

**UNIVERSITE DE FIANARANTSOA**

**ECOLE DE MANAGEMENT ET D’INNOVATION TECHNOLOGIQUE**

**MEMOIRE DE FIN DE CYCLE**

**EN VUE DE L’OBTENTION DU DIPLÔME DE LICENCE**

**Parcours : Développement d’Application Intranet et Internet**

Thème :

« E-CARRIER »

Conception et réalisation d’une application de gestion d’information des agents de la douane Malgache

Présenté par : **RASATAMALALANIAINA Hajatiana**

devant les membres de jury  composés de :

|  |  |
| --- | --- |
| Encadreur professionnel : | **Dr Paul Mazoto RAKOTOMAMONJY** |
| Encadreur pédagogique : | **---** |
| Examinateurs : | **---**  **---** |

***Année Universitaire  2021-2022***

AVANT-PROPOS

Chaque année, l’Ecole de Management et d’Innovation Technologique envoie ses étudiants en troisième année de licence en stage pratique pour une durée de trois (3) mois. Son principal objectif est de former les étudiants dans le domaine de travail. Le futur cadre doit être à la portée de savoir-faire, volonté, dynamisme et professionnalisme.

De ce fait, on a effectué la nôtre au sein de la douane Antaninarenina. Ce stage a pour but de renforcer et de consolider la connaissance acquise durant l’année de formation. Ainsi, on a comme thème de stage « CONCEPTION ET REALISATION D’UNE APPLICATION DE GESTION D’INFORMATION DES AGENTS DE LA DOUANE MALGACHE »

REMERCIEMENT

Cet ouvrage est le fruit des travaux effectués durant le stage de formation, au cours duquel plusieurs participants font leur part et apportent leur soutien à la réalisation.

Ainsi, je tiens à exprimer ici mes plus vifs remerciements à tous ceux qui ont contribué à la réussite de ce présent mémoire.

Mes remerciements s’adressent avant tout à l’Université de Fianarantsoa et plus particulièrement aux enseignants de l’Ecole de Management et d’Innovation Technologique (EMIT), spécialement ceux de la mention Informatique pour les cours qu’ils m’ont dispensés afin de me permettre d’approfondir mes connaissances et compétences.

* Professeur HAJALALAINA Aimé Richard, Président de l’Université de Fianarantsoa ;
* Docteur Hasina, directeur de l’Ecole de Management et d’Innovation Technologique, qui nous accepte à passer notre stage au sein de son établissement et notre collaboration durant le stage ;
* Spécialement à Madame RABEZANAHARY Hobiniaina, Responsable de Mention Développement d’Application Intranet - Internet.
* Mes sincères remerciements également à l’évaluateur qui ont accepté d’évaluer ce travail à sa juste valeur et avec leur aimable compréhension.
* Mes remerciements particuliers à Docteur RAKOTOMAMONJY Paul Mazoto pour ses conseils précieux et pour ses directives en qualité d’encadreur professionnelle.
* Les employés de la douane Malagache à Antaninarenina, je suis entièrement reconnaissant envers eux, pour leurs gentillesses que je puisse faire ce stage dans leur département.
* Sans oublier ma famille et mes amis, pour leurs soutiens moraux et financiers qu’ils m’ont accordés.

LISTE DE FIGURES

[Figure 1.4‑1: Les offres de formations de l'EMIT 4](#_Toc115689825)

[Figure 1.6‑1 : Organigramme de l'Emit 9](#_Toc115689826)

[Figure 2.2‑1: Organigramme de la direction général de la douane 10](#_Toc115689827)

[Figure 3.5‑1: diagramme de gantt 17](#_Toc115689828)

[Figure 4.3‑1: Cycle de vie en v 23](#_Toc115689829)

[Figure 4.3‑2:cycle de vie en spirale 25](#_Toc115689830)

[Figure 4.3‑3:Cycle de vie par incrément 27](#_Toc115689831)

[Figure 6.4‑1: Image qui représente l’acteur 39](#_Toc115689832)

[Figure 6.4‑2: Image qui représente le cas d’utilisation 40](#_Toc115689833)

[Figure 6.4‑3: Formalisme d’un cas d’utilisation 40](#_Toc115689834)

[Figure 6.4‑4: Formalisme d’une classe 41](#_Toc115689835)

[Figure 6.4‑5: Formalisme du diagramme de classe 45](#_Toc115689836)

[Figure 6.4‑6: Image qui représente l’agrégation 45](#_Toc115689837)

[Figure 6.4‑7: Image qui représente l’agrégation 45](#_Toc115689838)

[Figure 6.4‑8: Formalisme du diagramme de séquence 46](#_Toc115689839)

[Figure 6.4‑9: Formalisme du diagramme d’activité 47](#_Toc115689840)

[Figure 6.4‑10: Image d’un port 48](#_Toc115689841)

[Figure 6.4‑11: Formalisme d'un diagramme de composant 48](#_Toc115689842)

[Figure 6.4‑12: Diagramme de composant du système 48](#_Toc115689843)

LISTE DE TABLEAUX

[Tableau 1.2‑1 : Tableau récapitulatif 3](#_Toc115690372)

[Tableau 3.5‑1: Planification du projet 18](#_Toc115690373)

[Tableau 6.2‑1: Dictionnaire de données 39](#_Toc115690374)

[Tableau 6.4‑1: Les différents types de visibilité 42](#_Toc115690375)

[Tableau 6.4‑2: Tableau montrant les types de relation entre les Classes 44](#_Toc115690376)

[Tableau 6.4‑3: Tableau montrant les différentes multiplicités 45](#_Toc115690377)

LISTE D’ABREVIATIONS

Glossaires

**Application** : Un ensemble des programmes informatiques qui permet d’assurer une tâche.

**Base de données** : Un ensemble structuré de données enregistrées sur les supports accessibles par l’ordinateur.

**Identifiant** : Propriété immuable d’un objet permettant de le distinguer des autres et de retrouver son adresse en mémoire ou sur disque, ainsi que son type.

