



UNIVERSITE DE FIANARANTSOA  
ECOLE DE MANAGEMENT ET D'INNOVATION  
TECHNOLOGIQUE  
**MEMOIRE DE FIN DE CYCLE POUR L'OBTENTION  
DU DIPLOME DE LICENCE PROFESSIONNELLE**

**Mention** : Informatique

**Parcours** : Développement d'Application Internet-Intranet

MISE EN PLACE D'UNE SYSTÈME DE GESTION ET  
SUIVI DES PROJETS-CAMSON GROUPE

Présenté par :

➤ SOLOFONDRAIBE Donné Alphonse

Membres de jury :

Président :

Examineur :

Rapporteur :

Encadreur Professionnel : TAHIN'ANDRIAMBELONANDRO Ambininkasinirina

Année Universitaire 2020-2021



SOLOFONDRAIBE Donn  Alphonse  
N  le 05 juin 1997   Camp-Robin  
Adresse : P52, Cit  Fitiavana Andrainjato F/tsoa  
T l phone : +261 34 46 261 28  
Email : solofondraibedani@gmailm.com

## FORMATION

- **2020-2021** : Troisi me ann e de licence professionnelle en informatique,  cole de Management et d'Innovation Technologique (EMIT), Universit  de Fianarantsoa (UF),

**Parcours** : D veloppement d'Application Internet-Intranet

- **2019-2020** : Deuxi me ann e de licence professionnelle en informatique,  cole de Management et d'Innovation Technologique (EMIT), Universit  de Fianarantsoa (UF),

**Parcours** : D veloppement d'Application Internet-Intranet

- **2018-2019** : Premi re ann e de licence en D veloppement d'Application Intranet-Internet   l' cole de Management et d'Innovation Technologique de l'Universit  de Fianarantsoa.

**Parcours** : D veloppement d'Application Internet-Intranet

**2017-2018** : Obtention du dipl me de Baccalaur at de l'enseignement g n ral, « **s rie C** », Fianarantsoa, avec mention « **Assez-Bien** »

## EXPERIENCES PROFESSIONNELLES

- **Juillet 2022 - Maintenant** : D veloppeur FullStack aux TanIT Tech Antananarivo  
Technologies ( PHP/Symfony , ReactJS )

- **Juin 2022-Septembre 2022** : Stagiaire chez Camson Groupe Fianarantsoa

**Th me** : «Mise en place d'une syst me pour la suivi et gestion des projets Camson Groupe»

**Technologies** :

- Back-end : Symfony 5, NodeJs ,Socket.io ;

- Front-end : ReacJS ;

- SGBD : MySQL ;

- **Mars 2021-Mai 2021** : Stagiaire chez Upscale Business Solutions (UBS), Itaosy Antananarivo, **Th me** : « Mise en place d'une plateforme pour le suivi et  valuation de performance de TEAM-Projects »

**Technologies** :

- Back-end : NestJS 7 ;
- Front-end : Angulars 11, HTML5, Bootstrap 4 ;
- SGBD : MySQL ;
  - **2021-2022 :**
  - Conception et réalisation d'une application réseaux chat sous JAVA
  - Conception et réalisation d'une application de gestion des projets informatique sous J2EE, Spring boot/Angular et PostgreSQL
  - Conception et réalisation d'une application web Gestion de tâche sous PHP-Symfony et PostgreSQL
  - **2020-2021 :**
  - Conception et réalisation d'une application Desktop pour la Gestion des ventes de matériels Informatique sous JAVA-NetBeans-MySQL-Jasper Report
  - Conception et réalisation d'une application de gestion de parc touristique sous C#-MySQL-Visual Studio
  - Conception et réalisation d'une application de gestion de note sous PHP 7-MySQL, Bootstrap 4, JavaScript, Ajax
  - **2020 :**
  - Conception et réalisation d'une application web pour la Gestion des ventes de matériels Informatique sous PHP-MySQL
  - **2019 :**
  - Conception et réalisation d'une application de gestion de parc touristique sous Access

## **COMPETENCES EN INFORMATIQUE**

- Outils Informatique : Word, Excel, PowerPoint, Internet ;
- OS : Linux, Windows ;
- Langages : Java, Python, C#, PHP, JSP, J2EE, HTML5, JavaScript, CSS3, TypeScript, Ajax, NodeJS, JSON, XML
- Framework : NestJs, Ionic, Symfony, CodeIgniter 4, Spring Boot, Hibernate, Struts, ReactJs, Angular
- Web Service : API/SOAP ;
- Serveur d'Application : Apache Tomcat, Jboss, GlassFish ;
- Gestionnaire de Packages : NPM, yarn, Maven, Graddle ;
- SGBD : MySQL, PostgreSQL, MongoDB, MS Access ;
- Modélisation: MERISE II, UML ;

- Gestion de Projet : Méthode AGILE Scrum, Trello,GIT ;
- Réseaux ;
- Développement Mobile : Native et Hybride.

## CONNAISSANCE LINGUISTIQUES

<b>Langues</b>	<b>Compréhension à l'audition</b>	<b>Lecture</b>	<b>Rédaction</b>
Anglais	Passable	Passable	Assez-Bien
Français	Bien	Bien	Bien
Malagasy	Excellente	Excellente	Excellente

## DIVERS

- Sens du travail en équipe
- Dynamique
- Loisir : Surfez sur Internet, Écoutez des chansons

## AVANT-PROPOS

L'École de Management et d'Innovation Technologique, qui est un établissement rattaché à l'Université de Fianarantsoa, bascule actuellement vers le Système Licence-Master-Doctorat (LMD). Étant un de ces étudiants, de la mention Développement d'Application Internet Intranet (DA2I), l'obtention du diplôme de licence est donc conditionnée par un stage professionnel en entreprise d'une durée de trois mois au maximum.

Le principal objectif de l'école est de former ses étudiants dans le domaine de travail. Les futurs cadres doivent être à la portée du savoir-faire, volonté, dynamisme et professionnalisme. L'objectif est de renforcer et de consolider les connaissances acquises durant la formation dispensée à l'École.

## REMERCIEMENTS

Au terme de ce travail, nous tenons à remercier Dieu le tout puissant de nous avoir donné le courage, la volonté et la patience pour achever ce travail. Nous avons l'honneur et le plaisir de présenter nos profonde gratitude et nos sincères remerciements à :

- Professeur HAJALALAINA Aimé Richard Président de l'Université de Fianarantsoa ;
- Docteur RAKOTONIRAINY Hasina Lalaina, Directeur de l'Ecole de Management et d'Innovation Technologique, de nous avoir accueilli en tant qu'étudiant au sein de l'école ;
- Monsieur Camson, Directeur Général de la société Camson Groupe, de nous avoir accepté en tant que stagiaire au sein Camson Groupe ;
- Madame RABEZANAHARY Hoby, Responsable de Mention de l'Informatique à l'EMIT, qui a déployé tous ses efforts pour nous dispenser toutes les formations nécessaires durant trois ans à l'EMIT ;
- Par la même occasion, nous exprimons nos sincères remerciements les plus chaleureux à Madame TAHIN'ANDRIAMBELOANDRO Ambininkasinirina, Développeur web au Camson Groupe FIANARANTSOA, en tant qu'encadreur professionnel, non seulement pour son dévouement à l'organisation du stage mais surtout pour ses précieux conseils et suggestions à l'encadrement professionnel ;
- Tous les membres du jury pour avoir accepté d'examiner mon travail, pour leurs remarques et leurs recommandations qui ont permis améliorer ce mémoire. Ils m'ont fait l'honneur de participer à mon Jury.
- Tous les enseignants au sein de l'EMIT qui nous ont partagé leurs connaissances durant mon cursus universitaire ;
- Tous les personnels du Camson Groupe pour leur bienveillance durant la période de stage ;

Ces remerciements vont également à nos très chères et adorables familles qui sont toujours été présents, avec leurs soutiens et leurs précieuses aides, et à tous ceux qui sont concourus pour accumuler cet ouvrage.

Merci à vous tous !

LISTE DES FIGURES

LISTE DES TABLEAUX

## LISTE DES ABREVIATIONS



## GLOSSAIRE

# INTRODUCTION

Actuellement, l'utilisation des technologies prennent une place extrêmement importante dans le milieu professionnel. Chaque société cherche à être plus performante en automatisant presque la totalité de leurs tâches afin de rentabiliser leurs activités.

Camson groupe est l'un d'une entreprise de services du numérique à Madagascar plus précieusement c'est l'entreprise d'accueil de notre stage.

Or, le responsable de la direction de système d'information, au sein de cette Groupe, rencontre une difficulté face à la gestion du projet et aussi le suivi de tâche dans une équipe du développement, qui est actuellement se fait sous « Excel » qui n'est pas rentable. C'est pourquoi, le thème de notre stage s'adresse à la conception et la mise en place d'un système permettant gérer et de suivre le projet au sein de cette direction. Le principal objectif est donc de faciliter la gestion de projet et de suivre leur tâche d'un collaborateur.

Afin de bien mener ce projet à terme, notre mémoire est divisé en trois grandes parties. Pour la première partie, décrit la présentation générale de notre école qui est l'École de Management et de l'Innovation Technologique auprès de l'Université de Fianarantsoa, En suite présentation de lieu où on effectue notre stage, Camson Groupe Fianarantsoa. Et pour terminer la partie 1, nous allons présenter le projet. Pour la partie 2, tout d'abord, nous allons effectuer un état de l'art suivi par le choix de la méthode et notation utilisé, ensuite l'analyse du projet, l'analyse et études des besoins, l'analyse des faisabilités, et les contraintes du projet. Ainsi, nous parlerons de la modélisation de la plateforme avec l'outil de modélisation UML. Et enfin, dans la partie 3, nous allons parler de la mise en œuvre de l'application. Dans cette partie, on va parler les spécifications des outils de réalisations de l'ensemble de l'application web, suivi de l'implémentation de l'application et pour terminer cette partie, nous allons présenter des interfaces de réalisation de la plateforme.

## **PARTIE I : PRESENTATIONS GENERALES**

# **Chapitre 1 : PRESENTATION DE L'ECOLE DE MANAGEMENT ET D'INNOVATION TECHNOLOGIQUE**

## **1.1. Historique**

L'École de Management et d'Innovation Technologique (EMIT) est une école universitaire publique pluridisciplinaire, rattachée à l'Université de Fianarantsoa. La grande maturité au niveau de l'enseignement et la compétence des étudiants sortant de l'établissement ont permis aux dirigeants sous l'approbation du Ministère la conversion du Centre en École au sein de l'Université de Fianarantsoa par le Décret N°2016-1394 du 15 Novembre 2016. L'EMIT prépare d'une part le diplôme de Master en deux mentions en trois parcours et d'autre part le diplôme de Licence en trois mentions en cinq parcours. Auparavant, elle a été connue sous le nom du Centre Universitaire de Formation Professionnalisante (CUFP), créé par le Décret N°2005-205 du 26 Avril 2005 et dispensait le diplôme de Licence professionnelle en Administration ainsi qu'en Informatique. Mais avant cela, elle a été connue également sous le nom du Centre de Formation Continue (CFC), créé par l'Arrêté Rectoral N°99-23/UF/R du 10 Mars 1999 qui formait de diplôme de Technicien Supérieur.

L'EMIT a été sélectionnée « Meilleur Etablissement » pendant le Salon de la Recherche organisé par l'Organisation Internationale du Travail les 5 et 6 Juillet 2017. L'école est actuellement basculée vers le système Licence, Master et Doctorat (LMD). Toutes les offres de formation dispensée à l'EMIT sont habilitées par le Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.

## **1.2. Formations existantes**

L'école présente actuellement deux cycles : Licence et Master. Chaque cycle possède plusieurs parcours assurés par un chef de mention.

### **1.2.1. Cycle Licence**

L'école possède trois (03) mentions pour cinq (05) parcours en cycle Licence. Les étudiants toutes mentions confondues doivent effectuer un voyage d'études d'insertion en entreprise pour les premières années de Licence (L1), un stage de réalisation en entreprise avec un rapport de stage soutenu pour les étudiants en deuxième année de Licence (L2) et un stage de fin d'études suivi de la soutenance d'un mémoire pour les étudiants en troisième année de Licence (L3). La formation dure trois années universitaires et à la fin de la formation, les étudiants obtiennent des diplômes de Licence.

#### **1.2.1.1. Mention Management, Parcours Administration Economique et Sociale**

A l'issue de la formation, les étudiants ont les compétences de :

- Assister le Directeur Général, le Directeur des Ressources Humaines et le Directeur Administratif et Financier ;
- Gérer les Ressources Humaines ;
- Gérer une entreprise ou un projet.

#### **1.2.1.2. Mention Informatique, Parcours Développement d'Application Internet Intranet**

A l'issue de la formation, les étudiants sont compétents en :

- Administration des bases de données ;
- Administration des réseaux et systèmes informatiques ;
- Développement d'application client/serveur.

#### **1.2.1.3. Mention Relations Publiques et Multimédia, Parcours Communication Multimédia et Relations Publiques et Communication Organisationnelle**

A l'issue de la formation, les étudiants ont les compétences de :

- Rédiger un article dans un journal ;
- Occuper un poste d'un technicien de presse ;
- Travailler dans la revue de presse.

