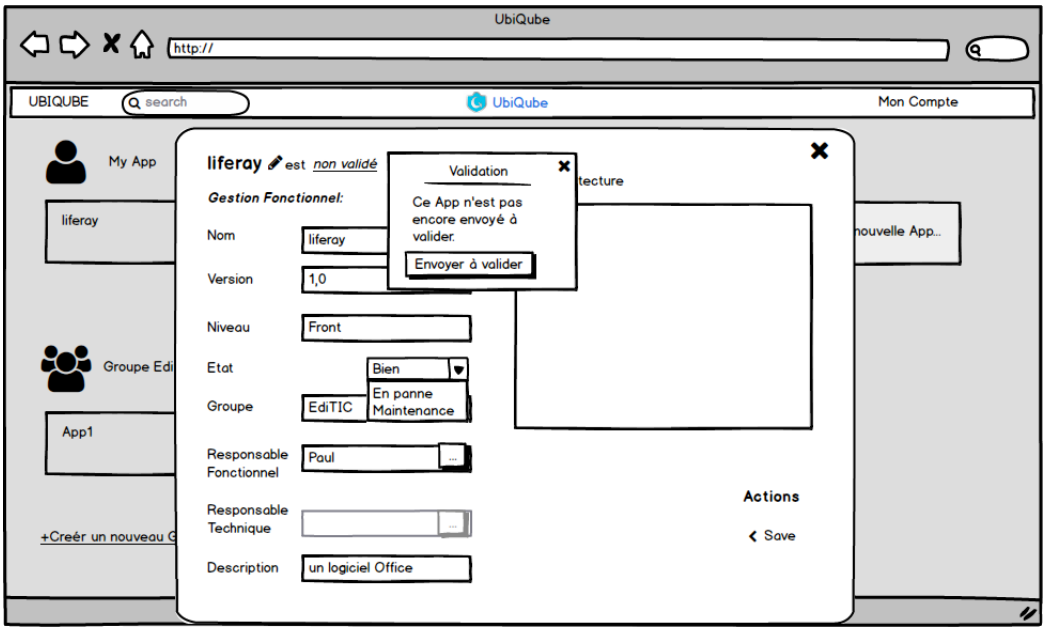

FIGURE 1.11 – Créer une App(avant validé)

1.5.2 Responsable pages

Le responsable technique et responsable fonctionnel peuvent éditer l'App après la validation en présentant les pages suivent fig.1.12, fig.1.13, fig.1.14.

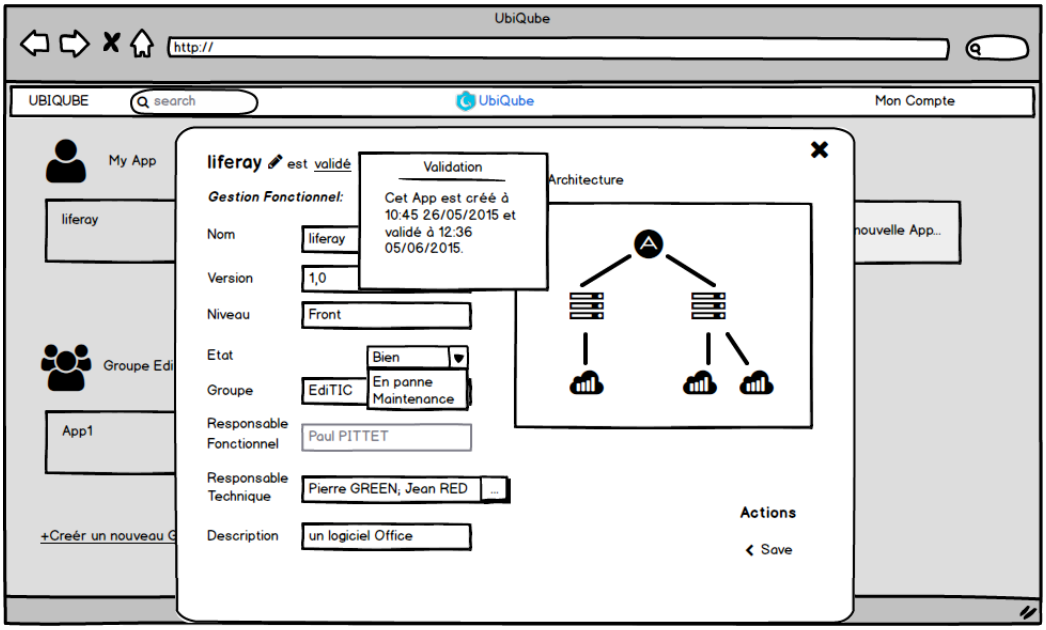

FIGURE 1.12 – Responsable fonctionnel édite une App(après validé)