**Net** : Littéralement, réseau, filet. Une façon familière de prononcer Internet. L'équivalent du Réseau (avec majuscule) en français, par rapport aux réseaux en général.

**Ordinateur portable** : Ordinateur conçu pour être déplacé manuellement et utilisé où l’on besoin et même dans des différents lieux.

**RAM** (**Random Access Memory**) : Ce qui signifie que l'on peut accéder instantanément à n'importe quelle partie de la mémoire, permet de stocker des informations pendant tout le temps de fonctionnement de l'ordinateur.

**Requête** : Dans une base de données, une requête donne la possibilité de rechercher des données en spécifiant des critères.

**Script** : Programme simple consistant en un ensemble d'instructions destinées à exécuter ou automatiser des tâches ou fonctions spécifiques.

**Système d'exploitation (SE)** : Logiciel qui contrôle l'affectation et l'utilisation de ressources matérielles telles que la mémoire, le temps processeur, l'espace disque et les périphériques. Un système d'exploitation est la base sur laquelle s'exécutent les applications. Prenons comme exemple Windows 98, Mac OS et UNIX.

**Système** : Tout ensemble d'éléments de composants fonctionnant ensemble pour l'exécution d'une tâche.

**Système de gestion de base de données (SGBD)** : un ensemble coordonné de logiciels qui  
permet de décrire, manipulé, traiter les ensembles de données formant la base. Il doit également assurer la sécurité et la confidentialité des données dans un environnement où de nombreux utilisateurs ayant des besoins variés peuvent interagir simultanément sur ces données.

TABLE DE MATIERE

[Chapitre 1 Présentation de l’EMIT 2](#_Toc115686950)

[1.2. Historique 2](#_Toc115686951)

[1.3. Objectif de l’école 3](#_Toc115686952)

[1.4. Formations existantes 4](#_Toc115686953)

[1.4.1. Cycle de Licence 5](#_Toc115686954)

[1.4.2. Cycle Master 6](#_Toc115686955)

[1.5. Partenaires de l’EMIT 7](#_Toc115686956)

[1.6. Organigramme de l’EMIT 8](#_Toc115686957)

[Chapitre 2 : Présentation de la douane 10](#_Toc115686958)

[2.1. Histoire de la douane malagasy 10](#_Toc115686959)

[2.2. La Direction général de la douane 10](#_Toc115686960)

[2.2.1. Organigramme de la direction général de la douane 10](#_Toc115686961)

[2.3. Plan stratégique 11](#_Toc115686962)

[2.3.1. Les axes 11](#_Toc115686963)

[2.3.2. Alignement stratégique et modalités de mise en œuvre 12](#_Toc115686964)

[Chapitre 3 : Présentation du projet 14](#_Toc115686965)

[3.1. Origine du projet 14](#_Toc115686966)

[3.2. Objectif du projet 14](#_Toc115686967)

[3.3. Problématique 14](#_Toc115686968)

[3.4. Conduites de projet 15](#_Toc115686969)

[3.4.1. Analyse des besoins 15](#_Toc115686970)

[3.4.2. Bilan critique de l’existant 15](#_Toc115686971)

[3.4.3. Etude de faisabilité 16](#_Toc115686972)

[3.4.4. Etude d’opportunité 16](#_Toc115686973)

[3.5. Cahier des charges 17](#_Toc115686974)

[3.5.1. Les trois volets encadrants le travail 17](#_Toc115686975)

[3.5.2. Les cible 17](#_Toc115686976)

[3.5.3. Le délais et planification de la réalisation du développement du projet 18](#_Toc115686977)

[Chapitre 4 : Méthodes et notations utilisées 20](#_Toc115686978)

[4.1. Présentation des modèles de cycle de vie 20](#_Toc115686979)

[4.2. Généralité sur le génie logiciel 20](#_Toc115686980)

[4.3. Notion de cycle de vie 20](#_Toc115686981)

[4.3.1. Présentations des différents modèles de cycle de vie. 21](#_Toc115686982)

[4.3.2. Choix du modèle de cycle de vie 28](#_Toc115686983)

[4.4. Présentation du langage UML 30](#_Toc115686984)

[4.4.1. Historique du langage UML 31](#_Toc115686985)

[4.4.2. Les avantages et inconvénients de l’UML 32](#_Toc115686986)

[4.4.3. Les grandes lignes essentielles 33](#_Toc115686987)

[Chapitre 5 : Analyse du projet 35](#_Toc115686988)

[5.1. Analyse des besoins 35](#_Toc115686989)

[5.1.1. Besoins de l’utilisateur 35](#_Toc115686990)

[5.1.2. Besoins fonctionnels 35](#_Toc115686991)

[5.1.3. Besoins non fonctionnels 36](#_Toc115686992)

[5.2. Etude de faisabilité 36](#_Toc115686993)

[5.3. Analyse de l’existant 36](#_Toc115686994)

