UNIVERSITÉ DE FIANARANTSOA

ECOLE DE MANAGEMENT ET D'INNOVATION TECHNOLOGIQUE

MINI-MÉMORE EN VUE DE PASSAGE EN TROISIÈME ANNÉE DE LICENCE

EN DÉVELOPPEMENT D'APPLICATION INTRANET ET INTERNET

Mention : Informatique

Conception et réalisation d'un système de gestion des courriers

Cas de la direction régional du budget Vakinankaratra

Présenté par : ANDRIANARIVELO Sitrakarinjaka Tovonirina

Sous la direction de : RAKOTOARISON Tsiorinantenaina René

Année universitaire : 2018-2019

ANDRIANARIVELO Sitrakarinjaka Tovonirina

Développeur informatique

Lot 0910 C 87 Mahafaly Vatofotsy
Tél: 034 93 605 22
Email: sitrakarinjakatovonirina@gmail.com

FORMATION

2018-aujourd'hui Fianarantsoa	En cours de Licence en développement d'application internet et intranet *EMIT (Ecole de Management et d'Innovation Technologique)*
2017 Antsirabe	Baccalauréat série C *Gallo Junior*
2014 Antsirabe	BEPC *Les orchidées*

Langues

Malagasy :

Anglais :

Français :

Logiciels: Word, Excel, Powerpoint,Adobe Photoshop, Adobe illustrator.

Languages: Java, Python, C#, HTML 5, CSS 3, PHP, JavaScript.

Frameworks: React, React Native, Symfony.

EXPERIENCE PROFESSIONNELLE

Juillet 2018 – septembre 2019 **EMIT**
Fianarantsoa

Tâches réalisées : Conception et réalisation d'un site e-commerce de location de vaisselle

CENTRES D'INTÉRÊT

Voyager, Membre d'une association répondant au nom de SELMADA (Solidarité Etudiant Lyon-Madagascar).

Sport, hobbies: Basket-ball, velo, natation, …

Liste des figures

Liste des tableaux

Liste des abréviations

AN : Alpha Numérique

API: Application Programming Interface

CSS: Cascading StyleSheet

IDE: Integrated Development Environment

MVC: Model View Controller

SGBD : Système de Gestion des Bases de Données

UML: Unified Modeling Language

ORM: Object-Relational Mapping

POO : Programmation Orientée Objet

RPM : Relation Publique et Multimédia

AES : Administration Economique et Sociale

SIGD : Système d'information Géomatique et Décisionnel

DA2I : Développement d'Application Inttranet et Internet

EMIT : Ecole de Management et d'Innovation Technologique

MERISE : Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise

DRB : Direction Régional du Budget

NASA : la National Aeronautics and Space Administration (Administration nationale de l'espace et de l'aéronautique, en français)

MERISE : Méthode d'étude et de réalisation informatique pour les systèmes d'entreprise

SADT: Structured Analysis and Design Technique

RAM: Random Access Memory

VCS: Version Control System

PHP: PHP Hypertext Preprocessor

TABLE DES MATIERES

AVANT-PROPOS

Les Nouvelles Technologies de l'Information et de la Communication (NTIC) sont en constante évolution et leur utilisation dans le monde du travail est devenue indispensable. Cela se traduit par la conception et la maintenance de ces technologies ou encore par l'investissement des entreprises dans ces nouveaux outils. L'importance croissante des NTIC dans le monde du travail est indéniable et leur domaine d'application ne cesse de s'étendre. Toutefois, l'utilisation de ces outils nécessite une compréhension approfondie de leurs caractéristiques, de leurs avantages et de leur influence.

Dans ce contexte, la gestion des courriers est une fonction clé dans les organisations publiques et privées. La mise en place d'un système de gestion des courriers efficace peut contribuer à améliorer la productivité et la transparence de l'administration. Néanmoins, cette fonction peut également être une source de problèmes et de dysfonctionnements si elle n'est pas gérée de manière optimale.

L'objectif de ce mini-mémoire est d'étudier la conception et la réalisation d'un système de gestion des courriers pour la représentation régionale du Ministère des Finances et du Budget. Plus précisément, ce travail vise à répondre aux questions suivantes :

Comment concevoir et réaliser un système de gestion des courriers efficace dans un contexte organisationnel spécifique ?

Quels sont les outils et les techniques disponibles pour la conception et la réalisation d'un tel système ?

Quels sont les avantages et les limites de la mise en place d'un système de gestion des courriers dans une organisation publique ?

Comment évaluer l'impact d'un tel système sur la productivité, la transparence et la qualité de service de l'administration ?

Notre méthodologie consiste en une étude de cas appliquée à la représentation régionale du Ministère des Finances et du Budget. Cette étude de cas s'appuiera sur une analyse documentaire et des entretiens semi-directifs avec des acteurs clés de l'organisation. Les résultats de cette étude seront présentés et discutés dans la suite de ce travail.

On vise à fournir des informations et des recommandations pratiques pour la conception et la réalisation d'un système de gestion des courriers dans une organisation publique. Nous

espérons que ce travail sera utile pour les praticiens de l'administration et pour les chercheurs intéressés par ce sujet.

REMERCIEMENTS

Nous remercions tout d'abord Dieu tout puissant de nous avoir permis de mener à terme ce projet qui présente pour nous le point de départ de notre carrière.

Il nous est agréable d'exprimer notre reconnaissance auprès de toutes les personnes, dont l'intervention au cours de ce projet, a favorisé son aboutissement.

Mes vifs remerciements s'adressent aussi à :

- Monsieur RAFAMANTANANTSOA Fontaine, Professeur, Président de l'université de Fianarantsoa de m'avoir permis d'effectuer mes études au sein l'Université ;
- Monsieur HAJALALAINA Aimé Richard, Docteur et Directeur de l'Ecole de Management et d'Innovation Technologique qui m'a donné l'opportunité d'effectuer mes études au sein de l'EMIT ;
- Madame RABEZANAHARY Hobiniaina responsable de la mention informatique en licence ;

Ainsi, nous tenons également à remercier tout le personnel de la directionrégional du budget Vakinankaratra pour leurs précieuses directives et leurs conseils pertinents qui ont été d'un appui considérable au cours de ce projet.

INTRODUCTION GÉNÉRALE

La direction régionale du budget dispose déjà de plusieurs solutions informatiques pour effectuer son travail efficacement, notamment grâce à un service informatique qui aide les gens dans leurs planifications financières et fiscales en utilisant des sites dédiés du ministère. Ce service forme les utilisateurs et crée des logins pour eux tout en les accompagnant dans le processus.

Cependant, malgré l'utilisation de ces outils, le service a identifié un besoin accru de fonctionnalités et d'outils de gestion pour répondre aux exigences croissantes de son travail. Il est donc nécessaire de développer une solution informatique sur mesure qui permettra au service d'optimiser ses activités et de fournir des services de qualité supérieure aux citoyens. Pour ce faire, le service dispose de plusieurs ordinateurs à la disposition des utilisateurs qui en ont besoin.

Notre objectif est de concevoir une solution qui aidera le service régional du budget à gérer plus efficacement leurs courriers, tout en améliorant l'expérience utilisateur et en offrant des fonctionnalités supplémentaires pour répondre aux besoins spécifiques du service.

La gestion des courriers est un élément essentiel de toute entreprise ou organisation, qu'il s'agisse de traiter les demandes des clients, de gérer la correspondance interne ou d'assurer le suivi des documents importants. Cependant, ce processus peut être fastidieux, chronophage et sujette aux erreurs humaines. C'est pourquoi la mise en place d'un système de gestion des courriers informatisé peut aider à simplifier et à rationaliser cette tâche. Dans ce texte, nous explorerons les étapes clés de la conception et de la réalisation d'un tel système, en nous concentrant sur les fonctionnalités clés, les avantages pour l'entreprise et les considérations techniques.

Notre projet de conception et réalisation d'un système de gestion des courriers sera organisé en trois grandes parties. Dans un premier temps, nous allons étudier le contexte général du projet pour mieux appréhender le cœur du travail, les besoins du client et les différentes fonctionnalités que notre application devra assurer. Nous exposerons également l'organisme d'accueil ainsi que notre établissement d'origine.

Par la suite, nous nous pencherons sur l'analyse des besoins fonctionnels du système futur. Nous commencerons par une étude préalable de l'existant afin de détailler les défaillances du système existant et de proposer des solutions adaptées. Nous concevrons ensuite les différents

modules de notre projet en utilisant le Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise (MERISE).

Dans la troisième et dernière partie, nous nous concentrerons sur l'étude technique et la mise en œuvre de notre projet. Nous décrirons l'architecture technique, les outils et les langages utilisés pour la réalisation du nouveau système d'information. Nous expliquerons également la démarche de développement et illustrerons certaines fonctionnalités assurées par notre application à travers une présentation de l'application via des captures d'écran. Enfin, nous discuterons des résultats obtenus.

En conclusion, nous récapitulerons le travail accompli dans le cadre de ce projet de fin d'études. Nous présenterons les connaissances acquises au cours de ce projet ainsi que les possibilités à considérer pour raffiner davantage l'outil développé.

Partie I: Présentation générale

Chapitre 1 : Présentation de l'EMIT

1.2. Historique

L'Ecole 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 Ecole 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. Depuis l'année universitaire 2013-2014, l'école est basculée totalement 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. L'école propose huit parcours répartis en trois mentions, représentés sur le tableau suivant :

Cycle	**Mention Management**	**Mention Informatique**	**Mention Relations Publiques et Multimédia**

Licence	Administration Economique et Sociale	Développement d'Application Internet/Intranet	Communication Multimédia
		Conception, Intégration et Gestion des Systèmes d'Information	Relations Publiques et Communication Organisationnelle
Master	Management Décisionnel	Système d'Information, Géomatique et Décision	Relations Publiques et Multimédia
		Modélisation et Ingénierie Informatique	

Tableau 1.2. *Tableau récapitulatif des formations au sein de l'EMIT*

1.3. Missions

L'école a pour mission, d'abord de dispenser de formations initiales et continues en informatique, en administration et en relations publiques et multimédia. Ensuite, elle offre des services connexes à l'informatique. Puis, elle forme des techniciens opérationnels immédiatement au sein des entreprises. Enfin, elle assure le perfectionnement professionnel des étudiants, des demandeurs d'emplois, des employés et des cadres d'entreprises.

1.4. Formations existantes

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

1.4.1. 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.

1.4.1.1 Mention Management, Parcours Administration Economique et Sociale

La condition d'accès en première année de Licence se fait par voie de concours pour les titulaires d'un Baccalauréat général de toutes séries (A, C, D) ou d'un Baccalauréat technique G1 et G2. 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.

1.4.1.2 Mention Informatique, Parcours Développement d'Application Internet Intranet et Parcours Conception, Intégration et Gestion des Systèmes d'Information

La condition d'accès en première année de Licence se fait par voie de concours pour les élèves titulaires du diplôme de Baccalauréat Général Scientifique (Série C et série D), Baccalauréat Technique Professionnelle et Baccalauréat Technique Technologique (Filière Industrielle, Maintenance Automobile, Ouvrage Bois, Ouvrage Métallique, Génie Civile). 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.

1.4.1.3 Mention Relations Publiques et Multimédia, Parcours Communication Multimédia et Relations Publiques et Communication Organisationnelle

La condition d'accès en première année de Licence se fait par voie de concours pour les titulaires d'un Baccalauréat général de toutes séries (A, C, D) ou d'un Baccalauréat technique G1 et G2. 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.

1.4.2. 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.

1.4.2.1 Mention Management, Parcours Management Décisionnel

La condition d'accès en première année de Master (M1) en S7 se fait par sélection de dossier après l'obtention du diplôme de Licence en Administration Economique et Sociale, en Gestion ou en Economie.

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.

Les sortants peuvent travailler dans les secteurs privés et publics des différentes régions de Madagascar en tant que chefs de conduite de travaux d'enquêtes communautaires, concepteurs de projets, chefs de services ou directeurs d'entreprises. Dans ce cas, les étudiants sortants sont capables de créer une petite entreprise, de monter un projet de développement rural et de gérer un grand projet.

1.4.2.2 Mention Informatique, Parcours Système d'Information, Géomatique et Décision et Parcours Modélisation et Ingénierie Informatique

La condition d'accès en première année de Master (M1) en S7 se fait par sélection de dossier après l'obtention du diplôme de Licence en Informatique et en Mathématique et Informatique pour les Sciences Sociales (MISS). Le Recrutement en S9 se fait par validation des crédits acquis.

Le parcours Système d'Information, Géomatique et Décision a pour objectif de donner un panorama des recherches actuelles et émergeantes 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 représentent 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 pout 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.

1.5. Partenaires de l'EMIT

L'EMIT travaille en collaboration avec plusieurs laboratoires de recherche, d'entreprises et d'autres écoles et universités. Parmi les organisations partenaires, citons à titre d'exemple les laboratoires de recherche tels que le LIMAD, LIMOS, IRD, CNRE, SPAD, Espace Dev, LRI et l'UPR-Green à travers le CIRAD.

Pour ce qui est des écoles et des universités partenaires, il y a entre autres : l'EDMI, l'Université de Toulouse Paul Sabatier, Université de Montpellier 2, l'ENI, GOUVSOMU, IOGA, l'Université de Clermont Auvergne, ESMIA, Université de Mahajanga, ISSTM et l'Université de Fianarantsoa.

L'Ecole est également en partenariat avec plusieurs entreprises, notamment dans le cadre des stages à effecteur à travers chaque parcours, telles que les entreprises Etech consulting, Orange, Lazan'i Betsileo, STAR, Alliance Française de Fianarantsoa, TELMA, BFV-SG, Bank of Africa (BOA), BNI Madagascar, Nelli Studio, YMAGOO, PREMIYA, JIRAMA, les assurances NY HAVANA, MAMA et ARO.

Des organismes gouvernementaux sont également partenaires de l'EMIT : la Région Haute Matsiatra, le Ministère de l'Enseignement Supérieur et de la Recherche Scientifique, le Ministère de l'Education Nationale, le Ministère des Travaux Publics, le Ministère des Finances et du Budget, le Ministère du Tourisme, le Ministère des Transports et de la Météorologie, le Ministère de la Poste, de la Télécommunication et des Technologies Numériques, la Banque Centre de Madagascar, le Foibe Taosaritanin'i Madagasikara (FTM), l'INSTAT.

1.6. 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 suivante.

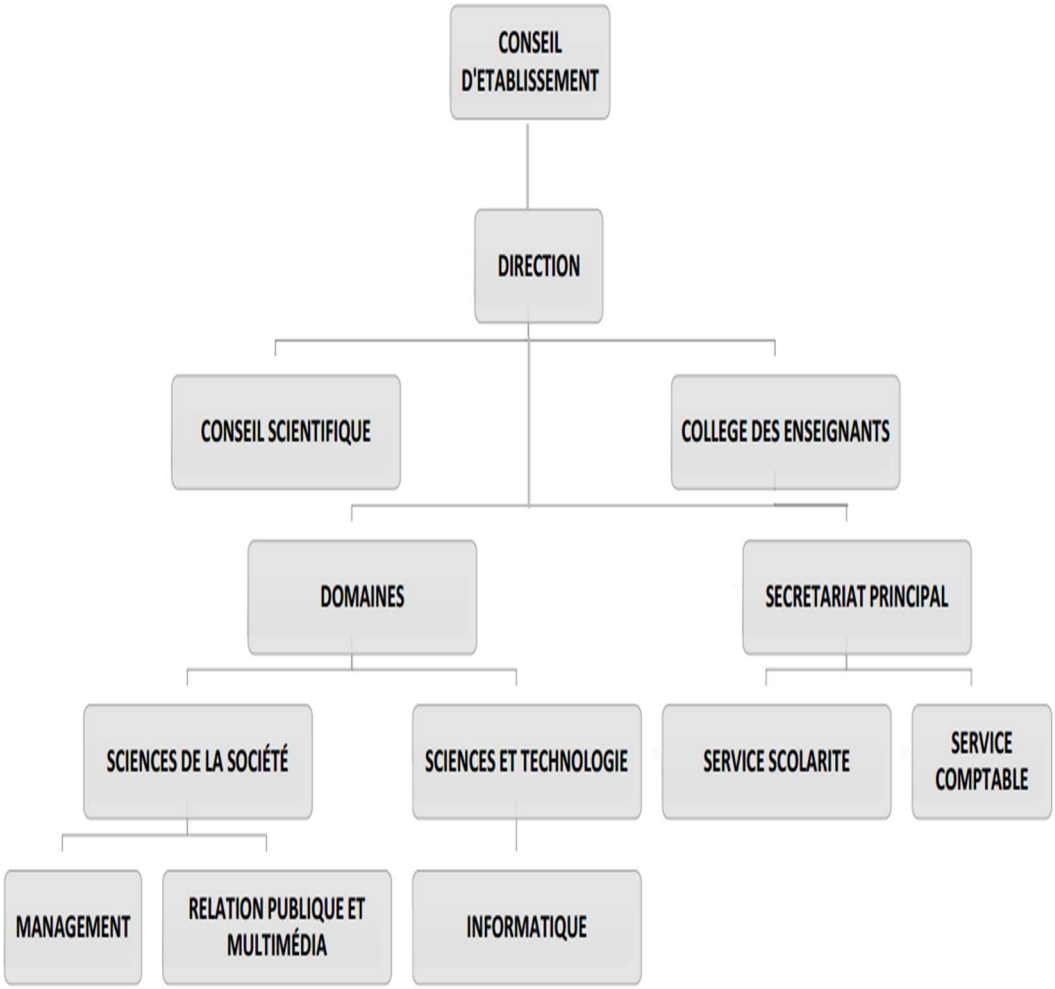

Figure 1.2. Organigramme de l'EMIT

Chapitre 2 : Présentation du service d'accueil

2.1. Généralité

La direction régionale du budget se situe sur la route d'Ambositra à Antsirabe, dans un bâtiment datant de la période coloniale.

2.2. Ses responsabilités

Le budget est un élément crucial de la gestion financière, et la responsabilité de la direction régionale du budget est de veiller à ce que les ressources financières soient utilisées de manière optimale. À cet égard, le Service Régional du Budget utilise une gamme de solutions informatiques pour répondre aux besoins des utilisateurs. Le département dispose d'un pôle informatique dédié à aider les gens dans leur planification financière et fiscale, en utilisant des sites du ministère conçus pour répondre à ces besoins.

Le rôle du pôle informatique ne se limite pas à la formation des utilisateurs et à la création de comptes d'accès. Il s'agit également d'assurer le bon fonctionnement des équipements informatiques mis à la disposition des utilisateurs, tels que les ordinateurs, afin qu'ils puissent bénéficier de solutions informatiques performantes pour gérer leur budget, leurs impôts et autres aspects financiers. Le pôle informatique de la direction régionale du budget joue donc un rôle clé dans l'accompagnement des utilisateurs tout au long de leur parcours, pour leur permettre d'optimiser l'utilisation des outils informatiques à leur disposition.

2.3. Organigramme

Dans de nombreuses organisations, il est essentiel de bien comprendre la structure hiérarchique et les relations fonctionnelles entre les différentes unités de l'entreprise. C'est là qu'intervient l'organigramme, un outil de représentation graphique qui permet de visualiser la structure de l'organisation, ses fonctions et les rapports entre les différents niveaux hiérarchiques. L'organigramme est un moyen efficace de communiquer cette information à toutes les parties prenantes, en particulier aux nouveaux employés qui ont besoin de comprendre la structure de l'entreprise. Dans ce contexte, nous allons examiner l'organigramme de la Direction Régionale du Budget, afin de mieux comprendre sa structure et son fonctionnement.

