|  |  |
| --- | --- |
| Dédicace |  |

|  |  |
| --- | --- |
| Remerciements |  |

|  |  |
| --- | --- |
| Sommaire |  |

[Dédicace i](#_Toc516944731)

[Remerciements ii](#_Toc516944732)

[Sommaire iii](#_Toc516944733)

[Liste des Sigles et Abréviations vii](#_Toc516944734)

[Résumé viii](#_Toc516944735)

[Abstract ix](#_Toc516944736)

[Liste des Tableaux x](#_Toc516944737)

[Liste des figures et illustrations xiii](#_Toc516944738)

[INTRODUCTION GENERALE 2](#_Toc516944739)

[Chapitre 1 : CONTEXTE ET PROBLEMATIQUE 3](#_Toc516944740)

[Introduction 3](#_Toc516944741)

[1.1. Création et missions 3](#_Toc516944742)

[1.2. Gouvernance et fonctionnement 4](#_Toc516944743)

[1.2.1. Gouvernance 4](#_Toc516944744)

[1.2.2. Fonctionnement 6](#_Toc516944745)

[1.3. Quelques réalisations du CARPA 7](#_Toc516944746)

[1.4. Situation géographique 9](#_Toc516944747)

[1.5. Accueil en institution 10](#_Toc516944748)

[1.6. Contexte et Problématique 10](#_Toc516944749)

[1.6.1. Contexte 11](#_Toc516944750)

[1.6.2. Problématique 11](#_Toc516944751)

[1.7. Objectifs et méthodologie 12](#_Toc516944752)

[1.7.1. Objectifs 12](#_Toc516944753)

[1.7.2. Méthodologie 12](#_Toc516944754)

[Conclusion 13](#_Toc516944755)

[Chapitre 2 : GENERALITES 15](#_Toc516944756)

[Introduction 15](#_Toc516944757)

[2.1. Rapid Application Development 15](#_Toc516944758)

[2.2. L’outil Ruby On Rails 16](#_Toc516944759)

[2.3. Le langage Ruby 17](#_Toc516944760)

[2.4. Architecture et création d’un projet avec Ruby On Rails 19](#_Toc516944761)

[2.4.1. Création des classes et modèles 19](#_Toc516944762)

[2.5. Scaffolding 20](#_Toc516944763)

[2.5.1. Running 21](#_Toc516944764)

[2.6. Les avantages de Ruby On Rails 21](#_Toc516944765)

[2.7. L’outil Git 22](#_Toc516944766)

[2.7.1. Git 22](#_Toc516944767)

[2.8. Le pattern MVC 23](#_Toc516944768)

[2.8.1. Définition 23](#_Toc516944769)

[2.8.2. Présentation du modèle MVC 23](#_Toc516944770)

[2.8.3. Le principe du MVC 24](#_Toc516944771)

[2.9. Constructeur d’un projet UML 25](#_Toc516944772)

[2.10. Développement Agile 26](#_Toc516944773)

[2.10.1. Principe du Manifeste Agile 26](#_Toc516944774)

[2.10.2. Modélisation Agile(AM) 27](#_Toc516944775)

[Conclusion 27](#_Toc516944776)

[Chapitre 3 : ANALYSE ET CONCEPTION 28](#_Toc516944777)

[Introduction 28](#_Toc516944778)

[3.1. Cahier de charges 29](#_Toc516944779)

[3.1.1. Fonctionnalités attendues de l’application 29](#_Toc516944780)

[3.1.2. Besoins de l’application 29](#_Toc516944781)

[3.1.3. Besoins fonctionnels 29](#_Toc516944782)

[3.1.4. Besoins non fonctionnels 42](#_Toc516944783)

[3.1.4. Étude de faisabilité 43](#_Toc516944784)

[3.2. Modélisation 46](#_Toc516944785)

[3.2.1. Outils de modélisation 46](#_Toc516944786)

[3.2.2. Acteurs du système 46](#_Toc516944787)

[3.2.3. Diagramme de contexte 47](#_Toc516944788)

[3.2.3. Diagramme de cas d’utilisation 49](#_Toc516944789)

[3.2.4. Diagramme de séquence 51](#_Toc516944790)

[3.2.5. Diagramme de classe 58](#_Toc516944791)

[3.2.6. Diagramme de déploiement 59](#_Toc516944792)

[Conclusion 60](#_Toc516944793)

[Chapitre 4 : IMPLEMENTATION 62](#_Toc516944794)

[Introduction 62](#_Toc516944795)

[4.1. Langage de programmation et environnement de développement 62](#_Toc516944796)

[4.1.1. Langage de programmation 62](#_Toc516944797)

[4.1.2. Outils de développement 65](#_Toc516944798)

[4.2. Présentation du projet 66](#_Toc516944799)

[4.3. Modèle Architecturale 68](#_Toc516944800)

[4.3.1. Synoptique de fonctionnement d’une application web 68](#_Toc516944801)

[4.3.2. Synoptique de fonctionnement d’une application web en Ruby On Rails 68](#_Toc516944802)

[Conclusion 69](#_Toc516944803)

[Chapitre 5 : RESULTATS ET COMMENTAIRES 70](#_Toc516944804)

|  |  |
| --- | --- |
| Liste des Sigles et Abréviations |  |

|  |  |
| --- | --- |
| Résumé |  |

|  |  |
| --- | --- |
| Abstract |  |

|  |  |
| --- | --- |
| Liste des Tableaux |  |

[Tableau 1 projets en cours de réalisation 7](#_Toc516942759)

[Tableau 2 projets avec contrat signé mais pas encore démarré 8](#_Toc516942760)

[Tableau 3 projets en cours de procédure 9](#_Toc516942761)

[Tableau 4 Les ancêtres de Ruby 19](#_Toc516942762)

[Tableau 5 Enregistrement d'un utilisateur 30](#_Toc516942763)

[Tableau 6 modifier un utilisateur 31](#_Toc516942764)

[Tableau 7 supprimer utilisateur 31](#_Toc516942765)

[Tableau 8 enregistrer un compte utilisateur 32](#_Toc516942766)

[**Tableau 9 modifier un compte utilisateur** 33](#_Toc516942767)

[Tableau 10 rechercher un utilisateur 33](#_Toc516942768)

[Tableau 11 enregistrer un courrier 34](#_Toc516942769)

[Tableau 12 modifier un courrier 34](#_Toc516942770)

[Tableau 13 supprimer un courrier 35](#_Toc516942771)

[Tableau 14 enregistrer projet 36](#_Toc516942772)

[Tableau 15 modifier projet 36](#_Toc516942773)

[Tableau 16 supprimer projet 37](#_Toc516942774)

[Tableau 17 enregistrer phase du projet 38](#_Toc516942775)

[Tableau 18 modifier phase du projet 38](#_Toc516942776)

[Tableau 19 supprimer phase du projet 39](#_Toc516942777)

[Tableau 20 enregistrer archive 39](#_Toc516942778)

[Tableau 21 modifier archive 40](#_Toc516942779)

[Tableau 22 supprimer archive 41](#_Toc516942780)

[Tableau 23 importer les données 41](#_Toc516942781)

[Tableau 24 exportation des données 42](#_Toc516942782)

[Tableau 25 Planning prévisionnel 46](#_Toc516942783)

[Tableau 26 Identification des messages échangés 48](#_Toc516942784)

|  |  |
| --- | --- |
| Liste des figures et illustrations |  |

[Figure 1 présentation de l’organigramme du CARPA 6](#_Toc516944537)

[Figure 2 Plan de localisation du CARPA 10](#_Toc516944538)

[Figure 3 Modèle en V 13](#_Toc516944539)

[Figure 4 Terminal 17](#_Toc516944540)

[Figure 5 Arbre généalogique de Ruby 18](#_Toc516944541)

[Figure 6 contenu d'un dossier ROR 20](#_Toc516944542)

[Figure 7 Première page d'un projet ROR 21](#_Toc516944543)

[Figure 8 Cycle de vie des états d'un fichier git 23](#_Toc516944544)

[Figure 9 Schématisation du MVC 24](#_Toc516944545)

[Figure 10 diagramme de contexte 47](#_Toc516944546)

[Figure 11 diagramme de cas d'utilisation s'authentifier et créer compte 49](#_Toc516944547)

[Figure 12 cas d'utilisation gestion et archivage d'un courrier 50](#_Toc516944548)

[Figure 13 cas d'utilisation gestion et archivage d'un projet 50](#_Toc516944549)

[Figure 14 diagramme des principaux cas d'utilisation 51](#_Toc516944550)

[Figure 15 Diagramme de séquence du cas d'utilisation « s'authentifier » 52](#_Toc516944551)

[Figure 16 Diagramme de séquence du cas d'utilisation enregistrer un utilisateur 53](#_Toc516944552)

[Figure 17 Diagramme de séquence du cas d'utilisation modifier utilisateur 54](#_Toc516944553)

[Figure 18 Diagramme séquence enregistrer un courrier 55](#_Toc516944554)

[Figure 19 Diagramme séquence Archiver un courrier 56](#_Toc516944555)

[Figure 20 Diagramme de séquence du cas d’utilisation enregistrer projet 57](#_Toc516944556)

[Figure 21 Diagramme séquence Archiver un projet 58](#_Toc516944557)

[Figure 22 Diagramme de classe 59](#_Toc516944558)

[Figure 23 Diagramme de déploiement 60](#_Toc516944559)

[Figure 24 Arborescence du projet dans RubyMine 67](#_Toc516944560)

[**Figure 25 Arborescence du projet dans l'explorateur Windows** 67](#_Toc516944561)

[Figure 26 Synoptique de l'application web 68](#_Toc516944562)

[Figure 27 Synoptique d'une application web en Ruby On Rails 69](#_Toc516944563)

|  |  |
| --- | --- |
| INTRODUCTION GENERALE |  |

Il se dégage aujourd’hui un consensus quant aux possibilités ouvertes par les technologies de l’information et de la communication (TIC) qui se développent rapidement dans tous les domaines de l’entreprise et plus largement de la société́. Elles permettent de manipuler de l’information pour la stocker, la convertir, la gérer, la transmettre et la retrouver.

L’Internet est considèré comme étant un moyen idéal de communication, d’échange de données ou encore d’apprentissage, et est également un outil efficace pour avoir des informations sur un service ou un produit. L’apprentissage est l’un des services privilégiés qu’Internet offre aux visiteurs. Dans cette optique, de nombreuses applications et sites Web dynamiques ont vu le jour.

Les institutions publiques gouvernementales sont parmi les établissements qui ont besoin d’un système informatique pour bien conduire leur travail en évitant la perte de temps. Ces institutions dépendent de plus en plus de l’informatique pour réaliser leurs objectifs, elles sont donc plus sensibles à la qualité́ des services informatiques fournis aux différentes catégories d’utilisateurs et sont à la recherche de moyens et des ressources pour améliorer leurs services. Ainsi, restaurer le plus rapidement possible le fonctionnement normal des services afin de minimiser l’impact négatif de celui-ci sur les activités métiers, et s’assurer que les meilleurs niveaux de qualité́ de service et de disponibilité́ sont maintenus, seront des atouts pour chaque entreprise.

L’objectif de notre travail est de concevoir et mettre en œuvre une application web de suivi du projet, du courrier et d’archivage numérique au sein **CARPA** dans le but de résoudre les problèmes liés au suivi des projets et à la gestion, le suivi et l’archivage du courrier.

|  |  |
| --- | --- |
| : CONTEXTE ET PROBLEMATIQUE |  |

## Introduction

Dans ce chapitre, nous allons présenter la structure dans laquelle nous avons effectué notre stage académique, pour cela nous présenterons premièrement de la création et des missions du **CARPA**, deuxièmement de sa gouvernance et de son fonctionnement, par la suite de quelques-unes de ses réalisations et enfin de sa situation géographique. Nous présenterons aussi les motivations de notre thème à travers le contexte de l’entreprise et la critique de l’existant. Nous terminerons cette partie les objectifs visés par la solution suggérée.

## 1.1. Création et missions

Crée et organisé par le décret **n° 2008/035 du 24 Janvier 2008** portant organisation et fonctionnement du Conseil d’Appui à la Réalisation des Contrats de Partenariat (**CARPA),** il s’agit d’un organisme de droit public placé sous la tutelle du Ministère en charge de l’économie, de la planification et de l’aménagement du territoire (MINEPAT). Cette structure est dotée d’une autonomie financière et de gestion et a pour mission de contribuer par son expertise, à la création, au renouvellement des infrastructures et des équipements publics, ainsi qu’à l’amélioration de la qualité du service public dans le cadre des projets de grande envergure technique et financière à réaliser à travers un contrat de partenariat.

A ce titre, il est notamment chargé :

* de l’élaboration des mécanismes de mise en œuvre des contrats de partenariat ;
* de l’évaluation préalable des projets à réaliser en partenariat public privé ;
* de la participation aux négociations, au contrôle et au suivi de l’exécution des contrats de partenariat ;
* de l’information et de la formation des administrations publiques, des milieux d’affaires et du public en général sur le concept et les mécanismes de PPP ;
* de la promotion, de la diffusion et de la vulgarisation du régime camerounais des contrats de partenariat ;
* de l’élaboration des instruments juridiques et techniques d’analyses des projets et sélection des partenaires de la personne publique ;
* du développement de l’expertise nationale et de la veille en matière d’innovation sur les modes de gestion des projets publics dans le cadre d’un contrat de partenariat.