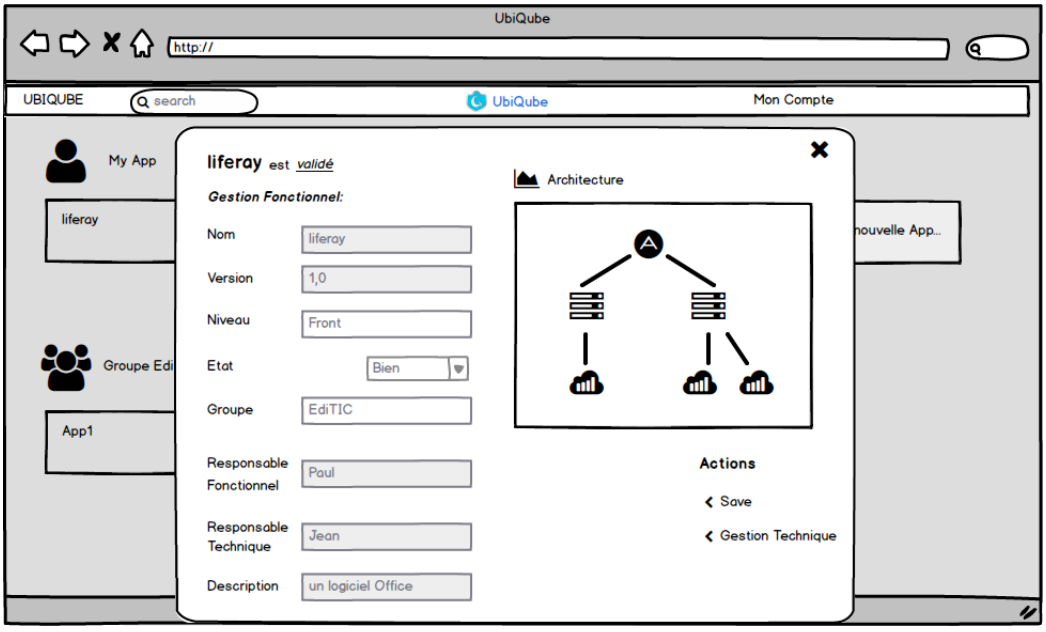

FIGURE 1.13 – Responsable technique édite une App(après validé)

1.5.3 Administrateur pages

Dans la zone d'administrateur, on peut configurer des paramètres d'ubiqube , gérer des droits de rôle et gérer des utilisateurs. Les maquettes sont comme les figures suivent fig.1.15, fig.1.16, fig.1.17.

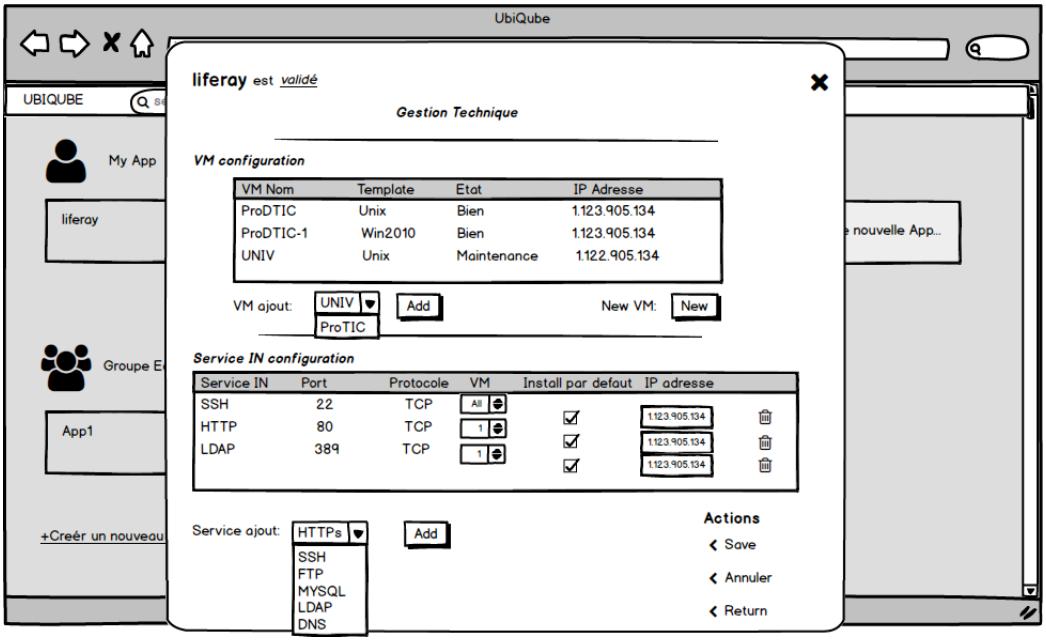

FIGURE 1.14 – Responsable technique gère des paramètres techniques

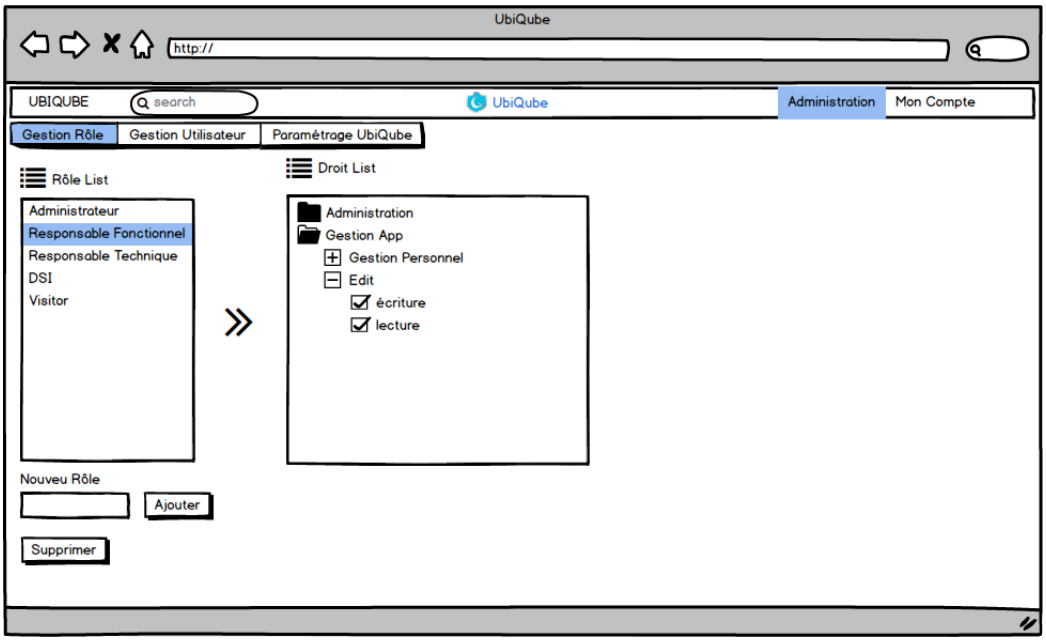

FIGURE 1.15 – Administrateur gère des droits

1.5.4 DSI pages

Dans la zone de DSI, il peut voir toutes les Apps rendues et les gère (fig.1.18, fig.1.19).