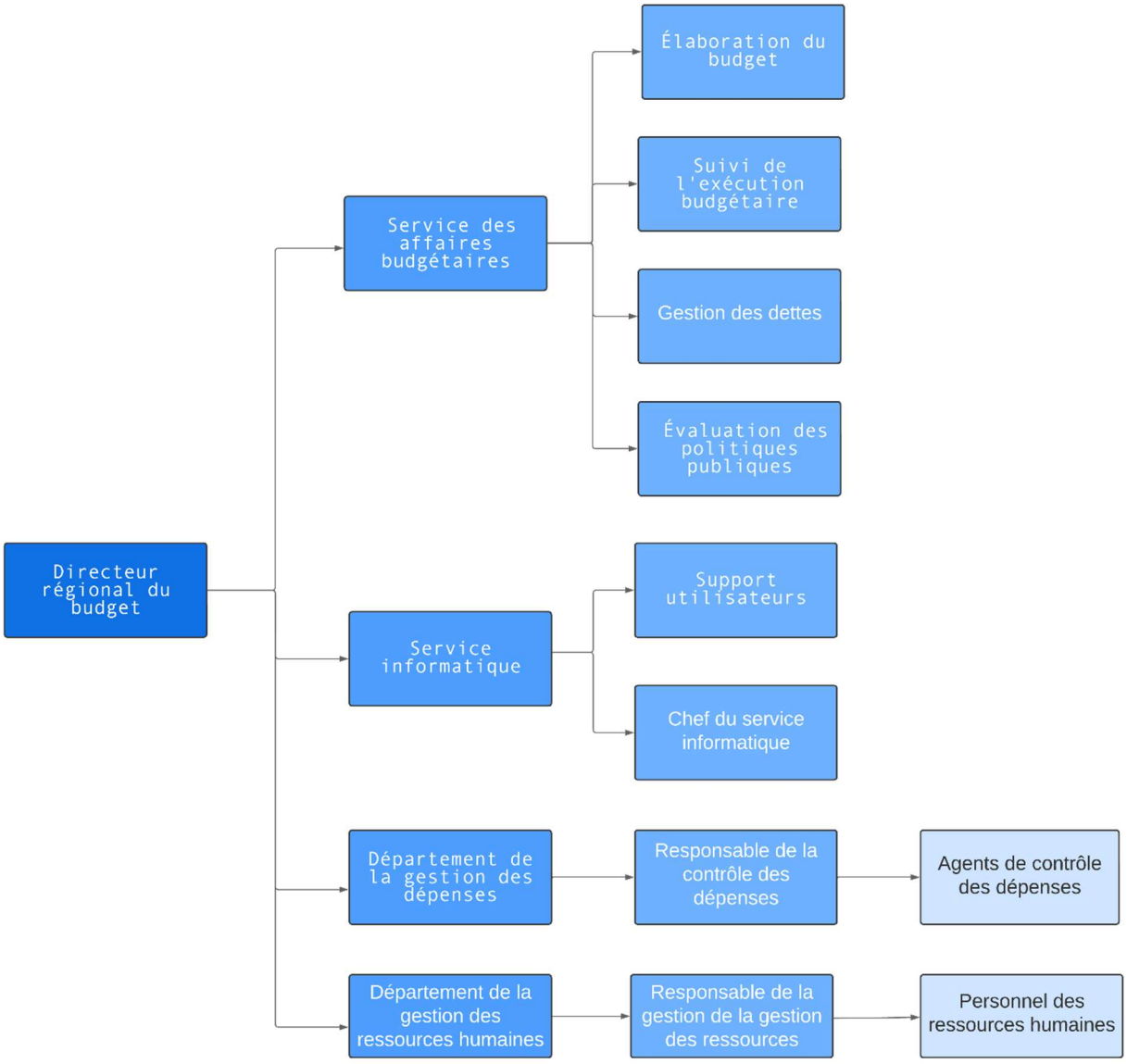

Figure 2.1. Organigramme de la direction régional du budget Vakinankaratra

Chapitre 3 : Présentation du projet

3.1. Généralité

L'informatique est une discipline omniprésente qui a un impact significatif sur de nombreux aspects de la vie, tels que le social, l'économique, le juridique, etc.

Il y a une tendance générale à l'harmonisation des supports utilisés dans les entreprises qui proposent des services de distribution de courriers (tels que la fiche de suivi des courriers, le bordereau de dépôt, le relevé des clients, etc.) afin de pallier les problèmes fastidieux et répétitifs et de répondre aux besoins d'évaluation et de suivi des activités de l'organisation. Le souci d'optimiser les méthodes de gestion des clients en rationalisant leur dossier est également présent.

L'informatisation vas permettre de suivre de manière détaillée la gestion de chaque courrier, ce qui permettra aux responsables de prendre de bonnes décisions et d'offrir la possibilité de retrouver les traces ou les opérations accomplies, tout en générant des statistiques.

En ce qui concerne la distribution de courriers, l'informatique offre la possibilité de suivre les différents courriers vers les clients destinataires de manière simple et efficace via internet, sans être limité par des contraintes de temps ou de lieu.

En somme, l'informatique touche presque tous les secteurs de la vie grâce au progrès constant observé dans chacune de ses branches, notamment avec la présence d'un site web dans notre cas.

Avec tous les bienfaits de l'informatisation le responsable informatique a pris pour projet d'informatiser la gestion des courriers entrants, sortants et internes de la Direction Régionale du Budget dans l'optique d'améliorer la gestion de son service de courrier.

3.2. Objectif et besoins de l'utilisateur

3.2.1. Objectifs

Nous avons comme objectif de concevoir et de réaliser un système de gestion de courriers pour la Direction Régionale du Budget de Vakinankaratra. Ce système permettra d'optimiser la gestion des courriers entrants, sortants et internes en automatisant les différentes étapes du processus, de la réception à l'archivage en passant par la distribution et le suivi des courriers. Le système visera également à améliorer la traçabilité et la sécurité des échanges de courriers,

en assurant notamment la confidentialité et l'intégrité des données. Enfin, notre objectif est de fournir un outil convivial et facile d'utilisation, qui permettra aux utilisateurs de gagner en efficacité et en productivité dans leur travail quotidien.

3.2.2. Enumération des besoins

Le tableau ci-dessous énumère les besoins selon le type de personne au sein du service :

Type d'utilisateur	**Les besoins**
Administrateur	- Permettre la gestion des courriers. - Offrir la possibilité de visualiser les courriers créés en affichant l'identité de leur créateur. - Fournir une consultation des profils utilisateurs. - Permettre la gestion des utilisateurs ainsi que les tâches qui leur sont attribuées dans l'application. - Offrir une consultation du profil de l'utilisateur actif. - Permettre la gestion des directions. - Fournir un accès au tableau de bord pour consultation.
Secrétaire	- Permettre la création, la modification, la suppression et la recherche des courriers - Offrir la possibilité de consulter son profil - Permettre la consultation d'un courrier - Fournir un tableau de bord pour une vue d'ensemble de l'activité de gestion des courriers.

	- Recevoir et distribuer les courriers vers leurs destinataires au sein du service
Utilisateur Simple	- Consultation de son profil - Consultation des courriers - Suivie des traitements des courriers envoyé ou reçu

Tableau 3.1. *Tableau démontrant les besoins selon le type d'utilisateur*

3.3. Moyens nécessaires à la réalisation du projet :

3.3.1. Moyens humains

Les personnes qui ont contribué pour la bonne réalisation de ce projet sont :

- Un encadreur professionnel qui a suivi de près l'état d'avancement du projet à réaliser.
- Un stagiaire qui est à la fois le concepteur et le réalisateur du projet

3.3.2. Moyens matériels

En termes de matériel, nous avons utilisé un ordinateur portable dont les caractéristiques sont mentionnées dans le tableau ci-dessous :

Marque et modèle	Processeur	RAM	Stockage	Système d'exploitation
DELL LATITUDE E5430	Intel Core i5 2320M 2.60GHz	4Go ddr3 de fréquence 1333mhz	320 Go en disque dur	Windows 10 Edition Professionnel Version 1909

Tableau 3.2. *Spécifications de l'ordinateur portable utilisé*

3.3.3. Moyens logiciels

Pour bien parfaire ce projet, nous avons eu recours à l'utilisation de divers logiciels :

- VSCode : Pour le développement de l'application
- MySql Workbench : pour la modélisation de la base de données
- Google Chrome : pour les différentes recherche et test de l'application

3.4. Résultats attendus

Le temps impartis ainsi terminer, nous devons fournir une application :

- Rapide et efficace
- Possédant toutes les fonctionnalités requises dans la gestion de tout type de courriers
- Fiable et sécurisé
- Facile à utiliser
- Permettant de suivre le traitement des courriers qui circule dans le service

Partie II: Analyse et Conception du projet

Chapitre 4 : Méthodes et notations utilisées

4.1. Introduction :

La phase de conception est essentielle pour la réussite d'un projet informatique. Dans ce chapitre, nous allons examiner en détail les différentes conceptions qui ont été effectuées jusqu'à présent pour répondre aux besoins des utilisateurs.

4.2. Méthode et notation de développement de projet informatique

4.2.1. Définition de la méthode

La méthode est une approche systématique et structurée pour concevoir un système logiciel. Elle comprend une série d'étapes et de techniques pour résoudre les problèmes liés à la conception, la planification, la mise en œuvre, la vérification et la maintenance du logiciel.

4.2.2. Définition de la notation

La notation, quant à elle, est un langage de représentation graphique utilisé pour décrire les différents aspects du système logiciel. La notation permet de modéliser les différents composants du système, leurs interactions et leurs relations, en utilisant des symboles et des diagrammes standardisés. Les notations les plus courantes pour la conception de logiciels sont UML (Unified Modeling Language), BPMN (Business Process Model and Notation), ERD (Entity-Relationship Diagram), etc.

Pour faire court, la méthode est une approche globale de conception de logiciel qui comprend un ensemble de techniques et d'étapes, tandis que la notation est un langage graphique standardisé utilisé pour représenter les différents aspects du système logiciel. Mais notre cas, la méthode en elle-même fournis aussi les notations nécessaires à la mise en place d’un projet informatique.

Donc pour la suite de ce document nous allons prendre en compte que des méthodes car il comporte déjà en soit les notations à utiliser.

4.2.3. Type de méthode de développement

Les méthodes de conception de logiciels les plus courantes sont la SADT, la MERISE, l’UML, le RAD, agile, etc. Pour mieux comprendre ces méthodes et mieux considérer le choix le plus appropriés nous allons voir la spécificité de quelques méthodes.

4.2.3.1 Présentation de SADT (Structured Analysis and Design Technique)

La méthode SADT a été développée dans les années 1970 par Douglas T. Ross et SofTech, Inc. pour la NASA dans le cadre du projet de navette spatiale américaine. Elle a été conçue pour répondre aux besoins de modélisation et de documentation des processus complexes de la NASA et est depuis lors largement utilisée dans l'industrie pour la conception de systèmes d'information. La méthode SADT est considérée comme l'une des premières méthodes de modélisation formelle utilisées dans le domaine de l'informatique.

a. Objectif :

SADT est une méthode de conception de systèmes d'information, qui vise à modéliser et analyser les processus métier.

b. Principes de base :

- La SADT est basée sur la modélisation graphique des processus métier à l'aide de diagrammes fonctionnels.
- Elle permet d'analyser les flux de données et de contrôle entre les différentes fonctions d'un processus métier.
- Elle s'appuie sur une approche descendante et hiérarchique de la modélisation, en décomposant les processus en sous-processus de plus en plus détaillés.
- Elle utilise des symboles graphiques standardisés pour représenter les différentes fonctions et leurs interactions.

c. Les différents éléments de SADT :

La méthode SADT se compose de plusieurs éléments clés qui permettent de décrire les processus de l'entreprise. Les principaux éléments sont les suivants :

- Les diagrammes de contexte : ils permettent de visualiser les relations entre les différents processus d'une entreprise et les entités externes qui interagissent avec elle.
- Les diagrammes de flux de données : ils représentent les flux de données entre les différents processus et les entités externes.
- Les diagrammes de processus : ils permettent de décrire en détail chaque processus et les différentes étapes qui le composent.

- Les diagrammes de structure : ils permettent de décrire les structures de données qui sont utilisées dans les processus

d. Phases de la méthode :

La SADT comporte plusieurs phases, de l'analyse du système à la conception détaillée :

- Analyse du système existant : elle permet de comprendre le fonctionnement actuel du système d'information, en identifiant les processus métier, les acteurs et les flux de données.
- Définition des besoins : elle permet de recueillir les besoins et les attentes des utilisateurs en matière de fonctionnalités et de performance du système.
- Modélisation fonctionnelle : elle permet de définir les fonctions du système, en utilisant des diagrammes fonctionnels qui représentent les processus métier et les interactions entre les fonctions.
- Conception détaillée : elle permet de définir les spécifications techniques du système, en précisant les règles de gestion, les algorithmes et les structures de données.

e. Avantages :

- La SADT permet de décrire les processus métier de manière claire et précise, en identifiant les flux de données et de contrôle entre les différentes fonctions.
- Elle permet de décomposer les processus en sous-processus de plus en plus détaillés, ce qui facilite la compréhension et l'analyse du système.
- Elle permet de standardiser la représentation graphique des processus, en utilisant des symboles standardisés qui facilitent la communication entre les différents acteurs du projet.

f. Limites :

- La méthode SADT peut être lourde et complexe à mettre en œuvre, en particulier pour les systèmes d'information complexes.
- Elle peut être trop théorique et abstraite, en ne prenant pas suffisamment en compte les aspects pratiques et opérationnels du système.
- Elle peut être centrée sur la modélisation des processus métier existants, au détriment de l'innovation et de la créativité.

En résumé, la méthode SADT est une méthode de conception de systèmes d'information qui repose sur une modélisation graphique des processus métier, en utilisant des diagrammes fonctionnels standardisés. Elle permet de décrire les processus de manière claire et précise, en identifiant les flux de données et de contrôle, mais peut être complexe et théorique à mettre en œuvre. La figure suivante démontre le processus à suivre en utilisant le méthode SADT :

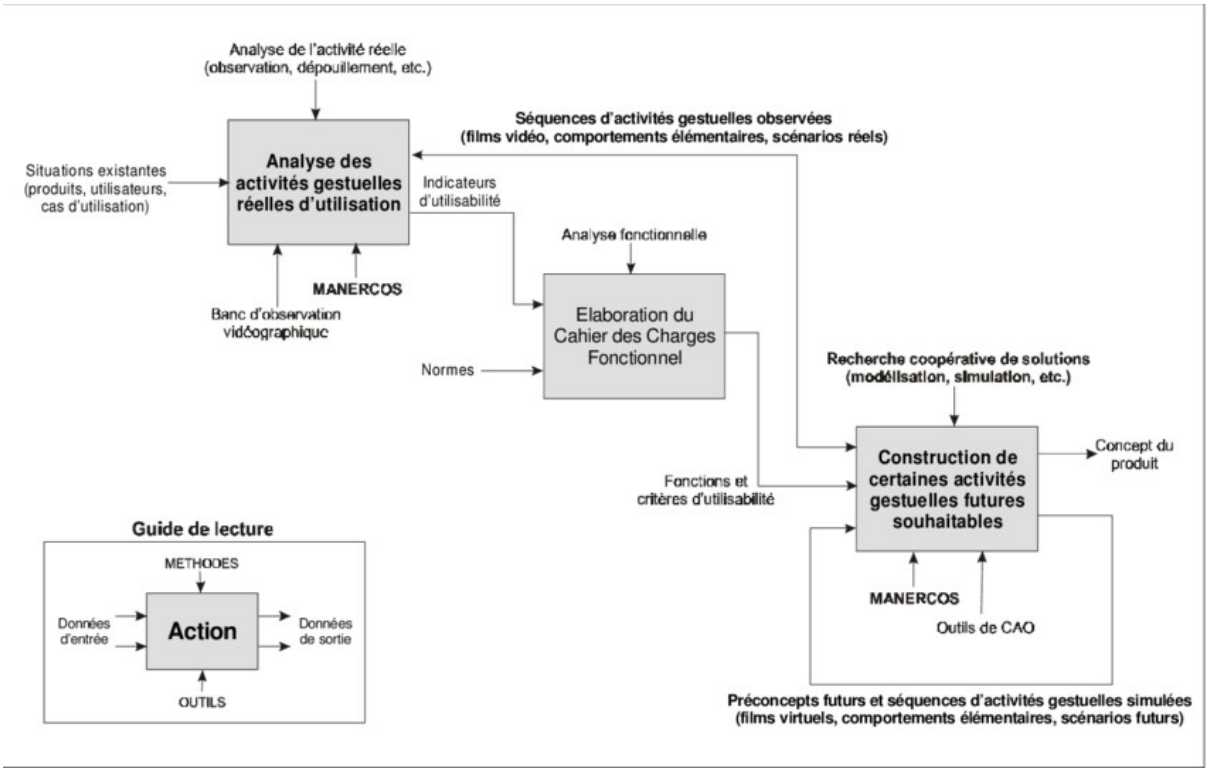

Figure 4.1. Méthode SADT

4.2.3.2 Présentation du MERISE

La méthode MERISE est une méthode de conception de systèmes d'information et elle est basée sur le modèle entité-association et utilise une approche descendante. Actuellement, la version 2 du MERISE est la version la plus récente et l'utilisé dans les projets informatiques. Pour mieux comprendre cette méthode, voici les informations importantes à connaître :

a. Origine et développement de MERISE :

La méthode MERISE a été développée par René Colletti dans les années 1970 pour répondre aux besoins de conception des systèmes d'information des entreprises. Et elle a été inspirée de la méthode américaine « Bachman, » qui utilisait également le modèle entité-association

b. Principes de la méthode :

- La méthode MERISE est basée sur le principe de la séparation des préoccupations, en séparant la description du domaine (modèle conceptuel) de la description du système informatique (modèle logique et physique).
- Elle utilise une approche descendante, en partant des besoins utilisateurs pour arriver à la définition des structures de données et des processus.
- Elle se base sur le modèle entité-association pour la modélisation conceptuelle, qui permet de représenter les entités du domaine, leurs attributs et leurs relations.
- Elle utilise le modèle relationnel pour la modélisation logique, qui permet de définir les tables et les relations entre elles.
- Elle propose une méthode de normalisation pour garantir l'intégrité des données et éviter les redondances.

c. Les étapes de la méthode :

La méthode MERISE se décompose en trois phases : l'analyse, la conception et la réalisation.

- L'analyse consiste à identifier les besoins des utilisateurs et à définir le modèle conceptuel des données.
- La conception consiste à traduire le modèle conceptuel en modèle logique et physique, et à définir les processus métier.
- La réalisation consiste à implémenter la solution informatique, en choisissant les outils et technologies les plus adaptés.

d. Avantages :

- La méthode est adaptée à la conception de systèmes d'information de grande taille.
- Elle permet de bien comprendre et modéliser les processus métier de l'entreprise.
- Elle favorise la communication entre les différents acteurs du projet.
- Elle permet de produire des modèles précis et complets.
- Elle facilite la maintenance et l'évolution du système d'information.

e. Limites :

- La méthode MERISE nécessite une forte implication des utilisateurs et une bonne connaissance de leur environnement.
- Elle peut être complexe à mettre en œuvre pour des projets de petite taille ou à faible complexité.
- Elle peut être rigide si elle est appliquée de manière trop stricte.
- Elle ne prend pas toujours en compte les aspects humains et organisationnels de l'entreprise.