[5.3.1. Matériels existants 36](#_Toc115686995)

[5.4. Critique de l’existant 37](#_Toc115686996)

[5.5. Solutions proposées 37](#_Toc115686997)

[5.6. Solution retenue 37](#_Toc115686998)

[Chapitre 6 : Conception du projet 37](#_Toc115686999)

[6.1. Conception d’un logiciel 37](#_Toc115687000)

[6.1.1. Présentation de l’outil de conception 37](#_Toc115687001)

[6.2. Dictionnaire de donnée 38](#_Toc115687002)

[6.3. Règle de gestion 39](#_Toc115687003)

[6.4. Modélisation 39](#_Toc115687004)

[6.4.1. Les diagrammes UML 39](#_Toc115687005)

[6.4.2. Description des diagrammes UML du projet 50](#_Toc115687006)

[Chapitre 7 : Spécification des outils de réalisation 61](#_Toc115687007)

[7.1. Choix technique 61](#_Toc115687008)

[7.1.1. Java spring 61](#_Toc115687009)

[7.1.2. Angular 62](#_Toc115687010)

[7.1.3. GitHub 63](#_Toc115687011)

INTRODUCTION GENERAL

La révolution informatique change l’image de la société. C’est aujourd’hui que naît le monde de demain, où l’ordinateur sera notre compagnon quotidien. L’ordinateur a trouvé aujourd’hui sa place à la maison et dans des bureaux.

Face à l’allure qu’elle prend actuellement, l’informatique est considéré comme un vecteur principal du développement de toutes les activités.

La meilleure solution pour s’échapper à ce dit embarras est donc l’informatisation des traitements des dossiers. Non seulement la rapidité qu’elle nous enchère, elle nous permet aussi la facilité de manipulation des données et nous exclut de l’erreur.

Justement nous avons eu la chance d’y avoir effectué un stage auprès de la douane Malagache en vue d’élargir les connaissances acquises au cours de quelques apprentissages.

En effet, la Direction de Ressource Humaine de la douane Malgache avait l’ambition d’informatiser la procédure de gestion d’information de la douane Malgache afin d’exercer un suivie des agents

Alors le travail consiste d’une part à la Conception et d’autre part à la réalisation d’une application de gestion d’information des agents de la douane malagasy.

Afin de mieux décortiquer le sujet, cet ouvrage se subdivise en 3 grandes parties :

* La première partie se rapporte à une présentation générale ;
* La deuxième partie consiste à une analyse et conception des données ;
* La troisième partie est la mise en œuvre de l’application au moyen du choix technologique et outils utilisés pour créer l’application.

PARTIE 1 : PRESENTATION GENERALE

# Chapitre 1 Présentation de l’EMIT

## 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 Professionnalisant (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 1.1 :

Tableau 1.2‑1 : Tableau récapitulatif

|  |  |  |  |
| --- | --- | --- | --- |
| Cycle | Mention Management | Mention Informatique | Mention Relation Publique 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 |

## Objectif de l’école

L’école a pour mission de :

* Former des techniciens supérieurs spécialisés en « Développement d’application Intranet Internet », en « Administration Economique et Sociale » et en « Relations Publiques et Multimédia » et des étudiants intellectifs sur les domaines « Système d’Information Géomatique et Décision » et « Management Décisionnel » opérationnel immédiatement dans les entreprises.
* Assurer le Perfectionnement professionnel des étudiants, des demandeurs d’emplois, des employés et des cadres d’entreprises.
* Assurer des Prestations de services.

## Formations existantes

L’école présente actuellement deux cycles : Licence et Master. Chaque cycle possède plusieurs parcours assurés par un chef de mention. La figure 1.1 offre une vue d’ensemble sur les formations dispensées au sein de l’EMIT.

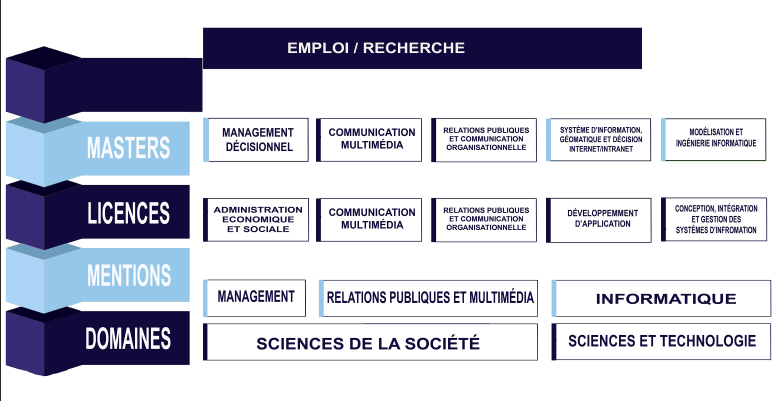


Figure 1.4‑1: Les offres de formations de l'EMIT

### Cycle de Licence

L’école possède trois (03) mentions pour cinq (05) parcours en cycle Licence. Les étudiants toute mention confondue 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.

#### 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.

#### 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 Licences 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.

#### 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.

#### Modalités de recrutements en L1

A chaque année, l’école effectue des recrutements des étudiants en Première Année de Licence Professionnelle. Les recrutements se font par le test de niveau. Les dates d’ouverture et de clôture des dossiers des candidatures sont fixées par le conseil d’établissement ainsi que les pièces à fournir.