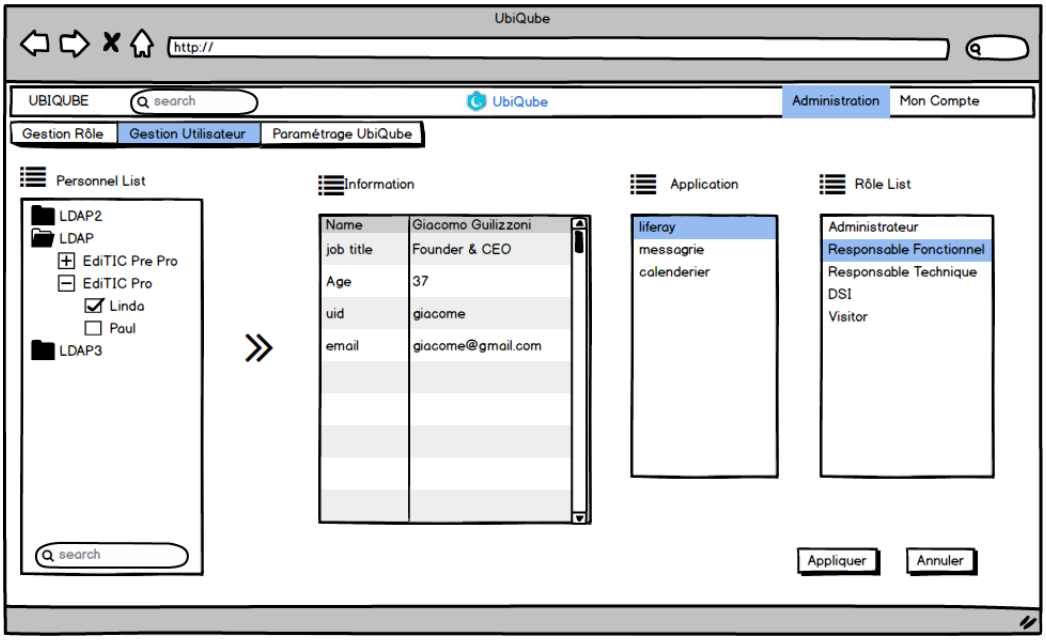

FIGURE 1.16 – Administrateur gère des utilisateur

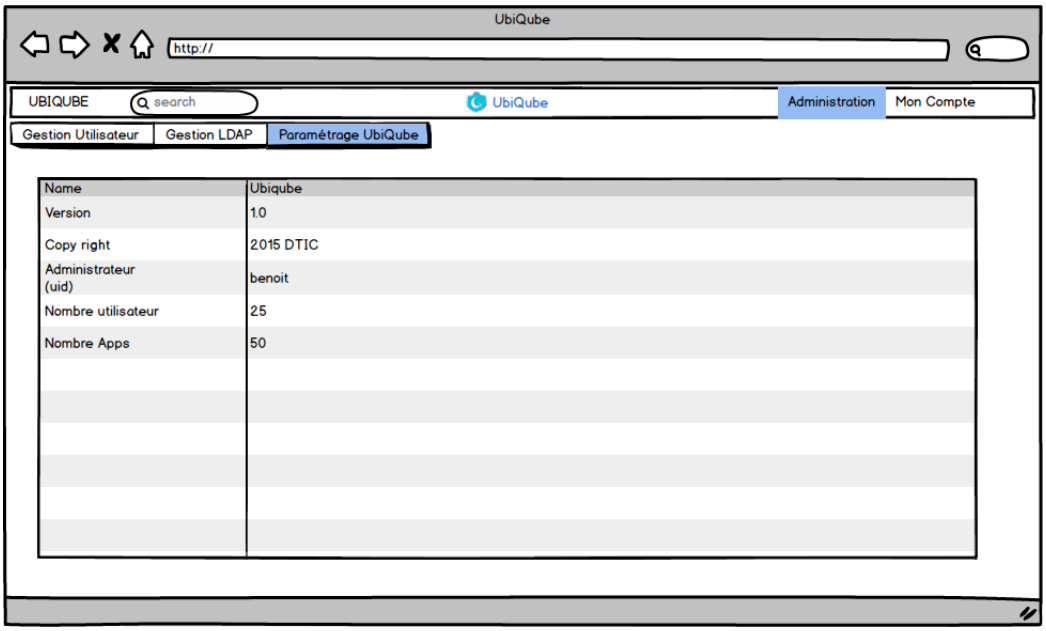

FIGURE 1.17 – Administrateur paramètre Ubiqube

1.6 Architecture générale du système

On monte un diagramme de SQL résumant la structure interne de **UbiQube** fig.1.20 identifie les principaux composants/éléments du système ainsi que leurs relations.