La figure suivante nous montre les différentes étapes de la démarche MERISE :

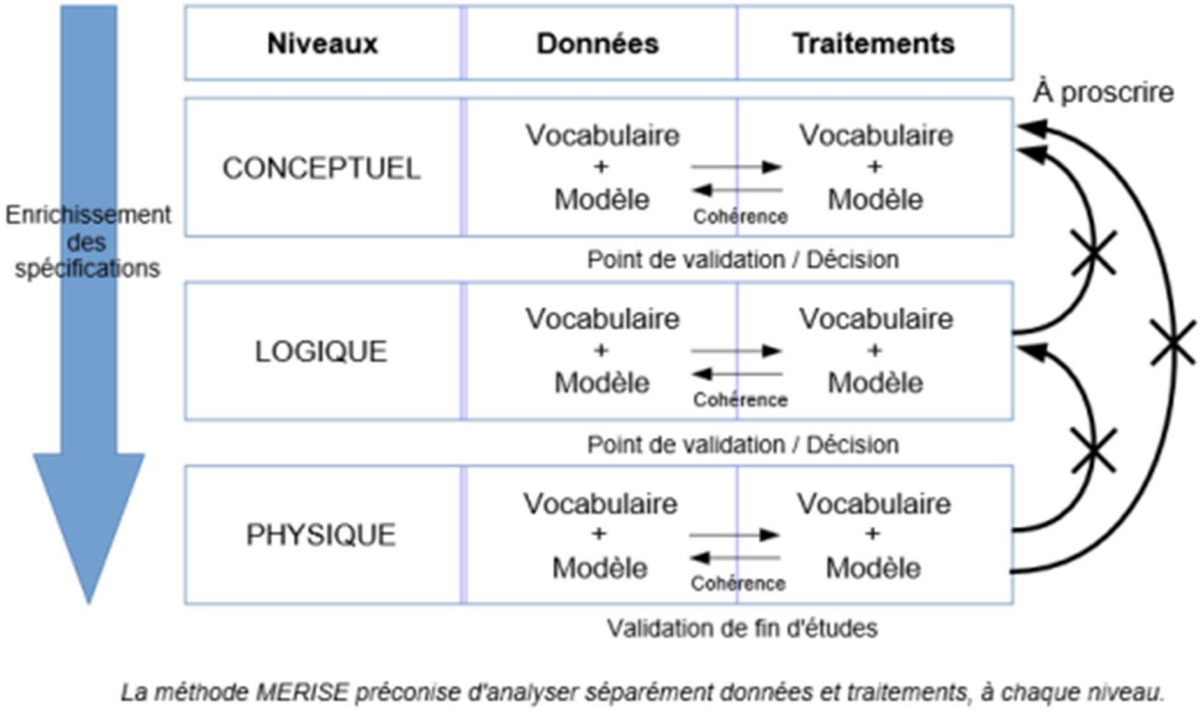

Figure 4.2. Démarche MERISE

4.2.3.3 Présentation de l'UML

UML (Unified Modeling Language) est un langage de modélisation graphique de logiciels. Elle est utilisée pour décrire, spécifier, concevoir et documenter les différents aspects d'un système informatique. UML a été développé par Grady Booch, James Rumbaugh et Ivar Jacobson en 1997. Comme son nom l'indique, UML est en effet un langage de modélisation graphique pour la spécification, la visualisation, la construction et la documentation de systèmes logiciels. Cependant, il peut être considéré comme une méthode de conception de logiciel car il fournit un cadre pour la modélisation et la documentation des différents aspects d'un système logiciel, y compris ses fonctionnalités, ses processus, ses données et ses

interactions avec d'autres systèmes. En utilisant UML, les concepteurs peuvent élaborer des modèles pour décrire les différentes perspectives du système, tels que les exigences, l'architecture, la conception détaillée, les tests et la mise en œuvre.

a. Concepts clés

Les diagrammes UML sont utilisés pour représenter différents aspects d'un système, tels que la structure, le comportement et les interactions entre les différents éléments. L'UML utilise une notation graphique pour représenter les concepts clés, tels que les classes, les objets, les relations, les états, les transitions, etc. Comme fondation il est basé sur un ensemble de concepts de modélisation, tels que l'abstraction, l'encapsulation, l'héritage, le polymorphisme, etc.

b. Les types de diagrammes UML

UML comprend plusieurs types de diagrammes qui permettent de modéliser différents aspects d'un système :

- Les diagrammes de classe : ils permettent de représenter la structure du système en utilisant des classes, des attributs, des méthodes et des relations entre les classes.
- Les diagrammes d'objets : ils permettent de représenter des instances de classes et les relations entre ces instances.
- Les diagrammes de séquence : ils permettent de modéliser les interactions entre les objets au fil du temps, en montrant les messages échangés entre les objets.
- Les diagrammes d'états-transitions : ils permettent de modéliser le comportement d'un objet ou d'un système en montrant les différents états et les transitions entre ces états.
- Les diagrammes d'activité : ils permettent de modéliser les processus métier en montrant les activités, les actions et les décisions nécessaires pour accomplir une tâche.
- Les diagrammes de déploiement : ils permettent de modéliser les différents composants matériels et logiciels nécessaires pour déployer le système.

c. Avantages

- C'est est un langage de modélisation standard qui est largement utilisé dans les industries.

- Son utilisation permet également de promouvoir les bonnes pratiques de conception de logiciels, telles que la modularité, la réutilisabilité, l'extensibilité et la maintenabilité
- Il permet d'élaborer des modèles de conception qui sont bien structurés, faciles à comprendre et à maintenir
- Il permet de communiquer efficacement les spécifications et la conception du système à toutes les parties prenantes, y compris les développeurs, les testeurs et les clients.
- Il facilite la migration d'une technologie à un autre grâce à la modélisation des classes, des objets et des relations qui est indépendante d'un quelconque langage de programmation ou technologie.
- C'est un langage flexible et peut être utilisé pour modéliser différents types de systèmes, tels que les systèmes embarqués, les applications d'entreprise, les systèmes d'information, etc.

d. Limites

- C'est un langage qui peut être complexe et difficile à maîtriser pour les débutants.
- Il peut être surutilisé, ce qui peut entraîner une documentation excessive et inutile.
- Il ne garantit pas la qualité du logiciel produit, car cela dépend de la manière dont la conception est mise en œuvre.
- Il ne prend pas en compte tous les aspects du développement logiciel, tels que la gestion de projet, la planification, le contrôle qualité, etc.

4.2.3.4 Présentation de la méthode RAD

La méthode RAD est une approche de développement rapide de logiciels qui a été introduite dans les années 1980. Elle est basée sur un processus de développement itératif et incrémental qui implique la participation active des utilisateurs finaux. L'objectif principal de cette méthode est de produire rapidement des logiciels de haute qualité tout en réduisant les coûts et les délais de développement.

a. Les étapes du processus RAD :

- Modélisation des processus métier : cette étape consiste à identifier les processus métier clés qui seront pris en charge par le logiciel.

- Conception : cette étape implique la création de modèles de données et de modèles de conception de l'interface utilisateur.
- Construction : cette étape implique la création du logiciel en utilisant des outils de développement rapides tels que des générateurs de code automatiques.
- Tests : cette étape consiste à tester le logiciel pour s'assurer qu'il répond aux exigences spécifiées.
- Déploiement : cette étape consiste à installer le logiciel sur les systèmes informatiques des utilisateurs finaux.

b. Les avantages :

- La méthode RAD permet de produire rapidement des logiciels fonctionnels.
- Les utilisateurs finaux sont impliqués tout au long du processus de développement, ce qui permet de mieux comprendre leurs besoins et d'améliorer la qualité du logiciel.
- L'utilisation d'outils de développement rapides permet de réduire les coûts et les délais de développement.

c. Les limites :

- La méthode RAD peut être difficile à mettre en œuvre pour les projets complexes qui nécessitent une grande quantité de personnalisation.
- Elle peut également entraîner des problèmes de qualité si les tests ne sont pas effectués de manière adéquate.
- Le processus RAD nécessite une forte participation des utilisateurs finaux, ce qui peut être difficile à gérer pour certains projets.

Voici une figure démontrant les étapes de conception en utilisant la méthode RAD :

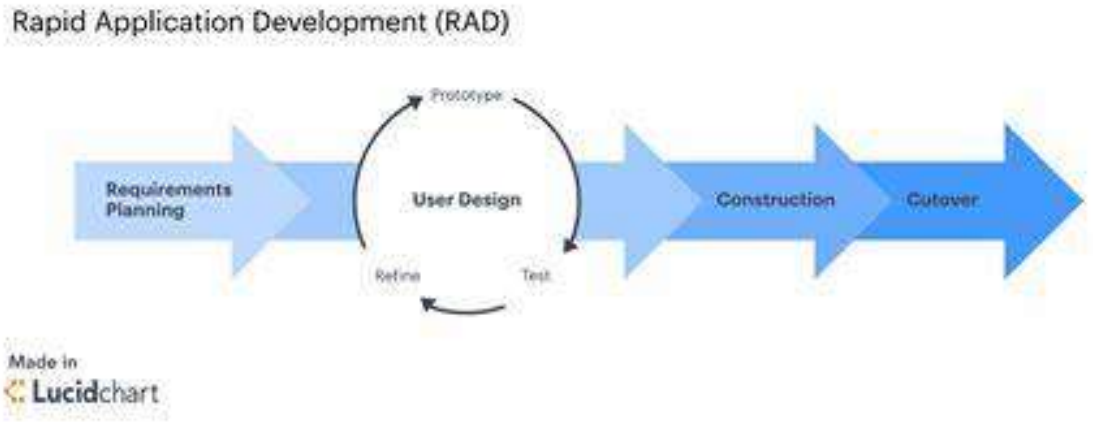

Figure 4.3. Méthode RAD

4.2.3.5 Présentation de la méthode agile

a. Description de la méthode

Il a été conçu pour être plus adaptable et flexible, avec les besoins du client au premier plan des priorités du projet. Cette méthode a été initialement développée pour des projets de développement web et informatique. Parce qu'elle peut être utilisée pour un large éventail de projets dans de nombreux secteurs, la technique Agile devient de plus en plus populaire. Des boucles de rétroaction plus courtes résultent d'un travail itératif, ce qui facilite la modification des articles en fonction de la technologie actuelle et/ou des préférences des clients. De nombreux chefs de projets informatiques et développeurs l'utilisent pour parfaire leur travail.

b. Fonctionnement de la méthode

La technique Agile est fondée sur un principe de base : planifier votre projet dans ses moindres détails avant de le développer est inefficace. Organiser chaque détail de votre projet est une perte de temps car c'est rare que tout se passe comme prévu. Les dangers se développent fréquemment, vous obligeant à ajuster vos plans. On part du principe que rien n'est gravé dans le marbre en utilisant la méthodologie Agile. C'est-à-dire que l'équipe doit pouvoir se remettre en question régulièrement et chercher à s'améliorer.

c. Principe clés de l'agilité

- La priorité est donnée à la satisfaction du client grâce à des livraisons fréquentes et régulières de logiciels fonctionnels.
- Les équipes sont autoorganisées et auto-dirigées, avec une forte communication entre les membres de l'équipe.

- Le changement est accueilli et le logiciel est conçu pour être facilement modifié afin de répondre aux besoins changeants du client.
- Les interactions face à face sont privilégiées pour faciliter la communication et la collaboration.

d. Méthodologies agiles courantes :

- Scrum : une méthode de gestion de projet agile qui implique la division du travail en sprints de courte durée (2 à 4 semaines), des réunions quotidiennes de stand-up, des revues de sprint et une rétrospective de sprint.
- Kanban : une méthode de gestion de projet agile qui utilise des tableaux pour suivre les tâches en cours, en attente et terminées. Le flux de travail est continu et le travail est terminé à mesure que les tâches sont prêtes.
- Lean : une méthode de gestion de projet agile qui se concentre sur la réduction des gaspillages et l'amélioration continue du processus de développement.
- Extreme Programming (XP) : une méthode de développement agile qui se concentre sur la qualité du logiciel grâce à des tests automatisés, une programmation en binôme et des mises en production fréquentes.

e. Avantages de l'agilité :

- Les clients sont impliqués dans le processus de développement, ce qui signifie que le produit final répond mieux à leurs besoins.
- Les équipes sont autoorganisées et motivées, ce qui peut conduire à une meilleure qualité et productivité.
- Les processus de développement sont plus flexibles et peuvent s'adapter rapidement aux changements de besoins du client.
- Les équipes travaillent en étroite collaboration, ce qui peut renforcer l'esprit d'équipe et la communication.

f. Limites de l'agilité :

- Les équipes doivent être très bien organisées et communiquer efficacement pour que la méthode fonctionne correctement.
- Les livraisons fréquentes de logiciels peuvent ne pas être appropriées pour tous les projets, par exemple pour des projets avec des contraintes de temps strictes.

- Les équipes peuvent se concentrer sur des fonctionnalités à court terme au détriment de la vision à long terme du projet.
- Le manque de documentation peut rendre le code difficile à comprendre pour les nouveaux membres de l'équipe.

La figure suivante illustre les processus à suivre en adoptant la méthode agile :

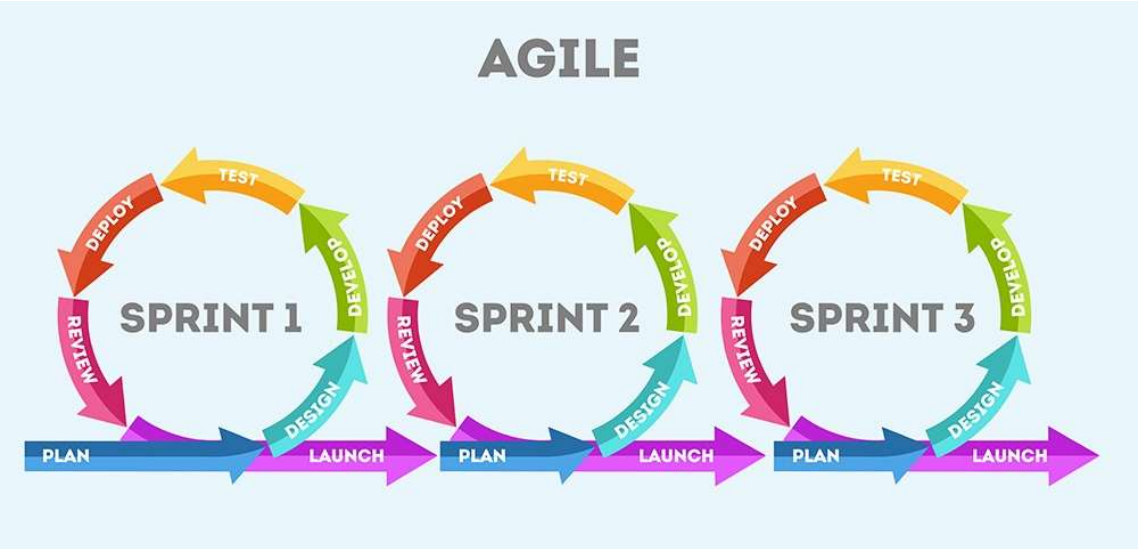

Figure 4.4. Cycle de développement selon la méthode Agile

4.2.4. Comparaison des différentes méthodes

Pour mieux considérer la méthode adéquate à notre projet, nous avons besoins de comparer les différentes méthodes citer ci-haut sur différents critères. Et pour ce faire faire nous allons voir le tableau de comparaison suivant :

Critères	SADT	MERISE	Agile	RAD
Complexité	Élevée	Moyenne	Faible	Moyenne
Flexibilité	Faible	Moyenne	Élevée	Élevée
Efficacité	Élevée	Élevée	Moyenne	Élevée
Adaptabilité	Faible	Moyenne	Élevée	Moyenne
Accessibilité	Moyenne	Élevée	Élevée	Moyenne

Tableau 4.1. *Tableau de comparaison des différentes méthodes de conception d'un projet informatique selon divers critères essentiels*

- Complexité : MERISE est plus simple que SADT et RAD, mais légèrement plus complexe qu'Agile.
- Flexibilité : MERISE est moins flexible qu'Agile, mais plus flexible que SADT et RAD.
- Efficacité : MERISE est efficace pour les projets de taille moyenne à grande, mais Agile est plus efficace pour les projets plus petits et SADT et RAD sont plus efficaces pour les projets plus grands.
- Adaptabilité : Agile est le plus adaptable, suivi de MERISE, RAD et SADT.
- Accessibilité : MERISE est facilement accessible et plus facile à apprendre que SADT et RAD, tandis qu'Agile est également facile à apprendre mais nécessite une certaine familiarité avec les principes de développement agile.

D'après les constats réaliser sur ce tableau de comparaison, il ressort que la méthode MERISE est la plus complète en termes de couverture de l'ensemble des phases de la conception d'un système de gestion de courriers. De plus, elle est la méthode que l'on maîtrise le mieux, ce qui peut nous permettre de gagner du temps en ne devant pas apprendre une nouvelle méthode. En conséquence, nous pouvons affirmer que MERISE est le meilleur choix à considérer pour la conception et la réalisation du système de gestion de courriers.

Chapitre 5 : Analyse du projet

5.1. Introduction

Au sein de la direction régionale du budget Vakinanakaratra, le traitement, l'enregistrement et l'envoi de courriers sont des tâches quotidiennes essentielles. Ces tâches sont confiées à l'agent en charge du bureau d'ordre du centre, qui y consacre une attention particulière chaque jour. En effet, le respect de ces tâches est crucial pour éviter l'accumulation de courriers en attente, qui peut rapidement devenir problématique.

5.2. Analyse de l'existant

5.2.1. Organisation actuelle

Après notre enquête sur l'organisation actuelle, il apparaît que chaque employé au sein de la direction dispose d'un ordinateur de bureau pour mener à bien ses tâches. En effet, tous les ordinateurs sont connectés à un réseau avec un domaine dédié. La direction d'ores et déjà divers application web pour parfaire leur travail et rester en synchronisation avec la direction générale du budget mais la gestion des courriers reste tout de même aux bonnes vieilles méthodes du notation sur papier dans un cahier. L'activité du bureau d'ordre s'articule autour des deux processus suivants :

- La gestion manuelle des courriers arrivés,
- La gestion manuelle des courriers sortants.
- La gestion manuelle des courriers internes

5.2.1.1 La gestion manuelle des courriers reçus

La méthode actuelle pour la gestion des courriers au bureau d'ordre est décrite par un processus qui se compose des étapes suivantes :

- Réception et enregistrement du courrier sur le registre "arrivée" du bureau d'ordre avec attribution d'un numéro d'ordre et ajout d'une fiche de circulation sur chaque courrier
- Distribution des courriers aux différentes unités administratives
- Recherche manuelle et suivi des courriers.

5.2.1.2 La gestion des manuelle des courriers émis

Le processus en place pour la gestion des courriers de départ se compose des étapes suivantes :

- Réception et enregistrement du courrier sur le registre "départ" du bureau d'ordre,
- Affranchissement des courriers,
- Recherche manuelle et suivi des courriers retournés. Ces étapes sont essentielles pour garantir la bonne gestion de tous les courriers sortants

5.2.1.3 La gestion manuelle des courriers internes

Le processus en place pour la gestion des courriers en internes est la suivante :

- Réception des courriers par le responsable et enregistrement des informations contenu dans le courrier dans le registre
- Attente de validation du contenu par les administrations supérieures
- Livraison du courrier vers l'unité correspondante

5.2.2. Inventaires des moyens matériels et logiciels

5.2.2.1 Moyens matériels

L'entreprise d'accueil dispose de différents moyens matériels qui lui permettent de réaliser ses activités quotidiennes. Ces moyens matériels sont essentiels pour le bon fonctionnement de l'entreprise et sont utilisés par les différents services de l'entreprise. Ils comprennent notamment des équipements informatiques, des machines de production, des véhicules de transport, des outils de communication, etc. Dans ce contexte, il est important d'identifier et d'analyser ces moyens matériels afin de comprendre leur impact sur le fonctionnement de l'entreprise et d'optimiser leur utilisation.