## 1.2. Gouvernance et fonctionnement

### 1.2.1. Gouvernance

Pour l’accomplissement de ses missions, le **CARPA** dispose :

* d’un Président ;
* d’un Comité d’Orientation ;
* d’un Secrétariat Technique.

#### Le président

Le Président du Conseil est nommé par décret du Président de la République parmi les personnalités de réputation professionnelle établie. Il assure la direction administrative, technique et financière du Conseil.

#### Le comité d’orientation

Il est présidé par le Président du Conseil, et est chargé de donner un Avis de faisabilité de chaque projet candidat au Régime des PPP. Ce comité est composé d’au moins douze représentants issus du secteur public et privé. Les avis et procès-verbaux des délibérations du Comité d’Orientation sont transmis au Premier Ministre qui dispose à cet égard d’un pouvoir de réformation. Le Comité d’Orientation se réunit en tant que de besoin sur convocation de son président.

#### Le secrétariat technique

Est l’équipe opérationnelle qui est composée du :

* du Coordonnateur Technique ;
* des experts ;
* du personnel administratif.

Le secrétariat est sous l’autorité du Président du Conseil et a à sa tête un Coordonnateur Technique chargé de préparer le budget, les programmes d’actions, les rapports d’activités, ainsi que les comptes et les états financiers du Conseil qu’il soumet au Comité d’Orientation pour approbation et arrêt. Il prépare et soumet au Comité d’Orientation la grille de rémunération et des avantages des personnels du Conseil. Le secrétariat technique compte, à titre permanent, un effectif de douze Experts, de huit personnels administratifs.

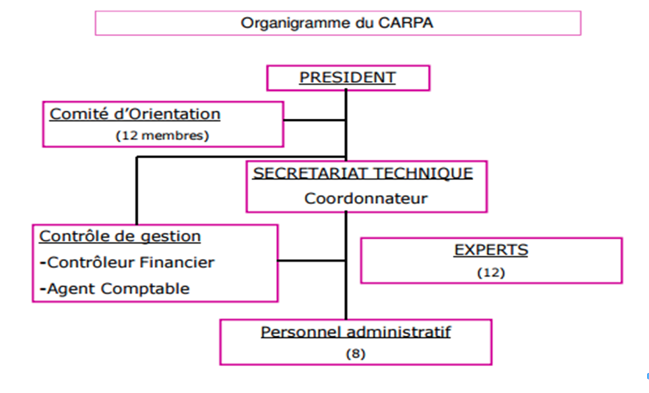


Figure 1 présentation de l’organigramme du CARPA

### 1.2.2. Fonctionnement

Pour fonctionner les ressources financières utilisées sont issues de diverses sources. Ces ressources permettent à ses différents organes de mener à bien les activités prévues pour l’accomplissement des missions qui leurs sont assignées entre autres on peut citer :

* la dotation annuelle inscrite au budget de l’État ;
* les contributions éventuelles du secteur privé et des partenaires au développement ;
* les dons et legs ;
* toutes autres ressources qui pourraient lui être affectées.

Ses ressources financières sont des deniers publics gérées suivant les règles prévues par le régime financier de l’État.

## 1.3. Quelques réalisations du CARPA

Bien que ses activités ne soient pas encore tout à fait connues du grand public, le **CARPA** est mobilisé autour d’un certain nombre important de projets rendus à différentes étapes de leur réalisation comme suit :

| **Code projet** | **Objet du projet** | **Situation du projet** | **Partenaire public** | **Partenaire privé** | **Montant de l’investissement** |
| --- | --- | --- | --- | --- | --- |
| P12-001 | Construction, gestion et maintenance du marché Congo à Douala | Phase de construction | Communauté Urbaine de Douala | SICC CONGO MANAGEMENT | **6 800 000 000 FCFA** |
| P12-002 | Fourniture et exploitation d’équipements d’imagerie à résonance magnétique (IRM) | Projet en exploitation | Hôpital Général de Douala | HITACHI MEDICAL SYSTEMS, représenté par NUMELEC-CMR | **760 millions F CFA** |
| P12-003 | Financement, Fourniture et maintenance des équipements de dialyse à l’HGD | Projet en exploitation | Hôpital Général de Douala | NUMELEC CAMEROUN | **220 000 000 F CFA** |
| P13-001 | Réhabilitation du laboratoire central de l’hôpital général de Yaoundé | Phase de construction | Hôpital Général de Yaoundé | NUMELEC CAMEROUN | **1.200.000.000 FCFA** |

Tableau 1 projets en cours de réalisation

| **Code projet** | **Objet du projet** | **Situation du projet** | **Partenaire public** | **Partenaire privé** | **Montant de l’investissementt** |
| --- | --- | --- | --- | --- | --- |
| **P11-002** | **Financement, construction et exploitation d’une tour de l’électricité (TOUREL)** | **Mobilisation des financements en cours** | **ARSEL-AER-EDC** | **UMNOMBO Consortium Partners** | **65 000 000 000 FCFA** |
| **P14-002** | **Approvisionnement supplémentaire de 50 000 m3 d’eau /jour dans la ville de Yaoundé** | **Levée en cours des conditions suspensives** | **ETAT/ Ministère de l’Eau et de l’Énergie** | **IMPERIAL HOLDING LTD (IHL)** | **26 milliards F CFA** |
| **P15-005** | **Modernisation du système informatique de la Douane au Cameroun** | **Levée en cours des conditions suspensives.** | **MINFI** | **CAMPASS S.A.** | **34 milliards FCFA** |

Tableau 2 projets avec contrat signé mais pas encore démarré

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Code projet** | **Objet du projet** | **Situation du projet** | **Partenaire public** | **Partenaire privé** | **Montant de l’investissement** |
| **P12-007** | **Construction, gestion et maintenance d’un Terminal hydrocarbures au Port de Kribi** | **Négociations du contrat en cours** | **ETAT /COPIL-CIPK** | **SCDP-BLAZE** |  |
| **P12-008** | **Exploitation du Terminal Polyvalent du port de Kribi** | **Finalisation des contrats** | **ETAT/COPIL-CIPK** | **NECOTRANS/KPMO** |  |
| **P12-009** | **Exploitation et maintenance du terminal à conteneurs au Port en Eau profonde de Kribi** | **Elaboration des annexes en cours** | **ETAT/COPIL-CIPK** | **BOLLORE/CGM-CMA/CHEC** |  |
| **P13-003** | **Autoroute Kribi-Lolabé** | **Analyse des offres** | **MINTP** |  |  |
| **P14-005** | **Construction et maintenance de lampadaires solaires pour éclairage public dans certaines villes du Cameroun** | **Analyse des offres** | **MINHDU/CTD** |  |  |
| **P13-005** | **Gare routière de SODIKO** | **Analyse des offres** | **CUD** |  |  |

Tableau 3 projets en cours de procédure

Plusieurs autres projets dont les études ont été récemment soumises au **CARPA** sont en phase d’évaluation préalable en vue de leur éligibilité ou non au régime des contrats de partenariat. Il faut également noter que certaines données des différents tableaux ci-dessus ont été mis à jour pendant la période de notre stage au **CARPA**

## 1.4. Situation géographique

Le **CARPA** a pour le moment une seule localisation dans la ville de **Yaoundé.** Les bureaux actuels se trouvent à **bastos** au lieu-dit « ancien CAMTEL bastos » et occupe le premier étage. On peut s’y rendre en suivant le plan suivant :

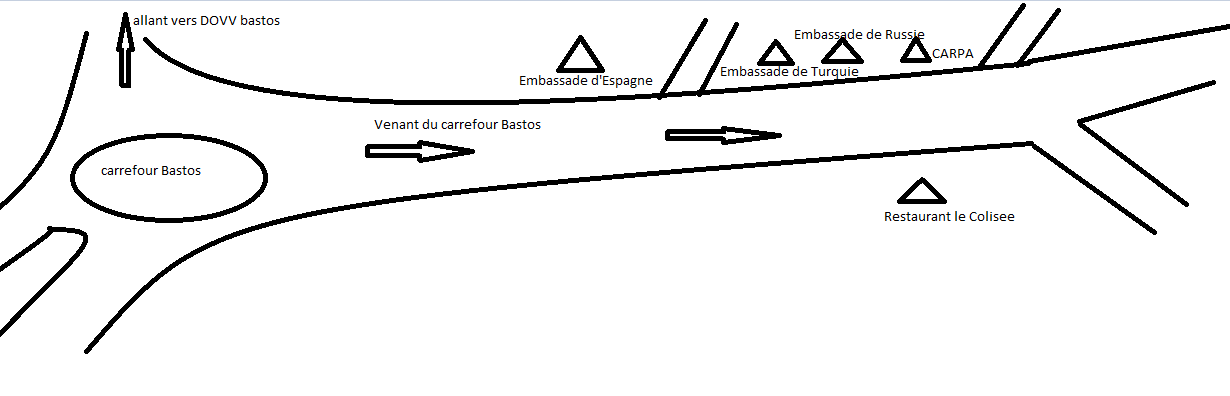


Figure 2 Plan de localisation du CARPA

## 1.5. Accueil en institution

Le Lundi **onze Mars 2018,** date de début de notre stage au **CARPA,** nous avons été accueillis par Monsieur **NDJANA Willy Freddie** Expert technique qui devait être notre encadreur professionnel durant toute cette période. Nous avons par la suite été présentés au Coordonnateur du Secrétariat Technique Monsieur **Marcel Blaise Mbella** qui nous a introduit auprès du Président du conseil.

Les premiers échanges avec ses responsables nous ont permis de disposer d’un certain nombre d’informations sur les missions de cette structure, l’attitude à adopter pendant toute cette période ainsi que les objectifs assignés à tout stagiaire.

## 1.6. Contexte et Problématique

L’analyse de l’existant est une étape importante dans le cycle de vie d’un système, il s’agit de connaitre la situation actuelle de l’organisation pour pouvoir porter un jugement juste. Ainsi, l’analyse du système existant doit nous fournir toute l’information nécessaire, afin d’établir une bonne conception et de proposer de bonnes solutions.

### 1.6.1. Contexte

L’étude de l’existant est une phase pour bien comprendre le système actuel et définir ses objectifs. En effet le CARPA ne dispose réellement pas d’un service informatique, il y’a une salle faisant office de salle serveur qui contient le routeur du fournisseur d’accès internet, un switch et un modem qui permet de fournir la connexion internet au personnel qui ont tous un ordinateur portable ou d’un pc fixe fournit en connexion internet et disposant d’outils nécessaires pour leurs tâches quotidiennes. Le **CARPA** pour le suivi des projets dispose d’un fichier concernant les informations relatives aux différents projets soumis aux **CARPA**. Pour ce qui est du courrier les assistantes disposent des registres ou elles notent les informations concernant ces différents courriers qui arrivent et sortent du **CARPA,** quant aux archives elles sont faites manuellement et classé dans les armoires.

### 1.6.2. Problématique

En effet, une analyse de l’existant au sein du CARPA a mis en exergue un certain nombre de problèmes liés au suivi des projets et à la gestion, le suivi et l’archivage du courrier.

S’agissant du suivi des projets, on a pu constater l’utilisation d’un tableau de bord de projet réalisé sur le logiciel Microsoft Word dont la consultation permet de renseigner l’état actuel d’un projet. Les problèmes identifiés à ce niveau sont les suivants :

* la perte de la mémoire des projets ;
* l’indisponibilité d’un outil d’archivage des principales données et documents de projet.

Pour ce qui est de la gestion, du suivi et de l’archivage du courrier, nous avons constaté la présence d’un registre dans lequel sont renseignées les informations concernant les courriers entrants et sortants, mais également leur affectation au sein du CARPA. Les problèmes identifiés ici sont les suivants :

* l’enregistrement et le suivi des courriers entrant et sortant du CARPA sont fastidieux en ce sens qu’il est possible que le traitement y relatif s’étale sur de longues minutes ;
* le risque de perte ou de détérioration registre entrainant la perte des informations ou données pendant une période.

## 1.7. Objectifs et méthodologie

### 1.7.1. Objectifs

Nous avons pour objectif général de mettre en place une **application web de suivi des projets, du courrier et d’archivage numérique au CARPA.** Les objectifs spécifiques sont :

* Offrir une interface conviviale pour les assistantes et Experts, dans le but de mieux organiser leur travail.
* Offrir une interface d’administration, qui permettra le suivi des utilisateurs et l’inspection du travail fait par ces derniers.
* Fournir une application qui facilitera l’archive.