### **1.2.2. Cycle Master**

L'Ecole de Management et d'Innovation Technologique (EMIT) possède deux (02) mentions pour trois (03) parcours en Master Professionnel qu'en Master Recherche. La durée de la formation est quatre semestres c'est-à-dire deux années universitaires.

#### **1.2.2.1. Mention Management, Parcours Management Décisionnel**

La mention Management propose un parcours Management Décisionnel ayant pour objectif de former et d'équiper les apprenants à la maîtrise des outils d'aide à la décision en matière de management et de leur donner les compétences requises dans ce domaine. Comme le management a besoin de se conformer en permanence aux diverses nouvelles exigences du marché, l'enseignement doit alors toujours viser pour mettre à jour les connaissances de l'apprenant par la formulation de programmes de cours qui tiennent compte de ces nouveautés. Ainsi, les objectifs principaux peuvent se résumer à former des acteurs de haut niveau en management décisionnel, de préparer des cadres capables de gérer et de créer un projet de développement économique régional et national.

### **1.2.2.2. Mention Informatique, Parcours Système d'Information, Géomatique et Décision et Parcours Modélisation et Ingénierie Informatique**

Le parcours Système d'Information, Géomatique et Décision a pour objectif de donner un panorama des recherches actuelles et émergentes en termes de système d'aide à la décision. En effet, les systèmes informatiques et la géomatique sont en plein essor, par les grilles de calcul et les multiples appareils mobiles intégrant des systèmes informatiques de plus en plus performants et complexes. Ces systèmes informatiques intégrant un parallélisme massif ou /et une mobilité des composants représente un défi pour le génie logiciel qui doit fournir de nouvelles méthodes et des outils de production de logiciel pour la description de l'architecture de ces systèmes complexes et pour leur validation et/ou certification. De plus, l'effervescence des techniques en géomatique qui fournissent des données spatiales et temporelles dans différents domaines représentent des moyens efficaces pour prendre les bonnes décisions.

### **1.3. Organigramme de l'EMIT**

La structure hiérarchique au sein de l'EMIT se compose d'un conseil (scientifique ou d'établissement), une direction, un collège des enseignants, des chefs de mentions et des services présents au sein de l'école. Cette structure est représentée de manière générale sur la figure 1.1.

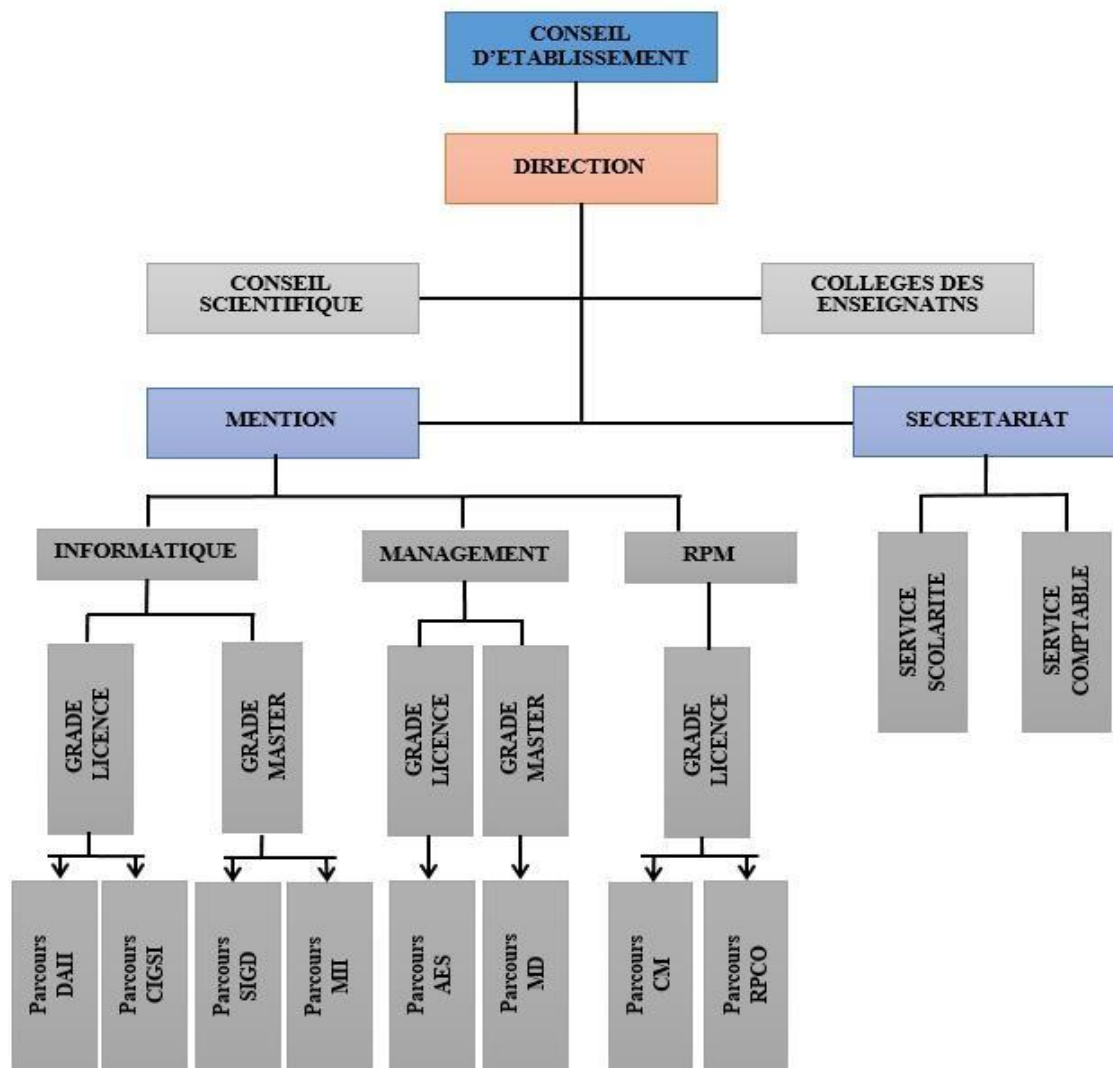


Figure 1 :1 Organigramme EMIT [EMIT]

## **Chapitre 2 : PRESENTATION DU CAMSON GROUPE**



## **Chapitre 3 : PRESENTATION DU PROJET**

### **3.1. Contexte du projet**

Grâce au développement des nouvelles technologies, la société qui nous accueille comme stagiaire, cherche toujours les moyens pour les rendre satisfait, autrement dit, des moyens pour répondre aux besoins de ses clients. La Direction du système d'Information, notamment « Développement » est l'une de direction au sein de cette société. Elle a pour rôle d'étudier et de réaliser les besoins des clients quel que soit le type de demande. Or les besoins des clients ne cessent pas de s'augmenter. C'est pourquoi, le responsable au sein de la direction en question veut avoir une application permettant de suivre et de gérer les projets réalisés par leurs équipes. Autrement dit, il veut avoir une application web pour le suivi de tâche et de gérer les projets au sein de la direction accessible à tout moment et n'importe où.

### **Description du projet**

Le client cherche toujours d'avoir la nouvelle fonctionnalité d'autre part, la société Camson Groupe ne cesse pas de le donner de satisfaire leurs clients. En ce moment les équipes au sein de la direction technique utilisent Excel pour effectuer des rapports de tâche terminés au responsable vu qu'Excel n'est pas rentable et lent. Tout le monde connaît que plusieurs outils en ligne peuvent faire la gestion de projet mais presque payant. En tant que stagiaire je propose une solution développée une application web pour gérer tout ça. C'est pourquoi, notre projet a pour thème « Système de suivi et gestion des projets CAMSON-GROUPE » qui consiste à développer une application web permettant de faciliter la gestion des projets au sein du CAMSON GROUPE. Cette application est destinée pour la direction, notamment pour le responsable de la Développement et leur équipe projets pour stocker les informations et mise à jour des tâches effectuées.

### **3.2. Objectif Général du projet**

La réalisation de ce projet a pour objectif de mettre en place une application web pour :

- Automatisation des rapports tâche effectués par l'équipe au responsable ;
- Mettre en place un système sécurisé ;
- Suivi de tâches effectuées de chaque subordonnée par le responsable de la direction ;
- Distribution de tâche au collaborateur ;

- Grouper les données dans une base de données ;
- 

### **3.3. Les acteurs du projet**

Les acteurs de ce projet sont le Maître d'œuvre et le Maître d'ouvrage.

#### **3.3.1. Maître d'ouvrage**

Le maître d'ouvrage (parfois maîtrise d'ouvrage) est une entité porteuse du besoin, définissant l'objectif du projet, son calendrier et le budget consacré à ce projet. Le résultat attendu du projet est la réalisation d'un produit, appelé ouvrage. Le responsable et le chef de projet au Camson Groupe sont les maîtres d'ouvrage de notre projet.

#### **3.3.2. Maître d'œuvre**

Le Maître d'œuvre est une personne morale garante de la bonne réalisation technique des solutions. Le maître d'œuvre (ou maîtrise d'œuvre) est aussi une entité retenue par le maître d'ouvrage pour réaliser l'ouvrage, dans les conditions de délais, de qualité et de coût fixés par ce dernier conformément à un contrat, dont il est notre rôle durant ce projet.

### **3.4. Résultats attendus**

A la fin du projet, on attend à obtenir une application web dynamique pour la gestion de projet qui améliore la manipulation des données en les stockant dans une base de données, qui élimine la répétition des tâches comme la recopie de mêmes données dans plusieurs fichiers, qui performe l'ajout d'un projet en offrant une interface facile à utiliser. Les utilisateurs de l'application attendent une application maniable et performante en même temps. Concernant la sécurité de l'application, la mise en place d'un système d'authentification a aussi été élaborée.

### **3.5. Déroulement de développement du projet**

Depuis l'étude des besoins jusqu'à la phase de développement, notre projet, comme tous les projets développés au sein de la société, que nous avons effectué notre stage, suit la norme et les processus réguliers défini par le chef de projet. Pour suivre les déroulements des projets, y inclus la nôtre, la société utilise la méthodologie Agile.

#### **3.5.1. Planification du projet**

La planification d'un projet est incontournable pour la gestion de projet quelques soit le types. Elle permet de : définir les travaux à réaliser, fixer les objectifs, coordonner les actions, maîtriser les moyens, diminuer les risques, suivre les actions en cours, rendre en compte de l'état d'avancement du projet. En d'autres termes, elle est importante et à mettre en place avant

la réalisation d'un projet afin que le projet soit à termes. La réalisation de ce projet a une durée de trois (03) mois. Nous avons planifié la réalisation selon le tableau 3.1.

Tableau 3.1 : Planification du déroulement de stage

Mois	Semaine 1	Semaine 2	Semaine 3	Semaine 4
Juin	Intégration à l'entreprise	Enquête et Recueil des informations au responsable	Rédaction du rapport de l'enquête	Analyses de besoins
Juillet	Spécification fonctionnelle	Spécification techniques	Conception du projet : Élaboration de différents diagrammes	
Août	Documentation sur les technologies à utiliser et configuration des environnements de travail et Codage de l'application			

### 3.5.2 Recueil des documents

Nous avons recueilli des documents pour commencer notre projet au sein de la direction. Nous avons utilisé des nombreux documents, du cours et surtout des recherches sur internet. Le recueil de documents tout au long de la réalisation du projet nous ont permis à l'amélioration du projet.

Nombreux sont les documents consultables pour la réalisation du projet, tel que des fichiers sur internet ayant différents formats : fichier .PDF, fichier vidéo, page web, etc.

La liste de documents recueillis ayant participé à la mise en place du projet se trouve à la bibliographie.

### 3.5.3 Inventaire et installation des logiciels nécessaires à la réalisation du projet

Des logiciels sont utiles pour la réalisation de notre projet, tel que :

- Le système de gestion de base de données : MySQL, qui a pour rôle de serveur de données
- L'éditeur Vs Code : assurant l'outil de développement de notre application, c'est-à-dire l'élaboration des pages web ;
- PHP v7.4, Symfony v5.4.2 qui assure le rôle du serveur web ;
- Composer v2.3.10 pour la gestion de dépendance PHP ;

- ReactJS occupé la partie client de l'application autrement dit l'IHM de l'application
- On a besoin de NodeJS installé, ici NodeJS v16.16 pour pouvoir utiliser npm ;
- npm v8.17.0 c'est la gestionnaire de dépendance du NodeJS ;
- Postman : C'est la chaîne d'outils essentielle pour les développeurs d'API de partager, tester, documenter et surveiller les API.

## **Partie II : ANALYSE ET CONCEPTION DU PROJET**

## **Chapitre 4 : METHODES ET NOTATIONS UTILISEES**

### **4.1. Présentation des modèles de cycles de vie**

Le « **cycle de vie d'un logiciel** » (en anglais software lifecycle), désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition. L'objectif d'un tel découpage est de permettre de définir des jalons intermédiaires permettant la validation du développement logiciel, c'est-à-dire la conformité du logiciel avec les besoins exprimés, et la vérification du processus de développement, c'est-à-dire l'adéquation des méthodes mises en œuvre.