La direction régionale possède plusieurs matériels informatiques, mais dans le contexte de notre travail nous allons mentionner uniquement les matériels du service informatique. Voici donc une liste non exhaustive des matériels du service informatique :

Type de machine	Processeur	RAM	Marque	Stockage	Nombre	Système d'exploitation
Ordinateur de bureau	Intel core i5 3,3GHz x6	8Go	HP PRODESK 400 G2 MT	500Go	4	Windows 10 Professional

Serveur	2 x Intel Xeon E5-2640v3 (2,6 GHz, 8 cœurs)	16Go	HP ProLiant DL360 Gen9	4 disques durs de 600 Go (SAS, 10K RPM)	1	Windows Server 2012 R2

Tableau 5.1. *Liste des ordinateurs et serveurs au sein du service informatique*

Le service informatique dispose aussi d'une imprimante HP LaserJet PRO G1005

5.2.2.2 Moyens logiciels

Le service informatique de la direction régional du budget dispose de nombreux moyens logiciels pour assurer le bon fonctionnement des différents services. Parmi ces moyens, on peut citer l'utilisation du pack Office qui est une suite de logiciels de bureautique largement utilisée pour la création de documents, tableurs, présentations, et autres tâches administratives. Le service informatique a également accès à de nombreuses applications web fournies par le ministère des finances et du budget, qui permettent de gérer les différentes opérations financières de la région du Vakinankaratra. Ces applications web sont essentielles pour la gestion des finances et la comptabilité. Le service informatique dispose également de nombreux autres logiciels spécialisés pour la gestion de projets, la sécurité informatique, la gestion des ressources humaines, et bien d'autres domaines. Tous ces moyens logiciels sont utilisés pour garantir la sécurité, l'efficacité et la productivité de la direction régionale du budget.

5.3. Critique de l'existant

La gestion du courrier est un enjeu majeur au sein de l'administration, qui nécessite une organisation rigoureuse pour traiter les volumes importants de courriers reçus quotidiennement. Les agents du bureau d'ordre doivent trier, consulter, imprimer, archiver et distribuer les documents avec précision et rapidité. Malgré cela, les opérations manuelles peuvent causer des pertes, des oublis et des retards, ainsi que des erreurs de classement et une perte de temps. Même si le traitement manuel est toujours sujet de critique, leurs procédures et façon de faire présente quand même quelques côtés positifs :

- Faible coût initial : le traitement manuel des courriers ne nécessite pas l'achat de logiciels, de matériel informatique coûteux et ne nécessite pas de formation approfondie du personnel.
- Plus grande flexibilité : les employés peuvent s'adapter rapidement aux changements de volume de courrier ou aux besoins spécifiques du traitement du courrier.
- Interaction humaine : le traitement manuel des courriers peut favoriser les relations interpersonnelles au sein de l'organisation, car les employés peuvent interagir directement avec les destinataires et les expéditeurs.
- Bonne organisation : les employés ne se contente pas de la présence d'une solution technologique mais s'organise de façon cohérente et productive

5.4. Analyse des besoins

La première étape du développement de toute application est la spécification des besoins, durant laquelle les différentes exigences sont déterminées afin de décrire clairement le programme qui sera développé. Cette étape se compose d'un ensemble de documents et de modèles, qui doivent être accessibles à toutes les personnes impliquées dans le projet. Les besoins sont distingués en deux catégories : les besoins fonctionnels, qui présentent les fonctionnalités attendues de l'application, et les besoins non fonctionnels, qui permettent d'éviter le développement d'une application qui ne répond pas vraiment aux besoins.

5.4.1. Spécification des besoins fonctionnels

Les besoins fonctionnels d'un système définissent les actions qu'il doit effectuer en réponse à des demandes, c'est-à-dire les sorties qu'il doit produire en fonction des entrées qu'il reçoit. Pour répondre aux exigences de l'entreprise, le système proposé doit offrir les fonctionnalités suivantes :

- Envoyer et recevoir des courriers
- Consulter les courriers
- Enregistrer les courriers en stockant les informations relatives dans une base de données
- Classer les courriers
- Permettre la recherche de courriers en fonction de critères basés sur les informations d'identification du courrier (référence, type, date, etc.)
- Avoir une vue d'ensemble sur le nombre de courrier circulant dans la direction pendant une période donnée et selon différent type de courrier

5.4.2. Spécification des besoins non fonctionnels

Les exigences non fonctionnelles se réfèrent aux limitations et contraintes internes et externes du système, sans être directement liées à son comportement. Les principales exigences non fonctionnelles de l'application sont les suivantes :

5.4.2.1 Besoins de Fiabilité :

La fiabilité est un critère crucial pour garantir l'efficacité de la gestion des courriers au sein de la direction régionale du budget. Les besoins de fiabilité comprennent :

- La sécurité : Les données des courriers doivent être stockées de manière sécurisée pour éviter toute perte ou divulgation non autorisée.
- La disponibilité : L'application de gestion de courriers doit être disponible à tout moment pour que les utilisateurs puissent y accéder et traiter les courriers en temps voulu.
- La stabilité : L'application doit être stable et ne pas présenter de dysfonctionnements, de bugs ou d'erreurs qui pourraient affecter la gestion des courriers.
- La sauvegarde : Les données des courriers doivent être sauvegardées régulièrement pour éviter toute perte de données en cas de panne ou de dysfonctionnement du système.
- La traçabilité : L'application doit être en mesure de tracer les actions effectuées sur les courriers, de manière à garantir la transparence et la responsabilité de la gestion des courriers.

5.4.2.2 Besoins d’efficacité

Les besoins d'efficacité de l'application de gestion de courriers sont les suivants :

- Le traitement des courriers doit être rapide et efficace, afin de réduire les délais de traitement et de distribution des courriers. Les utilisateurs doivent pouvoir accéder rapidement et facilement aux courriers qui leur sont destinés, et les courriers doivent être distribués aux destinataires dans les délais impartis.
- L'application doit permettre d'automatiser autant que possible les tâches de traitement de courriers, telles que l'enregistrement, le tri et la distribution des courriers. Cela permettra de réduire la charge de travail manuelle pour les agents du bureau d'ordre, et d'améliorer la productivité globale de l'organisation.
- L'application doit être facile à utiliser et intuitive, afin de minimiser les temps d'apprentissage pour les utilisateurs. Les fonctionnalités doivent être clairement

identifiées et facilement accessibles, et l'interface utilisateur doit être ergonomique et conviviale.

- L'application doit être capable de gérer de gros volumes de courriers, tout en maintenant des performances optimales. Les temps de réponse doivent être rapides, même lors de l'utilisation simultanée par de nombreux utilisateurs.
- L'application doit offrir des fonctionnalités de suivi et de « reporting », permettant aux utilisateurs de suivre l'état d'avancement des courriers, d'identifier les goulots d'étranglement et d'optimiser les processus de traitement de courriers. Cela permettra d'améliorer l'efficacité globale de l'organisation et de réduire les temps de traitement des courriers.

5.5. Identification des acteurs

Dans le cadre de notre projet de gestion de courriers au sein de la direction régionale du budget, tous les employés de différents services sont susceptibles d'utiliser l'application. Par conséquent, il est important d'identifier les acteurs clés qui interagiront avec le système. Ces acteurs incluent :

- Les agents du bureau d'ordre, qui seront chargés de recevoir, enregistrer et distribuer les courriers.
- Les chefs de service, qui seront responsables de la validation et de l'approbation des courriers.
- Les agents de secrétariat, qui seront chargés de saisir les informations relatives aux courriers dans l'application.
- Les utilisateurs finaux, qui consulteront les courriers à partir de l'application.

En identifiant ces acteurs clés et leur rôle dans le processus de gestion de courriers, nous pourrons concevoir un système qui répond aux besoins de chacun et facilite la collaboration entre les différents services.

5.6. Proposition de solution

À la lumière de l'analyse effectuée dans la section précédente concernant la gestion des courriers, nous allons proposer des solutions informatiques.

Après avoir étudié l'organisation en place, présentée dans les sections précédentes, nous sommes maintenant en mesure de proposer des solutions pour répondre aux besoins des utilisateurs et améliorer les points faibles identifiés.

La solution est le résultat d'une réflexion visant à résoudre un problème ou à surmonter une difficulté. Nous allons donc proposer des solutions potentielles pour les différentes tâches en examinant soigneusement leurs avantages et leurs inconvénients.

Pour résoudre le problème de gestion de courrier au sein de la direction régional, nous avons proposé trois (03) solutions qui sont les suivantes :

- Achat d'une application de gestion moderne payant
- Utilisation d'une application moderne de gestion libre de droit
- Création d'une application avec l'aide d'un stagiare

Dans le but de trouver la solution la plus appropriés nous allons comparer ces différentes solutions dans le tableau suivante :

Critère	Achat d'une application	Utilisation d'une application libre de droit	Création d'une application personnalisée
Coût initial	Élevé	Faible	Moyen
Coût de maintenance	Élevé	Faible	Moyen
Personnalisation	Limitée	Limitée	Élevée
Sécurité des données	Gérée par le fournisseur	Risquée	Gérée en interne
Intégration avec les systèmes existants	À vérifier	À vérifier	À priori facile
Formation des utilisateurs	Coûteuse	Facile	Facile
Délai de mise en place	Rapide	Rapide	Long

Tableau 5.2. *Comparaison des solutions proposés*

5.7. Solution retenue

Après avoir comparé les différentes solutions informatiques possibles pour la gestion des courriers au sein de la direction régionale du budget, il a été conclu qu'il serait préférable de créer une nouvelle application de gestion de courrier.

En effet, l'achat d'une nouvelle application moderne est difficile à élaborer dans le cadre d'un service public, car il doit être soumis à des étapes rigoureuses, ce qui peut entraîner des retards

considérables. D'autre part, l'utilisation d'une application libre de droit est risquée pour les données d'un État, car il n'y a aucune garantie de la qualité et de la sécurité de l'application.

La création d'une nouvelle application de gestion de courrier, avec l'entraide du service informatique, semble être la meilleure option, car elle permettra de répondre aux besoins spécifiques de la direction régionale du budget. Cette solution offrira également une plus grande flexibilité en termes de personnalisation et d'adaptation aux besoins futurs de l'organisation.

Bien que la création d'une nouvelle application nécessite des ressources considérables en termes de temps et de coûts, il est important de considérer cela comme un investissement à long terme qui permettra d'améliorer considérablement l'efficacité et la fiabilité de la gestion des courriers au sein de la direction régionale du budget.

Chapitre 6 : Conception du projet

6.1. Introduction

Dans le cadre de notre projet de gestion de courriers, l'étape conceptuelle correspond à la phase initiale de la conception et de la modélisation du système. Cette étape nous permet de définir les règles de gestion relatives à notre domaine d'activité. Cela nous permet également de mieux comprendre les relations entre les données et leur utilisation, en prenant en compte les contraintes liées à notre contexte d'utilisation.

Dans ce chapitre, nous allons aborder l'étape de la conception du projet de gestion de courriers au sein de la direction régionale du budget. Pour cela, nous allons décrire l'ensemble des données nécessaires à la mise en place de l'application et présenter les différents modèles conceptuels qui vont nous permettre de modéliser le domaine de gestion. Nous allons ainsi pouvoir définir les différentes entités, les relations entre ces entités, et les règles de gestion associées. Cette étape est cruciale pour la mise en place d'une application performante et efficace, et permettra de faciliter la manipulation et la gestion des courriers au sein de la direction.

6.2. Identification des acteurs

Dans notre application de gestion de courriers, il y aura deux types d'utilisateurs :

- L'administrateur
- L'utilisateur simple.

L'administrateur sera responsable de la configuration et de la maintenance du système, ainsi que de la gestion des comptes utilisateurs. Il aura également accès à toutes les fonctionnalités de l'application, y compris la création, la consultation et la modification des courriers.

L'utilisateur simple, quant à lui, sera en mesure de consulter les courriers et de soumettre des demandes de création ou de modification des courriers, mais il n'aura pas accès aux fonctions de configuration et de maintenance du système.

Un utilisateur est une personne qui utilise l'application pour accomplir une tâche ou accéder à des informations spécifiques. Dans le contexte de notre application de gestion de courriers, les utilisateurs peuvent être des employés de la direction régionale du budget, tels que des chefs de service ou des agents chargés de la gestion des courriers entrants et sortants.

1.1. Dictionnaire de données

Dans le cadre de notre projet, la conception d'une base de données est primordiale pour assurer la traçabilité et la sécurité des informations relatives aux courriers. Ainsi, nous allons présenter dans cette section la structure de la base de données ainsi que les différentes tables qui la composent.

Entité	Propriété	Description	Type	Taille	Observation
User	Id_user	Identification de l'utilisateur	N	3	
	Last_name	Nom de l'utilisateur	AN	30	
	First_name	Prénom de l'utilisateur	AN	30	
	Employee_number	Numéro de matricule de l'utilisateur	AN	15	
	User_access	Niveau d'accès de l'utilisateur	N	1	
	Username	Pseudonyme de l'utilisateur	AN	10	
	Password	Mot de passe de l'utilisateur	AN	20	
	User_Id_function	Fonction de l'utilisateur dans son département	N	3	
Depatment	Id_department	Identifiant du département	N	2	
	Label_department	Désignation du département	N	30	
Function	Id_function	Identifiant de l'enseignant	N	2	
	Label_function	Libellée de la fonction	N	12	

	Function_id_department	Identification du département dans lequel appartient la fonction	AN	2	
Mail	Id_mail	Identification du courriel	N	11	
	Mail_object	Objet du courriel	AN	60	
	Mail_origin	Origine du courriel	AN	10	
	Mail_pre_ref	Pré-référence du courriel	AN	15	
	Mail_Date	Date du courriel	D	10	dd/mm/yyyy
	Mail_id_direction	Numéro de direction à qui appartient le document	N	2	
	Id_outgoing_mail	Identification du courriel sortant	N	11	
	Id_incoming_mail	Identification du courriel entrant	N	11	
	Mail_id_user	Identification du compte propriétaire du courriel	N	3	
	Mail_ref	Référence du courriel	AN	15	
	Mail_classification	Le type du courriel	AN	20	
	Mail_arrival_date	Date d'arrivée du courriel	D	10	dd/mm/yyyy
	Mail_recipient	Destinataire du courriel	AN	128	
	Mail_processing_date	Date de traitement du courriel	D	10	dd/mm/yyyy
	Mail_status	Statut ajouté du courriel	AN	10	
	Mail_location	Numéro de référence à la salle, étagère et section de rangement de la version physique du courriel	AN	60	
	Mail_created_at	Date de création du courriel			
	Mail_created_by	Identification du compte qui a créé le courriel			

	Mail_updated_at	Date de modification des contenue du courriel			
	Mail_updated_by	L'identification du compte qui à effectuer une mise a jour sur le courriel			

Tableau 6.1. *Description des données utilisées*

AN : Alphanumérique N : Numérique D : Date

6.3. Règles de gestion

Les règles de gestion définissent les principes et les pratiques qui guident la manière dont les données sont collectées, stockées, traitées et utilisées dans le cadre de notre projet. Elles sont un élément clé pour assurer la qualité, la cohérence et la fiabilité des informations que nous manipulons. En établissant des règles de gestion claires et précises, nous pouvons garantir la pertinence de nos analyses, la prise de décision éclairée, ainsi que la conformité aux réglementations en vigueur. Voici donc les règle de gestion résultant des divers analyses et enquêtes au sein de la direction :

RG1: Les utilisateurs doivent être authentifiés et autorisés avant d'accéder aux données et fonctionnalités sensibles.

RG2: Les erreurs de saisie de données doivent être détectées et signalées à l'utilisateur.

RG3: Les données doivent être validées pour s'assurer qu'elles sont conformes aux spécifications et exigences de l'application.

RG4: Les enregistrements de données doivent être conservés de manière à garantir leur disponibilité et leur intégrité.

RG5: Les mises à jour des données doivent être enregistrées et horodatées pour garantir la traçabilité.

RG6: Aucun courriel ne doit être supprimer du système d'information

RG7: Les données et rapports doivent être accessibles en temps opportun pour permettre une prise de décision éclairée.

RG8: Les données doivent être présentées de manière claire, concise et compréhensible pour les utilisateurs finaux.

RG9: Les données doivent être stockées et sauvegardées conformément aux politiques de sécurité de l'information en vigueur.

RG10: Une personne à une fonction au sein de la direction régionale du budget

RG11: Un département propose différentes fonctions

RG12: Un courrier appartient soit à une personne soit à un département

RG13: Un courrier possède un type pour mieux le retrouver

RG14: Un compte peut enregistrer et modifier les courriels

RG15: Il y a deux types d'utilisateurs : les administrateurs et les utilisateurs simples.

RG16: Il y a deux types d'utilisateurs : les administrateurs et les utilisateurs simples.

RG17: L'emplacement de stockage est constitué d'une suite de chiffres et de lettres qui permettent de retrouver la version physique du courriel.

RG18: Les utilisateurs ne sont autorisés à accéder qu'aux courriels qui leur sont destinés ou qui leur ont été partagés.

6.4. Modélisation des données et des traitements

6.4.1. Modèle conceptuel de données

Le MCD (Modèle Conceptuel de Données) est une représentation des données d'un domaine sans tenir compte des aspects techniques et économiques de stockage et d'accès, ni des conditions d'utilisation. Son objectif est de formaliser les données qui seront utilisées dans le système d'information. Avant de construire ce modèle, il est important de faire un inventaire des données pour éliminer les redondances, synonymes et homonymies. Les éléments soulignés représentent les clés de chaque entité.