Le dossier d’inscription doit parvenir à MONSIEUR LE DIRECTEUR DE L’ECOLE DE MANAGEMENT ET D’INNOVATION TECHNOLOGIQUE-BP 1135-301 Fianarantsoa.

Le test de niveau se déroulera dans les centres d’examen qui sont fixés par le Conseil d’établissement ainsi que les épreuves à composer.

### 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.

#### 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.

#### 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 représente un défi pour le génie logiciel qui doit fournir de nouvelles méthodes et des outils de production de logiciel pour la description de l’architecture de ces systèmes complexes et 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.

## 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.

## Organigramme de l’EMIT

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.

La structure hiérarchique au sein de l'EMIT se compose d'un conseil (scientifique ou d'établissement), une direction, un collège des enseignants, des chefs de mentions et des services présents au sein de l'école. Cette structure est représentée de manière générale sur la figure 1.2.

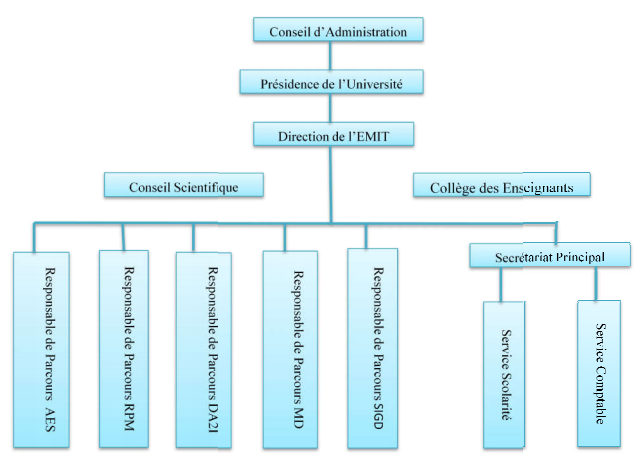


Figure 1.6‑1 : Organigramme de l'Emit

Source : EMIT

# Présentation de la douane

## Histoire de la douane malagasy

La douane malagache est une Administration bicentenaire, dont la naissance remonte à l’époque de la monarchie royale merina (groupe ethnique issu des « hauteurs » d’origine asiatique assez marqué, résident au centre de l’île) du début du 19ème siècle.

L’abolition de l’esclavage Anglo-merina du 23 octobre 1817 contribue à l’intensification du commerce extérieur sous le règne de Radama I et, par conséquent, à la mise en place de la douane pour alimenter les caisses de l’Etat monarchique. On assiste alors à la création des premières recettes des douanes sur le littoral est, ouest et sud. Des recettes qui auront plusieurs points communs :

1. Elles ont été confiées à des parents du roi
2. Elles ont été dirigées par des généraux (la douane était alors un service militarisé)
3. Elles ont pour principal objectif d’intensifier le commerce extérieur et de faire rentrer les devises.

## La Direction général de la douane

### Organigramme de la direction général de la douane

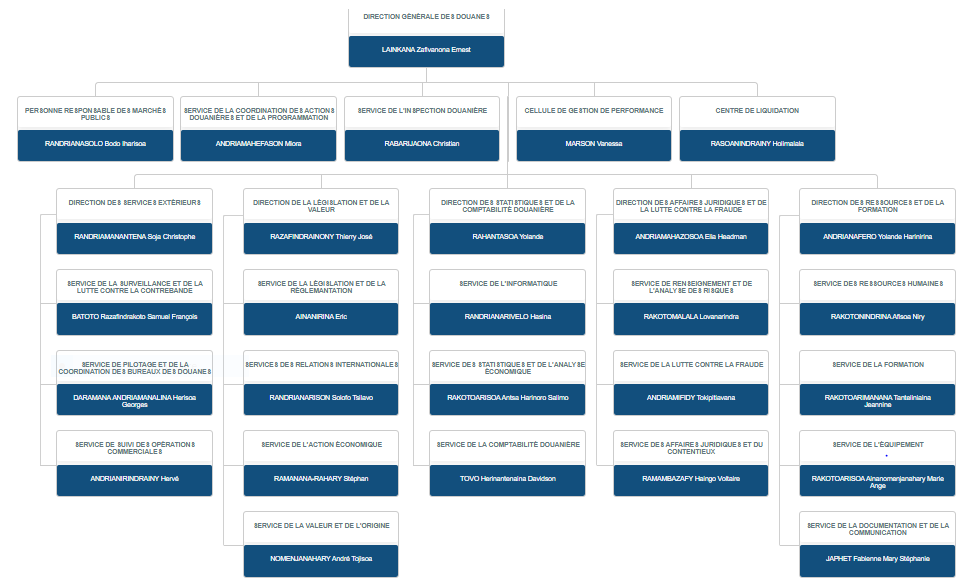


Figure 2.2‑1: Organigramme de la direction général de la douane

## Plan stratégique

Le nouveau Plan Stratégique 2020-2023, est bâti autour de la vision « **Douane, une administration innovante, redevable, partenaire pour l’émergence du pays »**

### Les axes

#### Axe 1 : Nous contribuons au financement du Plan Emergence de Madagascar.

* 1. : Optimisation de la collecte des recettes
  2. : Amélioration de la capacité de mobilisation des recettes douanières
  3. : Diminution significative des fraudes et irrégularités commises dans le cadre des régimes de suspension et d’exonération, des opérations douanières rigoureusement suivies
  4. : Renforcement de la collaboration entre le DGD et la DGI en vue de la mobilisation des recettes.