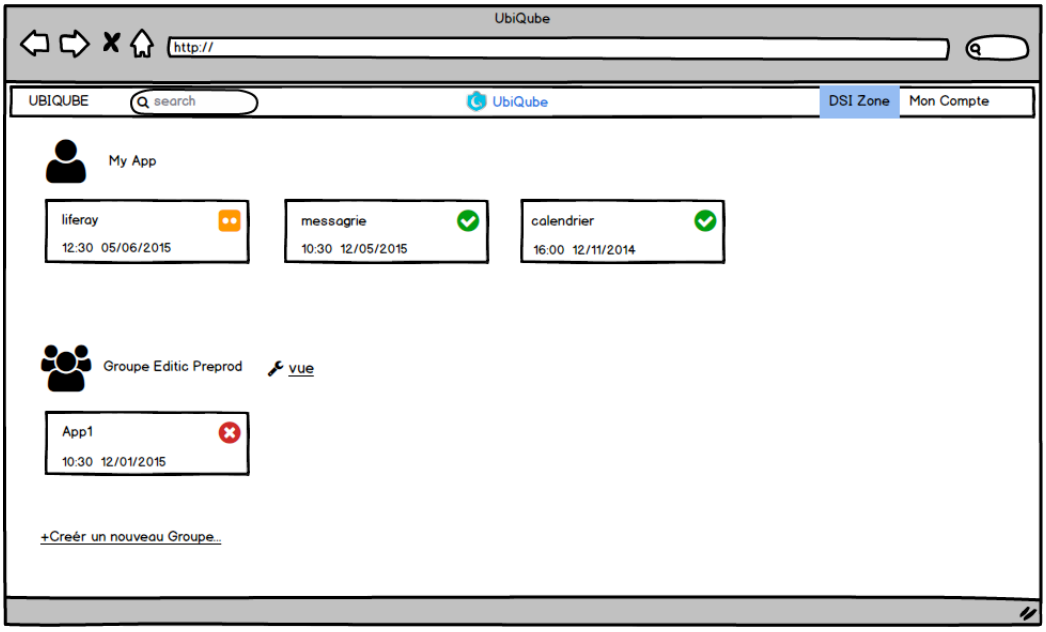

FIGURE 1.18 – La zone de DSI

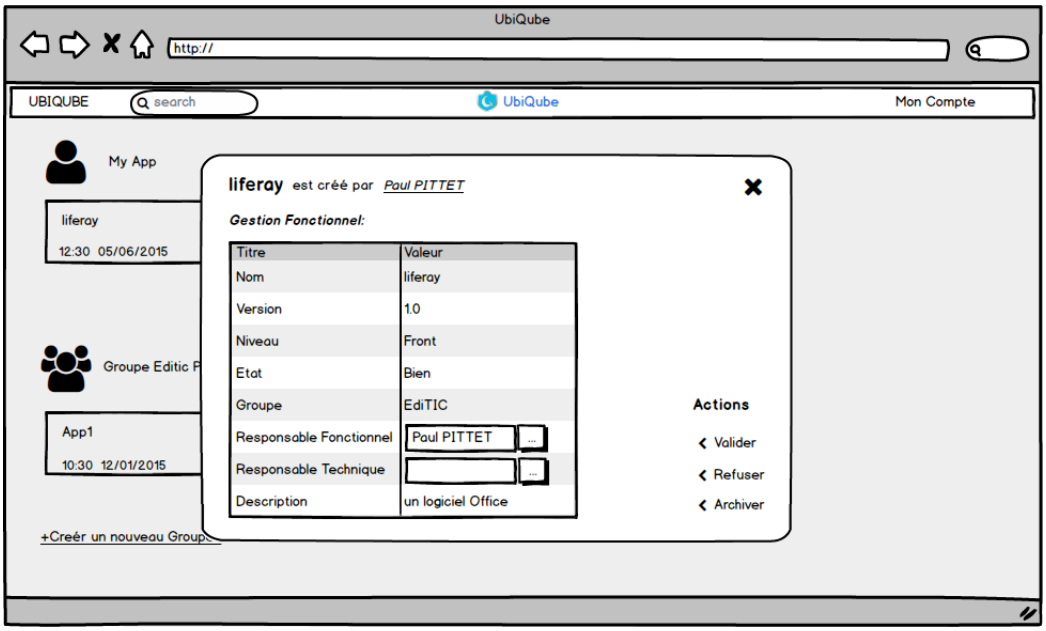

FIGURE 1.19 – DSI gère des Apps

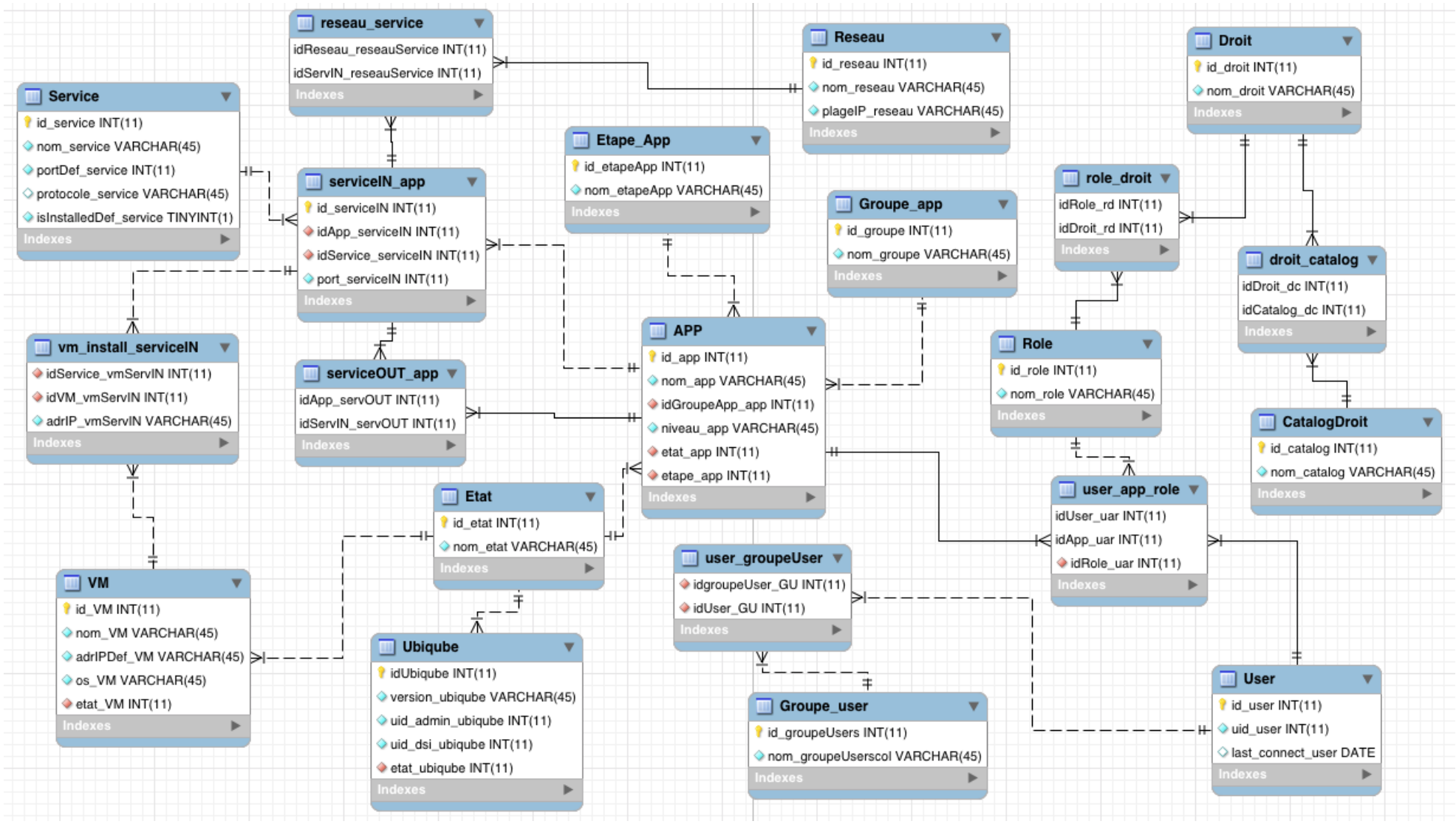

FIGURE 1.20 – La base de données d'ubiQube

1.7 Description des fonctionnalités

Il s'agit de l'expression des besoins fonctionnels. Cette partie a donc comme objectif de décrire l'ensemble des fonctionnalités du système en précisant avec quels composants de la partie 1.6 elles interagissent. Des diagrammes de cas d'utilisation plus détaillés que ceux présentés en 1.3.3. En outre, chaque fonctionnalité sera décrite précisément (cf.ci-dessous). Là encore, il s'agit d'une pré-analyse indispensable à l'évaluation de la complexité du projet et à la planification de sa réalisation. Toutes les fonctionnalités sont non développé.

1.7.1 Affichage les Apps après login

Description Les utilisateurs ne doivent pas avoir accès aux APPs auxquels ils ne sont pas rattachés. Ils ne doivent pas non plus pouvoir modifier (via des requêtes HTML POST par exemple) les APPs auxquelles ils n'ont pas accès. Les utilisateurs doivent se connecter pour accéder aux projets. L'authentification sur le site est opérationnelle. Il faut le rôle Admin pour accéder à la zone d'administration. Pour l'instant, l'accès n'est pas limité à ses propres APPs et chacun peut voir et modifier les APPs de tout le monde.

Priorité Forte.

Entrées L'utilisateur en session.

Sorties L'autorisation ou non d'accéder à telle ou telle page. Affichage des APPs à la page d'accueil après login.

1.7.2 Créer une nouvelle App

Description Tous les utilisateurs peuvent créer une nouvelle App dans la page d'accueil après login.

Priorité Forte

Entrées L'utilisateur en session.