6.4.1.1 Le MCD de notre projet

Sur la figure suivante, est le MCD de notre projet :

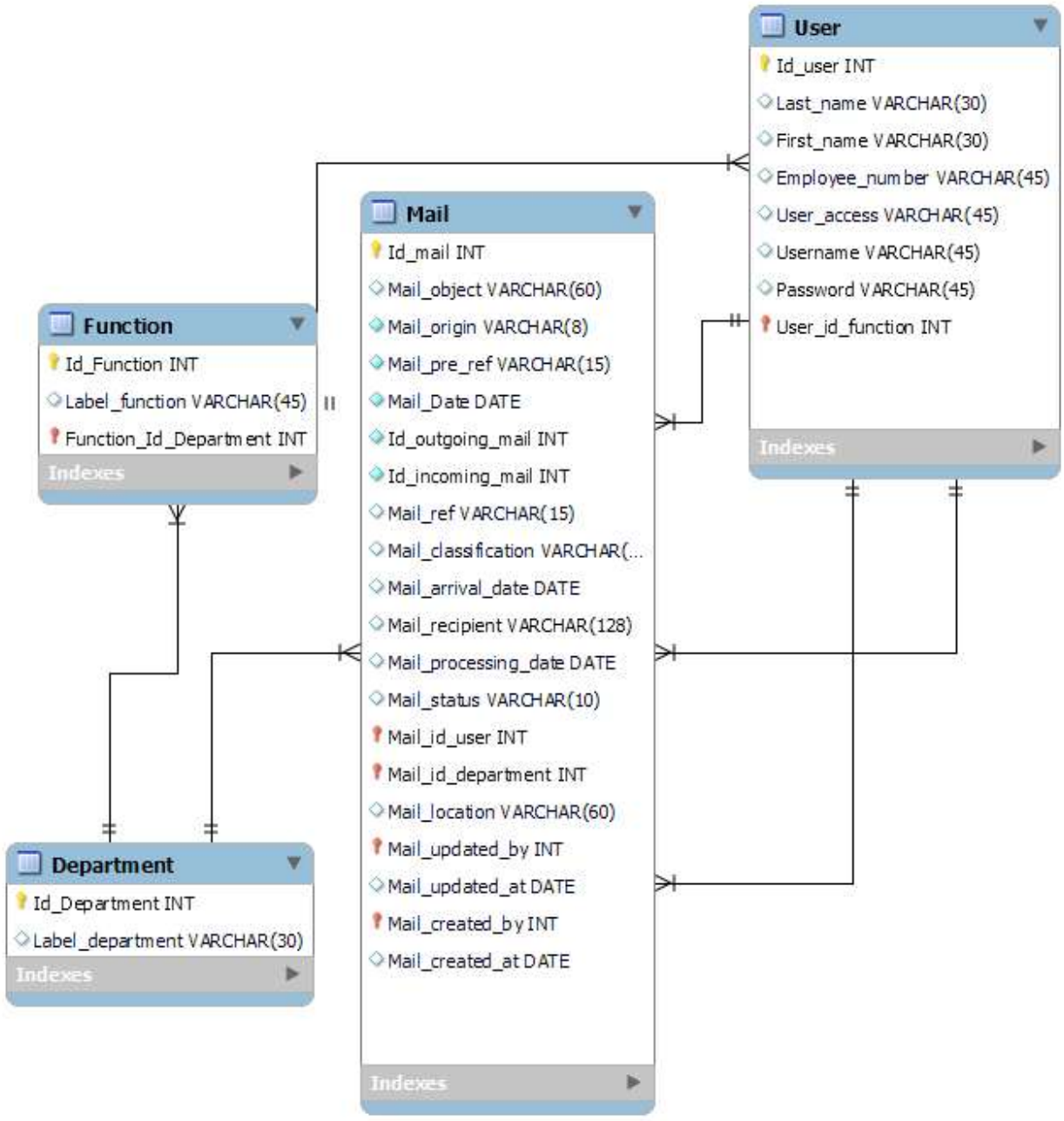

Figure 6.1. Modèle conceptuel de donnée de notre projet

6.4.2. Modèle logique de donnée

Le modèle logique de données MLD fournit une description des données tenant compte des moyens informatiques mis en œuvre. Il indique donc comment les données seront organisées.

Tandis que le modèle physique des données décrit la base des données où l'ensemble des fichiers correspond aux données gérées par le système d'information. Le MPD prépare le système de gestion des données. Les éléments soulignés représentent les clés primaires de chaque table. Les éléments devancer de # sont les clés étrangères.

6.4.2.1 Normalisation

L'objectif principal de la normalisation est d'empêcher les anomalies transactionnelles pouvant résulter d'une modélisation de données inexacte, ce qui permet d'éviter un certain nombre de problèmes potentiels tels que la redondance des données, les anomalies de lecture/écriture et les problèmes de performances.

a. Première forme normale :

- Une entité du MCD devient une relation, c'est à dire une table. Dans un SGBD de type relationnel, une table est structure tabulaire dont chaque ligne correspond aux données d'un objet enregistré (d'où le terme enregistrement) et où chaque colonne correspond à une propriété de cet objet. Une table contiendra donc un ensemble d'enregistrements. Une ligne correspond à un enregistrement. Une colonne correspond à un champ. La valeur prise par un champ pour un enregistrement donné est située à l'intersection ligne-colonne correspondant à enregistrement-champ. Il n'y a pas de limite théorique au nombre d'enregistrements que peut contenir une table. Par contre, la limite est liée à l'espace de stockage.
- Son identifiant devient la clé primaire de la relation. La clé primaire permet d'identifier de façon unique un enregistrement dans la table. Les valeurs de la clé primaire sont donc uniques. Les valeurs de la clé primaire sont obligatoirement non nulles. Dans la plupart des SGBDR, le fait de définir une clé primaire donne lieu automatiquement à la création d'un index.
- Les autres propriétés deviennent les attributs de la relation

b. Deuxième forme normale :

Une association de type 1 : N (c'est à dire qui a les cardinalités maximales positionnées à « 1 » d'un côté de l'association et à « n » de l'autre côté) se traduit par la création d'une clé étrangère dans la relation correspondante à l'entité côté « 1 ». Cette clé étrangère référence la clé primaire de la relation correspondant à l'autre entité.

c. Troisième forme normale

Une association de type N, N (c'est à dire qui a les cardinalités maximales positionnées à « N » des 2 côtés de l'association) se traduit par la création d'une relation dont la clé primaire est composée des clés étrangères référençant les relations correspondant aux entités liées par l'association. Les éventuelles propriétés de l'association deviennent des attributs de la relation.

6.4.2.2 MLD du projet

- **User** (Id_user, Last_name, First_name, Employee_number, User_access, Username, Password, #User_Id_function)
- **Mail (**Id_mail, Mail_object, Mail_origin, Mail_pre_ref, Mail_Date, #Mail_id_direction, Id_outgoing_mail, Id_incoming_mail, #Mail_Id_user, Mail_ref, Mail_classification, Mail_arrival_date, Mail_recipient, Mail_processing_date, Mail_status, Mail_location, Mail_created_at, #Mail_created_by, Mail_updated_at, #Mail_updated_by)
- **Department** (Id_department, Label_department)
- **Function** (Id_function, Label_function, #Function_id_department)

6.5. Conclusion

Dans ce chapitre, nous avons présenté une vision conceptuelle de la solution proposée en fonction des besoins de l'utilisateur et des améliorations suggérées, en utilisant MERISE. Cette étape nous permet de garantir le bon fonctionnement de notre application et nous aide dans la phase de développement de notre application de bureau.

Nous allons maintenant passer au dernier chapitre consacré à la réalisation, dans lequel nous décrirons l'environnement de travail de notre projet, ainsi que certaines interfaces représentant les fonctionnalités de notre application.

Partie III: Réalisation du projet

Chapitre 7 : Spécification des outils de réalisation

7.1. Introduction

Une fois l'étude et la conception de l'application terminées, il est temps de passer à l'étape cruciale de l'implémentation. Dans ce chapitre, nous allons explorer en profondeur le fruit de nos efforts déployés tout au long du projet. Nous allons également passer en revue l'environnement matériel ainsi que les outils de développement que nous avons utilisés pour la création de notre application.

7.2. Environnement de travail

7.2.1. Environnement matériel

Le matériel utilisé pour la réalisation de ce projet est composé d'un ordinateur portable dont la configuration est la suivante :

- Marque DELL et modèle LATTITUDE E5430
- Processeur Intel core i5 2320M 2.60Ghz
- RAM DDR3 4Go de fréquence
- Un stockage de 320Go

7.2.2. Environnement logiciel

L'environnement logiciel se réfère à l'ensemble des outils et des programmes utilisés pour le développement, la conception et la mise en œuvre d'une application. Cela comprend les environnements de développement intégrés (IDE) tels que Visual Studio, Eclipse, NetBeans, le pack JetBrains ainsi que des éditeurs de texte tels que Sublime text, Notepad++ et Visual Studio Code. Les IDE fournissent des fonctionnalités avancées telles que le débogage, la complétion de code, la gestion de versions et la construction d'applications, tandis que les éditeurs de texte offrent des fonctionnalités de base telles que la coloration syntaxique et la mise en forme de texte. Le choix de l'environnement logiciel dépend des besoins spécifiques du projet, des compétences de l'équipe de développement et de la plate-forme de déploiement. Un environnement logiciel bien adapté peut améliorer l'efficacité et la qualité du processus de développement. Pour utiliser le logiciel le plus appropriés à nos besoins et compétences, nous allons comparer quelques IDE et éditeur de texte dans le tableau suivant :

Fonctionnalité	VSCode	Pack Jetbrains	Eclipse	NetBeans	Notepad++	Sublime Text
Prise en main	Facile	Difficile	Facile	Facile	Moyenne	Facile
Personnalisation	Élevée	Élevée	Moyenne	Faible	Élevée	Faible
Intégration Git	Oui	Oui	Oui	Oui	Oui	Non
Débogage	Oui	Oui	Oui	Oui	Oui	Non
Extension et Plugin	Élevée	Élevée	Élevée	Moyenne	Moyenne	Faible
Support multi-langage	Oui	Oui	Oui	Oui	Oui	Non
Performance	Rapide	Rapide	Moyenne	Moyenne	Rapide	Rapide
Coût	Gratuit	Payant	Gratuit	Gratuit	Payant	Gratuit

Tableau 7.1. *Comparaison des différentes IDE et éditeur de texte*

D'après la comparaison effectuée entre les différents environnements logiciels, il ressort que Visual Studio Code (VSCode) présente plusieurs avantages par rapport aux autres outils étudiés. En effet, en plus d'être gratuit, VSCode est très performant et léger, ce qui le rend moins gourmand en ressources. De plus, il offre une grande flexibilité grâce à sa capacité d'extension, permettant ainsi de personnaliser l'interface en fonction des besoins de l'utilisateur.

Bien que les IDE présentent des fonctionnalités plus avancées, leur prise en main est souvent plus difficile et certaines fonctionnalités peuvent même ralentir le travail. En revanche, VSCode offre une interface simple et intuitive qui permet une prise en main rapide, tout en proposant un large éventail de fonctionnalités.

C'est pourquoi, dans le cadre de notre projet, nous avons décidé d'utiliser Visual Studio Code comme environnement logiciel. Sa légèreté, sa flexibilité et sa simplicité d'utilisation en font un outil idéal pour développer efficacement notre application.

7.3. Outils de développement

Pour arriver au terme de notre projet, nous avons besoins d’utiliser plusieurs outils qui sont catégoriser comme suit :

- Langage de programmation
- Frameworks
- Les outils de gestion de version de projet
- Les gestionnaires de paquet

- Les Système de gestion de base de données (ou SGBD)
- Outil de modélisation

Pour mieux faire notre choix d'outil à utiliser nous allons les voir un à un.

7.3.1. Le langage de programmation

Un langage de programmation est une notation conventionnelle utilisée pour écrire des programmes informatiques, qui sont des ensembles d'instructions exécutables par un ordinateur. Les programmes peuvent être écrits dans différents langages de programmation en fonction des besoins.

Voici quelques exemples de langages de programmation populaires :

- Python
- Java
- C++
- JavaScript
- PHP
- Ruby
- Swift
- Kotlin

Chacun de ces langages ont ses propres caractéristiques et est adapté à des types de projets spécifiques. Pour nous décide du langage de programmation a adopté, nous allons voir la comparaison de ces derniers dans le tableau suivant :

Langage de programmation	Frontend	Backend	Popularité	Avantages	Inconvénients
PHP	Oui	Oui	Très populaire	Bonne performance pour les sites web, grande communauté de développeurs, supporte de nombreux systèmes de	Code peu structuré, vulnérable aux failles de sécurité, difficulté à maintenir les grands projets

				gestion de contenu (CMS) tels que WordPress, Drupal, etc.	
JavaScript	Oui	Oui	Très populaire	Utilisation du même langage pour le frontend et le backend avec Node.js, grande variété de frameworks tels que React, Angular, Vue, etc.	Difficulté de débogage et de test, problèmes de performance pour les applications complexes
C++	Oui	Oui	Populaire	Haute performance, faible consommation de ressources, utilisé pour les systèmes embarqués et les jeux vidéo	Syntaxe complexe, nécessite une attention particulière pour éviter les erreurs, manque de bibliothèques pour les applications web
Swift	Oui	Oui	Populaire pour iOS	Syntaxe simplifiée par rapport à Objective-C, performance rapide	Limité à l'écosystème Apple, peu de bibliothèques tierces pour le développement web

Kotlin	Oui	Oui	Populaire pour Android	Compatibilité avec Java, moins de verbeux que Java, support pour les applications Android et les applications backend avec Spring	Communauté de développement plus petite que Java, documentation parfois insuffisante
Java	Oui	Oui	Populaire	Grande communauté de développeurs, utilisé pour les applications backend avec Spring, compatible avec Android	Syntaxe verbeuse, parfois lent pour les applications web, documentation parfois insuffisante
Ruby	Non	Oui	Moins populaire	Langage de script orienté objet, principalement utilisé avec le framework Ruby on Rails pour le développement web	Une communauté de petite taille et un langage seulement utilisé sur les grands projets et seulement si le client le demande

Tableau 7.2. *Comparaison des différents langages de programmation*

En fonction des besoins et des compétences de l'équipe de développement, PHP et JavaScript sont des choix populaires pour le développement web, car ils sont tous les deux largement utilisés et offrent une grande variété de frameworks et de bibliothèques pour faciliter le développement.

7.3.2. Les frameworks

Un framework est un ensemble d'outils, de bibliothèques et de conventions de codage qui fournissent un cadre de développement pour les applications. Il permet aux développeurs de se concentrer sur la logique de l'application sans avoir à se soucier des aspects plus techniques de la mise en place d'une architecture solide. Les frameworks permettent également d'accélérer le processus de développement en offrant des fonctionnalités prêtes à l'emploi, telles que des modèles de conception, des outils de validation de formulaire et des outils de gestion de bases de données.

L'utilisation d'un framework présente plusieurs avantages. Tout d'abord, cela peut accélérer le développement de l'application en fournissant des éléments clés tels que l'architecture et les bibliothèques nécessaires. De plus, les frameworks permettent de réduire les erreurs de code en fournissant des solutions testées et validées. Enfin, ils peuvent simplifier la maintenance de l'application en offrant des mises à jour régulières et en fournissant des outils de débogage avancés.

Cependant, l'utilisation d'un framework peut également présenter certains inconvénients. Tout d'abord, il peut y avoir une courbe d'apprentissage importante pour maîtriser le framework, ce qui peut ralentir le développement initial de l'application. En outre, certains frameworks peuvent avoir des limites en termes de personnalisation et de flexibilité, ce qui peut rendre plus difficile l'adaptation de l'application à des besoins spécifiques.

Dans notre projet de conception et réalisation d'un système de gestion de courrier, qui est limité en termes de temps et de ressources, l'utilisation d'un framework peut s'avérer bénéfique pour accélérer le développement et garantir une architecture solide. En effet, l'utilisation d'un framework peut fournir des fonctionnalités clés telles que la gestion de bases de données et l'interface utilisateur, ainsi que des outils de débogage pour faciliter la maintenance. Cela permettra également de minimiser les erreurs de code et d'optimiser le temps et les ressources nécessaires pour le développement de l'application.

7.3.3. Comparaison des différents frameworks

Etant donné que nous allons utiliser les langages de programmation PHP et Javascript, nous allons faire la comparaison entre les différentes framework disponible pour ces deux langages pour mieux faire notre choix.

7.3.3.1 Framework PHP

Framework	Facilité d'utilisation	Complétude	Mise à jour	Popularité
Symfony	Élevée	Élevée	Élevée	Très élevée
Laravel	Élevée	Élevée	Élevée	Très élevée
CodeIgniter	Moyenne	Moyenne	Moyenne	Moyenne
CakePHP	Moyenne	Élevée	Moyenne	Moyenne
Zend Framework	Faible	Élevée	Élevée	Faible
Yii	Moyenne	Élevée	Élevée	Faible

Tableau 7.3. *Comparaison des différents frameworks PHP*

Comme on peut le voir, Symfony se distingue en étant facile à utiliser, complet, régulièrement mis à jour et très populaire. Il est également soutenu par une grande communauté de développeurs, ce qui permet de bénéficier d'un support technique important. Dans le contexte de notre projet, qui est limité en termes de temps et de ressources, l'utilisation d'un framework tel que Symfony peut être très bénéfique car il permet d'accélérer considérablement le développement et de réduire les risques d'erreurs, tout en garantissant une haute qualité de code et une grande stabilité. La version de Symfony que nous allons utiliser est la version 4.

7.3.3.2 Framework Javascript

Framework	Description	Popularité	Avantages	Inconvénients
React	Framework open-source pour la création d'interfaces utilisateur, basé sur le principe de composant développée par Facebook.	Très populaire, utilisé par de nombreuses entreprises de renom, telles que Netflix, Airbnb, Instagram, etc.	- Utilisation d'une approche déclarative pour construire des interfaces utilisateur, ce qui rend le code plus facile à comprendre et à maintenir.	- Il n'est pas un framework complet, il faut donc utiliser des bibliothèques tierces pour des fonctionnalités supplémentaires. - La courbe d'apprentissage peut être raide pour les débutants, en particulier pour

				ceux qui ne sont pas familiers avec l'approche déclarative.
Angular	Framework open-source développé par Google pour la création d'applications web et mobiles.	Populaire, utilisé par de nombreuses entreprises de renom, telles que Google, IBM, Microsoft, etc.	- Utilisation d'une approche basée sur des composants pour la construction d'interfaces utilisateur. - Possibilité de générer du code automatiquement avec Angular CLI (Command Line Interface).	- Courbe d'apprentissage abrupte pour les débutants. - Une mise à jour majeure peut nécessiter des changements importants dans le code existant.
Vue.js	Framework open-source pour la création d'interfaces utilisateur et d'applications web, basé sur le modèle MVVM.	De plus en plus populaire, utilisé par des entreprises de renom telles que Xiaomi, Alibaba, GitLab, etc.	- Facilité d'utilisation et courbe d'apprentissage relativement douce. - Flexibilité pour utiliser des approches basées sur des composants ou des approches plus conventionnelles.	- La documentation peut être moins exhaustive que celle d'autres frameworks. - L'écosystème de la bibliothèque tierce est moins développé que celui de React et Angular.
Ember.js	Framework open-source pour le développement d'applications web.	Utilisé par un certain nombre d'entreprises de renom, telles que Netflix, Microsoft, LinkedIn, etc.	- Prise en charge d'une large gamme de fonctionnalités pour les applications web, y compris l'architecture basée sur des composants. - Courbe d'apprentissage	- Le framework peut être trop complexe pour des applications plus simples. - La documentation n'est pas aussi exhaustive que celle d'autres frameworks.

			relativement douce.	
Backbone.js	Bibliothèque JavaScript open-source pour la création d'applications web.	Utilisé par un certain nombre d'entreprises de renom, telles que Walmart, Airbnb, etc.	- Facilité d'utilisation et courbe d'apprentissage relativement douce. - Utilisation d	

Tableau 7.4. *Comparaison des différents framework javascript*

Après avoir comparé les différents frameworks JavaScript en fonction des critères importants tels que la popularité, la simplicité d'utilisation, la mise à jour régulière et la complétude, il est clair que React est la meilleure option pour notre projet de développement d'application web.

7.3.4. Les outils de gestion de version de projet

Un outil de gestion de version de logiciel (ou VCS, Version Control System en anglais) est un logiciel permettant de gérer les différentes versions d'un projet logiciel. Il permet aux développeurs de travailler en collaboration sur un même projet, de garder un historique des modifications apportées au code source, de revenir en arrière si nécessaire, et de travailler sur des fonctionnalités en parallèle sans affecter le travail des autres membres de l'équipe. Les VCS sont utilisés pour une grande variété de projets, allant des projets open-source aux projets commerciaux.