L'ensemble des étapes ou des phases qui interviennent dans le développement d'un logiciel de sa conception à sa disparition constitue le cycle de vie de celui-ci. Lorsque le découpage est ainsi effectué, la détection des erreurs se fait beaucoup plus tôt. En effet, il est constaté que plus les erreurs étaient détectées tardivement, plus grand était le coût de la réparation. Avec un tel découpage, le développeur maîtrise non seulement la qualité du logiciel mais aussi les délais et les coûts. Il peut donc ainsi assurer la validation du logiciel et la vérification du processus de développement.

### **4.2. Généralité sur le Génie Logiciel**

« Le terme **génie logiciel** désigne l'ensemble des méthodes, des techniques et outils concourant à la production d'un logiciel, au-delà de la seule activité de programmation. »

Le génie logiciel est apparu dans les années 1970 sous la coordination de l'OTAN. Le génie logiciel a été mis sur pied pour répondre à la crise du logiciel (1968). À cette époque les logiciels n'étaient pas fiables et il était difficile pour les développeurs de fournir dans les délais une application respectant les spécifications du cahier des charges.

### **4.3. Notion de processus et d'étape dans le cycle de vie**

C'est la description d'un processus couvrant les phases de création d'un produit, de distribution sur un marché et sa disparition.

Le but de ce découpage est de maîtriser les risques, de maîtriser au mieux les délais et les coûts, et d'obtenir une qualité conforme aux exigences. On distingue deux types de cycle de vie :

- Le cycle de vie des produits s'applique à tous les types de produits, et peut être considéré comme un outil de gestion ;
- Le cycle de développement des logiciels s'insère dans le précédent, on l'appelle souvent abusivement cycle de vie des logiciels.

Comme pour toutes les fabrications, il est important d'avoir un procédé de fabrication du logiciel bien défini et explicitement décrit et documenté.

En GL, il s'agit d'un type de fabrication un peu particulier : en un seul exemplaire, car la production en série est triviale (recopie). Les modèles de cycle de vie du logiciel décrivent à un niveau très abstrait et idéalisé les différentes manières d'organiser la production. Les étapes, leur ordonnancement, et parfois les critères pour passer d'une étape à une autre, sont explicités (critères de terminaison d'une étape - revue de documents -, critères de choix de l'étape suivante, critères de démarrage d'une étape).

#### **4.4. Présentations des différents modèles de cycle de vie**

Afin d'être en mesure d'avoir une méthodologie commune entre le client et la société de service réalisant le développement, des modèles de cycle de vie ont été mis au point définissant les étapes du développement ainsi que les documents à produire permettant de valider chacune des étapes avant de passer à la suivante. A la fin de chaque phase, des revues sont organisées avant de passer à la suivante.

Le cycle de vie du logiciel comprend généralement les minimales activités suivantes :

- **Définition des objectifs**, consistant à définir la finalité du projet et son inscription dans une stratégie globale.
- **Analyse des besoins et faisabilité**, c'est-à-dire l'expression, le recueil et la formalisation des besoins du demandeur (le client) et de l'ensemble des contraintes.
- **Conception générale**. Il s'agit de l'élaboration des spécifications de l'architecture générale du logiciel.
- **Conception détaillée**, consistant à définir précisément chaque sous-ensemble du logiciel.
- **Codage** (Implémentation ou programmation), soit la traduction dans un langage de programmation des fonctionnalités définies lors de phases de conception.

- **Tests unitaires**, permettant de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux spécifications.
- **Intégration**, dont l'objectif est de s'assurer de l'interfaçage des différents éléments (modules) du logiciel. Elle fait l'objet de tests d'intégration consignés dans un document.
- **Qualification** (ou recette), c'est-à-dire la vérification de la conformité du logiciel aux spécifications initiales.
- **Documentation**, visant à produire les informations nécessaires pour l'utilisation du logiciel et pour des développements ultérieurs.
- **Mise en production**, qui est le déploiement sursit du logiciel ;
- **Maintenance**, comprenant toutes les actions correctives (maintenance corrective) et évolutives (maintenance évolutive) sur le logiciel.

Pour bien organiser le processus de développement, plusieurs modèles ont été définis à savoir :

- Le modèle en cascade ;
- Le modèle en spirale ;
- Le modèle en V ;
- Le modèle par incrément.

#### **4.4.1. Modèle en cascade**

Le modèle de cycle de vie en cascade a été mis au point dès 1966, puis formalisé aux alentours de 1970. Il définit des phases séquentielles à l'issue de chacune desquelles des documents sont produits pour en vérifier la conformité avant de passer à la suivante

(Symbolisées dans le schéma par des flèches vers le haut) [Winston W& al, 1970].



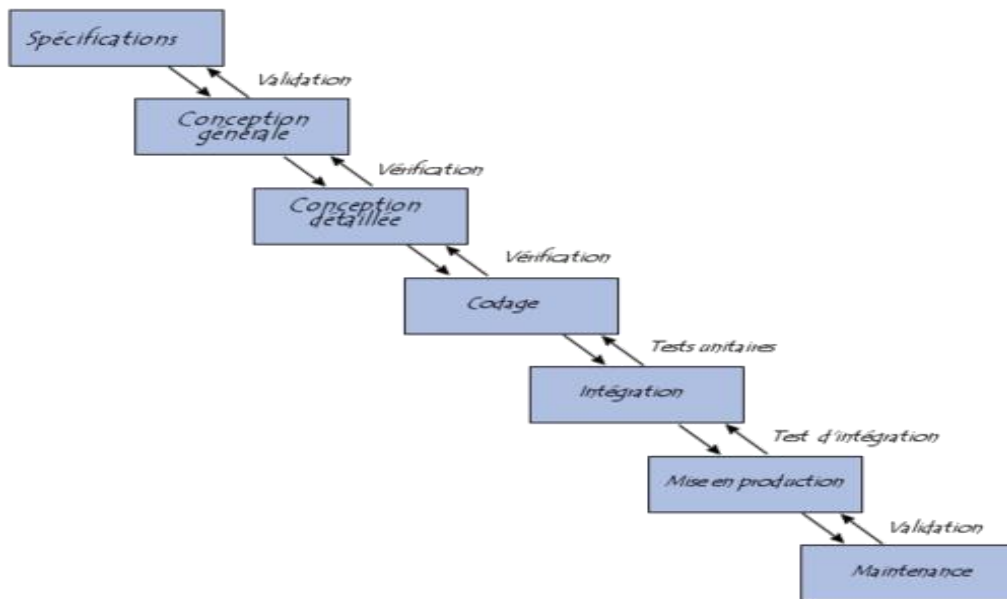


Figure 4.1 : Formalisme modèle de cycle de vie en cascade

#### 4.4.2. Modèle en V

Le modèle de cycle de vie en V part du principe que les procédures de vérification de la conformité du logiciel aux spécifications doivent être élaborées dès les phases de conception.

Il décrit les étapes essentielles du développement d'un logiciel, le cycle de vie du projet. Il est représenté par un V dont la branche descendante contient toutes les étapes de la conception du projet, et la branche montante toutes les étapes de tests du projet. La pointe du V, quant à elle, représente la réalisation concrète du projet, le codage ; on pourrait donc en déduire, de manière simpliste, que les deux branches montantes et descendantes ne sont que de la documentation.

En effet, chaque étape d'une branche a son pendant dans l'autre branche, c'est à dire qu'une étape de conception correspond à une étape de test qui lui est spécifique. A tel point, d'ailleurs, qu'une étape de test peut être élaborée dès que la phase de conception correspondante est terminée, indépendamment du reste du projet [Winston W &al., 1970].

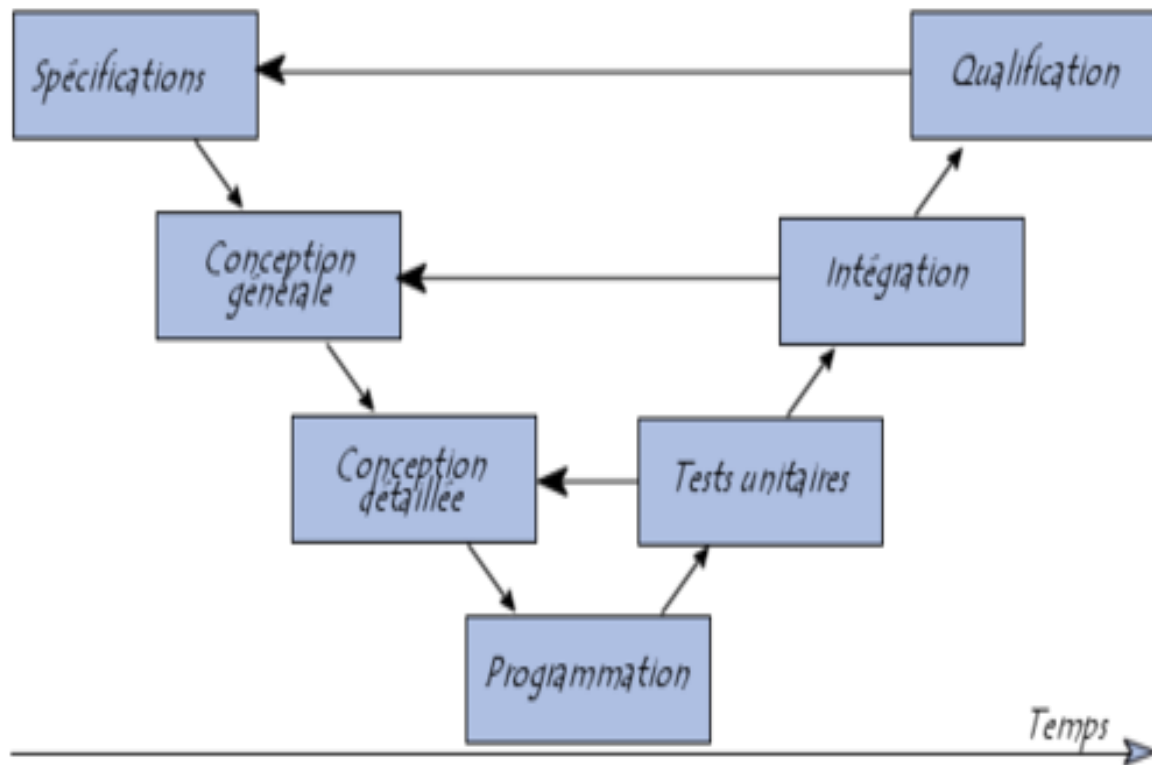


Figure 4.2 : Formalisme modèle de cycle de vie en V

#### 4.4.3. Modèle en Spirale

Le **modèle en spirale** (*spiral model*) est un modèle de cycle de développement logiciel qui reprend les différentes étapes du cycle en V. Par l'implémentation de versions successives, le cycle recommence en proposant un produit de plus en plus complet et dur. Le cycle en spirale met cependant plus l'accent sur la gestion des risques que le cycle en V.

On distingue [Barry W& al., 1988] quatre phases dans le déroulement du cycle en spirale :

1. Détermination des objectifs, des alternatives et des contraintes ;
2. Analyse des risques, évaluation des alternatives ;
3. Développement et vérification de la solution retenue ;
4. Revue des résultats et vérification du cycle suivant.

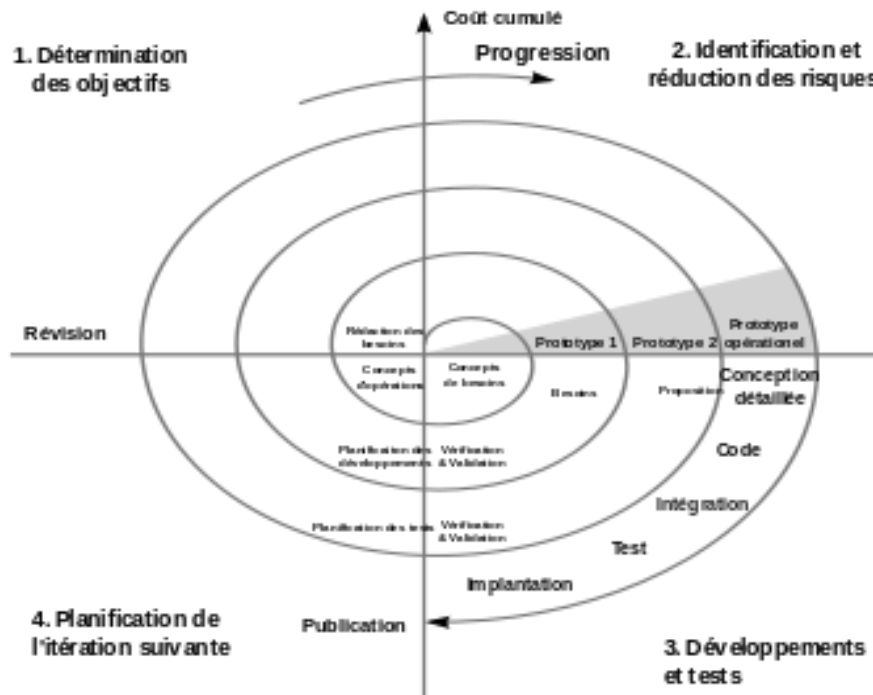


Figure 4.3 : Formalisme modèle de cycle de vie en Spirale

Ce modèle de cycle de vie tient compte de la possibilité de réévaluer les risques en cours de développement, il emprunte au prototypage incrémental mais lui adjoint une dimension relevant de la prise de décision managériale et non purement technique. Il couvre l'ensemble du cycle de développement d'un produit.