#### Axe 2 : Nous agissions en tant que Partenaire pour faciliter les échanges.

* 1. : Facilitation des échanges à travers une transparence et un partenariat accru
  2. : Favorisation, facilitation et promotion du civisme fiscal

#### Axe 3 : Nous contribuons à la sécurité et à la sûreté du pays, de son environnement et de sa population

3.1 : Contribution à la sécurité et à la sûreté du pays à travers une gestion coordonnée des frontières

3.2 : Amélioration de la capacité de la DGD à cibler et à réprimer la fraude organisée et la contrebande

#### Axe 4 : Nous investissons sur notre capital humain et notre organisation

* 1. : Mise en place des infrastructures et des équipements nécessaires pour fournir des services de qualité
  2. : Amélioration de l’image et de la transparence de l’administration douanière malgache
  3. : Promotion de l’éthique et lutte efficace contre la corruption afin de diminuer les atteintes à la déontologie professionnelle
  4. : Amélioration concrète significative constatée dans l’organisation et le déploiement des ressources par la DGD pour exécuter ses missions
  5. : Renforcement des capacités en matière de gestion stratégique et de gestion du changement
  6. : Etablissement de la douane en tant que centre d’excellence pour la recherche et l’innovation en matière douanière aux niveaux national et régional

### Alignement stratégique et modalités de mise en œuvre

Le monde change, notre douane s’adapte continuellement et de manière dynamique au nouvel environnement.

#### Un plan Stratégique pour contribuer à l’essor du pays aux niveaux international, régional et national

Sur le plan international, le plan stratégique 2020-2023 de la DGD s’aligne avec les Objectifs de Développement Durable de par ses objectifs de promotion d’une croissance économique durable, de préservation de la biodiversité terrestre, de promotion de l’avènement d’une société pacifique et de l’accès à une institution responsable. En outre, il concourt à la mise en œuvre de l’agenda 2063 de la Commission de l’Union africaine en misant sur la croissance inclusive, le développement durable, l’intégration régionale et l’instauration de la bonne gouvernance.

#### Une stratégie que sera exécutée selon les meilleures pratiques

Dans une logique de conduite du changement, la nouvelle stratégie a été développée en partenariat avec les parties prenantes internes et externes à la douane et en consultation avec les partenaires techniques et financiers, notamment le Fonds Monétaire International (Africa South). Aussi, des actions de conduite du changement ont été intégrées au plan stratégique pour faciliter l’adhésion des agents et usagers aux transformations induites par la nouvelle stratégie organisationnelle.

#### Programme de modernisation

Afin de mettre en œuvre notre Plan Stratégique et de répondre aux défis et opportunités d’un monde que change, nous avons développé un programme de modernisation ambitieux mais réaliste qui nous permettra d’atteindre nos objectifs stratégiques, et qui comprend de nombreux projets dont dix (10) considérés comme prioritaires.

# Présentation du projet

## Origine du projet

Afin de bien gérer tous les informations sur les agents, étant donné le nombre d’agents et le nombre de documents de chaque agent des douanes à gérer. Alors, le responsable de la ressource humaine fait appelle au service informatique de la douane pour créer une application adéquate pour gérer les informations, notamment les affectations, les avancements, les formations, les sanctions, les congés et les retraites des agents.

Non seulement ce projet permettra de faciliter le contrôle et le suivi de l’agent, mais il permettra aussi la rapidité du travail du responsable des agents.

## Objectif du projet

La réalisation de ce projet a pour but

* De créer une archive numérique pour stocker toutes les données concernant l’agent comme l’historique de leur affectation, leur avancement, leur retraite et leur congé.
* D’afficher les informations de façons complètes et en détaille de chaque agent.
* Car chaque activité a son propre responsable alors notre objectif est de mettre une interface spécialisée par chaque responsable pour interagir avec une seule base de données, et de partager les informations.
* De faciliter les recherches des informations de chaque agent celons leur attribut.
* De mettre en place un affichage dynamique qui permet de bien suivre les mouvements des agents en cas d’affectation, d’avancement, de retraite et de congé.

## Problématique

Les problèmes qui ont motivé la réalisation de ce projet :

* Les données sont stockées dans des fichiers Excel. Le responsable de Ressource Humaine n’a pas de logiciel spécifique.
* Difficulté de partage de l’information : car les tâche sont effectués par des différents responsable alors sous l’Excel on ne peut pas partager le contenu avec les autres responsables ; même lecture sur un réseau,
* Une absence de sécurité : Il est certes possible d’ajouter un code dans le tableur Excel, mais il est difficile de sécuriser les données qui circulent.
* Affichage des informations incomplètes : car les données sont séparées par plusieurs fichier Excel on ne peut pas donc simplement afficher toutes les informations par chaque agent.

## Conduites de projet

### Analyse des besoins

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

### Bilan critique de l’existant

#### Bilan critique de l’existant

Le responsable du service de la ressource humaine n’a pas encore utilisé de logiciel approprié pour le contrôle de ses services sauf le Microsoft Excel.

Le Microsoft Excel ne peut pas accomplir tous les besoins du responsable de la ressource humaine.

On ne peut pas donc afficher respectivement toutes les informations concernant les agents, leur affectation et leur avancement avec l’application qui est déjà utilisé.