### 1.7.2. Méthodologie

Pour atteindre les objectifs énoncés plus hauts, nous avons mis en œuvre la démarche incrémentale qui n’est rien d’autre que le modèle en V dont les détails sont les suivantes :

* **Analyse des besoins :** i l est question de collecter les besoins des futurs utilisateur de notre application, en ce qui concerne les données à traiter dans l’application et les exigences métier.
* **Modélisation et conception :** c’est à cette étape que nous avons exprimé notre vision du système en tenant compte des besoins des futurs utilisateurs. Pour ce faire , il a été question de modéliser le système à l’aide d’un langage de modélisation ou d’une méthode d’analyse spécifique. Tout au long de cette partie, nous tâcherons de répondre à la question « qu’est-ce qu’il faut ? » pour l’analyse et non au « comment ? » Pour la conception. Ainsi nous pourrons percevoir l’application sous toutes ses vues.(l’architecture, les fonctionnalités, la base de données de l’application)
* **L’implémentation :** on s’intéressera aux méthodes et outils nécessaires à la réalisation des vues. Il sera donc de construire les différentes entités que nous avons conçues. A l’aide d’un langage de programmation, de Framework et d’autres outils, nous allons mettre en œuvre les différentes vues de notre application.
* **Les tests et la maintenance :** il sera question ici de vérifier avec les utilisateurs du système, que les exigences sont respectées. En outre les livrables du projet devront aussi être testés. Ceci entrainera la correction des bugs et l’ajustement des fonctionnalités.

Nous avons la figure suivante qui présente ces étapes schématiquement.

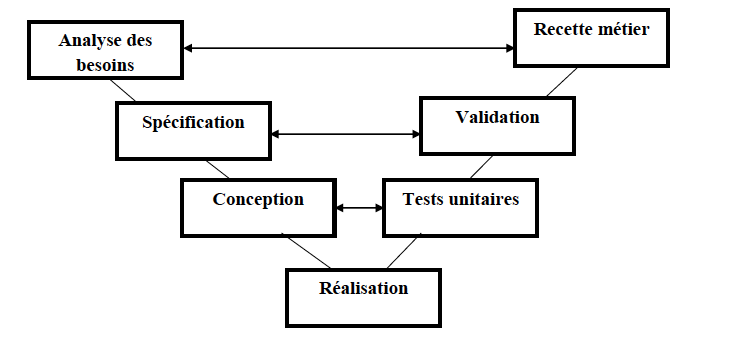


Figure 3 Modèle en V

## Conclusion

En somme il était question dans ce chapitre, de présenter le contexte de notre travail, sa problématique, sa méthodologie et les objectifs. Dans le chapitre suivant, il sera question pour nous de présenter les généralités liées à notre travail.

|  |  |
| --- | --- |
| : GENERALITES |  |

## Introduction

Dans l’accomplissement d’un travail scientifique, on s’appuie « toujours » sur des résultats d’autres recherches. Il devient donc très indispensable, dans un tel projet, de présenter les outils et techniques qui ont été́ utilisés pour parvenir à une bonne fin.

Dans ce chapitre des généralités, nous allons présenter la technique Rapid Application Développent, puis l’outil Ruby On Rails qui la met en valeur. Dans un souci de respect des principes du génie logiciel, pour éviter des erreurs de régression et pour mieux gérer les versions, nous utiliserons l’outil Git que nous présentons également dans ce chapitre. L’architecture MVC, le langage UML et la méthode de développement Agile feront l’objet des dernières présentations.

## 2.1. Rapid Application Development

Le Rapid Application Development (RAD), francisé en « Robot Automatique de Développement », est une méthode de développement rapide de logiciels. Il se fait généralement au moyen des outils tels que des AGL (Atelier de Génie Logiciel) ou des L4G (Langages de Quatrième Génération).

Partant du résultat de la modélisation, ils doivent pouvoir générer une application avec des fonctionnalités basiques telles que l’ajout, la modification et la suppression d’une entité́ ; également l’affichage des données par liste. On parle généralement de CRUD (Create, Read, Update, Delete).

Ces outils sont des ensembles de programmes permettant la conception de programmes, d’applications ou de systèmes parfois très complexes. Ils sont généralement formés par des langages puissants et évolués, accompagnés d’utilitaires de création d’interfaces graphiques des programmes générés. Exemple : Ruby On Rails, Sinatra, Visual Basic, Django.

Avec un L4G, on programme vite et c’est simple, mais le code généré́ est souvent lourd et très lent, sans véritable optimisation. De plus, on n’a que rarement accès aux entrailles de son programme, et s’il ne fonctionne pas, on peut mettre beaucoup de temps à diagnostiquer.

Il est donc nécessaire pour le programmeur de trouver l’outil qui répondra à ses attentes. Un outil qui soit compatible au langage utilisé, qui soit le plus « open source » possible et qui procure une architecture familière. Ruby On Rails et Sinatra sont des outils destinés à la plateforme Ruby et qui utilisent des architectures en MVC. Ruby On Rails a particulièrement attiré notre attention.

## 2.2. L’outil Ruby On Rails

Dans le monde Ruby, ROR, c’est-à-dire Ruby On Rails, apporte une réponse à la concurrence des Framework de développement rapide tels que Groovy on rails ou Django. Il permet de créer rapidement des applications WEB. Il a été́ initié en 2005 par David Heinemeier Hansson c’est un Framework web open source écrit en Ruby sous licence MIT. Il suit le modèle-vue-contrôleur(MVC). Il propose une structure qui de développer rapidement et intuitivement (référence :wikipedia.org/wiki/RubyOnRails). Il s’installe d’une manière très particulière notamment en ligne de commande après avoir installé́ Ruby. On peut l’utiliser en ligne de commande ou via des IDE qui l’intègrent. Sa commande principale est Rails. L’exécution de cette commande dans une invite de commandes Mac nous donne l’aperçu suivant :

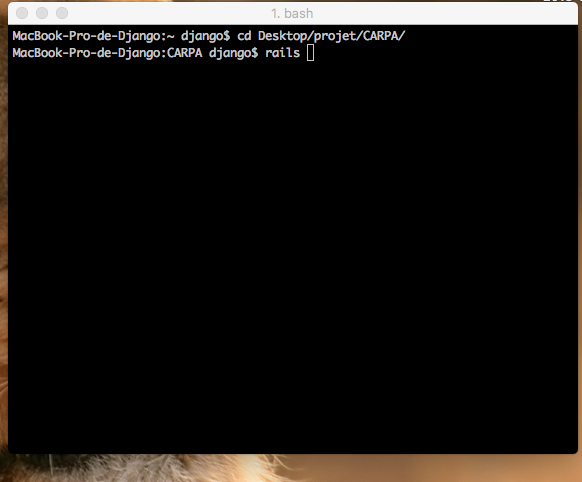


Figure 4 Terminal

* **Les modèles** sont des classes servant à modéliser les données et à établir les relations entre elles. Cela permet d’établir le mapping entre les Objets et la base de données, grâce aux outils tels qu’**Active Record**. Il est possible d’effectuer des requêtes de base sur les modèles. On trouve sur chaque modèles des méthodes save(), create(), find\_by\_name(),find() etc... Ces méthodes n’ont jamais été́ définies par le développeur, mais elles existent grâce au Framework. Il est également possible de contrôler la validation des formulaires depuis le fichier qui définit le domaine.
* **Les Contrôleurs**. Un contrôleur est une classe qui reçoit la requête de l’utilisateur et qui, en fonction de l’action demandée, va effectuer le traitement. Pour gérer quelle est l’action et quel contrôleur est appelé́, Rails se base sur le formatage de l’URL : **: http ://<. . . >/controller/action/**.
* **Les Vues** sont représentées par un moteur de template **erb**, ou du **Haml**, on peut insérer du code Ruby. On constate dans l’arborescence qu’il y’a une vue dédiée à chaque contrôleur et une vue dédiée à une action du contrôleur (Voir Annexe).

## 2.3. Le langage Ruby

Le langage Ruby a été́ conçu, au milieu des années 90, par Yukihiro Matsumoto, un programmeur Japonais. Son objectif était d’avoir un langage qui soit « plaisant » à utiliser : Ruby is **« made for developer happiness ! »** (reference).C’est un Langage Orienté Objet. La figure ci-dessous présente un arbre généalogique de Ruby. Quant au tableau, il représente les ancêtres de Ruby, avec les principales caractéristiques héritées de ces ancêtres.

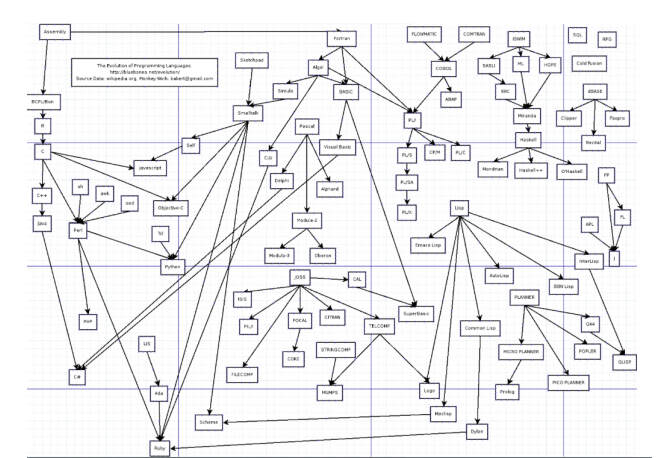


Figure 5 Arbre généalogique de Ruby

| **Langage** | **Année** | **Caractéristiques** |
| --- | --- | --- |
| LISP | 1958 | Approche fonctionnelle métaprogrammation |
| CLU | 1974 | Littérateurs |
| Smalltalk | 1980 | Langage objet pur, blocs de code GUI, sUnit |
| Eiffel | 1986 | Uniform Access Principle |
| Perl | 1987 | Expressions régulières et pattern matching |
| Ruby | 1993 |  |

Tableau 4 Les ancêtres de Ruby

La syntaxe de Ruby est faite pour apporter plus de flexibilité́ au framework Ruby On Rails.

## 2.4. Architecture et création d’un projet avec Ruby On Rails

La création d’un projet ROR se fait en trois étapes :

* Création des classes et modèles ;
* Scaffolding ;
* Running

### 2.4.1. Création des classes et modèles

Les « migrations » sont définitions du modèle. C’est la première étape dans la création d’un projet Rails. Nous avons utilisé dans ce projet la version 5.1.6. La commande rails new nomProjet permet de créer un projet nommé nomProjet. Rails va construire un nomProjet avec un contenu tel que représenté́ par la figure ci-après.

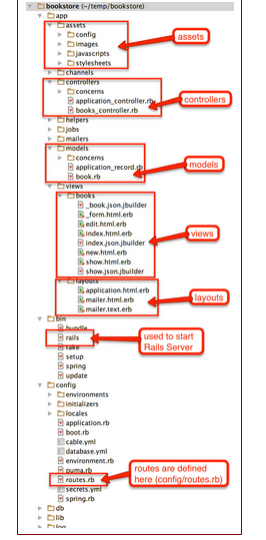


Figure 6 contenu d'un dossier ROR

## 2.5. Scaffolding

Le scaffolding est une action de mise sur pied d’un échafaudage. C’est là que repose la génération des futures fonctionnalités. Il se gère au niveau des controllers et permet de spécifier les actions du contrôleur. La commande rails qui permet de créer un contrôleur pour le domaine Étudiant est : rails generate controller Etudiants. Un fichier EtudiantsController sera créé́ dans le dossier app/controllers.

Le scaffold prend un domaine pour lequel le contrôleur définira les fonctionnalités CRUD. Ainsi, on pourra créer un Étudiant, modifier ses champs et le supprimer ; également, l’affichage de la liste des instances d’Étudiant crées sera disponible

### 2.5.1. Running

L’exécution d’un projet rails se fait par la commande rails server. On verra sur la console, le message Server Puma Booting http ://localhost :3000.

Rails nous invite ainsi à exécuter l’application sur un navigateur à l’adresse indiquée. Le navigateur nous présentera l’interface suivante :

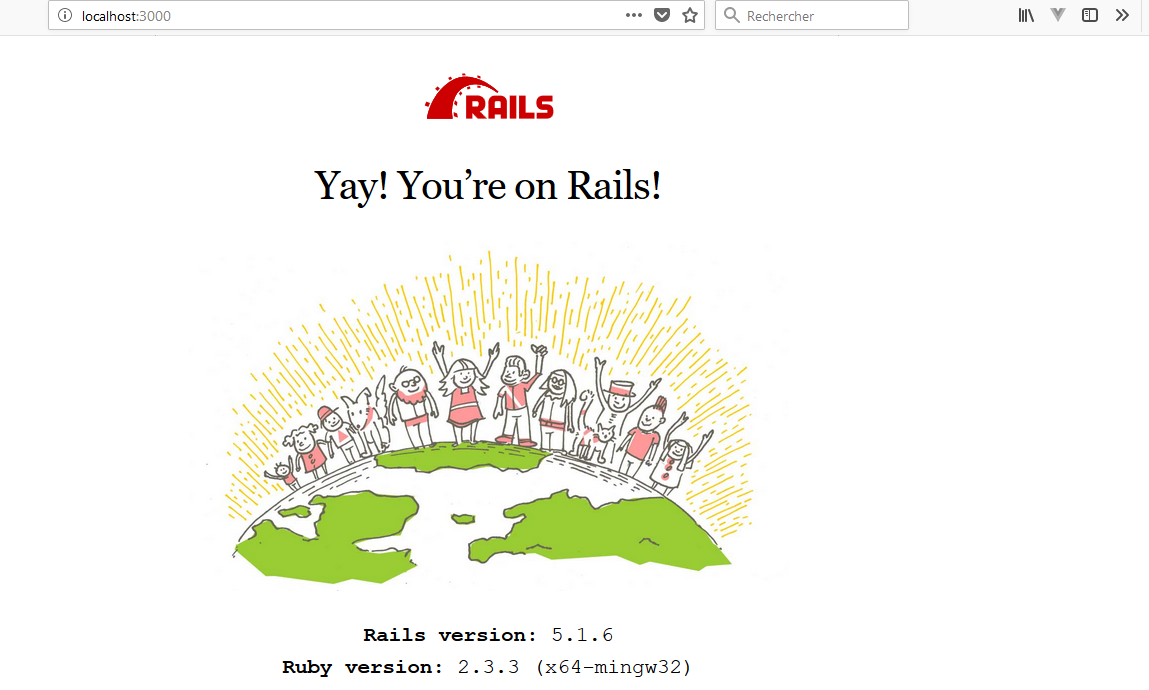
****

Figure Première page d'un projet ROR

On peut constater que son serveur embarqué utilise par défaut le port 3000. Mais, il peut être modifié si un autre serveur (tomcat par exemple) utilise déjà̀ ce port. Dans ce cas, la commande d’exécution, pour le port 3001 par exemple, sera rails server -p 3001.