#### 4.4.4. Modèle de cycle de vie par incrément

Dans les modèles précédents, un logiciel est décomposé en composants développés séparément et intégrés à la fin du processus.

Ce modèle de cycle de vie prend en compte le fait qu'un logiciel peut être construit étape par étape. Le logiciel est spécifié et conçu dans son ensemble. La réalisation se fait par incréments de fonctionnalités. Chaque incrément est intégré à l'ensemble des précédents et à chaque étape le produit est testé exploiter et maintenu dans son ensemble. Ce cycle de vie permet de prendre en compte l'analyse de risques et de faire accepter progressivement un logiciel par les utilisateurs plutôt que de faire un changement brutal des habitudes [Winston W, 1970], [Barry W. Boehm, 1988], [Barry W. Boehm, 1988].

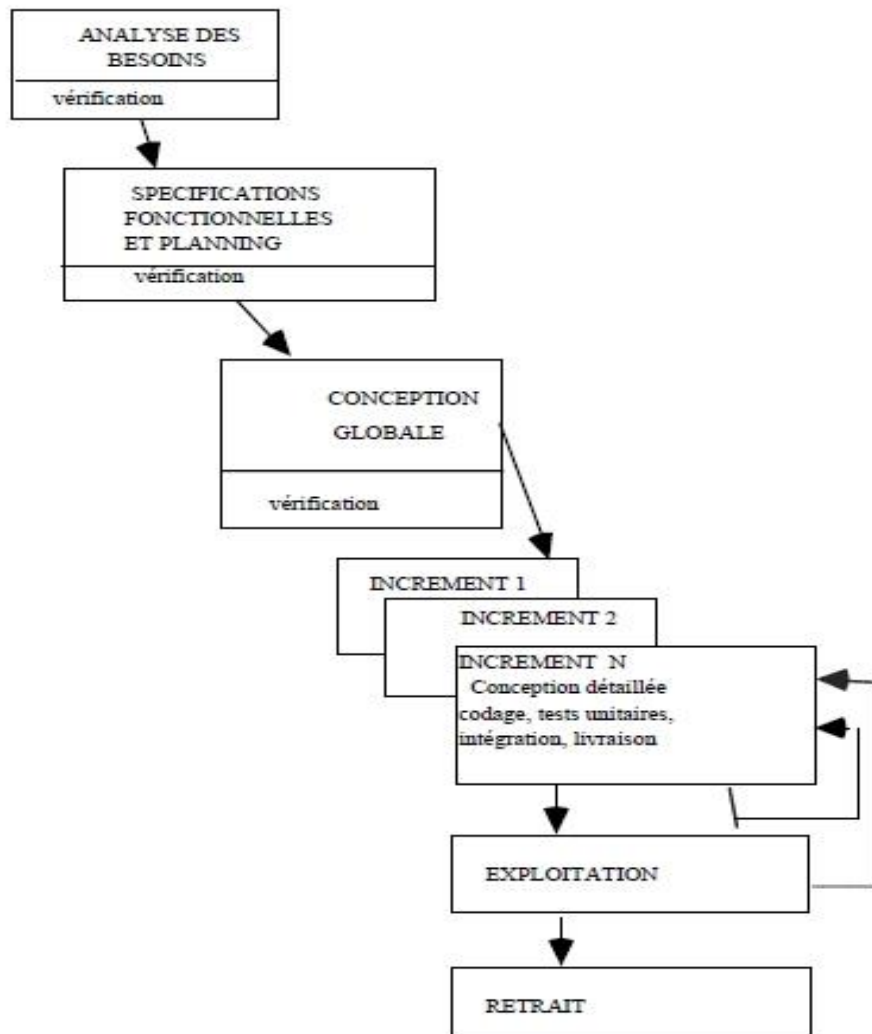


Figure 4.4 : Formalisme modèle de cycle de vie par incrément

#### 4.5. Comparaison de différent modèle de cycle de vie

Chacun de modèles de cycle de vie ont leurs avantages et leurs inconvénients.

Le tableau 4.1 [24] **représente** les différents modèles de cycles de vie d'un logiciel tels que : en cascade, en v, en spirale, par incrément :

Tableaux 4.1 : Comparaison cycle mode de cycle vie

Modèle de cycle de vie	Avantages	Inconvénients
EN V	<ul style="list-style-type: none"> <li>· Les limites de chaque étape sont visibles</li> <li>· Facilite un management du projet</li> <li>· La définition des besoins est non-évolutive</li> </ul>	<ul style="list-style-type: none"> <li>· Pas d'interaction entre les phases de développement</li> <li>· L'intégration n'a lieu qu'à la fin du cycle</li> <li>· Le client peut se retrouver non satisfait</li> </ul>

	<ul style="list-style-type: none"> <li>· La qualité prime sur le coût</li> </ul>	<ul style="list-style-type: none"> <li>· Pas de retour en arrière d'une phase à l'autre</li> </ul>
EN SPIRALE	<ul style="list-style-type: none"> <li>· Les utilisateurs finaux sont intimement associés à toutes les étapes du développement</li> <li>· Le développement se fait en interaction avec les clients</li> <li>· L'évolution du coût de développement est sous contrôle</li> <li>· Les utilisateurs ont dès le départ une vue globale du système</li> </ul>	<ul style="list-style-type: none"> <li>· Une expertise en évaluation des risques est nécessaire</li> <li>· La spirale peut être infinie</li> <li>· les développeurs travaillent par intermittence</li> <li>· il est difficile de définir les objectifs et les points de validation intermédiaires entre les différentes étapes</li> </ul>
PAR INCREMENT	<ul style="list-style-type: none"> <li>· Le client peut valider chaque étape du processus</li> <li>· Utilise la méthode Diviser Pour Régner</li> <li>· La délivrance du produit est rapide</li> <li>· Le coût de lancement du projet est moindre</li> <li>· Un produit exploitable peut être délivré à tout moment</li> <li>· Les clients obtiennent les fonctionnalités majeures du système très tôt</li> <li>· Le risque du changement des besoins est minimal</li> </ul>	<ul style="list-style-type: none"> <li>Requière une bonne planification et une bonne conception</li> <li>· Requière la définition complète des fonctionnalités du système pour une définition des différents incréments</li> <li>· Le coût total du développement du système n'est pas négligeable</li> <li>· Les différentes interfaces doivent être bien définies</li> </ul>

	· Développe les fonctions primordiales dès le départ	
--	--	--

#### 4.6. **Choix modèle de cycle de vie**

D'après le tableau 4.1, on choisit le modèle en V vu qu'il tient compte de la réalité et suit une procédure bien définie et facile entreprendre.

La représentation en V tient d'avantage compte de la réalité, le processus de développement n'est pas réduit à un enchaînement de tâches séquentielles.

Elle montre que :

- c'est en phase de spécification que l'on se préoccupe des procédures de qualification
- c'est en phase de conception globale que l'on se préoccupe des procédures d'intégration
- c'est en phase de conception détaillée que l'on prépare les tests unitaires.

Le modèle de cycle de vie en V permet d'anticiper sur les phases ultérieures de développement du produit. En particulier le modèle en V permet de commencer plus tôt :

- le plan de tests de qualification ;
- le plan d'évaluation de la performance.

#### 4.7. **Présentation d'UML**

Le langage de modélisation unifié, de l'anglais Unified Modeling Language (UML), est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.

##### 4.7.1. **Historique**

En Octobre 1994 G.Booch et J.Rumbaugh ont décidé de travailler ensemble pour unifier leurs méthodes au sein de la société Rational Software un an après, I.Jacobsoll a rejoint Rational Software pour travailler sur l'unification les travaux sur ce langage ont continué avec son adoption par de grands acteurs industriels dont Microsoft, Oracle et Unisys. Ce travail a abouti en janvier de l'année 1999 à UML 1.0. Le langage a été soumis par Rational Software et ses partenaires à l'OMG (Object Management Group) comme réponse à un appel d'offre sur la standardisation des langages de modélisation en Septembre 1997, et qui a été accepté à l'unanimité deux mois plus tard en Novembre 1997 dans sa version 1.1 qui devient de ce fait un standard. Des transformations continuelles ne cessent d'être effectuées pour supprimer les

incohérences, apporter des améliorations et ajouter de nouveaux concepts d'où l'apparition de la version 1.2 en Juin 1998 cette dernière n'a introduit aucun ajout d'ordre techniques, ces modifications par rapporta la version 1.1 porte uniquement sur un remaniement de la forme.

La version 1.3 apparue en Juin 1999 a apporté de nombreux changements qu'il s'agisse de corrections ou d'ajouts (modification des associations entre cas d'utilisation, simplification des stéréotypes, changement d'éléments de graphe d'activités et des automates d'états ...). D'autres versions sont apparues après la version 1.3 en Septembre 2001. La version 1.4 a été publiée, suivie par UML 1.5 en 2003. La version d'UML en cours à la fin 2004 est UML 2.0 et les travaux d'amélioration se poursuivent. UML est donc non seulement un outil intéressant mais une norme qui s'impose en technologie à objets et à laquelle se sont rangés tous les grands acteurs du domaine, acteurs qui ont d'ailleurs contribué à son élaboration.

#### **4.7.2. Avantages et Inconvénients d'UML**

Ces langages de modélisation n'est pas satisfaisante complètement, il possède ses points faibles. Tandis qu'il adopte aussi ses points forts.

Tableaux 4.2 : Avantages et inconvénients d'UML

Avantage	Inconvénient
<ul style="list-style-type: none"> <li>- Langage formel et normalisé ;</li> <li>- Permet une grande précision ;</li> <li>- Assure la pérennité ;</li> <li>- Facilite l'utilisation d'outils ;</li> <li>- Facilite la communication ;</li> <li>- Propose un cadre d'analyse ;</li> <li>- Permet la représentation d'éléments abstraits et complexes ;</li> <li>- Support de communication performant ;</li> <li>- Permet de bien définir les besoins clients, et ainsi d'éviter des surcoûts liés à la livraison d'un logiciel qui ne satisfait pas le client (Selon une étude du Standish Group en 1994, pour 53% des logiciels créés, le taux de délai de livraison non respecté est de 120%, on avait 90 % de budgets non tenus, et 60% de non disponibilité de certaines fonctionnalités) ;</li> <li>- Apporte une compréhension rapide du programme à d'autres développeurs externes en cas de reprise du logiciel et facilite sa maintenance.</li> <li>- Vulgarise les aspects liés à la conception et à l'architecture, propres au logiciel, au client.</li> </ul>	<ul style="list-style-type: none"> <li>- Son utilisation nécessite une formation préalable pour connaître ses normes standards (pour le client qui n'a pas de compétences dans ce domaine) ;</li> <li>- Il peut y avoir une mauvaise correspondance entre l'UML et le projet finalisé ;</li> <li>- Pour fonctionner correctement, les diagrammes UML doivent être synchronisés avec le code du logiciel qui nécessite du temps pour mettre en place et à entretenir, et ajoute du travail à un projet de développement logiciel. Les petites entreprises et les développeurs indépendants pourraient ne pas être en mesure de gérer la quantité supplémentaire de travail nécessaire pour synchroniser le code ;</li> <li>- UML n'est pas une méthode ;</li> <li>- Il nécessite un apprentissage et de l'expérience.</li> </ul>

#### 4.7.3. Les grandes lignes essentielles

UML est un langage de modélisation qui a pour objectif de clarifier et de simplifier la conception et la représentation d'un système donné et/ou de son fonctionnement. La caractéristique qui fait d'UML un langage de choix est la schématisation simple, compréhensible par l'homme et la machine de ses étapes essentielles.



UML 2.0 comporte treize types de diagrammes pour représenter des concepts particuliers du système d'information. Ils sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie. Ils se répartissent en deux grands groupes :

Les diagrammes structurels ou diagrammes statiques rassemblent cinq diagrammes :

- Le diagramme de classes qui représente les classes intervenant dans le système ;
- Le diagramme d'objets qui sert à représenter les instances de classes (objets) utilisées dans le système ;
- Le diagramme de composants qui permet de montrer les composants du système d'un point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données, ...) ;
- Le diagramme de déploiement qui sert à représenter les éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage, ...) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux ;
- Le diagramme de structures composites qui permet de décrire sous forme de boîte blanche les relations entre composants d'une classe.

Les diagrammes comportementaux ou diagrammes dynamiques rassemblent huit diagrammes :

- Le diagramme de cas d'utilisation qui permet d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système ;
- Le diagramme d'activités qui permet de décrire sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants ;
- Le diagramme d'états-transitions qui permet de décrire sous forme de machine à états finis le comportement du système ou de ses composants ;
- Le diagramme d'interaction qui permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences ;

- Le diagramme de séquence : représentation séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs ;
- Le diagramme de communication : représentation simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets ;
- Le diagramme global d'interaction qui permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences ;
- Le diagramme de temps qui permet de décrire les variations d'une donnée au cours du temps.