On ne peut pas partager et actualiser automatiquement les données entre les responsables

#### Bilan des besoins

Pour que l’utilisateur soit satisfait, il faut qu’il soit persuadé que le produit va lui permettre de faciliter ses activités. Ainsi, le service de la ressource humaine au niveau de la Direction des Ressources et de la Formation souhaite d’avoir une application de contrôle, permettant d’enregistrer les informations concernant les agents depuis sont entré à la douane jusqu’à son départ et leur mouvement en cas d’affectation, d’avancement, de nomination, de retraite et de congé et de mettre aussi ces informations dans une base de données bien sécurisé.

### Etude de faisabilité

L’étude de la faisabilité concerne l’initialisation du projet en fonction des ressources du service pour réussir la création de l’application. Concernant le développement de notre application, l’utilisation des outils informatiques sont importants

#### Etude de faisabilité technique

Tout d’abord, le projet est techniquement réalisable à 99%. L’utilisation de la technologie est facile, parce que Service Informatique de la douane a les ressources nécessaires à la réalisation de ce projet.

#### Etude de faisabilité organisationnelle

La douane dispose une organisation appropriée pour la réalisation. C’est le Service Informatique qui organise le plan du développement et gère toutes les taches jusqu’à la fin du projet.

### Etude d’opportunité

Cette étape d’avant-projet permet d’étudier la demande de projet et de décider si le concept est viable. Cette étape a pour enjeu de valider les besoins des utilisateurs par rapport aux objectifs.

Lors de la phase d’opportunité, les besoins généraux du maitre d’ouvrage doivent être identifiés. Il est nécessaire de s’assurer que ces besoins correspondent à une attente de l’ensemble des utilisateurs cibles et qu’ils prennent en compte les évolutions probables des besoins.

Dans l’étude d’opportunité aussi on parle des avantages attendus par le projet et ici les avantages attendus par ce projet sont les suivants :

* Information facile à consulter dans la mesure où l’information est affichée de façon complète et détaillée.
* Permet aux utilisateurs de partager et d’actualiser plus facilement les données.

## Cahier des charges

### Les trois volets encadrants le travail

Le cahier des charges comporte trois volets encadrants et orientant le travail : les recommandations portant sur l’interface graphique, les fonctionnalités attendues et les spécifications concernant l’architecture de l’application.

#### L’architecture de l’application

* Seul un responsable protégé par login et mot de passe peut ajouter, rechercher, vérifier, modifier ou supprimer des données sur l’application
* Le temps de réponse de l’application sera rapide.
* Cette application sera toujours disponible à utiliser sur les années à venir.
* Cette application sera très simple à utiliser pour les utilisateurs.
* Cette application sera compatible avec tous les versions de système d’exploitation

#### Fonctionnalités

* L’ensemble des fonctionnalités standard seront disponible (déplacement, interrogation d’objet, module de recherche, impression des données, affichage à échelle souhaitées).
* Possibilité d’effectuer des requêtes par attribut.
* La navigation et l’utilisation des fonctionnalités seront intuitives et améliorées ;

#### Interface graphique

* L’interface graphique sera simple, conviviale et ergonomique.

### Les cible

Notre application permettra de stocker toutes les données dans une base bien sécurisé et de partager les informations entre les responsables.

#### Structuration technique générale

L’idée de structuration technique est de proposer d’avancer en différentes briques ou modules de développement pour progresser pas à pas dans l’élaboration des fonctionnalités attendues pour ces outils numériques finaux.

### Le délais et planification de la réalisation du développement du projet

La réalisation de ce projet a une durée de trois (03) mois. La liste des activités de notre projet est reprise dans le tableau ci-dessous.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code Activité** | **Libellé Activité** | **Activité antérieur** | **Durée/jour** | **Activité successeur** |
| a | Enquête et recherche d’information | --- | 21 | b |
| b | Conception du système d’information | a | 16 | c |
| c | Développement | b | 25 | d |
| d | Teste | c | 7 | e |
| e | Formation | d | 5 | Fin |

Tableau 3.5‑1: Planification du projet

#### Présentation de la méthode Gant

Le diagramme de Gantt, couramment utilisé en gestion de projet, est l’un des outils les plus efficaces pour représenter visuellement l’état d’avancement des différentes activités qui constituent un projet.

#### Construction du diagramme

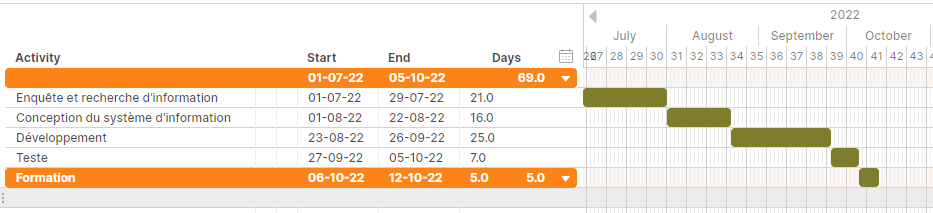


Figure 3.5‑1: diagramme de gantt

PARTIE 2 :   
Analyse et conception du projet

# Méthodes et notations utilisées

## Présentation des modèles de cycle de vie

Par définition, un cycle de vie logiciel est une représentation d’un processus depuis sa création, puis sa distribution jusqu’à sa disparition, cette représentation peut stipuler des tâches, des contraintes, des ressources ou une manière de réalisation. Chacun des modèles de cycle de vie du développement du logiciel n’est pas identiques, mais ils ont une idée fondamentale : la discrétion du développement du logiciel. Chacun ayant un résultat défini et un critère de terminaison défini, et chaque phase est achevée avant que la phase suivante commence. Le modèle de cycle de développement est éventuellement formé par les phases suivantes : analyse, conception, implémentation, test et installation.