## 2.6. Les avantages de Ruby On Rails

Les avantages que l’on peut avoir dans l’utilisation de Rails comme RAD, peuvent être les suivants :

* Web MVC : Facile à utiliser ;
* ERB : Langage de Template simple et complet ;
* Serveur embarqué ;
* Active Record : Modélisation et accès aux données ;
* Base de données simulée(développement) ;
* Internationalisation ;
* Tests & Tests unitaires ;
* Documentation très riche.

## 2.7. L’outil Git

Il n’est pas toujours évident de parvenir rapidement au but que l’on se fixe dans la réalisation programmes. Pour ne pas rendre la tâche encore plus complexe, il peut être nécessaire d’utiliser des méthodes et outils permettant de ne pas s’éloigner du but. L’outils, Git, que nous présentons dans cette partie nous permet de détecter les erreurs de régression et de gérer les versions, respectivement.

### 2.7.1. Git

Toujours dans un souci d’efficacité́ et de rapidité, le programmeur ne doit pas se permettre une mauvaise gestion des versions. Une version est tout simplement l’état du projet à un moment donné. On peut toujours avoir besoin, dans un projet, de retrouver un état antérieur. Git est un outil qui permet de sauvegarder et de restaurer des états.

Lorsque Git est bien installé, pour commencer à suivre un projet existant dans Git, il suffit de se positionner dans le répertoire du projet et exécuter la commande git init. Cela y va créer un sous-dossier nommé. git. La commande git add fichier permet d’indexer les fichiers du projet qui seront suivis en version.

Pour cloner un projet existant, on fait git clone urlProjet. La commande git commit permet de valider les modifications effectuées sur les fichiers indexés.

Le cycle de vie des états d’un fichier Git peut-être illustré par la figure ci-dessous :

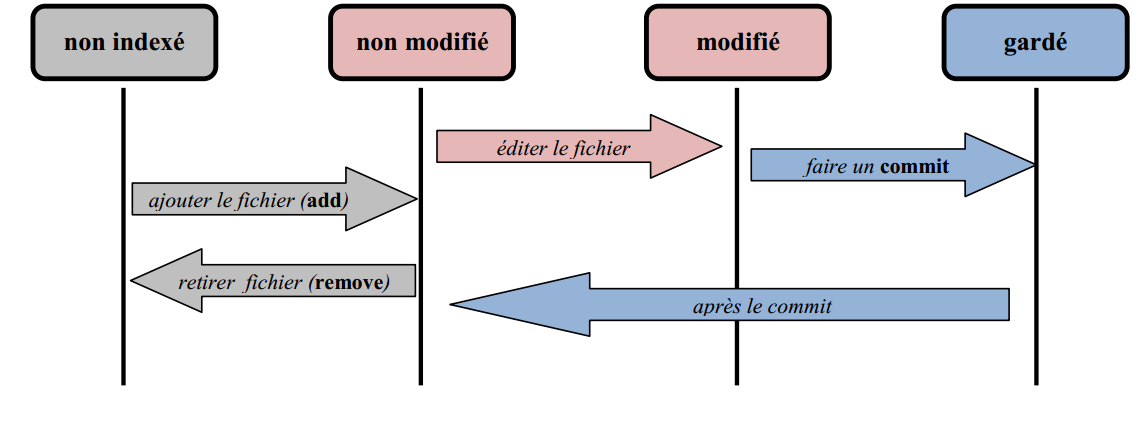


Figure 8 Cycle de vie des états d'un fichier git

## 2.8. Le pattern MVC

### 2.8.1. Définition

Un pattern est « une solution à un problème dans un contexte bien défini ». Ses quatre éléments d’un pattern sont :

* Son nom ;
* Son but ;
* Comment il résout le problème ;
* Les contraintes à considérer dans la solution.

Les patterns nous permettent de réutiliser les solutions qui ont marché pour les autres ; « pourquoi réinventer la roue ? ». Ils nous permettent également d’avoir un vocabulaire commun.

### 2.8.2. Présentation du modèle MVC

MVC (Modèle – Vue – Contrôleur) est un design de conception d’interface utilisateur permettant de découpler le modèle (logique métier et accès aux données) des vues (interfaces utilisateur).

Des modifications de l’un n’auront ainsi, idéalement, aucune conséquence sur l’autre ce qui facilitera grandement la maintenance.

### 2.8.3. Le principe du MVC

Modèle : gère les données et reprend la logique métier (le modèle lui-même peut être décomposé en plusieurs couches mais cette décomposition n’intervient pas au niveau de MVC). Le modèle ne prend en compte aucun élément de présentation !

Vue : elle affiche les données, provenant exclusivement du modèle, pour l’utilisateur et/ou reçoit ses actions. Aucun traitement, autre que la gestion de présentation, n’y est réalisé́.

Contrôleur : son rôle est de traiter les évènements en provenance de l’interface utilisateur et les transmet au modèle pour le faire évoluer ou à la vue pour modifier son aspect visuel (pas de modification des données affichées mais des modifications de présentation). Le contrôleur « connait » la (les) vue(s) qu’il contrôle ainsi que le modèle.

Il pourra appeler des méthodes du modèle pour réagir à des évènements, il pourra faire modifier à la vue son aspect visuel. Il pourra aussi instancier de nouvelles vues. Pour faire cela, le contrôleur sera à l’écoute d’évènements survenant sur les vues.

La vue observera le modèle qui l’avertira du fait qu’une modification est survenue. Dans ce cas, la vue interrogera le modèle pour obtenir son nouvel « état ».

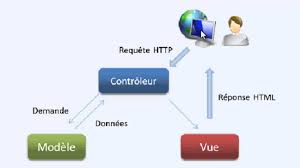


Figure 9 Schématisation du MVC

## 2.9. Constructeur d’un projet UML

UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue.

UML unifie à la fois les notations et les concepts orientés objet. Il ne s’agit pas d’une simple notation graphique, car les concepts transmis par un diagramme ont une sémantique précise et sont porteurs de sens au même titre que les mots d’un langage.

UML unifie également les notations nécessaires aux différentes activités d’un processus de développement et offre, par ce biais, le moyen d’établir le suivi des décisions prises, depuis l’expression de besoin jusqu’au codage.

Le fil tendu entre les différentes étapes de construction permet alors de remonter du code aux besoins et d’en comprendre les tenants et les aboutissants. En d’autres termes, on peut retrouver la nécessité́ d’un bloc de code en se référant à son origine dans le modèle des besoins.

UML 2 s’articule autour de treize types de diagrammes, chacun d’eux étant dédié́ à la représentation des concepts particuliers d’un système logiciel. Ces types de diagrammes sont repartis en deux grands groupes :

* **Diagrammes structurels**
* Diagramme de classe :
* Diagramme d’objets
* Diagramme de packages
* Diagramme de structure composite
* Diagramme de composants
* Diagramme de déploiement
* **Sept diagrammes comportementaux**
* Diagramme de cas d’utilisation
* Diagramme de vue des interactions
* Diagramme de communication
* Diagramme de temps
* Diagramme d’activité
* Diagramme d’état

## 2.10. Développement Agile

### 2.10.1. Principe du Manifeste Agile

La notion de méthode agile est née à travers un manifeste signé en 2001 par 17 personnalités du développement logiciel

Ce manifeste prône quatre valeurs fondamentales :

* « Personnes et interactions plutôt que processus et outils » : dans l’optique agile, l’équipe est bien plus importante que les moyens matériels ou les procédures. Il est préférable d’avoir une équipe soudée et qui communique, composée de développeurs moyens, plutôt qu’une équipe composée d’individualistes, même brillants. La communication est une notion fondamentale.
* « Logiciel fonctionnel plutôt que documentation complète » : il est vital que l’application fonctionne. Le reste, et notamment la documentation technique, est secondaire, même si une documentation succincte et précise est utile comme moyen de communication. La documentation représente une charge de travail importante et peut être néfaste si elle n’est pas à jour. Il est préférable de commenter abondamment le code lui-même, et surtout de transférer les compétences au sein de l’équipe (on en revient à l’importance de la communication).
* « Collaboration avec le client plutôt que négociation de contrat » : le client doit être impliqué dans le développement. On ne peut se contenter de négocier un contrat au début du projet, puis de négliger les demandes du client. Le client doit collaborer avec l’équipe et fournir un feedback continu sur l’adaptation du logiciel à ses attentes.

« Réagir au changement plutôt que suivre un plan » : la planification initiale et la structure du logiciel doivent être flexibles afin de permettre l’évolution de la demande du client tout au long du projet. Les premières releases du logiciel vont souvent provoquer des demandes d’évolution

### 2.10.2. Modélisation Agile(AM)

La « modélisation agile » prônée par Scott Ambler s’appuie sur des principes simples et de bon sens, parmi lesquels :

* Vous devriez avoir une grande palette de techniques à votre disposition et connaître les forces et les faiblesses de chacune de manière à pouvoir appliquer la meilleure au problème courant.
* N’hésitez pas à changer de diagramme quand vous sentez que vous n’avancez plus avec le modèle en cours. Le changement de perspective va vous permettre de voir le problème sous un autre angle et de mieux comprendre ce qui bloquait précédemment.
* Vous trouverez souvent que vous êtes plus productif si vous créez plusieurs modèles simultanément plutôt qu’en vous focalisant sur un seul type de diagramme.

## Conclusion

Dans ce chapitre des généralités, il a été́ question de présenter certains outils utilisés dans le cadre de notre travail. Nous avons à cet effet présenté la technique du RAD, les outils Rails et Git, l’architecture MVC, le langage UML et la méthode de développement Agile. Notons également que tout le long du projet, nous avons adopté́ une attitude « Agile », telle que décrite ci-dessus.

|  |  |
| --- | --- |
| : ANALYSE ET CONCEPTION |  |

## Introduction

Pour mettre sur pied un logiciel au sein d’une entreprise on passe par la réalisation d’un document appelé́ cahier de charges.

L’IEEE 1233 (ref ici) définit un cahier de charge comme étant l’expression d’un besoin à satisfaire. Le cahier de charge n’indique pas la manière de réaliser le besoin, ni un produit à fournir, il est en amont de la conception. C’est donc le seul document qui décrit les besoins d’un utilisateur en termes de fonctions à assurer et d’objectifs atteindre.

Dans ce chapitre nous allons en première phase spécifier les besoins fonctionnels pour la mise en place d’une application du suivi de projet, du courrier et d’archivage numérique. Cette phase décrit les exigences auxquelles la solution à mettre en place devra répondre, en termes de fonctionnalités, de caractéristiques fonctionnelles attendues et de la faisabilité́ du logiciel. La deuxième phase constituera essentiellement la conception de notre application. Il s’agira de l’établissement du diagramme des cas d’utilisation, des dia- grammes de séquences et des diagrammes de classes pour mieux comprendre son fonctionnement, afin de maitriser sa complexité́ et d’assurer sa cohérence.

## 3.1. Cahier de charges

### 3.1.1. Fonctionnalités attendues de l’application

Dans un début, il est important de recenser les propriétés fonctionnelles de de notre application. A la fin de ce projet notre application devrait être capable de :

### 3.1.2. Besoins de l’application

La spécification des besoins décrit sans ambiguïté́ l’application sécurisée à développer. L’énoncé d’un besoin exprime un comportement ou une propriété́ que le système doit respecter. Chaque énoncé́ traduit la présence d’un comportement très spécifique.

* Un identifiant unique ;
* La catégorie du besoin qui décrit si celui-ci est fonctionnel ou non fonctionnel ;
* La description ;
* Une liste de termes le référant ;
* La justification de la présence et de l’utilité́ du besoin ;
* La priorité́ du besoin par l’une des appréciations suivantes : haute, moyenne, faible ;
* La vérification qui décrit un moyen avec lequel le client peut se rassurer tel qu’implémenté, le besoin est rempli.