Sorties Créer une nouvelle App dans la liste.

1.7.3 Gestion fonctionnel d'App

Description Le responsable technique met les paramètres de VMs : le nombre, l'adresse IP, service installé etc.

Priorité Forte

Entrées une APP concernée

Sorties paramétrage les informations fonctionnelles d'APP

1.7.4 Valider/Refuser/Archiver App

Description Étant en DSI, il recevra les notations de nouvelle App à valider. Dans une zone DSI, il peut
Refuser : le créateur va modifier des descriptions d'App et re-envoie à valider.
Valider : le créateur peut configurer les paramètres d'App à la suivent.
Archiver : DSI refuse l'App et l'archive dans un fichier XML.

Priorité Forte

Entrées DSI en session

Sorties les Apps non validées dans la page de DSI.

1.7.5 Gestion technique d'App

Description Le responsable technique met les paramètres de VMs : le nombre, l'adresse IP, service installé etc.

Priorité Forte

Entrées une APP concernée

Sorties paramétrage les informations techniques d'APP

1.7.6 Le zone d'administration

Description La zone d'administration n'est accessible que par les administrateurs. Elle permettra d'afficher quelques données générales et d'exécuter plusieurs tâches spécifiques : Lister, ajouter ou intégrer l'utilisateur à une APP ; Gestion les droits du rôle ; Paramétrage UbiQube.

Priorité Forte

Entrées La liste des utilisateurs et des APPs.

Sorties /

1.7.7 Envoyer une nouvelle App à valider

Description Quand utilisateur finit de remplir les informations, il peut l'envoie à DSI pour validation. Pour un site plus dynamique, des requêtes AJAX seront envoyées à intervalle régulier pour rafraîchir la liste des notifications.

Priorité Moyenne

Entrées L'utilisateur en session

Sorties les notifications non lues associées à l'utilisateur.