Il existe plusieurs VCS, chacun ayant ses propres fonctionnalités et avantages. Les plus populaires sont :

- GIT
- SVN (Subversion)
- Mercurial
- Perforce

7.3.4.1 GIT

Créé par Linus Torvalds en 2005, Git est aujourd'hui l'outil de gestion de version le plus utilisé. Il est distribué et décentralisé, ce qui signifie que chaque développeur possède une copie complète de l'historique du projet sur son ordinateur, et peut travailler de manière autonome sans avoir besoin d'un serveur centralisé.

7.3.4.2 SubVersion (SVN)

SVN est un outil de gestion de version centralisé, ce qui signifie qu'il nécessite un serveur centralisé pour stocker l'historique du projet. Il a été populaire avant l'arrivée de Git et reste encore utilisé dans certains projets.

7.3.4.3 Mercurial

Créé en 2005, Mercurial est un outil de gestion de version distribué similaire à Git. Il est moins populaire que Git, mais reste utilisé dans certains projets.

7.3.4.4 Perforce

Un outil de gestion de version commercial utilisé principalement pour les projets de grande envergure. Il est centralisé et est souvent utilisé dans l'industrie des jeux vidéo.

7.3.4.5 Choix de la gestion de version

Chaque outil a ses avantages et ses inconvénients, et le choix d'un outil dépendra des besoins spécifiques du projet. Cependant, il est fortement recommandé d'utiliser un outil de gestion de version dans tout projet de développement de logiciel pour assurer une collaboration efficace et un suivi de l'historique des modifications. Pour mieux choisir le système de gestion de version, nous allons voir le tableau de comparaison suivante :

Système de gestion de versions	Avantages	Inconvénients
SVN	- Bien adapté aux projets de taille moyenne - Gratuit	- Ne supporte pas le renommage et le déplacement de fichiers - Ne gère pas très bien les branches et les fusions
Git	- Très flexible et performant - Gère très bien les branches et les fusions - Possibilité de travailler hors-ligne	- Peut être difficile à comprendre pour les débutants - Peut être compliqué à utiliser pour les gros projets

Mercurial	- Très similaire à Git, mais avec une courbe d'apprentissage plus facile	- Moins populaire que Git - Peut être lent sur les gros projets
Perforce	- Très performant pour les gros projets - Gère très bien les fichiers binaires	- Peut être coûteux pour les entreprises - Peut être difficile à utiliser pour les petits projets

Tableau 7.5. *Comparaison des différents systèmes de gestion de version de logiciel (Versioning)*

Il est important de choisir un système de gestion de versions adapté à ses besoins. Dans notre projet de conception et réalisation d'un système de gestion de courrier, nous avons choisi d'utiliser Git en raison de sa flexibilité, de sa performance et de sa capacité à gérer les branches et les fusions, tout en permettant de travailler hors-ligne.

7.3.5. Les gestionnaires de paquet

Un gestionnaire de paquets est un outil logiciel qui facilite l'installation, la mise à jour et la gestion des bibliothèques et des dépendances nécessaires à un projet. Il permet aux développeurs de spécifier les dépendances de leur projet dans un fichier de configuration, et de gérer ces dépendances de manière cohérente sur différents environnements.

7.3.5.1 Gestionnaire de paquet pour PHP

Fonctionnalités	Composer	PEAR	PECL
Gestion de dépendances	Oui	Oui	Non
Installation	Facile	Difficile	Facile
Mise à jour	Oui	Oui	Oui
Compatibilité	PSR-0, PSR-4	Non	Non
Popularité	Très élevée	Élevée	Moyenne

Tableau 7.6. *Comparaison des différentes gestionnaires de paquet de PHP*

Comme on peut le voir, Composer offre des fonctionnalités complètes de gestion de dépendances, avec une installation et une mise à jour faciles, une compatibilité avec les normes PSR-0 et PSR-4, ainsi qu'une popularité très élevée dans la communauté PHP. PEAR et PECL ont certaines fonctionnalités similaires, mais ont des défauts, comme une installation plus difficile ou une compatibilité limitée avec les normes actuelles. Ainsi, pour notre projet, il est recommandé d'utiliser Composer pour gérer les dépendances de notre application PHP.

7.3.5.2 Gestionnaire de paquet de javasript

Fonctionnalités	npm	Yarn
Installation rapide	Oui	Oui
Vitesse	Moyen	Très elevé
Sécurité	Elevé	Elevé
Mise en cache	Oui	Oui
Verrouillage des dépendances	Oui	Oui
Installation parallèle	Oui	Oui
Gestion des scripts	Oui	Oui
Visibilité des vulnérabilités	Non	Oui
Répétabilité des builds	Non	Oui
Choix des registres	Oui	Oui
Intégration des workspaces	Non	Oui

Tableau 7.7. *Comparaison des différentes gestionnaires de paquet javascript*

Comme on peut le constater dans le tableau, Yarn offre des avantages significatifs par rapport à npm, notamment en termes de vitesse, de sécurité et de fonctionnalités telles que la répétabilité des builds et la visibilité des vulnérabilités. En outre, Yarn permet de choisir facilement entre différents registres et intègre la gestion des workspaces. Pour ces raisons, Yarn est généralement considéré comme le meilleur choix pour la gestion des paquets JavaScript.

7.3.6. Les Système de gestion de base de données (ou SGBD)

Les systèmes de gestion de base de données (SGBD) sont des logiciels qui permettent de stocker et de manipuler des données de manière structurée. Il existe plusieurs types de SGBD en fonction du type d'information qu'ils manipulent :

- Les SGBD relationnels sont les plus couramment utilisés et sont basés sur des tables qui contiennent des données organisées en lignes et colonnes. Les exemples les plus connus sont MySQL, PostgreSQL et Microsoft SQL Server.
- Les SGBD NoSQL sont des SGBD non-relationnels qui peuvent stocker des données de différents types tels que des documents, des graphes, des paires clé-valeur, etc. Les exemples les plus connus sont MongoDB, Cassandra et Redis.
- Les SGBD orientés objets sont basés sur le modèle objet et permettent de stocker des objets tels qu'on les utilise dans les langages de programmation orientée objet. Les exemples les plus connus sont db4o et ObjectStore.
- Les SGBD temps réel sont conçus pour stocker et traiter des données en temps réel, comme des données de capteurs, de transactions financières, ou de réseaux sociaux. Les exemples les plus connus sont Apache Kafka, Apache Cassandra, et Redis.
- Les SGBD géospatiaux sont conçus pour stocker et interroger des données géospatiales, telles que les cartes et les coordonnées GPS. Les exemples les plus connus sont PostGIS, MongoDB et GeoServer.

Chaque type de SGBD a ses propres avantages et inconvénients en fonction de l'application spécifique à laquelle il est destiné.

7.3.6.1 Choix du type de la SGBD

Nous avons décidé d'utiliser un Système de Gestion de Base de Données (SGBD) relationnel pour notre application de système de gestion de courrier. Ce choix est basé sur le fait que les données à stocker sont principalement des informations transactionnelles structurées, avec des relations entre les différentes entités. Les SGBD relationnels offrent des avantages tels que la capacité à gérer efficacement de grands volumes de données structurées, la capacité à gérer les relations entre les données et la disponibilité de requêtes SQL avancées pour interroger les données. Parmi les exemples de SGBD relationnels, nous pouvons citer MySQL, PostgreSQL, Oracle, SQL Server et SQLite. Avec ce choix, nous pourrons garantir la fiabilité, la stabilité et la sécurité de notre application de gestion de courrier.

7.3.6.2 Choix du SGBD

Fonctionnalité	MySQL	Oracle	PostgreSQL	SQL Server
Licence	Open source / Commercial	Commercial	Open source	Commercial
Langage de requête	SQL	SQL	SQL	SQL
Prise en charge des transactions	Oui	Oui	Oui	Oui
Réplication	Oui	Oui	Oui	Oui
Intégrité référentielle	Oui	Oui	Oui	Oui
Gestion des utilisateurs	Oui	Oui	Oui	Oui
Sécurité	Moyenne	Forte	Forte	Forte
Performance	Rapide	Rapide	Rapide	Rapide
Facilité d'utilisation	Très facile	Moyenne	Moyenne	Moyenne

Tableau 7.8. *Comparaison des différents systèmes de gestion de base de données relationnels*

Dans ce tableau, nous avons comparé les quatre SGBD relationnels les plus populaires, à savoir MySQL, Oracle, PostgreSQL et SQL Server. Bien que tous ces SGBD offrent des fonctionnalités similaires, MySQL se démarque par sa simplicité et sa facilité d'utilisation. De plus, MySQL est open source, ce qui signifie que son utilisation est gratuite pour la plupart des utilisateurs.

7.3.7. Les outils de modélisation

Un outil de modélisation est un logiciel qui permet de créer des modèles ou des schémas de base de données pour les SGBD relationnels. Il permet de concevoir, visualiser, modifier et documenter les structures de données et leurs relations. MySQL Workbench et Visual Paradigm en sont des exemples courants.

Dans le cadre de notre projet de système de gestion de courrier utilisant MySQL comme SGBD, nous avons opté pour l'utilisation de MySQL Workbench en tant qu'outil de modélisation. Cette

décision est motivée par la simplicité et la facilité d'utilisation de MySQL Workbench, ainsi que par la cohérence entre l'outil de modélisation et le SGBD utilisé. En utilisant ces deux outils ensemble, nous pourrons mieux gérer la conception et la mise en place de notre base de données.

Chapitre 8 : Mise en œuvre et implémentation

8.1. Installation et configuration des outils

8.1.1. Installation de Visual Studio Code

La figure suivante nous montre la première étape d'installation de VSCode :

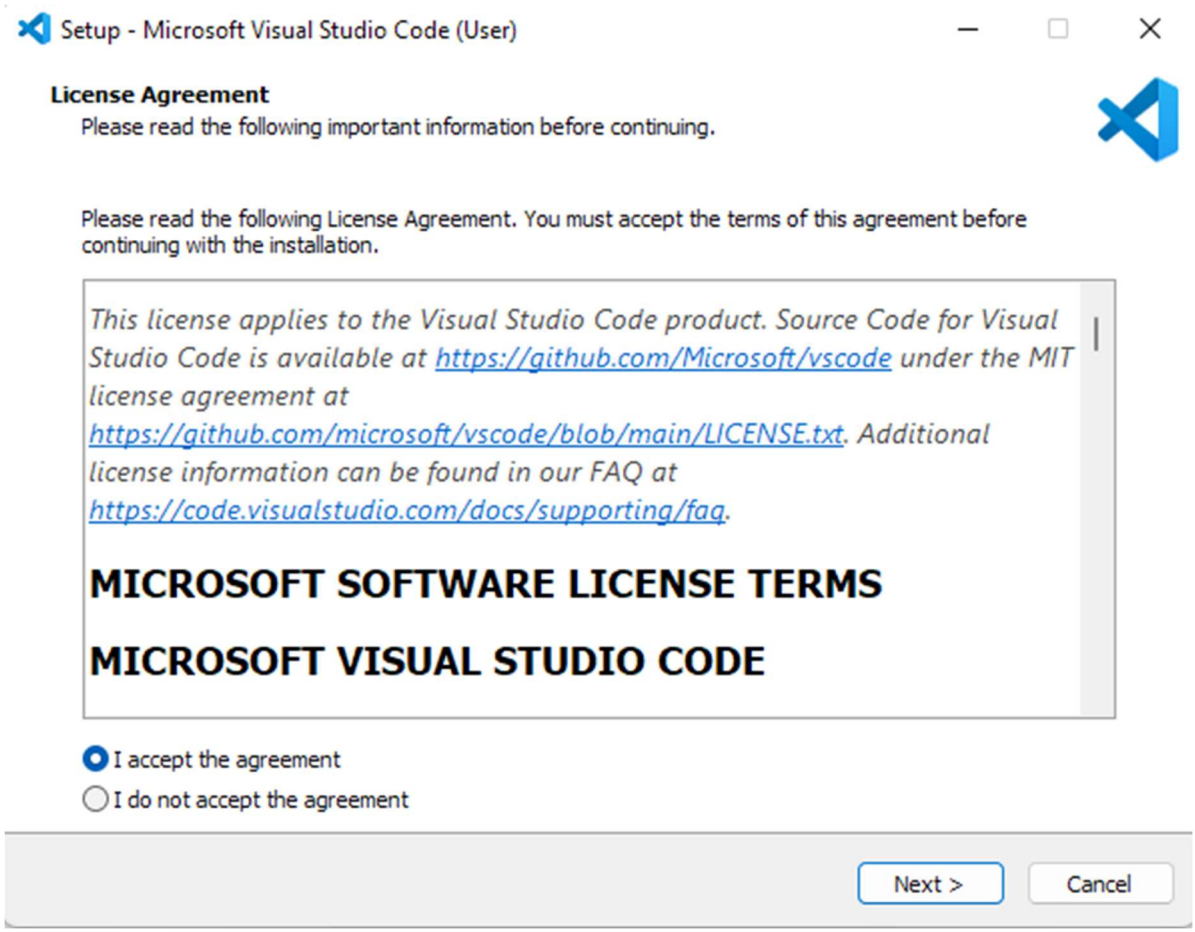

Figure 8.1. Installation de VSCode

8.1.2. Installation de composer

Voici la première page qui s'ouvre quand on veut installer composer, le gestionnaire de paquet de PHP :

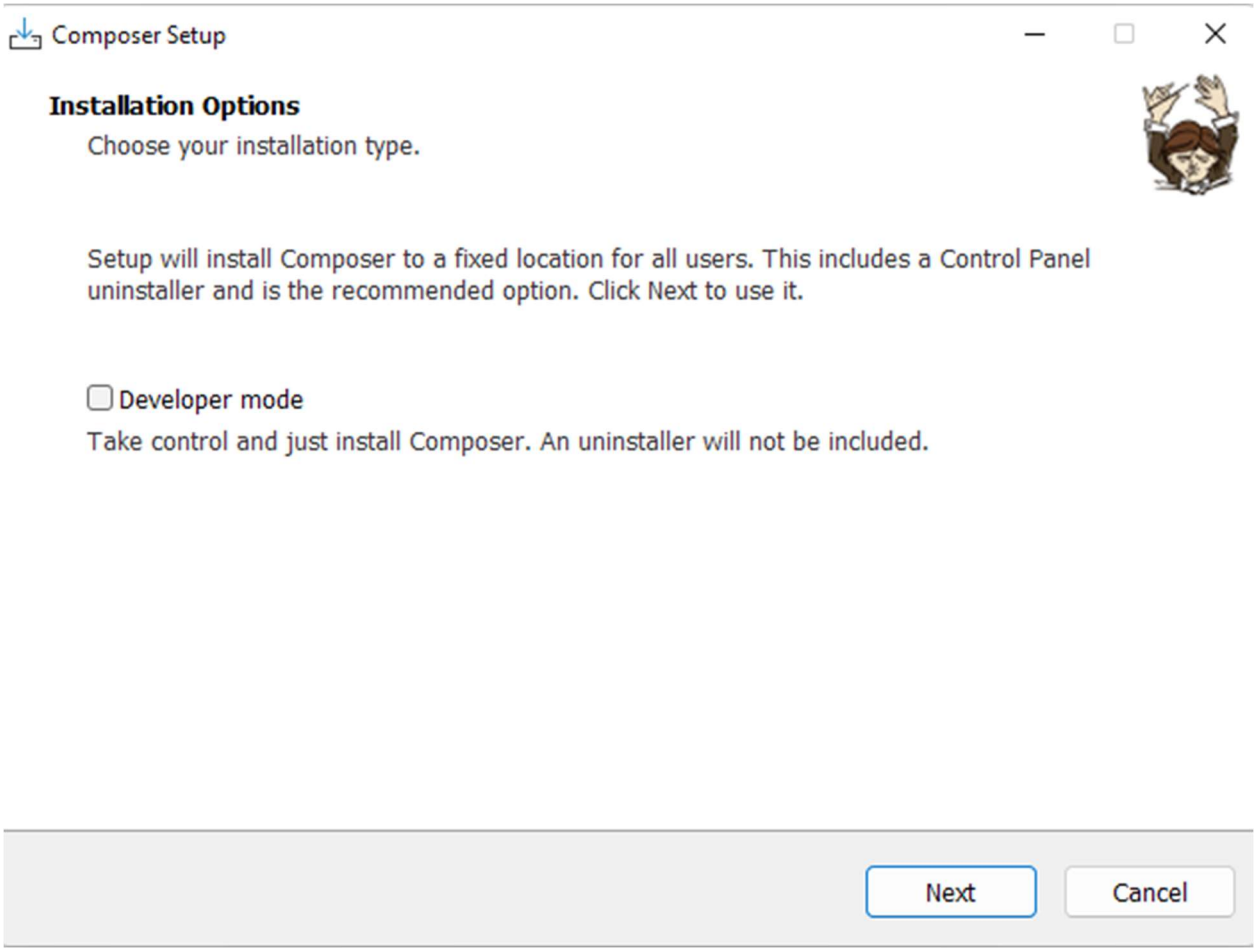

Figure 8.2. Installation de composer

8.1.3. Installation de nodejs

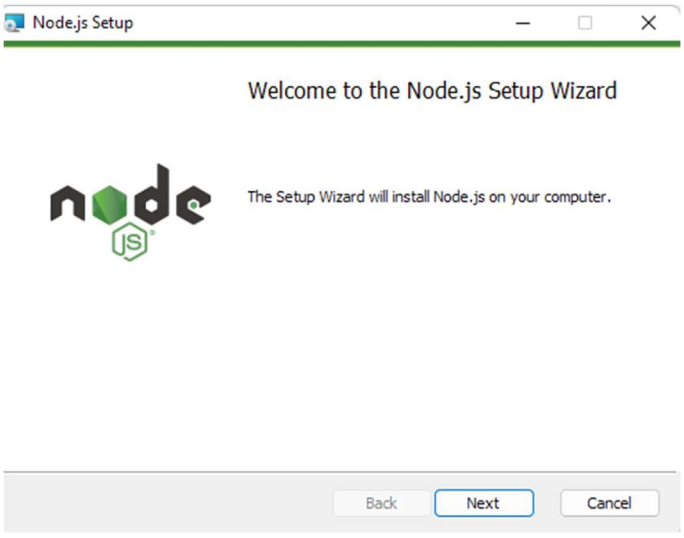

Figure 8.3. Installation de nodejs

8.1.4. Installation de Yarn

Etant donné que yarn n'est pas installé avec nodejs, mais seulement npm , il est nécéssaire d'installer yarn avec l'aide de npm. La figure suivante nous montre comment l'installer :

```
C:\Users\sitra>npm install -g yarn
```