Les plus utilisés pour la maîtrise d'ouvrage sont les diagrammes de cas d'utilisation, de classes, de séquence, de composants et les diagrammes de déploiement. Les diagrammes de composants, de déploiement et de communication sont surtout utiles pour la maîtrise d'œuvre à qui ils permettent de formaliser les contraintes de la réalisation et la solution technique.

## **Chapitre 5 : ANALYSE DU PROJET**

Cette phase d'analyse permet d'étudier et de critiquer ce qui existe dans la direction concernée par le thème du stage. L'étude de l'existant est le point de passage obligé qui matérialise le premier contact du concepteur avec les utilisateurs.

### **5.1. Étude des besoins**

Le recueil des informations concernant l'utilisation du système est plus important. Il est nécessaire d'identifier et d'analyser les besoins de l'utilisateur afin que sa structure du formulaire, ainsi que les fonctionnalités souhaitées. De ce fait, nous avons déterminé le domaine du projet en recueillant les enquêtes et analyses effectués auprès du responsable de la direction.

#### **5.1.1. Besoins de l'utilisateur**

En principe, l'utilisateur a besoin d'améliorer leurs tâches en appliquant un nouvel système d'application :

- Automatisation des rapport tâche effectué par l'équipe au responsable ;
- Mettre en place un système sécurisé ;
- Distribution de tâches au collaborateur
- Suivi des taches effectuées par chaque employé.
- Enregistrement des projet, utilisateur, l'action à faire pour chaque information dans une base de données
- Sécurisation des données dans la base des donnés

#### **5.1.2 Besoins fonctionnels**

Les besoins non fonctionnels identifient les besoins de la société qui ne concernent pas le comportement du système mais identifient des contraintes internes et externes du système. Ils sont importants car ils agissent de façon indirecte sur le résultat et le rendement de l'utilisateur. Notre application doit être simple et avoir une intuitivité de son interface graphique, utilise des termes techniques métiers sur les formulaires à remplir, assure une authentification avant que l'utilisateur accède aux fonctionnalités correspond à son profil dans le but de sécurité. Le temps de recherche doit être rapide. On peut citer, parmi les besoins non fonctionnels, tel que :

- La sécurité ;
- La fiabilité (la gestion des erreurs) ;
- La qualité ;

## **5.2 Moyens pour la mise en œuvre**

Pour la réalisation de notre projet, nous voulons des configurations matérielles et logicielles compatible avec le système. Nous allons parler des ressources matérielles et logicielles dont le moyen nécessaire est un ordinateur utilisé comme poste de travail, qui possède :

- Un logiciel de modélisation ;
- Un logiciel bureautique pour l'élaboration du mémoire ;
- Un environnement de développement ;
- Une connexion internet pour la documentation ;
- Un serveur d'application ;
- Navigateur pour l'exécution de l'application ;
- Un serveur de la base de données.

## **5.3 Etude de faisabilité**

L'étude de faisabilité dans la gestion de projets est une étude qui s'attache à vérifier que le projet soit techniquement faisable et économiquement viable.

### **5.3.1 Analyse de l'existant**

L'analyse de l'existant est les réponses obtenues à partir des questions que nous avons posées aux responsables. L'objectif de l'analyse est de savoir l'organisation existant dans le service.

Le responsable de la direction effectue tous les traitements sous Microsoft Excel pour le suivi de tâche

### **5.3.2 Critique de l'existant**

En ce moment, le responsable utilise Excel pour effectuer le système de suivi. Notons que l'utilisation du logiciel Excel demande beaucoup de temps pour la saisie des informations. De plus, les informations saisies ne sont pas sauvegardées dans une base de données, il faut alors faire des recherches dans tous les fichiers Excel pour trouver ce que l'on cherche. La méthode de recherche est alors fastidieuse alors qu'il n'est pas sûr de trouver ce qu'il veut vraiment. De plus, la sauvegarde des informations dans plusieurs fichiers EXCEL nécessite beaucoup d'espace de disque à cause de la lourdeur des fichiers et de la recopie des fichiers dans plusieurs ordinateurs. Par ailleurs, il y a de nombreuses fonctionnalités du logiciel Excel qui n'est même pas nécessaire pour le système de suivi. Enfin, ce logiciel ne possède pas de vérification précise

des champs pour la sécurité des données, cela peut entraîner une erreur sur le type de données saisi par l'utilisateur

## 5.4 Proposition de solutions

Face aux différents problèmes cités, nous avons proposé les solutions dans le tableau

	Propositions	Avantages	Inconvénients
Solution 1	La première solution consiste à utiliser des logiciels de gestion de projet comme par exemple « Trello ».	-Logiciel a de fonctionnalité libre - Logiciel avec une grande communauté d'utilisateur	- Logiciel besoin de connexion internet -Logiciel est moins sûr en termes de sécurité
Solution 2	La deuxième solution consiste à développer une application web pour la suivi et gestion de projet	- Logiciel plus adapté à la situation -Logiciel facile à manipuler -Logiciel moins couteux	-Logiciel pas encore testé après comparaison des différents avantages et inconvénients,

**Tableau : Solutions proposées**

Le tableau nous montre les différentes solutions proposées face à cette situation. Nous avons choisi la deuxième solution. Vu qu'il est le plus adapté à la situation, de plus on est compétant pour le développement de cette application

## 5.5 Etude d'opportunité

En utilisant l'application, la direction peut travailler à la chaîne tout en gérant et en contrôlant de manière automatique les équipes ou le subordonné qui a effectué la tâche. L'avantage de l'utilisation de l'application est la rapidité du travail, la fiabilité de sauvegarde. L'application permet de faire des mises à jour (suppression, modification et ajout) de manière rapide et permet aussi d'effectuer d'autres opérations comme le suivi des équipes, la recherche et la liste selon les conditions de recherche

# **Chapitre 6 : CONCEPTION DU PROJET**

La conception de logiciel met en œuvre un ensemble d'activités qui à partir d'une demande d'informatisation d'un processus (demande qui peut aller de la simple question orale jusqu'au cahier des charges complet) permettent la conception, l'écriture et la mise au point d'un logiciel (et donc de programmes informatiques) jusqu'à sa livraison au demandeur.

## **6.1. Présentation de l'outil de conception**

Il existe plusieurs outils de conception. Pour concevoir un système d'information, nous avons adopté un outil de conception tel que : Visual Paradigm, Win'Design, Paradigm Plus, Poseidon for UML, Visual UML. Nous allons présenter deux outils de conception.

### **6.1.1. Visual paradigm**

C'est un logiciel de création de diagrammes dans le cadre d'une programmation. Tout en un, il possède plusieurs options permettant une large possibilité de modélisation en UML.

Il offre de nombreux outils pour créer différents types de schémas comme les diagrammes d'exigences et de cas d'utilisation.

Il possède bon nombre de navigateurs permettant de personnaliser chaque élément. Il permet aussi de générer des codes sources en divers langages comme le Java ou C++ à partir du modèle créé.

Inversement, il permet de produire un modèle à partir de codes sources. Ce logiciel permet d'utiliser la souris pour manipuler facilement le diagramme à créer. Tous les types d'actions sont pris en charge dont le traçage et la commande ainsi que la connexion des éléments

### **6.1.2. Win 'design**

Il est probablement l'outil le plus facile à prendre en main et son interface est très intuitive. Un très grand choix de SGBD est pris en compte, un grand nombre d'options de transformation de modèles (à tous les niveaux), la réactivité et l'efficacité du support.

## 6.2. Choix d'outils de conception

Nous allons dresser le tableau 6.1 pour comparer les outils de conception pour mieux faire le choix entre eux

Caractéristiques	Visual Paradigm	Win'design
Fiabilité	Oui	Non
Efficacité	Oui	Non (gourmant en mémoire)
Validité	Oui	Non (pas de modèle d'activité)
Portabilité	Oui (non sur certaine Windows SP3)	Non

**Tableau 6.1 : Choix d'outil de conception**

A partir de ce tableau, nous pouvons conclure que Visual Paradigm est meilleur pour la conception de notre application.

## 6.3. Règles de gestion

Comme toute application, une application dynamique devra aussi suivre certaines règles de gestion. Ces règles sont associées au niveau conceptuel et décrivent le « QUOI » de la société, en d'autres termes, on recueille dans l'interview surtout celle de la direction les règles traduisant soient les objectifs anciens, soient les objectifs nouveaux avec leurs contraintes respectives.

Pour commencer la conception de ce système, prenons en compte les règles de gestion suivantes :

RG 0 : L'utilisateur a pour rôle admin avoir le droit de créer un ou plusieurs projets.

RG 1 : Un projet est créé par un utilisateur.

RG 2 : Un projet avoir un ou plusieurs tâches.

RG 3 : Chaque carte peut avoir un ou plusieurs étiquettes

RG 4 : Une carte contenir un ou plusieurs tâches.

RG 5 : Une carte est placée dans un tableau.

RG 6 : Chaque carte a de titre, description, date limite et de(s) tâche(s)

RG 7 : Le responsable du service doit avoir un compte pour accéder à l'application.

RG 8 : le responsable gère les utilisateurs (il peut ajouter, modifier, supprimer des utilisateurs).



RG 9 : Chaque utilisateur a son propre login et mot de passe.

RG 10 : Le responsable seulement a le droit d'ajouter ou de modifier des projets, de même pour la carte et la tâche

RG 11 : chaque utilisateur peut assigner à une ou plusieurs cartes.

RG 12 : le responsable peut faire le suivi des tâches effectuées par leur subordonnée.

RG 13 : Au moment de l'inscription de l'utilisateur, il a obtenu un mot de passe par défaut    envoyer par email

## **6.4 Dictionnaire des données**

Le dictionnaire des données décrit la totalité de données manipulées, il est représenté sous forme d'un tableau.

Dans ce tableau, nous avons trouvé alors les listes symbolisées (rubrique du dictionnaire) pour éviter l'usage du mot trop long (qui se trouve à la description). Il suffit de classer par entité les rubriques en évoquant les autres caractéristiques. Nous allons détailler dans le tableau.

Tableau 6.2: Dictionnaire de données

## **6.5. Modélisation**

Modéliser un système avant sa réalisation permet de mieux comprendre le fonctionnement du système. C'est également un bon moyen de maîtriser sa complexité et d'assurer sa cohérence.

Un modèle est un langage commun, précis, qui est connu par tous les membres de l'équipe et il est donc, à ce titre, un vecteur privilégié pour communiquer. Dans le domaine de L'ingénierie du logiciel, le modèle permet de mieux répartir les tâches et d'automatiser certaines d'entre elles. C'est également un facteur de réduction des coûts et des délais. Le modèle est enfin indispensable pour assurer un bon niveau de qualité et une maintenance efficace.

Dans notre cas, nous allons utiliser le langage UML pour la modélisation de notre application, en construisant les différents diagrammes UML correspondant à notre système

### **6.5.1. Cas d'utilisation et description des séquences**

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des

présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés.

#### **6.5.1.1 Acteurs**

Ils sont des entités externes qui interagissent avec le système, comme une personne humaine ou un robot. Une même personne (ou robot) peut être plusieurs acteurs pour un système, c'est pourquoi les acteurs doivent surtout être décrits par leur rôle, ce rôle décrit les besoins et les capacités de l'acteur. Un acteur agit sur le système. L'activité du système a pour objectif de satisfaire les besoins de l'acteur. Les acteurs sont représentés par un pictogramme humanoïde (stick man) sous-titré par le nom de l'acteur.

La figure 6.1 représente le formalisme d'un acteur.

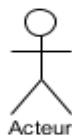


Figure 6.1: Formalisme d'un acteur

#### **6.5.1.2 Cas d'utilisation**

Le cas d'utilisation décrit les exigences fonctionnelles du système. C'est l'étape qui devrait être conçue avant tout. C'est l'unité cohérente qui représente la fonctionnalité visible de l'extérieur. Dans cette étape, le recensement des acteurs et la description des actions liées au système sont les plus reconnus de sa description soit en diagramme soit textuellement.

La figure 6.2 indique le formalisme du cas d'utilisation.

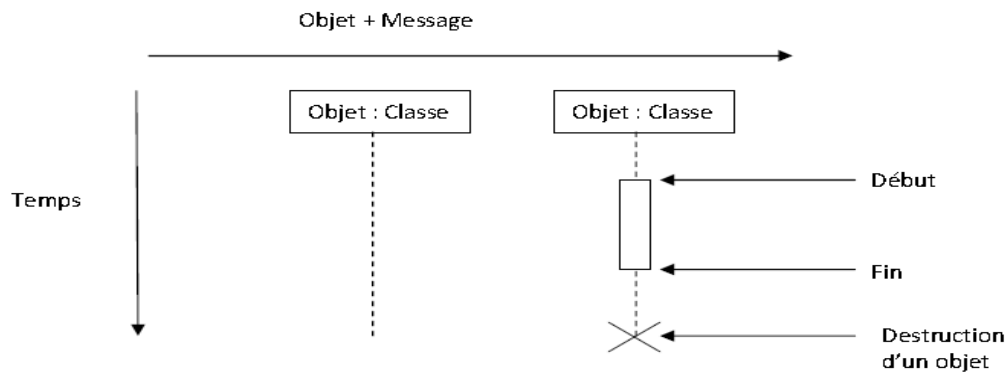


Figure 6.2: Formalisme d'un cas d'utilisation

##### **6.5.1.1. Relation entre cas d'utilisation**

L'intrusion « include » qui montre qu'un cas d'utilisation utilise un autre cas d'utilisation. Tandis que l'extension « extend » montre qu'un cas d'utilisation complète un autre cas d'utilisation.