## Généralité sur le génie logiciel

C’est *« l’ensemble des méthodes, des techniques et outils concourant à la production d’un logiciel, au-delà de la seule activité de programmation »* [Université Nongo Conarky]. Le génie logiciel est un domaine de recherche qui a été défini du 07 au 11 octobre 1968, à Garmisch-Partenkirchen, sous le parrainage de l’OTAN. Il a pour objectif de répondre à un problème qui s’énonçait en deux constatations : d’une part le logiciel n’était pas fiable, d’autre part, il était incroyablement difficile de réaliser dans des délais prévus des logiciels satisfaisants leur cahier des charges. Le génie logiciel est « l’application pratique de la connaissance scientifique dans la conception et l’élaboration de programmes informatiques et de la documentation associée nécessaire pour les développer, les mettre en œuvre et les maintenir » [Barry W., 1988]. Le génie logiciel consiste à appliquer des méthodologies, c’est-à-dire à développer et à utiliser des méthodes et des outils dans le but de produire des logiciels de qualité en respectant des contraintes de temps et de coût.

## Notion de cycle de vie

Le cycle de vie est la description d'un processus couvrant les phases de création d'un produit, sa distribution sur un marché, et sa disparition. Le but de ce découpage est de maîtriser les risques, maîtriser au mieux les délais et les coûts, obtenir une qualité conforme aux exigences. L’origine de ce découpage provient du constat que les erreurs ont un coût d’autant plus élevé qu’elles sont détectées tardivement dans le processus de réalisation. Le cycle de vie permet de détecter les erreurs au plus tôt et ainsi de maîtriser la qualité du logiciel, les délais de sa réalisation et les coûts associés. On distingue deux types de cycle de vie : le cycle de vie des produits s'applique à tous les types de produits, et peut être considéré comme un outil de gestion ; et le cycle de développement des logiciels s'insère dans le précédent, on l'appelle souvent abusivement cycle de vie des logiciels [Anne-Marie H, 2002].

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

Un modèle est une représentation abstraite et simplifiée de la réalité en vue de la décrire. Le modèle exclut certains détails de la réalité qui n'influent pas de manière significative sur le phénomène étudié. Modéliser un système revient donc à représenter de façon abstraite ce que le concepteur du système croit important pour la compréhension du phénomène réel étudié [Djallel B.]. La plupart du temps, les logiciels sont réalisés en équipe. La modélisation permet de répartir  
efficacement les tâches entre les membres de l'équipe et même d'automatiser certaines de ces tâches. Un modèle permet d'assurer la bonne compréhension du fonctionnement du système et de maîtriser la complexité de celui-ci. Par ailleurs les coûts, les délais et la qualité du logiciel sont maîtrisés.

Le cycle de vie du logiciel comprend généralement au minimum les onze (11) étapes suivantes :

* La définition des objectifs qui consiste à définir la finalité du projet et son inscription dans une stratégie globale ;
* L’analyse des besoins et faisabilité qui est l’expression, le recueil et la formalisation des besoins du demandeur et de l’ensemble des contraintes ;
* L’estimation de la faisabilité de ces besoins ;
* La spécification ou conception générale qui s’agit de l’élaboration des spécifications de l’architecture générale du logiciel ;
* La conception détaillée consiste à définir précisément chaque sous-ensemble du logiciel ;
* Le codage ou l’implémentation ou programmation d’où la traduction dans un langage de programmation des fonctionnalités définies lors de la phase de conception ;
* Les tests unitaires permettent de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux spécifications ;
* L’intégration d’où son objectif est de s’assurer de l’interfaçage des différents éléments du logiciel, elle fait l’objet de tests d’intégration consignés dans un document ;
* La qualification ou recette : C’est-à-dire la vérification de la conformité du logiciel aux spécifications initiales ;
* La documentation vise à produire les informations nécessaires pour l’utilisation du logiciel et pour des développements ultérieurs ;
* La mise en production qui est le déploiement sur site du logiciel ;
* La maintenance qui comprend toutes les actions correctives (maintenance corrective) et évolutives (maintenance évolutive) sur le logiciel.

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

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

#### Le modèle de cycle de vie en cascade

Le modèle de cycle de vie en cascade a été mis au point dès 1966, puis formalisé aux alentours de 1970. Il définit des phases séquentielles à l'issue de chacune desquelles des documents sont produits pour en vérifier la conformité avant de passer à la suivante [Jean François P., 2005].

À la fin de chaque étape, des documents ou alors des logiciels sont produits. Ces résultats sont revisités de façon approfondie et on ne passe à la phase suivante que si ces résultats sont jugés satisfaisants. Après quelques modifications du modèle en cascade original, les possibilités de retour arrière ont été ajoutées. Seulement une phase ne peut remettre en cause que la phase précédente ce qui est insuffisant du point de vue pratique. Ce modèle repose sur l'hypothèse souvent idéaliste que l'on peut dès le départ, fixer en détail et complètement ce que l'on veut réaliser. Un autre inconvénient de ce modèle est que la vérification du bon fonctionnement du système s'effectue très tardivement.

**Principes**

* Processus Linéaire
* On convient d’un certain nombre d’étapes, se terminant a des dates précises par la production de document ou logiciels.
* Les résultats de l’étape sont examinés attentivement avant de passer ò l’étape suivante.
* Itération : une étape ne remet en cause que l’étape précédente (validation, vérification).