Figure 8.4. Installation de yarn

8.1.5. Installation de Mysql workbench

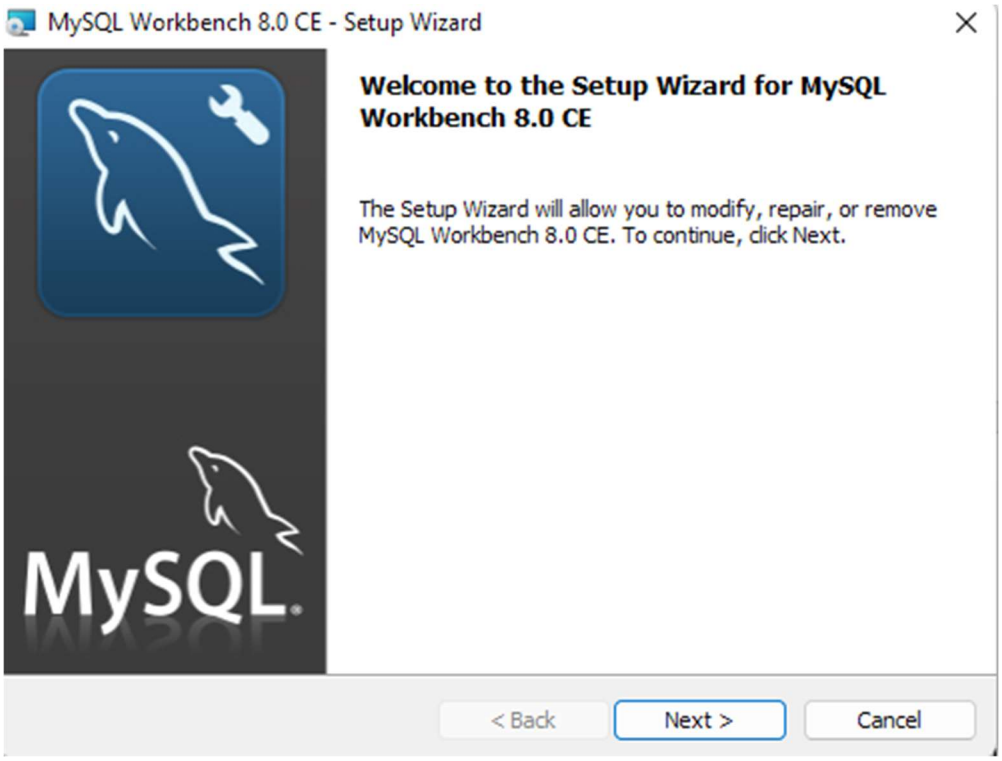

Figure 8.5. Installation de MySql Workbench

8.1.6. Installation de XAMPP, PHP et mysql

Le serveur comme XAMPP comprend déjà les outils nécessaires comme Mysql et PHP, donc il n'est plus nécessaire de les installer séparément. Les figures suivantes nous montrent son installation :

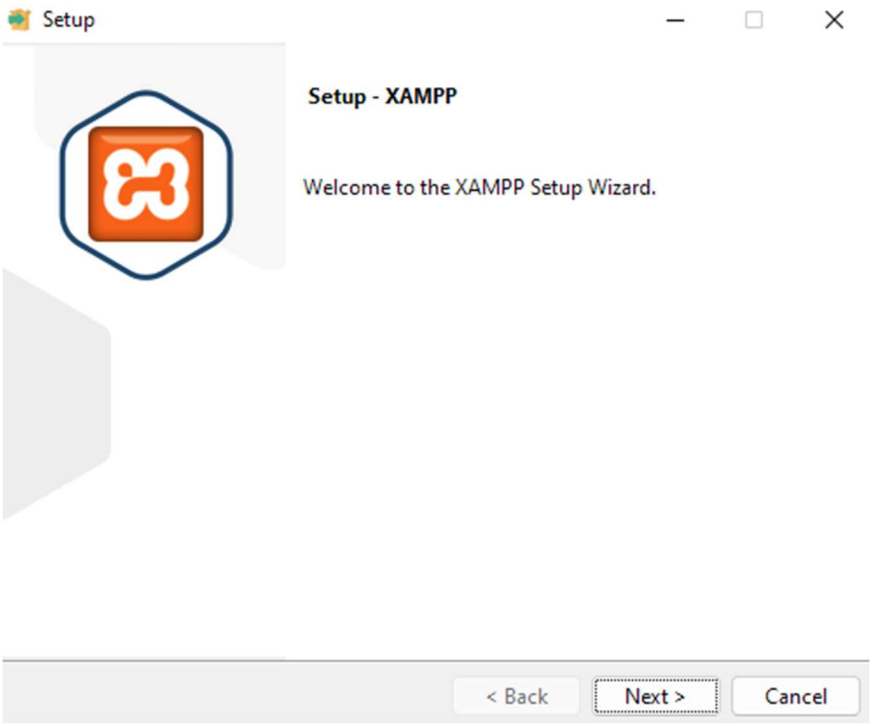

Figure 8.6. Installation de XAMPP

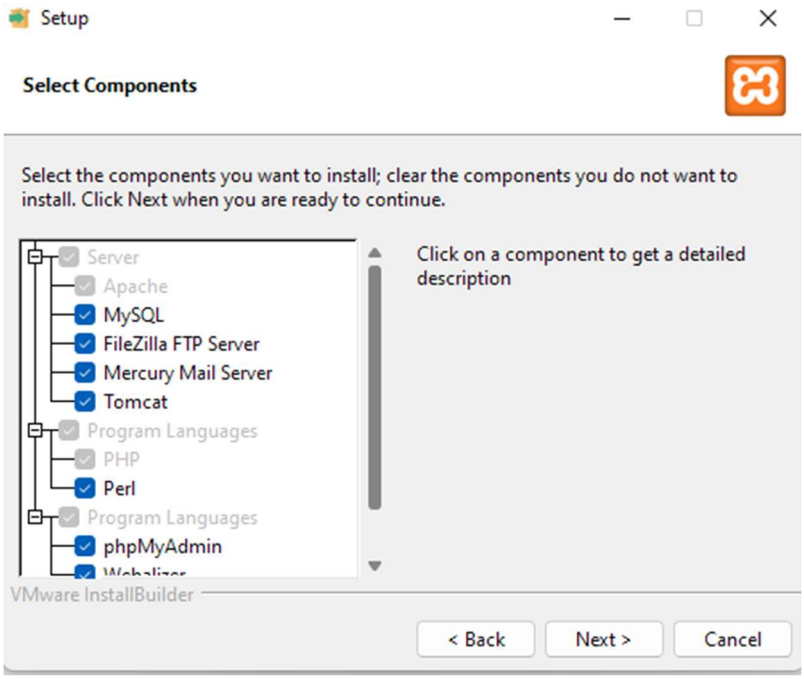

Figure 8.7. Installation des langages comme PHP et Mysql

8.2. Architecture logicielle

8.2.1. Architecture Client-serveur ou architecture en deux-tiers

8.2.1.1 Définition

Cette architecture implique la séparation des fonctionnalités entre une partie cliente (frontend) et une partie serveur (backend), reliées par le biais d'une interface de programmation (API) qui permet la communication et l'échange de données entre les deux parties

8.2.1.2 Composants

L'architecture client-serveur est composée de deux projets web distincts : le projet frontend et le projet backend.

- Le projet **frontend** : Il s'agit de la partie client de l'application qui est développée en utilisant des technologies web telles que HTML, CSS et JavaScript. Son rôle est de fournir une interface utilisateur conviviale pour les utilisateurs finaux.
- Le projet **backend** : Il s'agit de la partie serveur de l'application qui est développée en utilisant un langage de programmation tel que PHP, Java ou Python. Son rôle est de gérer les données et de fournir des API aux clients.
- L'**API** : C'est une interface de programmation d'application qui permet aux deux parties de communiquer entre elles. L'API peut être développée en utilisant des technologies telles que REST ou SOAP.

Nous avons décidé d'utiliser REST comme technologie d'API pour notre application car sachant que nous avons choisi Symfony comme framework pour notre projet backend, nous avons constaté que celui-ci fournit un bundle appelé API Platform, qui facilite exponentiellement la définition d'un REST API en offrant des fonctionnalités de documentation automatique, de validation des données, de pagination, de filtrage et de tri des résultats, entre autres. En utilisant API Platform, nous pourrons développer notre API plus rapidement et plus efficacement, tout en garantissant une documentation précise et à jour pour les utilisateurs de l'API. C'est pourquoi nous sommes convaincus que REST et API Platform sont les meilleurs choix pour notre projet.

8.2.1.3 Avantages

L'architecture client-serveur présente plusieurs avantages, notamment :

- Séparation claire des responsabilités entre le frontend et le backend.
- Meilleure évolutivité et maintenabilité de l'application.
- Possibilité de développer les deux parties de manière indépendante.
- Possibilité de réutiliser les API dans d'autres projets.

8.2.1.4 Inconvénients

L'architecture client-serveur présente également certains inconvénients, tels que :

- Complexité accrue en raison de la nécessité de développer deux projets distincts.
- Plus grande difficulté à déboguer en raison de la nature distribuée de l'application.

8.2.2. Architecture MVC (Model-View-Controller)

Nous allons utiliser cette architecture dans le projet frontend etant donné que c'est même le fondement d'un framework moderne

8.2.2.1 Définition

L'architecture MVC est une méthode d'organisation de l'interface graphique d'un programme. Elle consiste à distinguer trois entités distinctes : le modèle, la vue et le contrôleur, chacune ayant un objectif précis dans l'interface.

L'organisation globale d'une interface graphique est souvent difficile. Bien que la façon dont MVC organise une interface ne soit pas une solution miracle, elle fournit souvent un point de départ qui peut ensuite être adapté. Il fournit également un cadre pour l'organisation d'une application.

Les rôles des trois entités dans l'architecture MVC sont les suivants :

- Modèle : données (accès et mise à jour)
- Vue : interface utilisateur (entrées et sorties)
- Contrôleur : gestion des événements et synchronisation

8.2.2.2 Rôle du modèle

Le composant "Modèle" regroupe et collecte toutes les données. Il contient, après tout, les classes dont les instances doivent être visualisées et traitées. Le modèle spécifie quelles données doivent être incluses dans l'application. Si l'état de ces données change, le modèle avertira généralement la vue (afin que l'affichage puisse changer au besoin) et occasionnellement le contrôleur (si une logique différente est requise pour contrôler le changement de vue). Il encapsule l'accès aux données.

8.2.2.3 Rôle de la vue

La vue sert d'interface utilisateur. Sa première tâche consiste à afficher les données du modèle qu'il a récupérées. Sa deuxième tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d'entrée, boutons, etc.) Ses différents événements sont transmis au contrôleur.

La vue peut également fournir plusieurs vues, partielles ou complètes. Par exemple, la donnée unique de l'application de conversion de base de données est un entier. Ce même entier est affiché de diverses manières (sous forme de texte dans diverses bases de données, bit par bit avec des cases à cocher et avec des curseurs). La vue peut également offrir à l'utilisateur la possibilité de modifier la vue.

8.2.2.4 Rôle du contrôleur

Le contrôleur synchronise le modèle avec la vue. Tous les événements utilisateurs sont reçus et les actions à effectuer sont initiées. Si une modification de données est nécessaire pour une action de données, le contrôleur demande que les données du modèle soient modifiées, puis informe que les données ont été modifiées pour mettre à jour la vue. Certains événements utilisateur concernent la vue, pas les données. Le contrôleur demande un changement dans ce cas.

8.2.2.5 Fonctionnement

Le fonctionnement de ces différentes parties sont illustrées par la figure suivante :

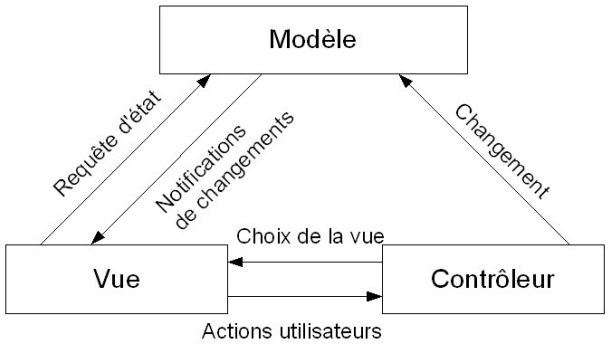

Figure 8.8. Architecture MVC.

8.3. Démarche de mise en place du projet

Etant donnée l'architecture de notre application, nous avons besoins de crée deux projets distincts qui sont le frontend et le backend. Nous allons donc voir les démarches de création de ces deux différents projets :

8.3.1. Création du projet backend

Pour pouvoir crée une application Symfony, nous avons besoins d'utiliser les commandes fournies par composer. La commande à faire dans un terminal est la suivante :

```
C:\Users\sitra>composer create-project symfony/skeleton mail_manager
```

Figure 8.9. Création du projet Symfony

8.3.1.2 Création d'une entité dans symfony

Pour créer les tables, plus communément appelé entité dans Symfony, il faut exécuter dans un terminal, dans le répertoire du projet Symfony, la commande suivante :

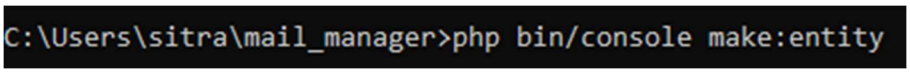

Figure 8.10. Création d'une entité dans Symfony

Cette commande va lancer un assistant en ligne de commande pour vous guider dans la création de votre entité. Vous devrez répondre à quelques questions sur le nom de l'entité, les propriétés et les relations avec d'autres entités.

8.3.1.3 Lancement du projet Symfony

Pour lancer une application

Symonfy, nous dévons exécuter la commande suivante, dans une terminale ouverte dans le répertoire du projet :

Figure 8.11. Lancement d'un projet Symfony

8.3.2. Création du projet frontend

Pour pouvoir crée un projet en react, il faut taper dans le terminal la commande suivante :

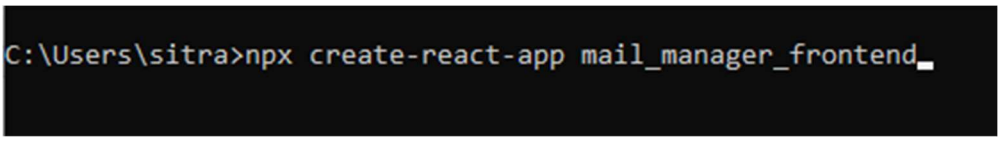

Figure 8.12. Création d'un projet react

8.3.2.2 Lancement d'un projet react

Pour lancer un projet react en utilisant npm ou yarn, il faut exécuter respectivement l'un des commandes suivantes dans un terminale ouvert dans le répertoire du projet :

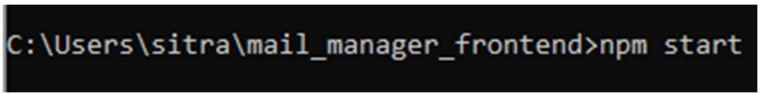

Figure 8.13. Lancement d'un projet react avec NPM

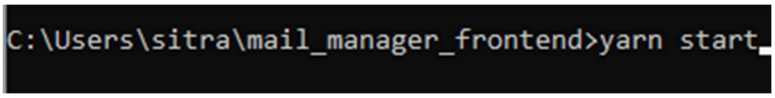

Figure 8.14. Lancement d'un projet react avec YAR

8.3.3. Création du REST API avec API platform

Pour créer le REST API de chaque entité du projet, il suffit d'installer le bundle api platform et ensuite de faire des configurations, la figure suivante illustre comment installer api plateform à l'aide de composer :

```
C:\Users\sitra\mail_manager>composer require api
```

Figure 8.15. Installation d'API platform

Chapitre 9 : Présentation de l'application développée

Dans ce chapitre nous allons voir les extraits de code des différentes fonctionnalités de l'application développée ainsi que quelques captures d'écran de son interface utilisateur.

9.1. Extrait de code et interface utilisateur

9.1.1. La fonctionnalité d'authentification

9.1.1.1 Extrait de code

Dans cette section, nous allons examiner les extraits de code qui gèrent l'authentification au sein de notre application. Plus précisément, nous allons examiner le code React et Symfony qui communiquent entre eux pour permettre aux utilisateurs de se connecter et d'accéder aux zones protégées de l'application. En examinant en détail le processus d'authentification, nous pouvons mieux comprendre comment les différents composants de notre application travaillent ensemble pour offrir une expérience utilisateur transparente. Nous commencerons par examiner le code React pour le composant de connexion, puis nous passerons au code Symfony qui gère l'authentification côté serveur.

```javascript
function Login() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const history = useHistory();

  const handleSubmit = async (event) => {
    event.preventDefault();
    const response = await axios.post('http://localhost:8000/login_check', {
      username,
      password
    });

    // Vérifier si l'authentification est réussie en vérifiant si le token est renvoyé
    if (response.data.token) {
      // Stocker le token dans le local storage pour les futurs appels d'API
      localStorage.setItem('token', response.data.token);

      // Rediriger vers la page d'accueil
      history.push('/');
    } else {
      // Afficher un message d'erreur si l'authentification a échoué
      alert('Identifiants incorrects');
    }
  };
  return {...}
  }
```

Figure 9.1. Extrait du code en javascript qui s'occupe de l'authentification

```php
class AuthController extends AbstractController
{
    private $jwtManager;

    public function __construct(JWTTokenManagerInterface $jwtManager)
    {
        $this->jwtManager = $jwtManager;
    }

    public function login(Request $request)
    {
        $username = $request->request->get('username');
        $password = $request->request->get('password');

        // Vérifier les identifiants de l'utilisateur
        $user = $this->getDoctrine()->getRepository(User::class)->findOneBy(['username' => $username]);
        if (!$user || !password_verify($password, $user->getPassword())) {
            throw new BadCredentialsException();
        }

        // Générer un token JWT pour l'utilisateur
        $token = $this->jwtManager->create($user);

        // Retourner le token sous forme de réponse JSON
        return new JsonResponse(['token' => $token]);
    }
}
```

Figure 9.2. Extrait du code en PHP qui s'occupe de l'authentification

9.1.1.2 Interface utilisateur

Maintenant, dans la figure suivante, nous allons voir l'interface utilisateur de la page de connexion :

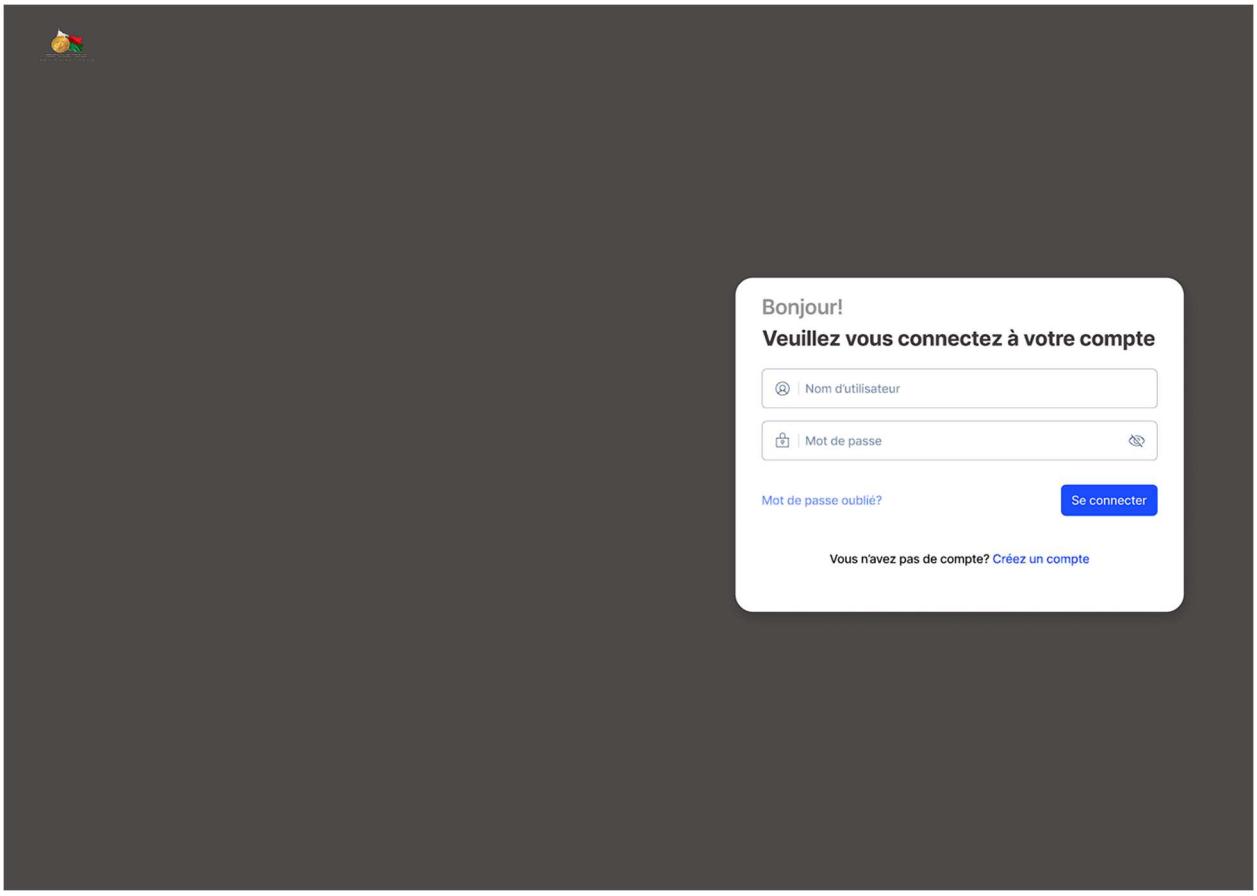

Figure 9.3. Interface Utilisateur de la page de connexion

9.1.2. La fonctionnalité de lister les utilisateurs

9.1.2.1 Extrait de code

```
function UserList() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    axios.get('/api/users')
      .then(response => {
        setUsers(response.data);
      })
      .catch(error => {
        console.log(error);
      });
  }, []);

  return (...);
}
```