### 3.1.3. Besoins fonctionnels

Les besoins fonctionnels expriment des actions que doit effectuer l’application en réponse à une demande (sorties qui sont produites pour un ensemble donné d’entrées) . Pour mieux recenser ces besoins nous avons opté pour un découpage en module.

#### Le module de gestion des utilisateurs

Dans ce module nous pouvons énumérer les besoins suivants :

* Créer un utilisateur

|  |  |
| --- | --- |
| **Identifiant** | F1 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra l’ajout des données des utilisateurs qui seront les futurs utilisateurs du système |
| **Termes** | Administration, données, utilisateur |
| **Justification** | Plusieurs responsables du CARPA utiliseront l’application pour cela, nous devons les identifier afin de leur créer des comptes pour une bonne traçabilité |
| **Priorité** | Haute |
| **Vérification** | Après enregistrement d’un utilisateur, on vérifie dans la liste des utilisateurs si ce dernier existe |

Tableau 5 Enregistrement d'un utilisateur

* Modifier un utilisateur

|  |  |
| --- | --- |
| **Identifiant** | F2 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de modifier les données d’un utilisateur. |
| **Termes** | Administration, données, utilisateur, modification |
| **Justification** | L’administrateur peut mal saisir les données |
| **Priorité** | Moyenne |
| **Vérification** | Après la modification d’un utilisateur, on vérifie dans la liste des utilisateurs si la modification est effective. |

Tableau 6 modifier un utilisateur

* Supprimer un utilisateur

|  |  |
| --- | --- |
| **Identifiant** | F3 |
| **Catégorie** | Fonctionnelle |
| **Description** | Supprimer un utilisateur |
| **Termes** | Administration, données, utilisateur, suppression |
| **Justification** | Le CARPA peut décider de rompre le contrat d’un des responsable ou ce dernier peut être mis à la retraite |
| **Priorité** | Moyenne |
| **Vérification** | Après la suppression d’un utilisateur, on vérifie si l’utilisateur n’est plus dans a liste |

Tableau 7 supprimer utilisateur

* Créer un compte utilisateur

|  |  |
| --- | --- |
| **Identifiant** | F4 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra l’ajout des comptes d’utilisateurs. |
| **Termes** | Administration, donnée, compte |
| **Justification** | Plusieurs utilisateurs sont dans le système et chacun doit avoir un compte pour se connecter à l’application. |
| **Priorité** | Haute |
| **Vérification** | Après enregistrement d’un compte, on vérifie dans la liste des comptes si ce dernier existe. |

Tableau 8 enregistrer un compte utilisateur

* Modifier un compte

|  |  |
| --- | --- |
| **Identifiant** | F5 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de modifier les données d’un compte. |
| **Termes** | Administration, donnée, compte, modification |
| **Justification** | L’administrateur peut mal saisir les données, ou alors le profil de l’utilisateur peut changer. |
| **Priorité** | Moyenne |
| **Vérification** | Après la modification d’un compte, on vérifie dans la liste des comptes si ce dernier existe. |

**Tableau 9 modifier un compte utilisateur**

* Effectuer des recherches sur les utilisateurs

|  |  |
| --- | --- |
| **Identifiant** | F6 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire une recherche rapide sur les utilisateurs. |
| **Termes** | Administration, données, utilisateur, modification, recherche. |
| **Justification** | L’administrateur peut avoir besoin des informations sur une personne. |
| **Priorité** | Moyenne |
| **Vérification** | Si la personne existe dans le système alors elle sera affichée. |

Tableau 10 rechercher un utilisateur

#### Module gestion du courrier

* Créer un courrier

|  |  |
| --- | --- |
| **Identifiant** | F7 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire une recherche rapide sur les courriers. |
| **Termes** | Administration, données, courrier. |
| **Justification** | L’administrateur ou l’utilisateur peut avoir besoin des informations sur un courrier. |
| **Priorité** | Haute |
| **Vérification** | Si le courrier existe dans le système alors il sera affiché. |

Tableau 11 enregistrer un courrier

* Modifier un courrier

|  |  |
| --- | --- |
| **Identifiant** | F8 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire une modification sur les courriers. |
| **Termes** | Administration, données, courrier, modification. |
| **Justification** | L’administrateur ou l’utilisateur peut avoir besoin des informations sur un courrier. |
| **Priorité** | Moyenne |
| **Vérification** | Après la modification, on vérifie dans la liste des courriers si ce dernier est à jour. |

Tableau 12 modifier un courrier

* Supprimer un courrier

|  |  |
| --- | --- |
| **Identifiant** | F9 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire la suppression d'un courrier. |
| **Termes** | Administration, données, courrier, suppression. |
| **Justification** | CARPA peut décider de ne plus avoir besoin des informations d'un courrier. |
| **Priorité** | Moyenne |
| **Vérification** | Après la suppression, on vérifie dans la liste des courriers si ce dernier a été supprimer. |

Tableau 13 supprimer un courrier

#### Module suivi des projets

* Enregistrer un projet

|  |  |
| --- | --- |
| **Identifiant** | F10 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire l’enregistrement d'un projet. |
| **Termes** | Administration, données, projet. |
| **Justification** | L'administrateur a besoin des données qui permettrons de faire le suivi des projets |
| **Priorité** | Haute |
| **Vérification** | Après l’enregistrement, on se rassure que ce dernier est bien dans la liste. |

Tableau 14 enregistrer projet

* Modifier un projet

|  |  |
| --- | --- |
| **Identifiant** | F11 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire la modification d'un projet. |
| **Termes** | Administration, données, projet, modification. |
| **Justification** | L'administrateur ou l'utilisateur peut faire une erreur dans la saisie d'une d'un projet et souhaite faire une modification |
| **Priorité** | Moyenne |
| **Vérification** | Après la modification d'une phase du projet, on vérifie s'il est dans la liste des phases du projet. |

Tableau 15 modifier projet

* Supprimer un projet

|  |  |
| --- | --- |
| **Identifiant** | F12 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire la suppression d'un projet. |
| **Termes** | Administration, données, projet, suppression. |
| **Justification** | CARPA peut décider de ne plus avoir besoin des informations d'un projet |
| **Priorité** | Moyenne |
| **Vérification** | Après la suppression, on vérifie dans la liste des projets si ce dernier a disparu. |

Tableau 16 supprimer projet

#### Module gestion phase du projet

* Enregistrer une phase du projet

|  |  |
| --- | --- |
| **Identifiant** | F13 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire l'enregistrement d'une phase du projet. |
| **Termes** | Administration, données, phase projet. |
| **Justification** | L'administrateur ou l’utilisateur a besoin des données concernant les phase du projet pour le suivi des projets. |
| **Priorité** | Haute |
| **Vérification** | Après l’enregistrement, on vérifie dans la liste des phases du projet ce dernier est bien enregistrée. |

Tableau 17 enregistrer phase du projet

* Modifier phase du projet

|  |  |
| --- | --- |
| **Identifiant** | F14 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire la modification d'une phase du projet. |
| **Termes** | Administration, données, phase du projet, modification. |
| **Justification** | L'administrateur ou l'utilisateur peut faire une erreur dans la saisie d'une phase du projet et souhaite une modification |
| **Priorité** | Moyenne |
| **Vérification** | Après la modification d'une phase du projet, on vérifie s'il est dans la liste des phases du projet. |

Tableau 18 modifier phase du projet

* Supprimer phase du projet

|  |  |
| --- | --- |
| **Identifiant** | F15 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire la suppression d'une phase du projet. |
| **Termes** | Administration, données, phase du projet, suppression. |
| **Justification** | CARPA peut décider de ne plus avoir besoin des informations d'une phase du projet |
| **Priorité** | Moyenne |
| **Vérification** | Après la suppression, on vérifie dans la liste des phases du projet si ce dernier a disparu. |

Tableau 19 supprimer phase du projet

#### Module archivage

* Enregistrer une archive

|  |  |
| --- | --- |
| **Identifiant** | F16 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire l’enregistrement d'une archive. |
| **Termes** | Administration, données, archive. |
| **Justification** | L'administrateur ou l’utilisateur a besoin des données concernant les archives. |
| **Priorité** | Haute |
| **Vérification** | Après l’enregistrement, on vérifie dans la liste des archives si ce dernier est bien enregistrée . |

Tableau 20 enregistrer archive

* Modifier une archive

|  |  |
| --- | --- |
| **Identifiant** | F17 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire la modification d'une archive. |
| **Termes** | Administration, données, archive, modification. |
| **Justification** | L'administrateur ou l'utilisateur peut faire une erreur dans l'enregistrement d'une archive et souhaite une modification |
| **Priorité** | Moyenne |
| **Vérification** | Après la modification d'une archive, on vérifie s'il est dans la liste des archives. |

Tableau 21 modifier archive

* Supprimer une archive

|  |  |
| --- | --- |
| **Identifiant** | F18 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra de faire la suppression d'une archive |
| **Termes** | Administration, données, archive, suppression. |
| **Justification** | CARPA peut décider de ne plus avoir besoin des informations concernant une archive. |
| **Priorité** | Moyenne |
| **Vérification** | Après la suppression, on vérifie dans la liste des archives si ce dernier a disparu. |

Tableau 22 supprimer archive

#### Module administration des données

* Importer les données sous format Excel, PDF, DOC etc

|  |  |
| --- | --- |
| **Identifiant** | F19 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra aux utilisateurs d'importer les données sous format Excel, PDF, DOC etc. |
| **Termes** | Administration, données, importé. |
| **Justification** | Ceci facilite l'administration des données et la production des rapports. |
| **Priorité** | Moyenne |
| **Vérification** | Après importation des données, ces dernières apparaissent dans la liste. |

Tableau 23 importer les données

* Exporter les données sous format excel, pdf, json, xml

|  |  |
| --- | --- |
| **Identifiant** | F20 |
| **Catégorie** | Fonctionnelle |
| **Description** | Le système permettra aux utilisateurs d'exporter les données sous format Excel, PDF, json, XML. |
| **Termes** | Administration, données, exporté. |
| **Justification** | Ceci facilite la production des rapports. |
| **Priorité** | Moyenne |
| **Vérification** | Après l'exportation un document est produit en fonction du format choisis. |

Tableau 24 exportation des données

### 3.1.4. Besoins non fonctionnels

Les besoins non spécifiques sont des besoins qui font références aux aspects généraux de l’interface utilisateur. Ce sont des besoins en matière de performance, de type de matériel ou le type de conception.

* **Ergonomie et convivialité**

L’application devra disposer d’une interface conviviale, facile et agréable à utiliser et à comprendre, permettant ainsi une appropriation rapide et intuitive par ses utilisateurs. Le portail doit être simple et transparent, laissant les utilisateurs prendre le contrôle de l’application.

* **Sécurité et fiabilité**

L’application devra envoyer des notifications pour toutes tentatives d’usurpation des droits de privilèges ou lorsqu’un champ est mal rempli. L’application devra restreindre l’utilisation de ses fonctionnalités sur le principe de responsabilité des utilisateurs et des moindres privilèges. Le droit d’accès aux ressources doit se fonder sur le principe de moindres privilèges et de la responsabilité des utilisateurs.

* **Documentation**

L’application devra être documentée :

* Au niveau du code proprement dit (commentaire des lignes de code), pour permettre en cas de besoin de comprendre, modifier ou changer le code ; ceci nous permet d'assurer la maintenance qui est l'une des bonnes qualités d'un logiciel.
* Au niveau de l'interface, considérée comme aide dans notre cas, elle permet à tout utilisateur de mieux s'accommoder aux portails afin d'assurer l'utilisabilité.

Les documents qui accompagneront la livraison de l’application sont :

* Document de conception ;
* Document de modélisation;
* Guide de l'utilisation.
* **Évolution, maintenance et réutilisation**

Pour faciliter l’évolution, la réutilisation et la maintenance, l’application devra

être développée suivant une architecture choisie sur la base des technologies utilisées

pour son développement ; mais aussi, la séparation en module devra être prise en compte

pour mieux cerner les fonctionnalités qui y sont implémentées. Et enfin les conventions

de codage telles que l’utilisation des noms significatifs, majuscules pour les constantes

etc. devront être respectées.

### 3.1.4. Étude de faisabilité

#### Justification du projet

La mise en place de cette plateforme permettra à l'entreprise de profiter au maximum des avantages suivants :

* **Gagner en temps et limiter les coûts et dépenses :** Tous les membres de l’entreprise signaleront à temps les différents problèmes rencontrés et ceux-ci seront pris en charge et résolus dans les brefs délais
* **Augmenter la productivité :** est clair que la résolution des problèmes à temps et dans les brefs délais augmente la productivité de l’entreprise.

#### Analyse des coûts et bénéfice

Les bénéfices de l’utilisation d’une telle application ont déjà été énumérés dans le paragraphe précédent. Cependant, en termes de coûts :

* L’application devra disposer d’une administration permanente ;
* Pour chaque version de l’application, les utilisateurs devront recevoir une formation sur l’utilisation de celui -ci.