1.7.8 Le panel du groupe

Description DSI peut créer un groupe pour partager des Apps. Celui ci contient des membres d'un groupe et des Apps qui sont visibles par tous les membres.

Priorité Faible

Entrées La liste d'utilisateur

Sorties Un nouveau groupe

1.7.9 Supervision des VMs

Description Pour chaque App, on peut superviser l'état du VM et les services associés.

Priorité Faible

Entrées une App concernée

Sorties Les informations techniques d'App

1.8 Product Backlog

Parce que ce système est utilisé dans l'entreprise intérieur donc actuellement les utilisateurs sont des personnels du DTIC. Mais il y a 4 groupes principaux. On détaille dans le tableau suivant tab.1.2

User type	Besoins	Recette	Priorité
Utilisateur	Affichage les Apps après login	Afficher mes Apps après login	Forte
Utilisateur	Créer une nouvelle App	Ajouter une nouvelle App dans ma liste	Forte
Utilisateur	Envoyer une nouvelle App à valider	envoyer la nouvelle App à DSI pour valider	Moyenne
Responsable fonctionnel	Gestion fonctionnel d'App	Remplir des information fonctionnel d'APP	Forte
Responsable technique	Supervision des VMs	Savoir l'etat du VM et de services	Faible
Responsable technique	Gestion technique d'App	Remplir des information technique d'APP	Forte
Administrateur	Accéder à la zone d'administration	Administrer les comptes utilisateur, les Apps et l'application	Forte
DSI	Valider/Refuser/Archiver App	Contrôle le cycle vie d'App	Forte
DSI	Le panel du groupe	Donner des droits d'accès à un ensemble de personne	Faible

TABLE 1.2 – Product Backlog

1.9 Conditions de fonctionnement

1.9.1 Performances

La plupart des pages du site étant dynamiques (avec du Javascript et de l'AJAX), il est nécessaire que le temps de réponse entre une action de l'utilisateur et la modification de l'interface soit négligeable (au mieux instantané, sinon inférieur à une seconde). Néanmoins, si l'utilisateur a une mauvaise connexion à internet (au moins, au réseau hébergeant le serveur), il se peut que le temps de réponse soit plus long. Dans ce cas, un délai d'attente maximum (timeout) de 10 secondes sera défini. Au delà, l'utilisateur se verra notifié d'un message d'erreur de connexion au serveur.

Le contenu des pages les plus importantes comme le tableau de bord d'un projet, sera réactualisé régulièrement et automatiquement (par exemple toutes les 30 secondes), sans recharger toute la page. On appliquera la même règle de timeout en cas de serveur inaccessible.

1.9.2 Capacités

Le système est prévu pour accueillir un petit nombre d'utilisateur simultanés, sans nombre maximum défini. Néanmoins, si de plusieurs utilisateurs modifient en même temps la même information, il est possible que certaines modifications ne soient pas prises en compte. La capacité maximum de stockage dépend deux éléments :

- La quantité d'informations maximum stockées en base de données : La taille maximum de MySQL est égale à ma maximum d'un fichier sur le système de fichier qui est utilisé, En notre cas, un serveur Linux en ext3 ou ext4 aura une taille de fichier maximum de 2 To.
- La capacité de stockage réelle du serveur, principalement pour les fichiers joints aux Apps.

1.9.3 Contrôlabilité

L'administrateur peut contrôler la liste des utilisateurs, des Apps et des paramètres de Ubiqube.

Ubiqube : le projet RoR a des logs du développement(development.log) et du test(test.log) dans le répertoire : /log

serveur NginX : l'accès log est /var/log/nginx/nginx.access.log

HTTP server Unicorn : Les logs d'unicorn mettent dans le répertoire /shared/log : unicorn.stderr.log, unicorn.stdout.log.

1.9.4 Sécurité

Il y a 4 types d'utilisateurs : utilisateur ordinaire, responsable fonctionnel, responsable technique et administrateur. L'authentification via le serveur LDAP de L'université Rabelais limitera la connexion au site aux étudiants, professeurs et membres du personnel.

Plan de développement

2.1 Découpage du projet en tâches

Afin de répondre aux objectifs du présent projet, le projet est découpé en tâches. Cette section décrit chacune des tâches en précisant leurs éventuels livrables et leurs estimation de charge en jours/homme.

2.1.1 Étude du contexte et des besoins

Description de la tâche

Cette tâche à pour objectif de comprendre le contexte dans lequel se place Ubiqube, d'étudier précisément les besoins motivant ce projet, les contraintes, d'exprimer des objectifs et de reporter l'ensemble dans un premier rapport. Cette tâche comporte également une étude d'ensemble portant sur ce que la concurrence propose à l'heure actuelle.

Livrables

Les chapitres sur le contexte, les besoins et l'état d'art dans ce rapport.

Estimation de charge

3 jour/homme

2.1.2 Étude de matériaux

Description de la tâche

Cette tâche à pour comprendre des web techniques cœur et des outils utilisés pendant ce projet. Par exemple :

- Sécurité :CAS, Central Authentication Service
- BDD : LDAP (site référenciel [3])
- Framework : Ruby on Rails(site officiel [4])
- Deploiement : Ansible(site officiel [5])

Livrables

Le rapport d'étude nommé "Étude d'outils"

Estimation de charge

10 jour/homme

Statu de la tâche

Réalisée

2.1.3 Proposer des résolutions

Description de la tâche

Suite à la comparaison des solutions, on décide les manières de travail :

- Définir l'architecture du système
- Construire la base de données
- Designer des maquettes du site web

Livrables

Les maquettes du site web

Estimation de charge

5 jour/homme

Statu de la tâche

Réalisée

2.1.4 Rédaction du cahier de spécification

Description de la tâche

La rédaction du cahier de spécifications dans l'intérêt de gestion projet est pour communiquer les fonctionnalités de l'application et les tâches à accomplir durant ce projet. Notamment on précise l'environnement, les objectifs, les utilisateurs cibles, les contraintes, l'architecture générale, le découpage de tâches et le planning.