Formalisme du diagramme de séquence



**Figure 6.3 Formalisme du diagramme de séquence**

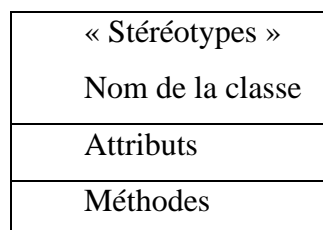
### 6.5.1.2. Le diagramme de classe

Le diagramme de classes constitue un élément très important de la modélisation : il permet de définir quelles seront les composantes du système final : il ne permet en revanche pas de définir le nombre et l'état des instances individuelles. Néanmoins, on constate souvent qu'un diagramme de classe proprement réalisé permet de structurer le travail de développement de manière très efficace ; il permet aussi, dans le cas de travaux réalisés en groupe (ce qui est pratiquement toujours le cas dans les milieux industriels), de séparer les composantes de manière à pouvoir répartir le travail de développement entre les membres du groupe. Un diagramme de classes fait abstraction des aspects dynamiques et temporels.

Il sert aussi à décrire la structure statique du logiciel avec la collection d'éléments qui sont :

- Classes (attributs, opérations, ...) ;
- Relations : associations, agrégations, compositions, héritage, dépendance ;
- Paquetages.

La notation pour les classes est représentée par la figure 6.4



**Figure 6.4 : Notation pour les classes**

Les méthodes représentent les actions qui peuvent être effectuées sur la classe.

Syntaxe des attributs :

Visibilité nom : type=valeur
------------------------------

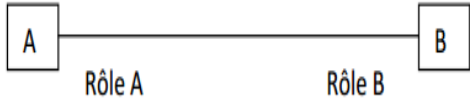

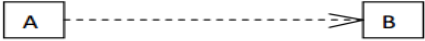


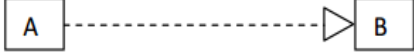
Les différents types de visibilité sont montrés dans le tableau 6.3

**Tableau Les différents types de visibilité**

Modificateur	Visibilité
Privé (-)	Seuls les membres de la classe uniquement
Public (+)	Accessibles par toutes les classes
Package (~)	Accessibles par toutes les classes de même package
Protégé (#)	A l'intérieur de la classe et dans la sous classe

Il y a divers types de relation entre les classes, dont nous allons résumer dans le tableau 6.4.

**Tableau 6.4: Les différents types de relations entre les classes**

Type de relation	Représentation graphique
<u>Association</u> : Elles représentent un lien durable ou ponctuel entre deux objets, une appartenance, ou une collaboration. L'association permet à A d'atteindre B.	
<u>Généralisation et héritage</u> : B est la classe de base et A est la classe dérivée (spécialisée). La classe A peut posséder toutes les caractéristiques de sa classe parent B.	
<u>Dépendance</u> : C'est une relation unidirectionnelle. La modification de B peut entraîner la modification de A.	
<u>Agrégation</u> : Lorsqu'un objet en contient d'autres, on parle d'agrégation. A est inclus dans B.	
<u>Composition</u> (agrégation composite) : Quand le composite (B) est détruit, le composant (A) l'est aussi	
<u>Implémentation</u> : Une classe peut implémenter une interface.	

Les interfaces se différencient des autres classes par le stéréotype <<interface>>.	
---	--

Ces relations peuvent aussi être additionnées par des cardinalités, qui sont représentés dans le tableau 6.5.

Tableau 6.5 : Type de relations

Cardinalité	Signification
1	Une et une seule instance
0..1	Zéro ou un
m.. n	De m à n (où m et n sont des entiers naturels)
n	Exactement n instances
*	Plusieurs
0.. *	De zéro à plusieurs
1.. *	D'une à plusieurs

#### 6.5.1.4.1. Héritage

##### a) Spécialisation

Les hiérarchies de classes permettent de gérer la complexité, en ordonnant les objets au sein d'arborescence de classes, d'abstraction croissante.

##### b) Généralisation

Démarche ascendante, qui consiste à capturer les particularités communes d'un ensemble d'objets, issus de classes différentes. Elle consiste à factoriser les propriétés d'un ensemble de classes, sous forme d'une superclasse, plus abstraite (permet de gagner en généralité).

##### c) Classification

L'héritage (spécialisation et généralisation) permet la classification des objets.

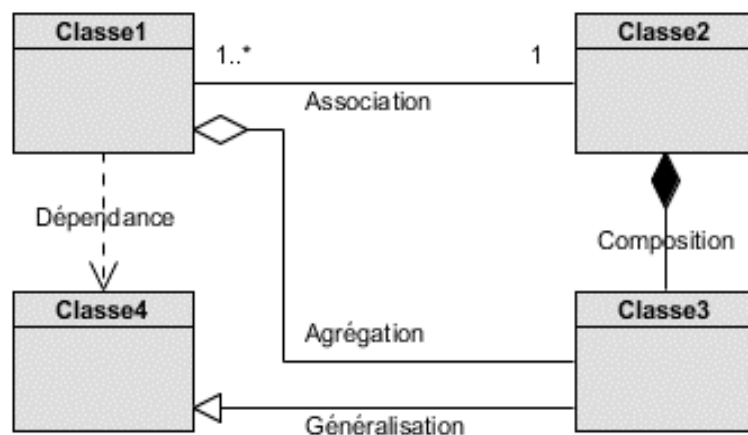
Une bonne classification est stable et extensible : ne classifiez pas les objets selon des critères instables (selon ce qui caractérise leur état) ou trop vagues (car cela génère trop de sous-classes). Les critères de classification sont « subjectifs ». Le principe de substitution permet de déterminer si une relation d'héritage est bien employée pour la classification. Il doit être possible de substituer n'importe quelle instance d'une superclasse, par n'importe quelle

instance d'une de ses sous-classes, sans que la sémantique d'un programme écrit dans les termes de la superclasse ne soit affectée.

Si Y hérite de X, cela signifie qu'Y est une sorte de X (analogie entre classification et théorie des ensembles).

#### d) Formalisme

Le formalisme du diagramme de classe est représenté par la figure 6.5.



**Figure 6.5 : Formalisme du diagramme de classe**

##### 6.5.1.3. Diagramme de cas d'utilisation global du système

Le diagramme de cas d'utilisation global résume toutes les fonctionnalités possibles de notre système. En partant des formalismes cités des acteurs selon la figure 6.1 et des cas d'utilisations, figure 6.2, nous pouvons obtenir la figure 6.6 représentant le diagramme de cas d'utilisation global de notre projet.

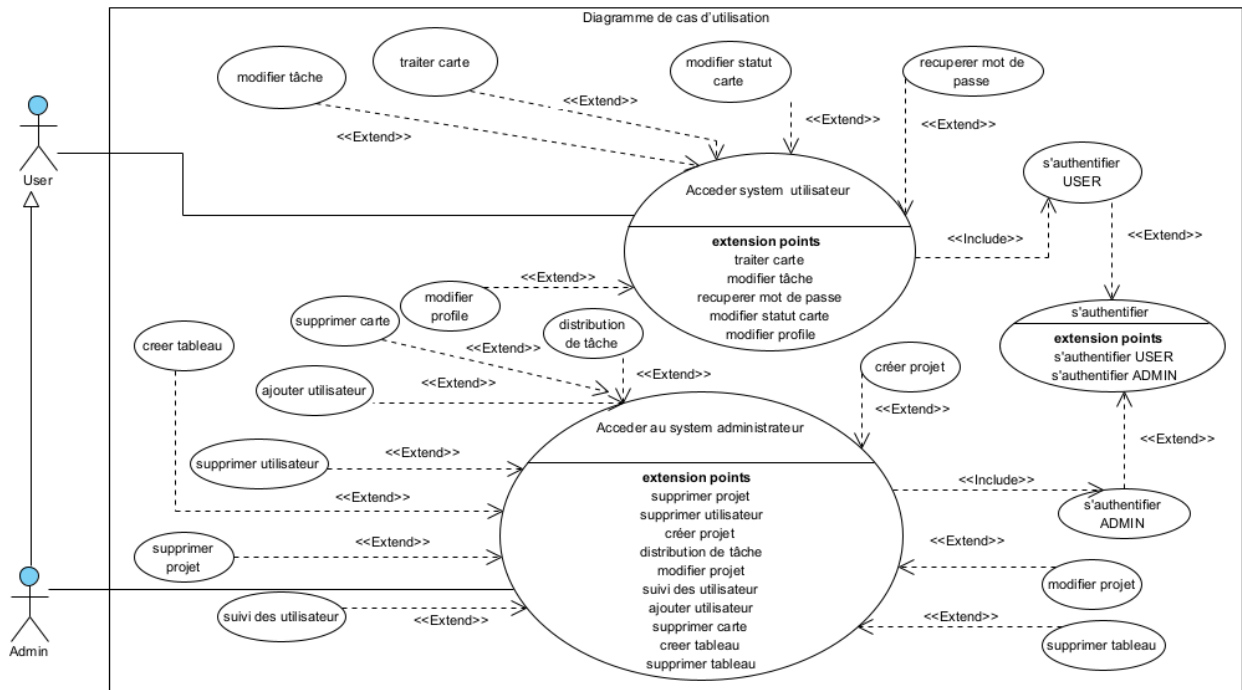


Figure 6.6 : Diagramme de cas d'utilisation du projet

#### 6.5.1.4. Description textuelle des cas d'utilisation

Les descriptions textuelles des cas d'utilisation servent à résumer de manière plus convenable la description des cas d'utilisation. Dans cette optique, chaque description textuelle est composée des grandes lignes décrites par le tableau 6.6.

Tableau 6.6 : Les éléments d'une description textuelle

Ligne	Description
Cas d'utilisation	Le nom du cas d'utilisation étudié
Précondition	Les conditions nécessaires au préalable du cas d'utilisation pour qu'il soit exécutable
Scénario nominal	Scénario normal du cas d'utilisation
Scénario alternatif	Le pire des scénarios
Postconditions	Les conditions nécessaires à l'achèvement du cas d'utilisation

##### a) Cas d'utilisation : « accéder système utilisateur »

Tableau 6.7 : Description textuelle cas d'utilisation « accéder système utilisateur »

Cas d'utilisation	Accéder système utilisateur
Acteur	Utilisateur
But	Avoir base de connaissance sur Trello
Résumé du métier	L'utilisateur est chargé d'effectuer un traitement de tâche qui lui est assignés
Précondition	Avoir login et mot de passe
Post condition	Accès au système ou ouverture page d'accueil

### Scenario Nominale

Tableau 6.8: Scenario Nominale cas d'utilisation « accéder système utilisateur »

N° d'enchaînement	Action
1	L'utilisateur demande page d'authentification
2	Le système envoyé le formulaire
3	L'utilisateur remplit le formulaire
3	Le système vérifie les informations saisies
6	L'utilisateur peut voir la carte qui lui est assigné
7	Traiter la tâche
8	Modifier statut carte
9	Modification profile

### Scenario Alternatif

Tableau 6.9: Scenario Alternatif cas d'utilisation « accéder système utilisateur »

N° d'enchaînement	Action
4	Erreur informations
5	5.1. Le système renvoie le formulaire d'authentification 5.2. L'utilisateur ressaisi les informations



### b) Cas d'utilisation « accéder système administrateur »

Tableau6.10 : Description textuelle cas d'utilisation « accéder système administrateur »

Cas d'utilisation	Accéder système administrateur
Acteur	Le responsable
But	Administrer les informations concernant
Résumé du métier	L'utilisateur est chargé de créer, projet carte, utilisateur, distribué le tâche et d'effectuer le suivi des utilisateurs
Précondition	Avoir login et mot de passe
Post condition	Accès au système ou ouverture page d'administration

### Scenario Nominale

Tableau6.11 : Scenario Nominale cas d'utilisation « accéder système administrateur »

N° d'enchaînement	Action
1	Demander page d'authentification
2	Le système envoyé le formulaire
3	L'utilisateur rempli le formulaire
3	Le système vérifie les informations saisies
6	Créer projet
7	Créer tableau
8	Créer carte
9	Créer tâche
10	Distribution tâche
11	Supprimer projet
12	Supprimer tableau
13	Supprimer carte
14	Modifier carte

15	Créer compte utilisateur
16	Modifier utilisateur
17	Supprimer compte utilisateur
18	Suivre les activités des utilisateurs

### Scenario Alternatif

Tableau 6.12: Scenario Alternatif cas d'utilisation « accéder système administrateur »