#### Planning prévisionnel

Le projet se déroulera en plusieurs étapes. Les étapes, auront des activités, celles-ci seront à leur tour constituées des tâches et les durées de ces tâches, tous cela est décrit dans le tableau ci-après.

| Étapes | Activités | Tâches | Durée |
| --- | --- | --- | --- |
| la conception et la modélisation la conception et  la modélisation de l’application web | Analyse des besoins | Collecte information sur les besoins fonctionnels concernant relatif au courrier | Un jour |
| Collecte information sur les besoins fonctionnels concernant relatif au courrier | Une semaine |
| Modélisation | Diagramme de classe | Une semaine |
| Validation du Diagramme de classe | Un jour |
| Diagramme d’objet | Trois jours |
| Validation du Diagramme d’objet | Un jour |
| Diagramme de séquence | Trois jours |
| Validation du Diagramme de séquence | Un jour |
| Implémentation | Codage | Développement des codes de l’application (Back – End) | Trois semaines |
| Développement « vue utilisateur » (Front-end) | Trois semaines |
| Tests | Tests de la fonctionnalité de l’application | Une semaine |
| Présentation, formation et finalisation de l’application | Présentation de l’application | Présentation du module suivi et archivage du courrier | Un jour |
| Présentation du module suivi des projets et archivage des documents de projet | Un jour |
| Finalisation de l’application | Prise en compte des observations effectuées au cours de la présentation | Une semaine |
| Rédaction du guide d’utilisation |  | Une semaine |

Tableau 25 Planning prévisionnel

## 3.2. Modélisation

### 3.2.1. Outils de modélisation

#### UML comme Langage de modélisation

UML est un langage graphique qui permet de représenter et de communiquer les divers aspects d'un système d'informations. Aux graphiques, sont associés des textes qui expliquent leur contenu. UML est donc un métalangage, car il fournit les éléments

permettant de construire le modèle qui, lui, sera le langage du projet.

#### StarUML comme logiciel de modélisation utilisée

StarUML est un logiciel de modélisation UML, cédé comme open source par son éditeur, à la fin de son exploitation commerciale, sous une licence modifiée de GNU GPL. Il est écrit en Delphi, et dépend de composants Delphi propriétaires (non open-source), ce qui explique peut-être pourquoi il n'est plus mis à jour. StarUML gère la plupart des diagrammes spécifiés dans la norme UML 2.0.

### 3.2.2. Acteurs du système

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données. (référence ici)

Dans le cadre de notre application nous avons distingué trois acteurs principaux

* **Administrateur :** est chargée de la gestion des utilisateurs, de la gestion du suivi des projets, de la gestion du courrier et la gestion d'archivage.
* **Les experts :** Sont chargés de faire le suivi des projets, leur mise en œuvre et peuvent aussi faire leurs archivages
* **Assistant(e) :** est chargé(e) de valider le courrier avant leur archivage, consulte la liste des courriers et la liste d'archive.

### 3.2.3. Diagramme de contexte

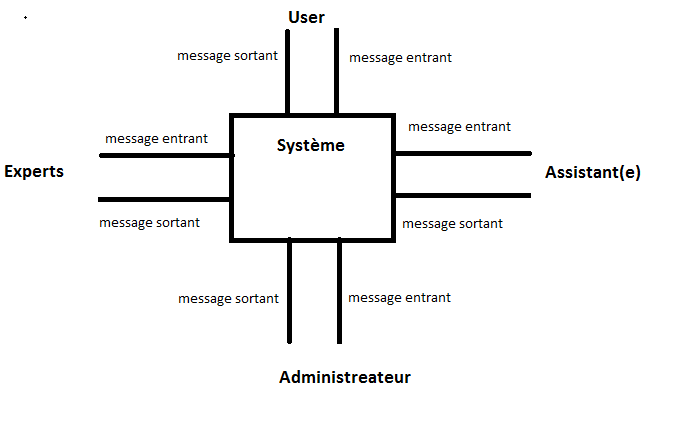


Figure 10 diagramme de contexte

**Identification des messages échangés entre les acteurs et le système :**

Les différents acteurs de notre système, peuvent échanger des messages entrants ou bien sortants comme suit :

|  |  |  |
| --- | --- | --- |
| **Acteur** | **Messages entrants** | **Messages sortants** |
| User | M1 : Demande de connecter | M2 : L’affichage de l’espace d’accueil |
| Administrateur | M3 : demande d’ajouter, modifier où  supprimer un utilisateur  M5 : valider le formulaire  M7 : demande d’authentification  M9 : insertion de login et le mot de passe | M4 : affichage du formulaire  M6 : confirmation de l’opération  M8 : demande de login et le mot de passe  M10 : vérification des informations |
| Experts | M3 : demande d’ajouter, modifier un projet  M5 : demande d’authentification  M9 : insertion de login et mot de passe | M4 : affichage du formulaire  M8 : demande de login et password  M10 : vérification du mot de passe |
| Assistant(e) | M3 : demande de consulter, de faire une  recherche et d’enregistrer un courrier  M5 : demande d’authentification  M9 : insertion de login et mot de passe | M4 : affichage du formulaire  M8 : demande de login et password  M10 : vérification des informations |

Tableau 26 Identification des messages échangés

Les messages entrants représentent les demandes qu’un acteur effectue tandis que les messages sortants représentent la réponse du système à une demande donnée.

### 3.2.3. Diagramme de cas d’utilisation

Un cas d'utilisation est une unité cohérente représentant une fonctionnalité visible de l'extérieur. Il réalise un service de bout en bout, avec un déclenchement, un déroulement et une fin, pour l'acteur ui l'initie. Un cas d'utilisateur modélise donc un service rendu par le système, sans imposer le mode de réalisation de ce service. (ici référence)

#### Cas d'utilisation préliminaire

**L’authentification :** permet d’identifier chaque utilisateur, et de lui donner l’accès aux fonctionnalités propices.



Figure 11 diagramme de cas d'utilisation s'authentifier et créer compte

#### Gestion et archivage d’un courrier

Ce diagramme fait intervenir un acteur. Lorsqu’un courrier arrive l’assistante est chargée d’assurer son suivi de son arrivé à la fin de son traitement et archive ce dernier y compris les dossiers relatifs à ce dernier.

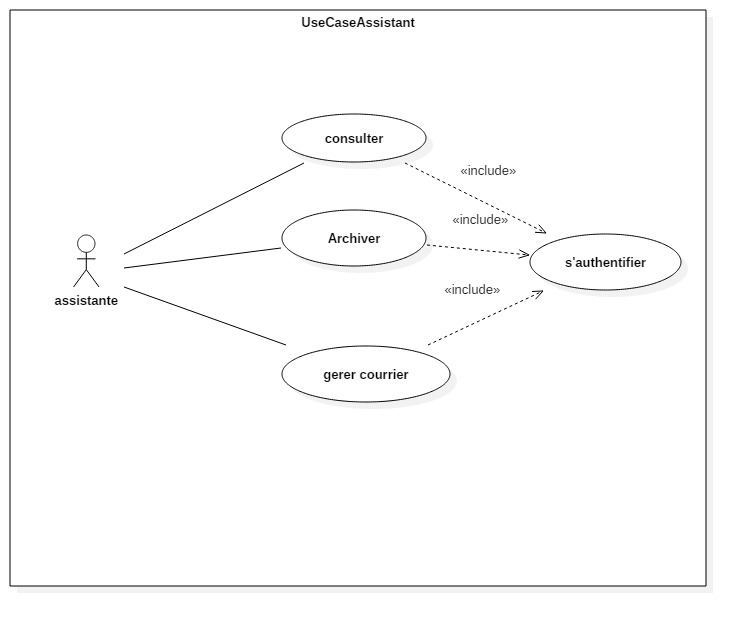


Figure 12 cas d'utilisation gestion et archivage d'un courrier

#### Gestion et archivage d’un projet

Dans le diagramme ci-dessous nous avons les experts qui sont les principaux intervenant dans la gestion, le suivi et l’archivage des projets.

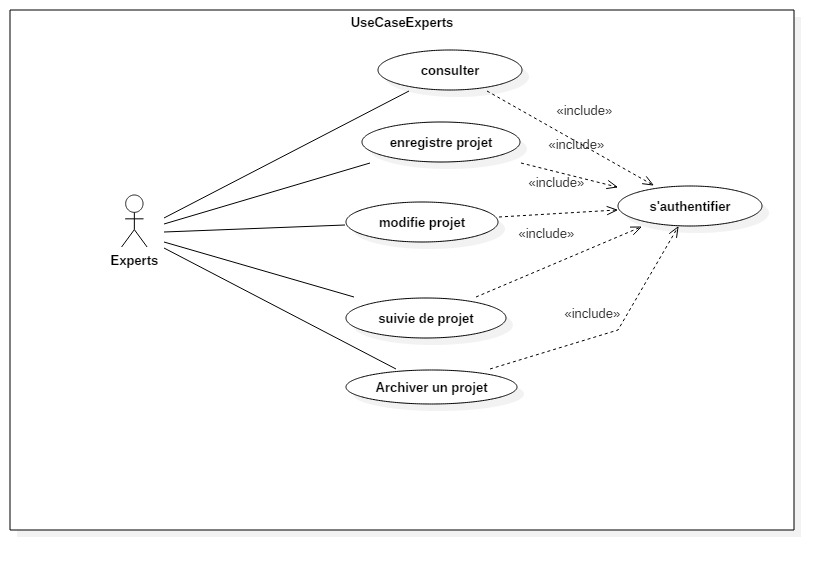


Figure 13 cas d'utilisation gestion et archivage d'un projet

#### Gestion des principaux cas d’utilisation

Ce diagramme fait intervenir l’administrateur comme principal acteur qui a tous les droits, car il est chargé de faire le suivi des projets, des courriers et l’archivage.

#### 

Figure 14 diagramme des principaux cas d'utilisation

### 3.2.4. Diagramme de séquence

Les diagrammes de séquences mettent en valeur les échanges de messages (déclenchant des événements) entre acteurs et objets (ou entre objets et objets) de manière chronologique, l’évolution du temps se lisant de haut en bas. (ref ici)

#### Diagramme de séquence s’authentifier

L’authentification consiste à assurer la confidentialité des données, elle se base sur la vérification des informations associées à un utilisateur (généralement un login et un mot de passe). Ces informations sont préétablies dans une base de données. Lors d’une authentification deux cas se présentent : les informations introduites par l’utilisateur sont incomplètes, dans ce cas un message d’erreur s’affiche, ou les informations saisies sont complètes et le système procède à leur vérification. Ceci explique l’utilisation de l’opérateur « alt ». Le même opérateur illustre les deux réactions du système, après la vérification des informations saisies par l’utilisateur, soit par l’affichage d’un message d’erreur, ou de l’interface correspondante.

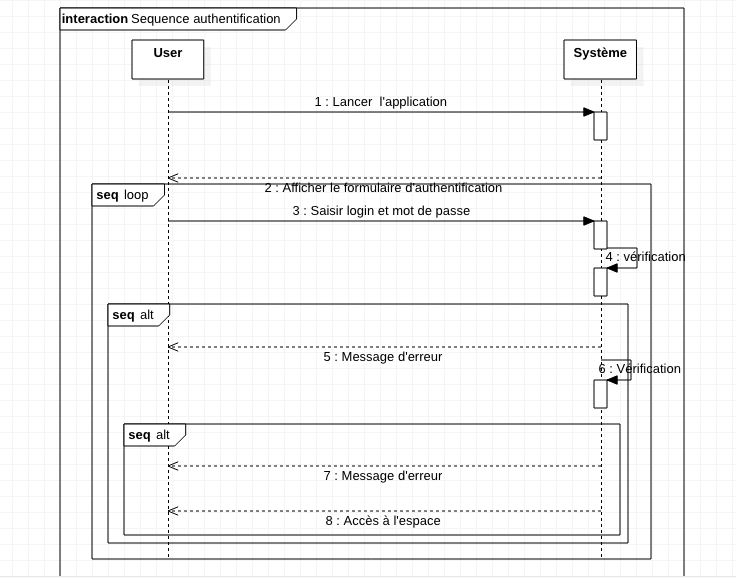


Figure 15 Diagramme de séquence du cas d'utilisation « s'authentifier »

#### Diagramme de séquence du cas d’utilisation « Enregistrer un utilisateur »

Le diagramme de séquence enregistre un utilisateur, permet de voir les étapes lorsqu’un utilisateur veut s’enregistrer dans notre système. Lorsqu’un administrateur ou un utilisateur quelconque fait une demande d’enregistrement, le système lui répond par

l’affichage d’un formulaire qui sera validé après remplissage.