Livrables

Le cahier de spécifications.

Estimation de charge

5 jour/homme

2.1.5 Revue de code complète

Description de la tâche

Revue de code sur Githude, configuration l'environnement et change la base de donnée au Mysql.

Statu de la tâche

Réalisée

Estimation de charge

5 jour/homme

2.1.6 Affichage les Apps après login

Description de la tâche

Après login, l'utilisateur peut accéder directement aux APPs auxquels ils sont rattachés. En cliquant l'App, on peut parcourir l'information d'App. Le rôle Administrateur peux accéder aux tous les Apps. Maquette correspond fig.1.9 et fig.1.10.

Estimation de charge

5 jour/homme, jusqu'à 19/06/2015

2.1.7 Créer une nouvelle App

Description de la tâche

Dans la page de tous les Apps, tous les utilisateurs peuvent créer une nouvelle App. Maquette correspond fig.1.9 et fig.1.11.

Estimation de charge

5 jour/homme, jusqu'à 26/06/2015

2.1.8 Gestion fonctionnel et technique de App

Description de la tâche

Pour responsable technique, il peut gère des paramètres technique d'App après la validation. Maquette correspond fig.1.14 et fig.1.14

Estimation de charge

10 jour/homme, jusqu'à 10/07/2015

2.1.9 Le panel du App pour DSI

Description de la tâche

DSI charge le processus de vie d'App. Il peut valider, refuser ou archiver une nouvelle App. Maquette correspond fig.1.18 et fig.1.19.

Estimation de charge

5 jour/homme, jusqu'à 17/07/2015

2.1.10 Le zone d'administration(partie 1)

Description de la tâche

L'administrateur peut gérer des droits d'utilisateur. Il peut choisir un utilisateur dans la liste d'utilisateur et puis afficher toutes les informations d'utilisateur, les Apps auxquelles utilisateur rattache et leur rôle. Maquette correspond fig.**??**.

Estimation de charge

10 jour/homme, jusqu'à 31/07/2015

2.1.11 Le zone d'administration(partie 2)

Description de la tâche

L'administrateur peut accéder aux tous les Apps et le paramètre. Maquette correspond fig.**??**. L'administrateur peut aussi paramètrer UbiQube. Maquette correspond fig.1.17

Estimation de charge

5 jour/homme, jusqu'à 07/08/2015

2.1.12 Automatisation de serveur

Description de la tâche

Ubiqube génère le playbook de l'Ansible pour le déploiement de services(Apps) à partir les paramètres techniques d'App.

Estimation de charge

10 jour/homme, jusqu'à 21/08/2015

2.1.13 Supervision des VMs

Description de la tâche

Superviser l'etat du VM dans le module de gestion technique. On peut superviser les IN services et OUT services d'App. IN service est des flux d'entrées d'App, et on peut voir l'information de source et le port. OUT service est des flux de sortie d'App. Grâce à cette fonction, on peut voir les relations entre VM(App).

Estimation de charge

5 jour/homme, jusqu'à 28/08/2015

2.1.14 Envoyer une nouvelle App à valider

Description de la tâche

Après la création d'App, on peut l'envoyer à DSI pour la validation.

Estimation de charge

5 jour/homme, jusqu'à 04/09/2015

2.1.15 Le panel du groupe

Description de la tâche

DSI peut créer un nouveau groupe qui a un ensemble d'App visible pour les membres.

Estimation de charge

5 jour/homme, jusqu'à 11/09/2015

2.1.16 Test

Description de la tâche

Test de fonctionnalités, test de charge et tests de déploiement.

Livrables

Un rapport de tests sera rédigé en expliquant ce qui a été fait. Un script permettant d'automatiser les tests sera également livré.

Estimation de charge

10 jour/homme, jusqu'à 26/09/2015

2.1.17 Manuels techniques et utilisateurs

Description de la tâche

le manuel technique pour développeur regroupe des informations générales qu'il est nécessaire de lire avant de commencer toute action de développement sur Ubiqube. Le manuel utilisateur indique la manière d'utilisation pour différent utilisateur.

Livrables

Guide d'utilisateur et Guide de développeur.

Estimation de charge

la charge est répartie sur toute la durée du projet

2.1.18 Présentation oral finale

Description de la tâche

Une présentation finale conclut le stage. Cette présentation est réalisée à partir de la constitution des besoins, des objectifs. On explique alors clairement ce qui a été réalisé et comment. On évalue la réussite du projet et on chiffre. La présentation doit faire part de la difficulté du sujet. Elle aussi traite de la gestion de projet : expliquer les éventuelles modifications de planning et leurs conséquences, ou bien encore expliquer la démarche et les méthodologies adoptées.