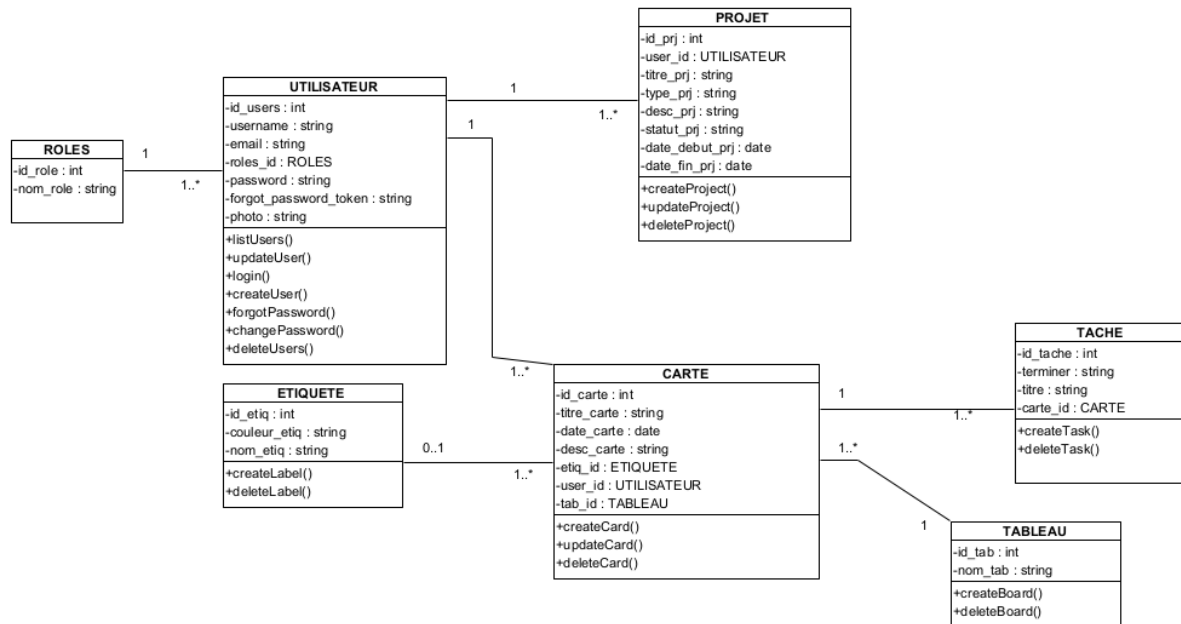
N° d'enchaînement	Action
4	Erreur informations
5	5.1. Le système renvoie le formulaire d'authentification 5.1. Le responsable ressaisie les informations

#### 6.5.1.5. Diagramme de séquence

Le diagramme de séquence permet de montrer les interactions d'objets dans le cadre d'un scénario d'une utilisation. Conformément à formalisme sur la Figure 6.3, le diagramme de séquence utilisateur de notre application se présente comme dans la figure 6.6

### 6.5.1.6. Diagramme de classe du projet

Le diagramme de classes est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Selon le formalisme de la figure 6. 4, le diagramme de classe de notre application se présente comme sur la Figure 6.8.



## **PARTIE III : MISE EN ŒUVRE DE L'APPLICATION**

### **Chapitre 7 : SPECIFICATION DES OUTILS DE REALISATION**

#### **7.1. Le système de gestion de base de données**

Un logiciel qui permet d'interagir avec une base de données s'appelle un Système de Gestion de Base de Données (SGBD).

Gérer une base de données consiste à contrôler les données stockées, regroupées sous forme de tables. Pour cela, nous avons besoin de choisir un SGBD qui permet une facilité à l'accès aux données, une autorisation de multiples utilisateurs à y accéder, et une manipulation de ces données (insertion, modification, suppression).

##### **7.1.1. Définition d'une base de données**

Une base de données est une entité dans laquelle il est possible de stocker des données de façon structurer et avec le moins de redondance possible. Ces données doivent pouvoir être

utilisées par des programmes, par des utilisateurs différents. Ainsi, la notation de base de données est généralement couplée à celle de réseau, afin de pouvoir mettre en commun ces informations, d'où le nom de la base. On parle généralement le système d'information pour désigner toute la structure regroupant les moyens mis en place pour pouvoir partager des données.

### **7.1.2. Utilité d'une base de données**

Une base de données permet de regrouper des données au sein d'un même enregistrement. Cela est d'autant plus utile que les données informatiques sont de plus en plus nombreuses. Une base de données peut être locale, c'est-à-dire utilisable sur une machine par un utilisateur, ou bien répartie, c'est-à-dire que les informations sont stockées sur des machines distantes et accessibles par réseau. L'avantage majeur de l'utilisateur d'une base de données est la possibilité de pouvoir être accédées par plusieurs utilisateurs simultanément.

## **7.2. Caractéristique d'un SGBD**

L'architecture à trois niveaux définis par le standard ANSI/PARC permet d'avoir une indépendance entre les données et les traitements. D'une manière générale, un SGBD doit avoir les caractéristiques suivantes :

- Indépendance physique : le niveau physique peut être modifié indépendamment du niveau conception. Cela signifie que tous les aspects matériels de la base de données n'apparaissent pas pour l'utilisateur, il s'agit simplement d'une structure transparente de représentation des informations.
- Indépendance logique : le niveau conceptuel doit pouvoir être modifié sans remettre en cause le niveau physique, c'est-à-dire que l'administrateur de la base doit pouvoir la faire évoluer sans que cela gêne les utilisateurs.
- Rapidité d'accès : le système doit pouvoir fournir les réponses à la requête le plus vite possible, cela implique des algorithmes de recherche rapides.
- Administration centralisée : le SGBD doit pouvoir éviter dans la mesure du possible des informations redondantes, afin d'éviter d'une part un gaspillage d'espace mémoire mais aussi des erreurs.
- Sécurisation des données : le SGBD doit représenter des mécanismes permettant de gérer les droits d'accès aux données selon les utilisateurs.

## **7.3. Points forts et faibles des SGBD**

Considérons quatre SGBD le plus souvent, tel que : MySQL, PostgreSQL. Les avantages et les inconvénients de chaque SGBD sont figurés dans le tableau 7.1.

**Tableau 7.1: Avantages et inconvénients des SGBD**

Base de données	Avantages	Inconvénients
MySQL	<ul style="list-style-type: none"> <li>- Multi-utilisateurs ;</li> <li>- Gratuit (sauf si vous commercialisez un service ou un logiciel qui utilise MySQL) ;</li> <li>- Interfaces de programmation (API) : C, Perl, PHP, Python et Java ;</li> <li>- Langage de requête : SQL (langage de requête le plus répandu) ;</li> <li>- Portabilité ;</li> <li>- Multi pilotage.</li> </ul>	<ul style="list-style-type: none"> <li>- Pas d'héritage de tables ;</li> <li>- Support incomplet de triggers et procédure stockées ;</li> <li>- Manque de robustesse avec forte volumétrie.</li> <li>- N'importe pas des références d'intégrité relationnelle.</li> </ul>
PostgreSQL	<ul style="list-style-type: none"> <li>- Multi plate-forme ;</li> <li>- Libre ;</li> <li>- Largement reconnu, comportement stable et plus proche de l'Oracle.</li> </ul>	<ul style="list-style-type: none"> <li>- Les tables sont obligatoirement transactionnelles ;</li> <li>- Les commandes INSERT, DELETE et UPDATE sont plus lente.</li> </ul>

Nous dressons le tableau 7.2 pour comparer les quatre SGBD cités ci-dessus.

Tableau 7.2 : Comparaison des SGBD

Caractéristique	MySQL	PostgreSQL
<b>Rapidité</b>	OUI	OUI
<b>Robustesse</b>	Moyen	Elevé
<b>Multiplateforme</b>	OUI	OUI
<b>Facilité d'utilisation</b>	OUI	OUI
<b>Sécurité</b>	OUI	OUI
<b>Fonctionnalités</b>	Moyen	Moyen

D'après le tableau 7.2, qui nous présente la comparaison des SGBD, nous choisissons d'utiliser MySQL comme SGBD, vu qu'il est facile à manipulé, multiplateforme, sécurisé et robuste. Il est fiable et relativement performant Puis il est très riche fonctionnellement et possède une multitude de modules. Enfin, il est facile à utiliser et à administrer.

## **7.4. Présentation de MySQL.**

MySQL est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisées. Il est multi-thread et multi-utilisateur.

C'est un logiciel libre, open source, développé sous double licence selon qu'il est distribué avec un produit libre ou avec un produit propriétaire. Dans ce dernier cas, la licence est payante, sinon c'est la licence publique générale GNU (GPL) qui s'applique. Un logiciel qui intègre du code MySQL ou intègre MySQL lors de son installation devra donc être libre ou acquérir une licence payante. Cependant, si la base de données est séparée du logiciel propriétaire qui ne fait qu'utiliser des API tierces (par exemple en C# ou PHP), alors il n'y a pas besoin d'acquérir une licence payante MySQL. Ce type de licence double est utilisé par d'autres produits comme le framework de développement de logiciels Qt (pour les versions antérieures à la 4.5).

MySQL fonctionne sur de nombreux systèmes d'exploitation différents

## **7.5. Choix de langage de Programmation web**

### **Choix d'utilisation framework PHP**

#### **Principales différences entre Symfony & Laravel**

- Laravel utilise un modèle MVC très répandu alors que Symfony se base sur une approche de “composants réutilisables”.
- Les 2 frameworks utilisent tous les 2 un ORM (Object Relational Mapping) pour gérer les requêtes de manière native, mais ils utilisent chacun un ORM différent. Doctrine pour Symfony et Eloquent pour Laravel.
- Les migrations de la base de données sont automatiques avec Symfony, grâce aux modèles définies dans le code. Avec Laravel, les migrations doivent se faire manuellement, ce qui implique d'avoir des connaissances plus poussées en SQL.
- Symfony et Laravel possède un moteur de template différent : Twig pour Symfony et Blade pour Laravel.
- Une application Laravel peut permettre de développer un POC (Proof of Concept) beaucoup plus rapidement grâce à son système de composants réutilisables.

- Symfony possède une plus grande communauté open-source que Laravel. On peut donc plus facilement trouver de l'aide suite à un problème ou bien voir ses remontés de bugs corrigées plus vite.

## **7.6. Présentation ReactJS**

# **Chapitre 8: MISE EN ŒUVRE ET IMPLEMENTATION**

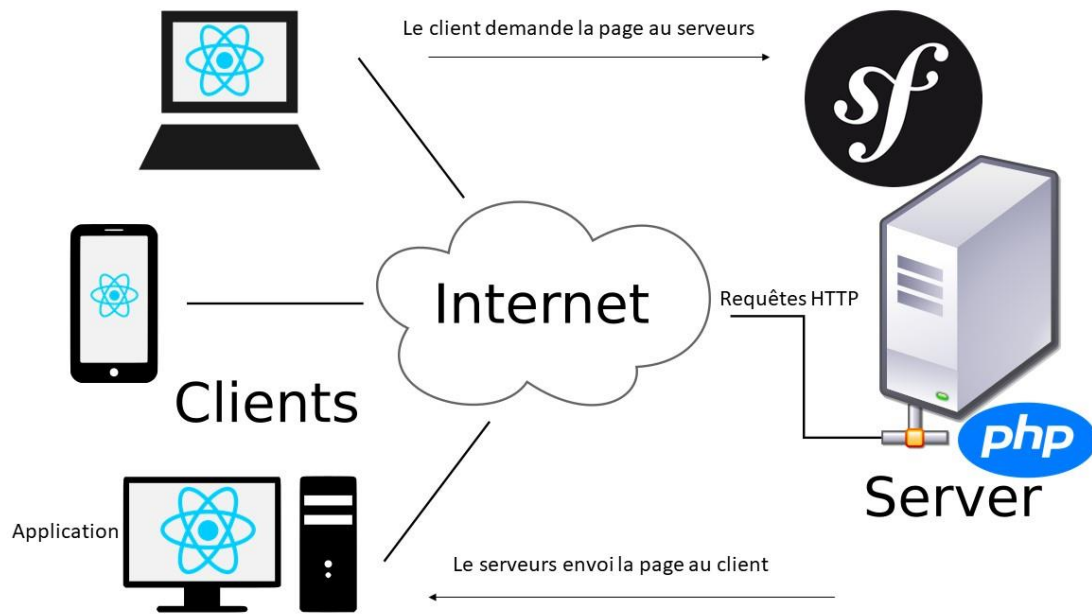
## **8.1 : Architecture logicielle de l'application**

L'architecture logicielle est une vue tournée sur l'organisation interne et le découpage d'un logiciel en modules. Dans les logiciels les caractéristiques communes concernent les interfaces, c'est-à-dire la connectique qui permet la communication entre les modules, ainsi que les caractéristiques du matériel informatique et du système d'exploitation sur lequel le logiciel s'exécutera et les caractéristiques du réseau informatique qui sera utilisé

L'architecture logicielle explique le déroulement d'une requête en tapant une adresse dans le navigateur.