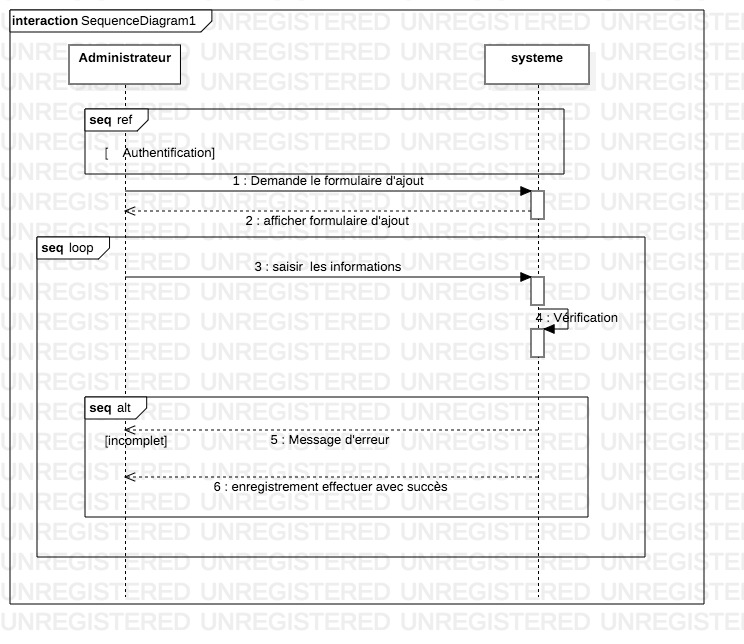


Figure 16 Diagramme de séquence du cas d'utilisation enregistrer un utilisateur

#### Diagramme de séquence du cas d’utilisation « modifier un utilisateur »

Lorsque l’administrateur envoi une demande de modification d’un utilisateur, le système lui répond en affichant la liste des utilisateurs. Une fois l’utilisateur à modifier est choisi, un formulaire s’affiche afin d’apporter les modifications souhaitées.

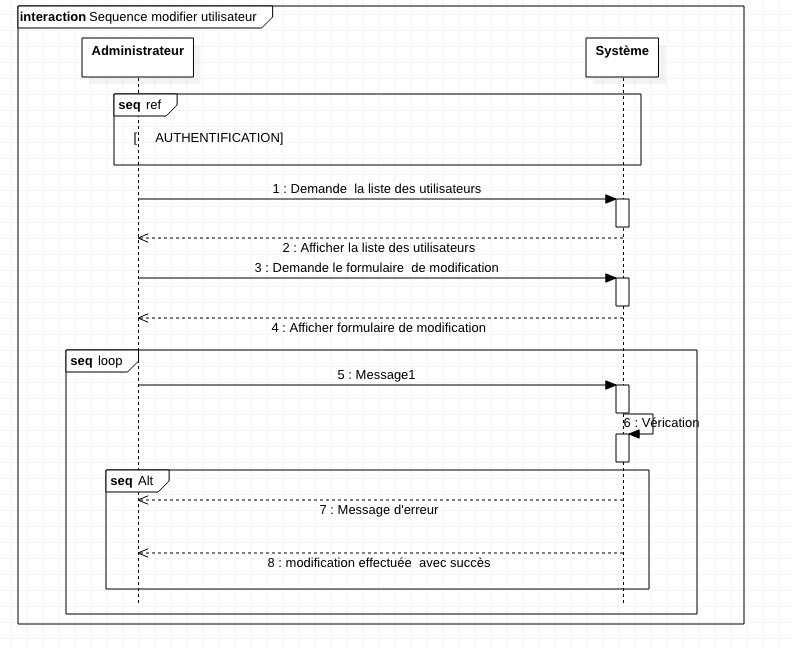


Figure 17 Diagramme de séquence du cas d'utilisation modifier utilisateur

#### Diagramme de séquence du cas d’utilisation « enregistrer courrier »

Lorsqu’une Assistante veut faire l’enregistrement d’un nouveau courrier, il fait une demande d’ajout au système. Ce dernier va recevoir un formulaire qui sera validé après remplissage.

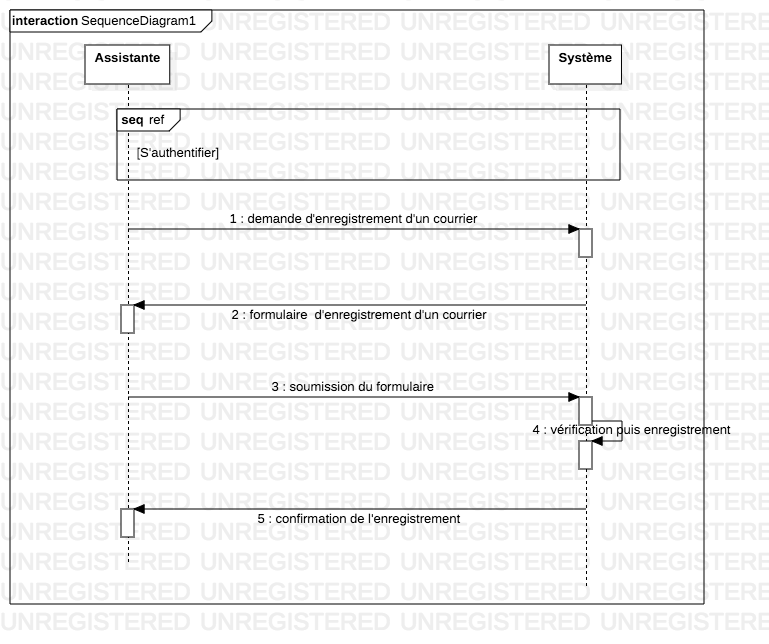


Figure 18 Diagramme séquence enregistrer un courrier

#### Diagramme de séquence du cas d’un cas d’utilisation « Archiver un courrier »

Lorsque l’administrateur ou veulent ajouter un courrier aux archives, ils ou elles envoient une demande d’ajout au système. Ce dernier va répond au système en affichant le formulaire d’ajout qui sera validé après son remplissage.

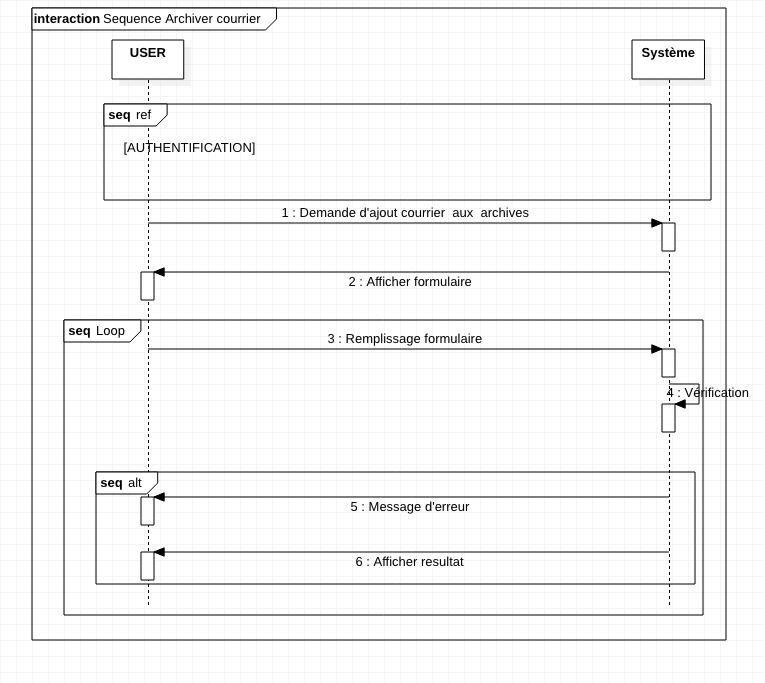


Figure 19 Diagramme séquence Archiver un courrier

#### Diagramme de séquence du cas d’utilisation « enregistrer projet »

Lorsqu’un Expert veut faire l’enregistrement d’un nouveau projet, il effectue une demande d’ajout au système. Ce dernier va recevoir un formulaire qui sera validé après remplissage.

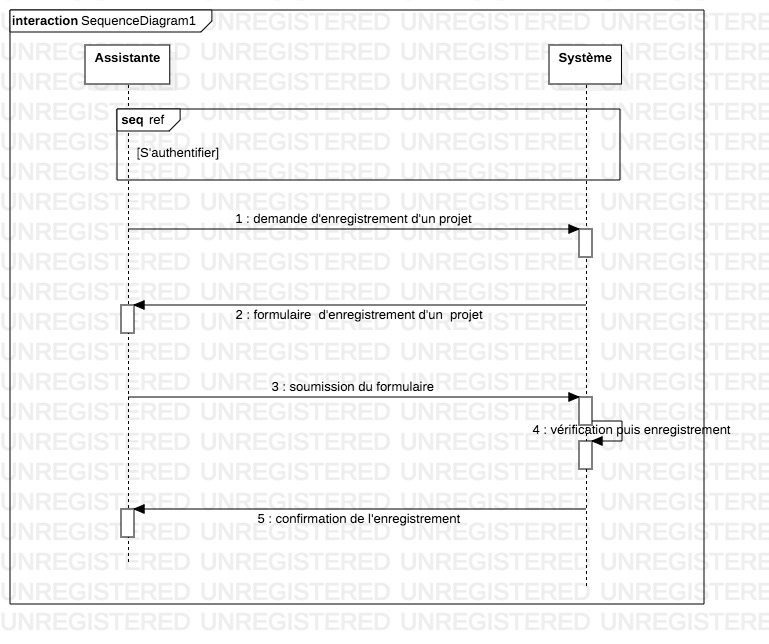


Figure 20 Diagramme de séquence du cas d’utilisation enregistrer projet

#### Diagramme de séquence du cas d’un cas d’utilisation « Archiver un projet »

Lorsque l’administrateur ou un Expert veulent ajouter un projet aux archives, ils ou elles envoient une demande d’ajout au système. Ce dernier va répond au système en affichant le formulaire d’ajout qui sera validé après son remplissage.

#### 

Figure 21 Diagramme séquence Archiver un projet

### 3.2.5. Diagramme de classe

Le diagramme de classe constitue l’un des pivots essentiels de la modélisation avec UML. En effet, ce diagramme permet de donner la représentation statique du système à développer. Cette représentation est centrée sur les concepts de classe et d’association. Chaque classe se décrit par les données et les traitements dont elle est responsable pour elle-même et vis-à-vis des autres classes. Les traitements sont matérialisés par des opérations. Le détail des traitements n’est pas représenté directement dans le diagramme de classe ; seul l’algorithme général et le pseudo-code correspondant peuvent être associés à la modélisation (ref ici). La description du diagramme de classe est fondée sur :

* Le concept d’objet
* Le concept de classe comprenant les attributs et les opérations ;
* Les différents types d’association entre classes.

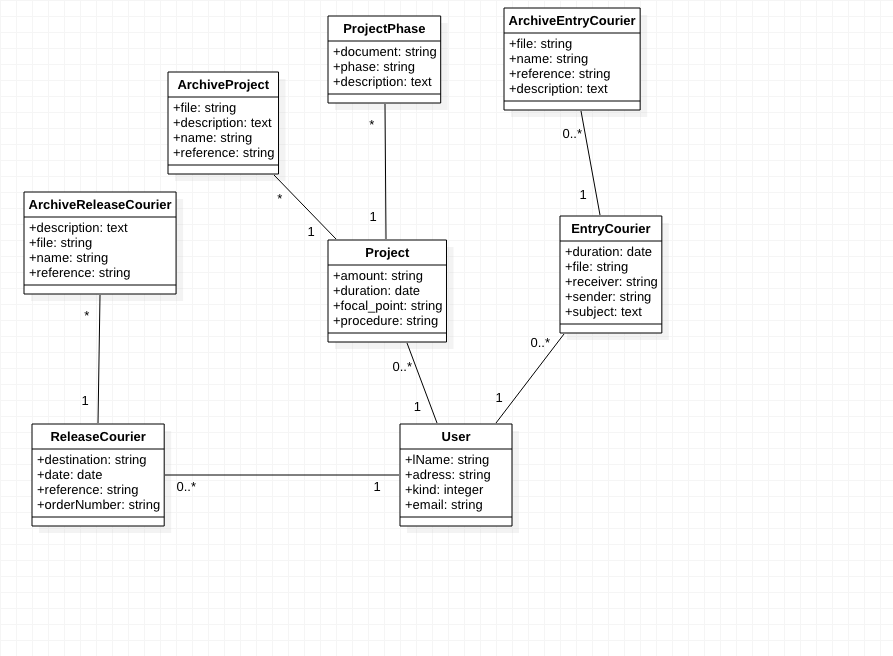


Figure 22 Diagramme de classe

### 3.2.6. Diagramme de déploiement

Le diagramme de déploiement est une vue statique qui sert à représenter l’utilisation de l’infrastructure physique par le système et la manière dont les composantes du système sont réparties ainsi que leurs relations entre eux. Le CGI (Common Gateway Interface), le langage de script de script exécuté sur le serveur, permet de manipuler les données avant de renvoyer à l’ordinateur client. Dans la documentation de SQLITE, le terme connector fais référence à un logiciel qui permet à une application de se connecter sur une base de données SQLITE.