Livrables

Une présentation orale de 15 minutes

Estimation de charge

3 jours/homme

Contraintes temporelles

La présentation à partir du 25 Septembre 2015

2.1.19 Rapport final

Description de la tâche

A ce poit, il s'agit Rapport du stage

Livrables

Rapport du stage et silde de soutenance

Estimation de charge

la charge est répartie sur toute la durée du projet

Contraintes temporelles

La tâche a débuté dès le premier jour du stage et s'étendra jusqu'à la date limite qui est fixée au 10 Septembre 2015

2.2 Planning

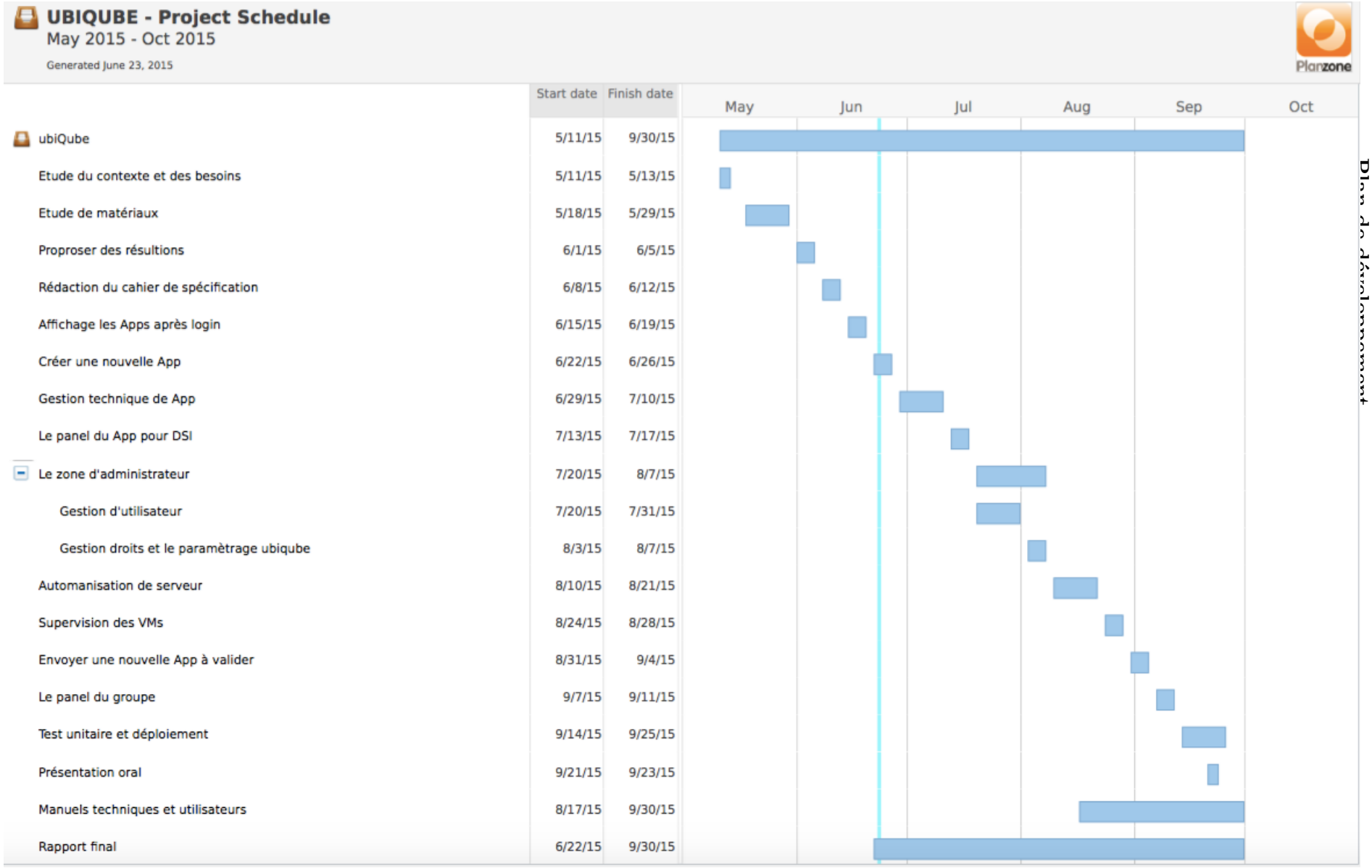

Figure 2.21 – Diagramme Gantt

Glossaire

API(Application P Interface) Une API est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. Elle est offerte par une bibliothèque logicielle ou un service web, le plus souvent accompagnée d'une description qui spécifie comment des programmes consommateurs peuvent se servir des fonctionnalités du programme fournisseur.

RoR(Ruby on Rails) Il est un framework web libre écrit en Ruby. Il suit le motif de conception modèle-vue-contrôleur aussi nommé MVC. En tant que framework, il propose une structure au programmeur qui lui permet de développer plus vite et plus intuitivement. Il ajoute aussi un grand niveau d'abstraction dans la programmation de l'application par un ensemble de fonctions de haut niveau qui lui offre ainsi l'économie d'écrire lui-même la plupart des routines obligatoires d'une application web.

ORM(Object Relation Mapping) Ensemble de classes faisant le lien entre la base de données et les classes d'un programme. Cela permet de transformer une table en un objet facilement manipulable via ses attributs.

LDAP(Lightweight Directory Access Protocol Protocole d'annuaire réseau. Il permet le partage de bases de données sur un réseau. Ces bases de données contiennent des informations sur les personnels, organisations et matériels à gérer.

CAS(Central Authentication Service) Service central d'authentification est un système d'authentification unique : on s'authentifie sur un site Web, et on est alors authentifié sur tous les sites Web qui utilisent le même serveur CAS. Il évite de s'authentifier à chaque fois qu'on accède à une application en mettant en place un système de ticket.

DTIC Direction des Technologies de l'Information et de la Communication

DSI Direction des systèmes d'information

Bibliographie

[1] Officiel site web - ubiqube. http://ubiqube.univ-tours.fr.

[2] Github - ubiqube. https://github.com/CHENJing88/UbiQube.git.

[3] Le protocole ldap. http://www-sop.inria.fr/members/Laurent.Mirtain/ldap-livre.html#I1.

[4] Officiel site web - ruby on rails guides (v4.2.1). http://guides.rubyonrails.org/index.html.

[5] Ansible is simple it automation. http://www.ansible.com/home.