La figure 8.1 représente l'architecture logicielle de notre application

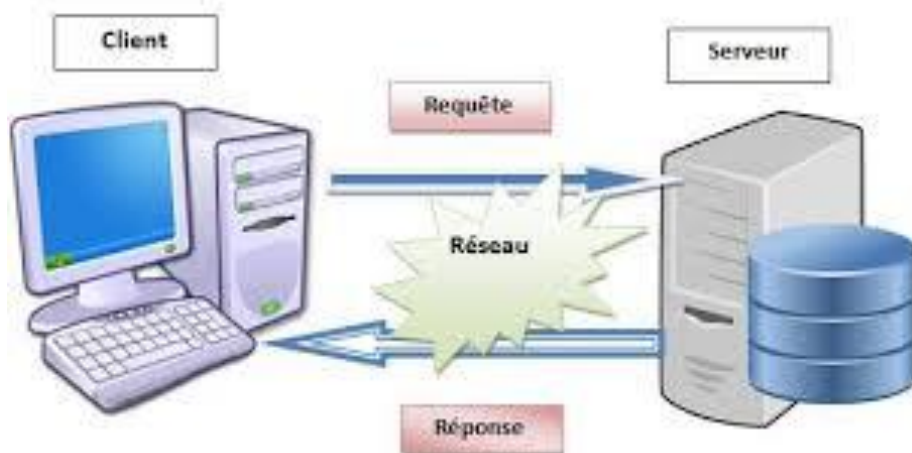




**Figure 8.1: Architecture logicielle de l'application**

## 8.2 : Architecture matérielle

L'architecture matérielle [Marc L ,2009], est une vue tournée sur le choix et l'organisation des différents composants électroniques d'un appareil informatique. L'architecture matérielle est donc une définition des composants matériels pour la réalisation d'une application. Notre application a une architecture matérielle comme la figure 8.2 nous montre.



**Figure 8.2: Architecture matérielle de l'application**

Le client demande l'affichage d'une page au serveur web (soit en cliquant sur le lien ou en validant un formulaire après avoir rempli). Le serveur reçoit la demande et va charger la page.

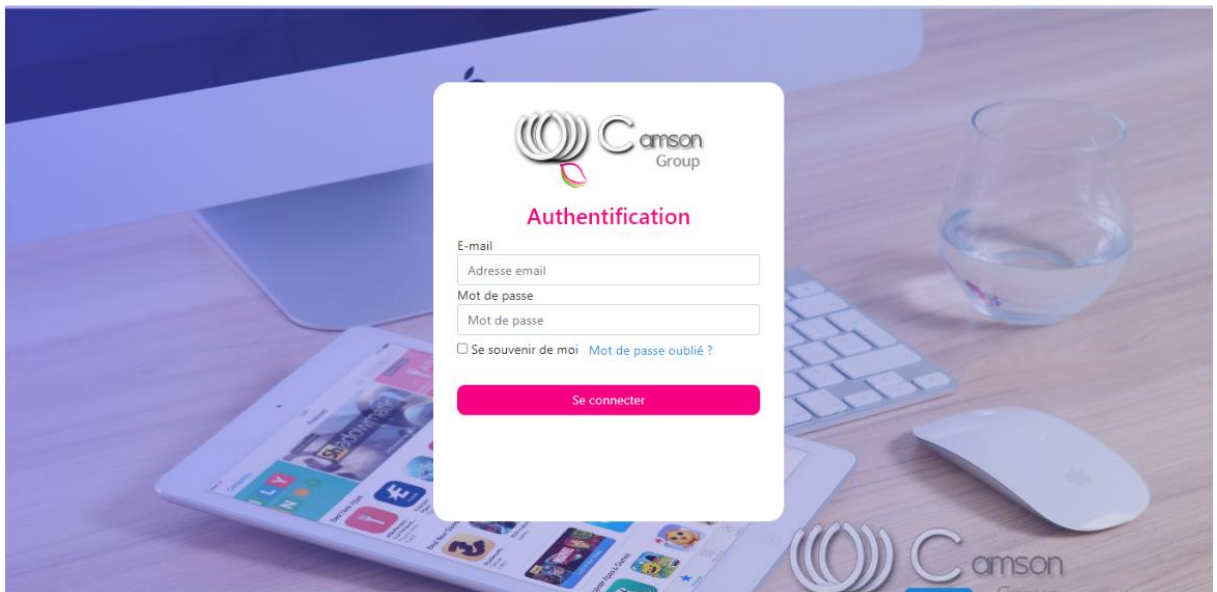
## Chapitre 9 : PRESENTATION DE L'APPLICATION

### 9.1 : Aperçus de l'application

Nous avons presque terminé la réalisation de l'application, y compris les fonctionnalités souhaitées. Les figures suivantes nous montrent quelques interfaces réalisées.

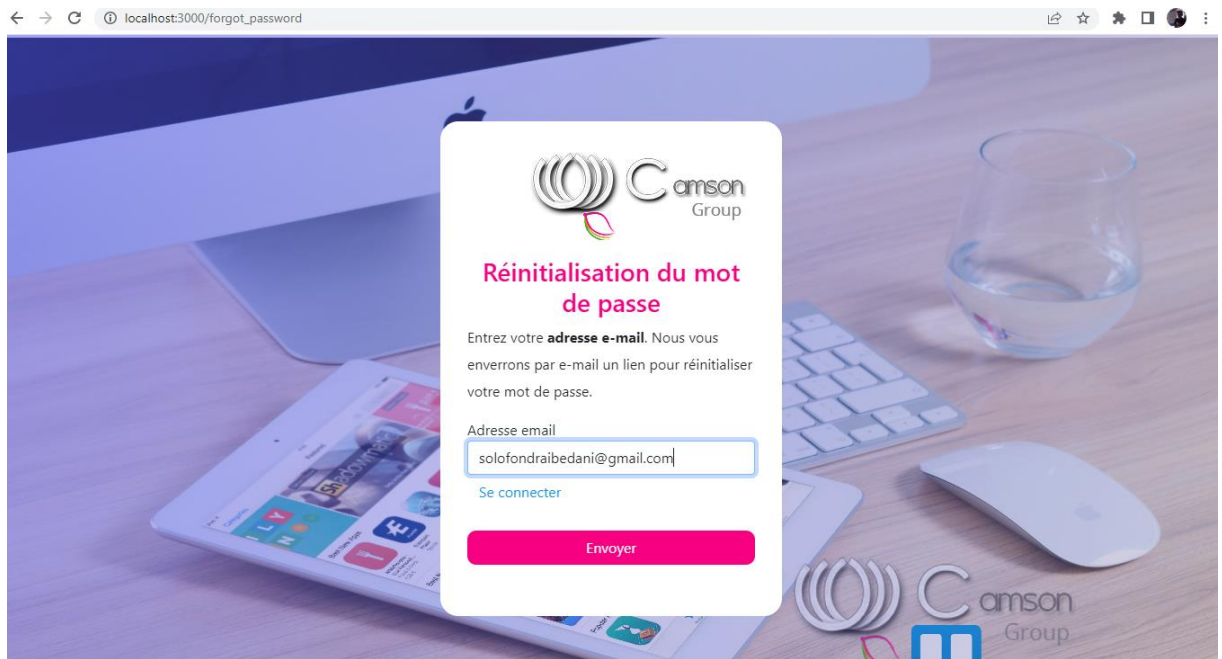
#### 9.1.1. Page d'authentification

Pour bien sécuriser notre application, surtout nos données manipulées, nous devons s'authentifier avant d'accéder aux fonctionnalités de l'application. L'utilisateur doit avoir un compte.



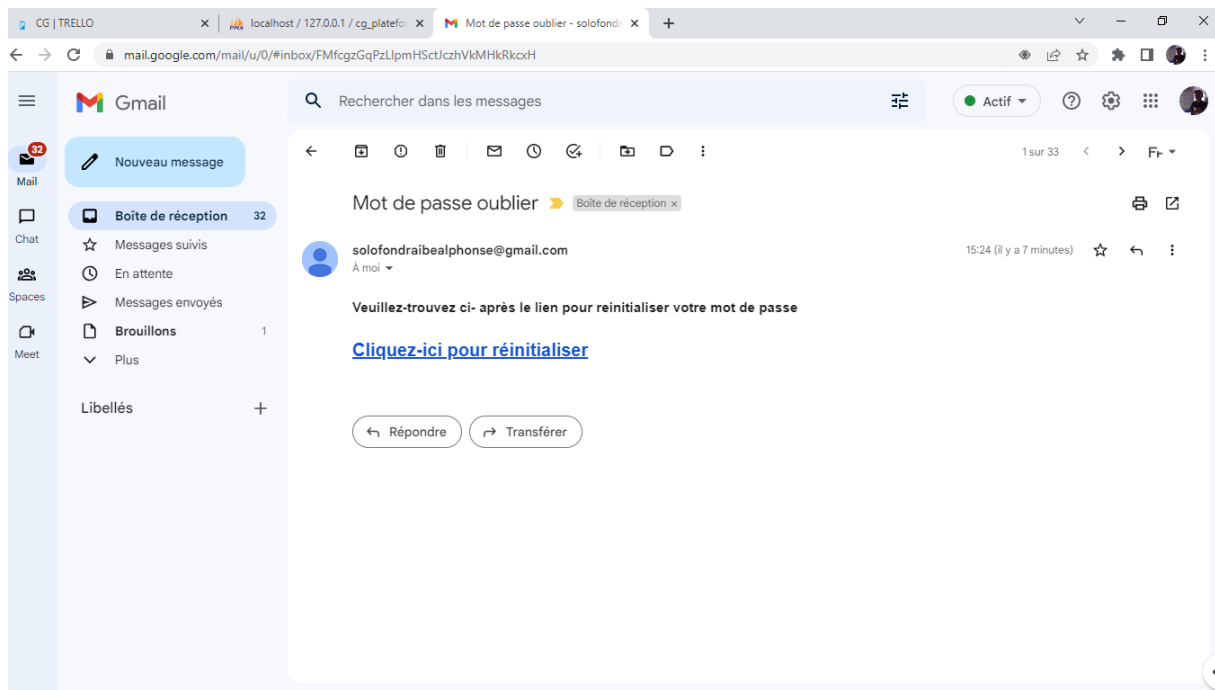
**Figure 9.1: Page d'authentification**

D'après la figure 9.1, l'utilisateur doit remplir les champs de leur adresse email et son mot de passe. S'il oublie son mot de passe, il suffit de cliquer sur le lien 'mot de passe oublié'.

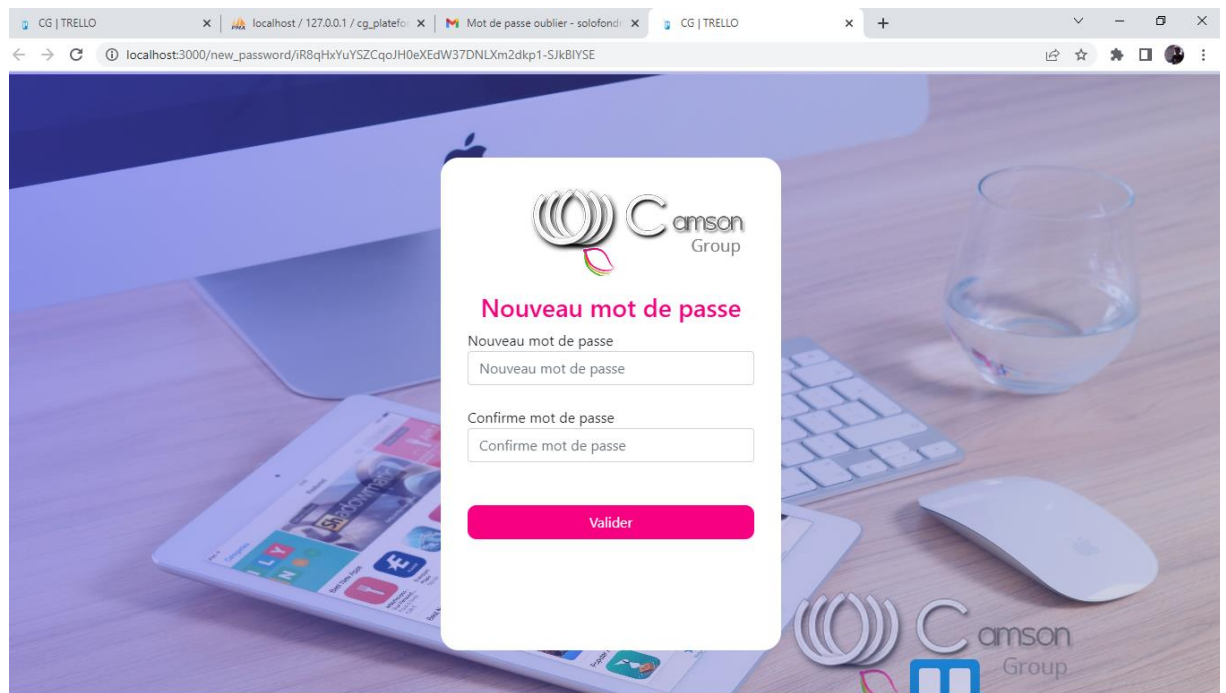


**Figure 9.2: Page de réinitialisation de mot de passe**

D'après la figure 9.2, l'utilisateur doit remplir saisir sont adresse email, le programme cherche l'adresse email dans la base des données, si on trouve il recevra un lien pour pouvoir réinitialiser votre mot de passe, et ce lien expirera dans 24 heure, sinon il affiche le message comme « L'adresse email que vous avez saisissez n'existe pas dans la base des données »



**Figure 9.3 Lien par email pour réinitialiser mot de passe**



**Figure 9.4 Nouveau mot de passe**