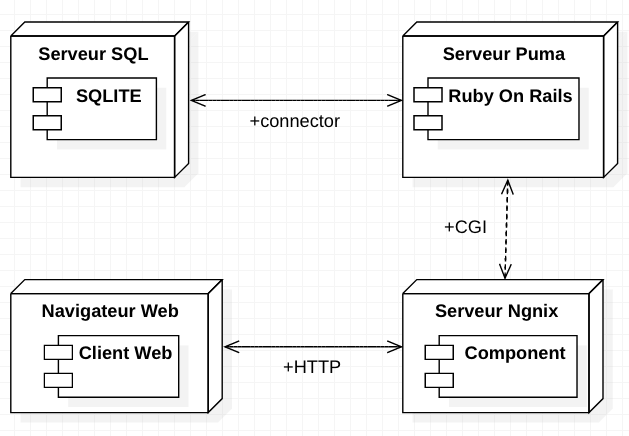


Figure 23 Diagramme de déploiement

## Conclusion

Tout au long de ce chapitre, il était question pour nous de décrire de manière les différentes étapes de conception de notre plateforme. De faire une présentation de façon succincte les scenarios les plus importantes de notre système tout en faisant usage des diagrammes de séquences et le diagramme de classe. Et tout cela constituera ainsi notre cahier de charge.

|  |  |
| --- | --- |
| : IMPLEMENTATION |  |

## Introduction

Dans ce chapitre nous entamons la partie pratique, ou nous allons présenter l’environnement et les outils de développement utilisé, l’architecture de l’application et quelques interfaces de celui-ci. Vue la complexité́ de notre système après analyse, conception et modélisation il ne sera évident pour nous de l’implémenter comme cela a été́ prévu dans le cahier de charge tout en respectant les délais prescrit. En se basant sur l’étude de l’existant faite au chapitre 2 dans les généralités, nous avons opté́ pour l’utilisation d’une solution open source qui couvre le maximum de fonctionnalités exigées, et la modification pour satisfaire aux besoins de l’entreprise.

## 4.1. Langage de programmation et environnement de développement

### 4.1.1. Langage de programmation

#### Ruby

Le langage Ruby a été́ conçu, au milieu des années 90, par Yukihiro Matsumoto, un programmeur Japonais. Son objectif était d’avoir un langage qui soit « plaisant » à utiliser, (référence ici) c’est un langage open source dynamique et orienté objet avec une syntaxe flexible. La version utilisée est la 2.5.1.

Le langage Ruby compte diverses manières d’utilisation (référence ici) :

* Pour une interface web : c’est l’utilisation la plus courante ;
* En ligne de commandes (CLI "Command Line Interface") ;
* Pour produire une interface desktop (GUI "Graphical User Interface ").

#### JavaScript

JavaScript est un langage de script orienté objet principalement utilisé dans les pages HTML. A l’opposé des langages serveurs (qui s’exécutent sur le site), JavaScript est exécuté́ sur l’ordinateur de l’internaute par le navigateur lui-même. Ainsi, ce langage permet une interaction avec l’utilisateur en fonction de ses actions (lors du passage de la souris au-dessus d’un élément, du redimensionnement de la page, etc.). La version standardisée de JavaScript est l’ECMAScript. (référence ici)

#### HTML/XHTML

Le HTML et sa variante plus stricte XHTML sont des langages de balisage des pages Web. Il n’y a pas si longtemps, le HTML servait à définir aussi bien la structure des pages que leur présentation visuelle. Aujourd’hui, ces deux aspects doivent être bien distincts et le X/HTML est destiné́ unique- ment à représenter la structure d’une page : titres, sous-titres, paragraphes, images, formulaires de saisie, liens hypertextes, etc.

#### CSS

CSS (Cascading Style Sheets, ou feuilles de styles en cascade) permet de modifier la présentation des éléments X/HTML : couleur, taille, police de caractères, mais aussi position sur la page, largeur, hauteur, empilement, bref tout ce qui touche à la mise en page d’un document X/HTML.

#### Bootstrap

Bootstrap est un Framework CSS, mais pas seulement, puisqu’il embarque également des composants HTML et JavaScript. Il comporte un système de grille simple et efficace pour mettre en ordre l’aspect visuel d’une page web. Il apporte du style pour les boutons, les formulaires, la navigation...etc. Il permet ainsi de concevoir un site web rapidement et avec peu de lignes de code ajoutées. Le Framework le plus proche de Bootstrap est sans doute Foundation qui est présenté́ comme « The most advanced responsive front-end Framework in the world » (référence ici)

#### Ruby On Rails

Ruby On Rails est un Framework libre distribué sous licence MIT destiné au prototypage et au développement d’application Web diverses et variées, il est multiplateforme. ROR (Ruby On Rails) est orienté RAD (Rapid Application Development) (référence ici), il reprend de nombreux principes communs aux méthodes "Agiles" et est basé sur l’architecture MVC et ceci se voit dès l’arborescence d’un projet ROR.

#### JQUERY

JQuery est un Framework JavaScript libre et Open Source, implanté côté́ client, qui porte sur l’interaction entre le DOM (Document Object Model), JavaScript, AJAX et le Html. Cette librairie JavaScript a pour but de simplifier les commandes communes du JavaScript. La devise de jQuery est en effet, "Écrire moins pour faire plus" (write less do more).

JQuery, du moins à l’origine, est l’œuvre d’un seul homme : John Resig. Ce jeune surdoué́ de JavaScript développa la première version de jQuery en janvier 2006.

Les spécificités de jQuery sont nombreuses mais l’essentielle est assurément la souplesse qu’il apporte pour accéder à tous les éléments du document Html grâce à la multitude de sélecteurs mis en place. Cette caractéristique fut d’ailleurs retenue pour donner un nom à ce Framework : j pour JavaScript et Query pour chercher ou accéder aux éléments. (référence ici)

#### XML

XML (eXtensible Markup Language) est un langage de balisage, qui se distingue de HTML par le fait qu’il permet de spécifier la structure du contenu d’un document plutôt que la façon de le présenter.

#### SQL

SQL (Structured Query Language) est un langage d’interrogation de base de données très populaire. Il constitue aujourd’hui une norme implémentée par de nombreux SGBD (Systèmes de Gestion de Bases de Données).

### 4.1.2. Outils de développement

#### SQLITE

Sqlite est le système de gestion de base de données (SGBD) que nous avons utilisé pour gérer notre base de données car il respecte le modèle d’architecture 3-tiers. Il permet de créer facilement nos tables, il est sécurisé, rapide et léger, qui fonctionne sur de nombreux systèmes d’exploitation (dont Linux, Mac OS, Windows, Solaris, FreeBSD ... ) et qui est accessible en écriture par de nombreux langages de programmation, incluant notamment Ruby, C, C++ ,PHP, Java, .NET, Python .

#### RubyMine

RubyMine est un environnement de développement intégré (IDE), qui est cross-Platform (utilisable sur n’importe quel système), il comprend toutes les caractéristiques d’un IDE moderne (éditeur couleur, projet multiplateforme et de page web) intègre le système de contrôle de version et bien d’autre outils à son sein, il est payant mais néanmoins donne une version d’essai de 30 jours et une licence pour étudiant.

#### ImageMagick

ImageMagick est un logiciel en ligne de commande très puissant de manipulation d’image dans pratiquement tous les formats existants. Sa licence est compatible avec la licence GPL (general public licence).

#### StarUML

StarUML est un logiciel de modélisation UML, qui nous a permis de modéliser nos différents diagrammes

## 4.2. Présentation du projet

L’arborescence du projet représente ici l’empaquetage et l’organisation des différents fichiers du projet. Le projet Java que nous avons créé sur RubyMine est constitué en module, en fonction de leurs utilités. Les captures ci-dessous présentent cette arborescence, d’abord dans un explorateur Windows, et ensuite dans l’IDE RubyMine.

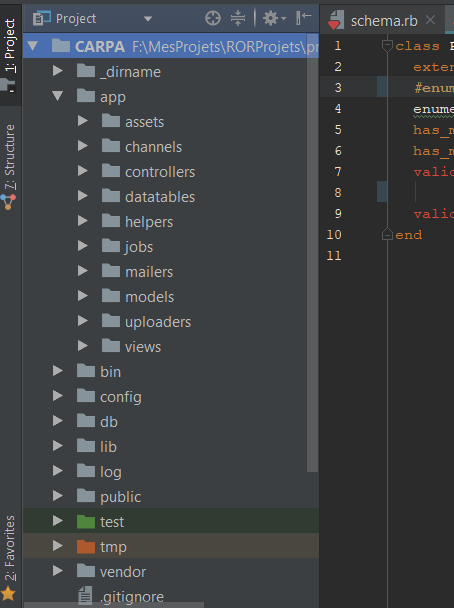
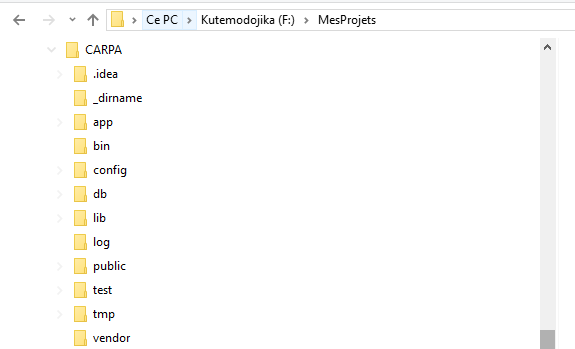


Figure 24 Arborescence du projet dans RubyMine



**Figure 25 Arborescence du projet dans l'explorateur Windows**

## 4.3. Modèle Architecturale

### 4.3.1. Synoptique de fonctionnement d’une application web

La figure ci-dessous présente le principe de fonctionnement de notre application web. Dans cette architecture, nous avons d’un côté, un ordinateur ayant notre application web et de l’autre, un serveur qui communique avec notre ordinateur à travers des web services.

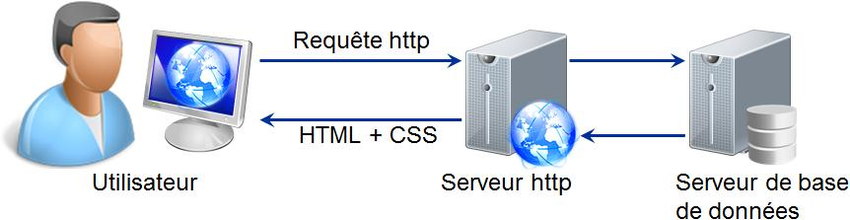
****

Figure 26 Synoptique de l'application web

### 4.3.2. Synoptique de fonctionnement d’une application web en Ruby On Rails

La figure ci-dessous présente l’architecture générale du système en Ruby On Rails. On y ressort une ORM Active Record dans notre cas, qui permet de faire le mapping relationnel consistant à modéliser la base de données sous forme d’objets pour une manipulation plus simple à travers le code Ruby.

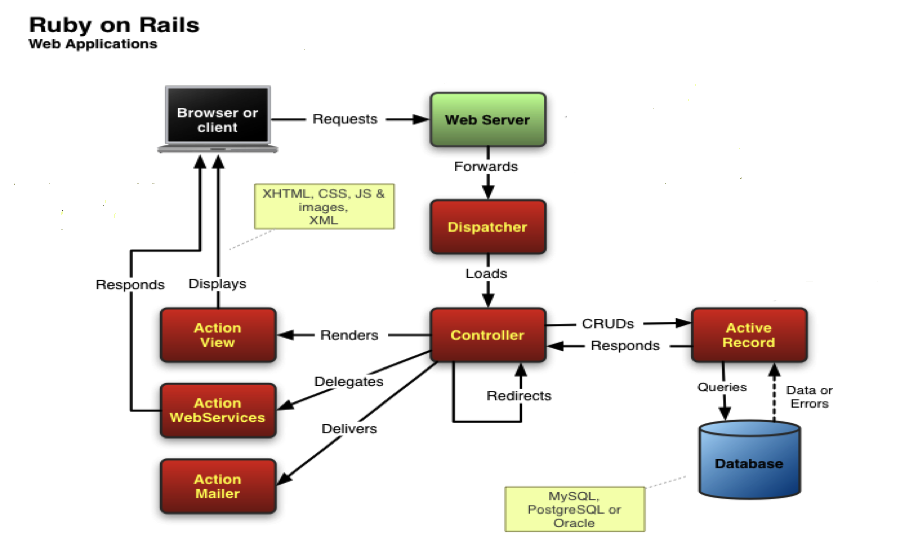


Figure 27 Synoptique d'une application web en Ruby On Rails

Le contrôleur s’occupe de l’accès aux données. Nous implémentons aussi dans cette couche les algorithmes métier de notre système. Elle ne change pas si on modifie l’interface utilisateur ou la façon d’accéder aux données nécessaires au fonctionnement de l’application.

L’interface utilisateur est l’interface web qui permet à l’utilisateur de piloter l’application et d’en recevoir des informations.

## Conclusion

Nous avons dans ce chapitre dévoilé ce qui a été utilisé pour résoudre la problématique générale du suivi des projets du courrier et d’archivage. Mais aussi, pour la satisfaction de chaque besoin émis par le cahier de charges. Il ne nous reste plus qu’à présenter le résultat de tout cela dans le chapitre qui suit.

|  |  |
| --- | --- |
| : RESULTATS ET COMMENTAIRES |  |