Figure 9.4. Extrait de code qui s'occupe de la liste des utilisateurs en javascript

```php
/**
 * @Route("/api")
 */
class UserController extends AbstractController
{
    /**
     * @Route("/users", methods={"GET"})
     */
    public function listUsers(UserRepository $userRepository): JsonResponse
    {
        $users = $userRepository->findAll();
        return $this->json($users);
    }
}
```

Figure 9.5. Extrait de code qui s'occupe de la liste des utilisateurs en PHP

9.1.2.2 Interface utilisateur

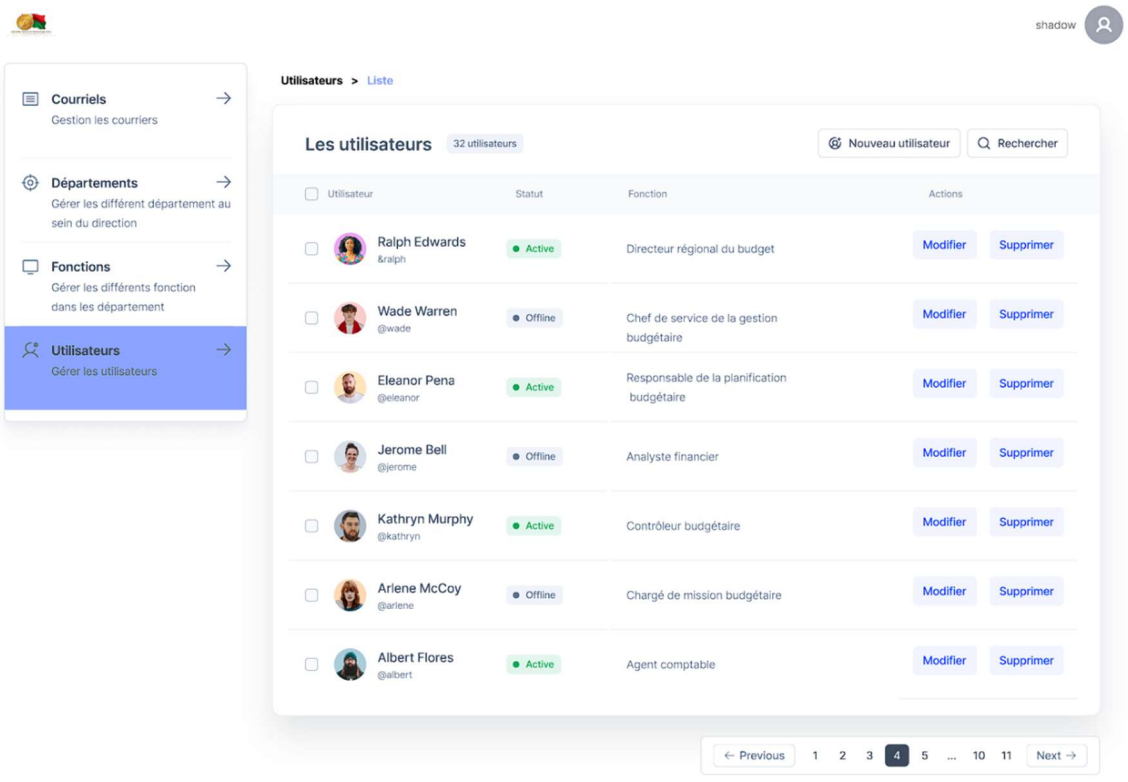

Figure 9.6. Interface utilisateur de liste des utilisateurs

9.1.3. Fonctionnalité création d'un nouveau courriel

9.1.3.1 Extrait de code

```javascript
axios.post('http://localhost:8000/api/mails', formData)
  .then(response => {
    console.log(response.data);
    // Réinitialiser le formulaire
    setMail({
      Mail_object: '',
      Mail_origin: '',
      Mail_pre_ref: '',
      Mail_Date: '',
      Mail_id_direction: '',
      Id_outgoing_mail: '',
      Id_incoming_mail: '',
      Mail_Id_user: '',
      Mail_ref: '',
      Mail_classification: '',
      Mail_arrival_date: '',
      Mail_recipient: '',
      Mail_processing_date: '',
      Mail_status: '',
      Mail_location: '',
    });
    // Récupérer la liste des courriels mise à jour
    axios.get('http://localhost:8000/api/mails')
      .then(response => {
        setMails(response.data);
      })
      .catch(error => {
        console.error(error);
      });
  })
  .catch(error => {
    console.error(error);
  });
```

Figure 9.7. Extrait de code qui s'occupe de la création d'un courriel en javascript

```php
/**
     * @Route("/mails", name="create_mail", methods={"POST"})
     */
    public function createMail(Request $request): Response
    {
        $entityManager = $this->getDoctrine()->getManager();

        // Create new Mail object with POST data
        $mail = new Mail();
        $mail->setMailObject($request->request->get('Mail_object'));
        $mail->setMailOrigin($request->request->get('Mail_origin'));
        $mail->setMailPreRef($request->request->get('Mail_pre_ref'));
        $mail->setMailDate(new \DateTime($request->request->get('Mail_Date')));
        $mail->setMailIdDirection($request->request->get('Mail_id_direction'));
        $mail->setIdOutgoingMail($request->request->get('Id_outgoing_mail'));
        $mail->setIdIncomingMail($request->request->get('Id_incoming_mail'));
        $mail->setMailIdUser($request->request->get('Mail_Id_user'));
        $mail->setMailRef($request->request->get('Mail_ref'));
        $mail->setMailClassification($request->request->get('Mail_classification'));
        $mail->setMailArrivalDate(new \DateTime($request->request->get('Mail_arrival_date')));
        $mail->setMailRecipient($request->request->get('Mail_recipient'));
        $mail->setMailProcessingDate(new \DateTime($request->request->get('Mail_processing_date')));
        $mail->setMailStatus($request->request->get('Mail_status'));
        $mail->setMailLocation($request->request->get('Mail_location'));
        $mail->setMailCreatedAt(new \DateTime('now'));
        $mail->setMailCreatedBy($request->request->get('Mail_created_by'));
        $mail->setMailUpdatedAt(new \DateTime('now'));
        $mail->setMailUpdatedBy($request->request->get('Mail_updated_by'));

        // Save new Mail object to database
        $entityManager->persist($mail);
        $entityManager->flush();

        // Return a success response
        return $this->json([
            'message' => 'Mail created successfully!',
            'mail' => $mail,
        ]);
    }
```

Figure 9.8. Extrait de code qui s’occupe de la création d’un courriel en PHP

9.1.3.2 Interface utilisateur

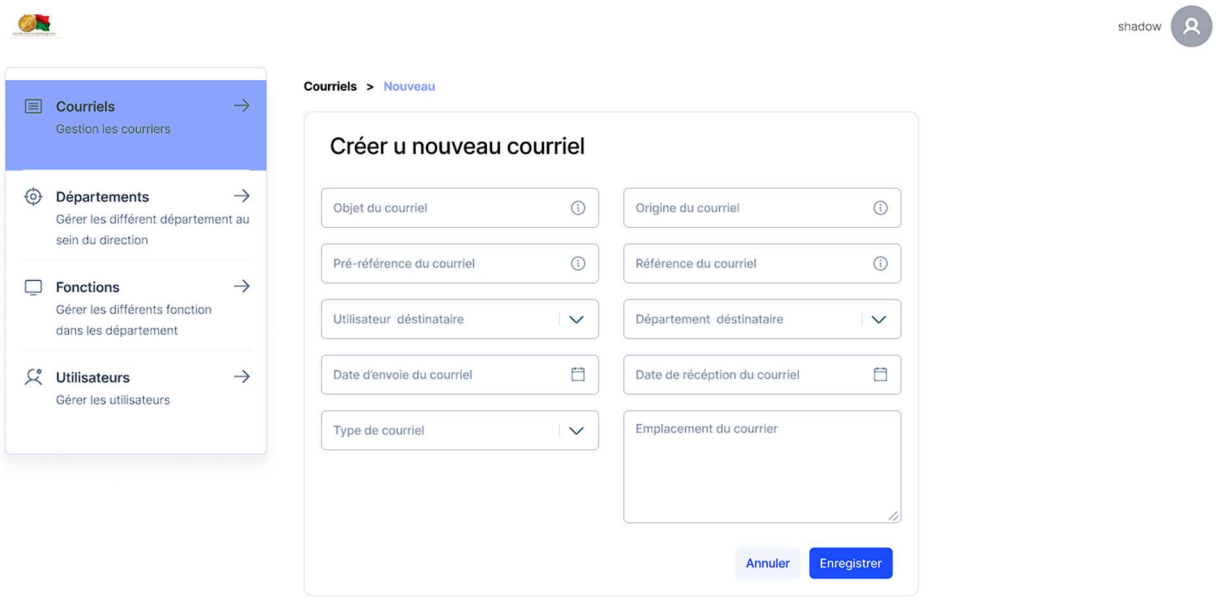

Figure 9.9. Interface utilisateur de la création de courriel

9.1.4. Fonctionnalité d'affichage de la liste des courriers

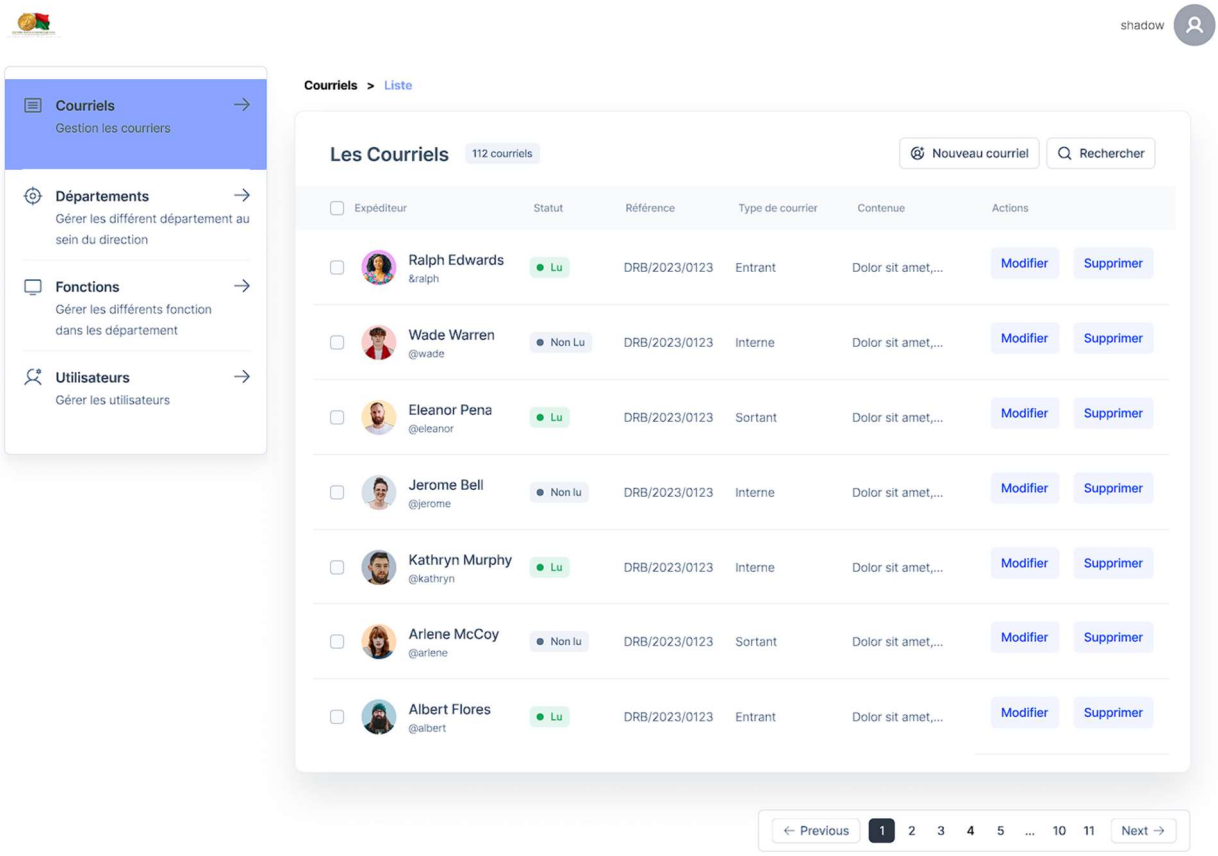

Figure 9.10. Interface utilisateur de la liste des courriels

9.2. Conclusion

En somme, ce chapitre avait pour but de présenter l'environnement technique et les différentes étapes nécessaires à la création de l'application. Nous avons commencé par décrire l'architecture technique, puis identifié les différentes phases du processus de développement. Enfin, nous avons présenté quelques interfaces pour illustrer certaines des fonctionnalités de l'application.

.

CONCLUSION GÉNÉRALE

En conclusion de ce rapport de stage, nous avons présenté la conception et la réalisation d'un système de gestion de courrier pour la Direction Régionale du Budget de Vakinankaratra. Ce projet avait pour objectif de faciliter la gestion des courriers, une tâche importante et fastidieuse pour les employés de la direction, qui jusqu'à présent utilisaient des méthodes manuelles pour suivre les courriers.

Nous avons utilisé la méthode MERISE pour concevoir le système, qui a été développé en utilisant les dernières technologies du développement web afin d'obtenir une application performante et à la pointe de la technologie. Tout au long de ce stage, nous avons été confrontés à de nombreux défis, notamment la compréhension de la façon dont les courriers étaient stockés et suivis au sein de la direction, mais nous avons réussi à les surmonter grâce à notre persévérance et notre détermination.

Nous sommes fiers d'avoir pu établir un lien entre nos connaissances universitaires et celle du monde professionnel à travers ce stage. Nous avons également acquis de nouvelles compétences techniques, approfondi et mis en pratique nos connaissances théoriques en matière de conception et de développement de systèmes informatiques.

Il est important de noter que même si notre système de gestion de courrier répond aux besoins actuels de la direction, il reste toujours des améliorations possibles. Nous encourageons donc la direction à continuer à évaluer les besoins des utilisateurs et à apporter les améliorations nécessaires pour garantir l'efficacité et la performance du système.

En somme, ce stage nous a permis d'acquérir une expérience précieuse dans le développement de systèmes d'information, mais également de contribuer à l'amélioration des processus de travail de la Direction Régionale du Budget de Vakinankaratra.

BIBLIOGRAPHIE

Nous tous chez monday.com, 2021. Guide de la gestion de projet informatique [en ligne].

Mondayfrancais. Disponible sur : <https://monday.com/blog/fr/the-newbies-guide-to-it-project-management/> (consulté le 15/04/2021)

Larreur G.,2021. Qu'est-ce-qu'un cahier des charges informatique [en ligne]. Ozytis. Disponible sur : <https://ozytis.fr/cahier-des-charges-informatique/ (consulté le 15/04/2021)

Client lourd et client léger », https://www.supinfo.com/articles/single/3424-client-lourdclient-leger, du 05 Mars 2021

« Architecture d'un système d'information », http://www.guillaumeriviere.name/ estia/si/pub/SI_COURS-01_2012_introduction.pdf, (consulté le 5/04/2021).

« UML », http://openclassrooms.com /courses/26832-apprenez-a-programmer-enjava/21530-modeliser-ses-objets-grace-a-uml, (consulté le 17/04/2021).

Benslimane, D., Dustdar, S., Sheth, A. (eds.), 2010. Services Mashups: Concepts, Challenges, and Applications. Springer-Verlag Berlin Heidelberg. 262 p.

Booch, G., Rumbaugh, J., Jacobson, I., 1999. The Unified Modeling Language User Guide. Addison-Wesley Professional. 688 p.

Fowler, M., 2004. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional. 208 p.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional. 416 p.

Larman, C., 2004. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall. 736 p.

Martin, R.C., 2002. Agile Software Development: Principles, Patterns, and Practices. Prentice Hall. 768 p.

Martin, R.C., 2003. UML for Java Programmers. Prentice Hall. 208 p.

Reenskaug, T., 1979. Working with Objects: The OOram Software Engineering Method. W. W. Norton & Company. 208 p.

Shalloway, A., Trott, J.R., 2011. Design Patterns Explained: A New Perspective on Object-Oriented Design. Addison-Wesley Professional. 480 p.

Sommerville, I., 2011. Software Engineering. Pearson Education Limited. 792 p.

RÉSUMÉ

Le présent rapport décrit la conception et la mise en œuvre d'une application web pour la gestion des courriers, offrant un exemple concret de la gestion d'un projet informatique dans un contexte professionnel. Nous avons opté pour la méthode MERISE, une étape incontournable pour assurer la réussite d'un tel projet, et nous avons utilisé les technologies React et Symfony, qui reposent sur une architecture client-serveur et les langages PHP et JavaScript. VSCode a été l'éditeur de texte de choix, tandis que MySql a été le SGBD choisi. Les défis ont été surmontés grâce à notre détermination et notre persévérance, et nous avons élaboré une solution efficace qui répond aux besoins de la direction, tout en sachant que des améliorations pourront être apportées au fil du temps pour maintenir la pertinence de l'application en fonction des besoins évolutifs de la direction.

Mots-clés : MVC, SGBD, MERISE, react, Symfony, PHP, Javascript, VSCode, client-serveur ;

ABSTRACT

This report describes the design and implementation of a web application for mail management, providing a concrete example of managing an IT project in a professional setting. We chose the MERISE method, an essential step to ensure the success of such a project, and we used the React and Symfony technologies, which are based on a client-server architecture and the PHP and JavaScript languages. VSCode was the text editor of choice, while MySql was the chosen SGBD. Challenges were overcome through our determination and perseverance, and we developed an effective solution that meets the needs of the direction, knowing that improvements may be made over time to maintain the relevance of the application as the direction's needs evolve.

Keywords: MVC, SGBD, MERISE, React, Symfony, PHP, JavaScript, VSCode, client-server.