**Inconvénients**

* Validation limitée à un pas d’itération
* Augmentation des risques : erreur d’analyse ou de conception très couteuse si détecté trop tard.
* Difficile d’effectuer de modification encours de route
* Solution limitée aux petits projets
* Bien adapté lorsque les besoins sont clairement identifiés et stables
* Projet court avec peu de participants.

#### Le modèle de cycle de vie en v

Le modèle de cycle de vie en V a été imaginé pour pallier le problème de réactivité du modèle en cascade. Ce modèle est une amélioration du modèle en cascade qui permet en cas d’anomalie, de limiter un retour aux étapes précédentes.

Il est représenté par un V dont la branche descendante contient toutes les étapes de la conception du projet, et la branche montante, toutes les étapes de test du projet. La pointe du V représente la réalisation concrète du projet, le codage. On pourrait donc en déduire, de manière simple, que les deux banches du V n’est que de la documentation[b].

Le modèle en V est actuellement le cycle de vie le plus connu, et certainement le plus utilisé. Ce modèle consiste à reposer sur une interdépendance des étapes soumises à une validation avant la prochaine étape et une vérification anticipatoire du produit final.

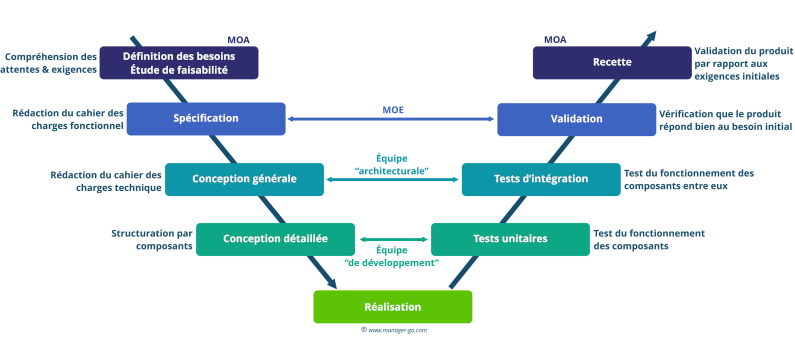


Figure 4.3‑1: Cycle de vie en v

**Intérêt**

* Validation intermédiaire :
* Prévention des erreurs : validation de produits à chaque sortie d’étape descendante.
* Préparation des protocoles de validation finaux à chaque étape descendante
* Validation finale montante : confirmation de la pertinence de l’analyse descendante.
* Limitation des risques en cascade par validation de chaque étape.
* Existence d’outils support.
* Modèles éprouvés très utilisé pour de grands projets

**Limitation**

* Un modèle toujours séquentiel…
* Prédominance de la documentation sur l’intégration : validation tardive du système par lui-même
* Les validation intermédiaires n‘empêchent pas la transmission des insuffisances des étapes précédentes.
* Manque d’adaptabilité.
* Maintenance non intégrée : syndrome du logiciel jetable.
* Adaptées aux problèmes sans zones d’ombres.
* Idéal quand les besoins sont bien connus, quand l’analyse et la conception sont claires.

#### Le modèle de cycle de vie en spirale

Le modèle en spirale est un modèle de cycle de développement logiciel qui reprend les différentes étapes du cycle en V. Par l’implémentation de version successive, le cycle recommence en proposant un produit de plus en plus complet et dur. Le cycle en spirale met cependant plus l’accent sur la gestion des risques que le cycle en V [Djallel B.].

Le modèle en spirale est un générateur de modèle qui se base sur le risque pour des projets. Le modèle guide une équipe à adopter les éléments d’un ou plusieurs modèles de processus.

On distingue quatre (04) phases dans le déroulement du cycle :

* Détermination des objectifs, des alternatives, des contraintes ;
* Analyse des risques, évaluations des alternatives ;
* Développement et vérification de la solution retenue ;
* Revue des résultats et vérification du cycle suivant.

Le processus de développement est représenté par une spirale plutôt qu’une séquence d’activités avec retour arrière éventuels. Chaque boucle dans la spirale représente une étape du processus de développement. Les risques sont explicitement adressés et résolu tout au long du processus.

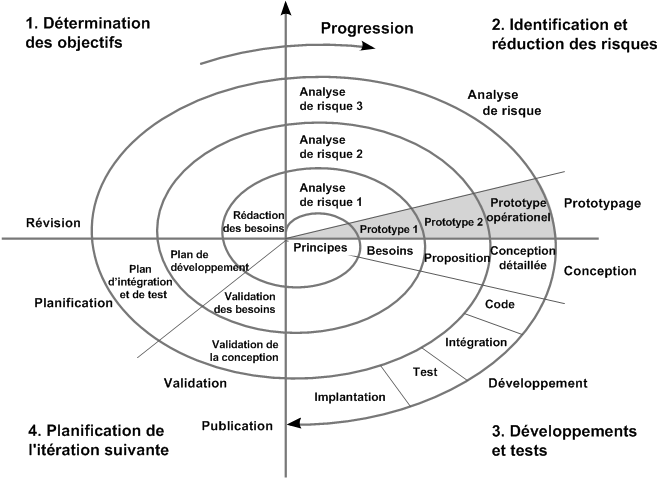


Figure 4.3‑2:cycle de vie en spirale

**Formulaire d’un tour de spirale**

* Objectif
* Contraintes
* Alternatives
* Risques
* Résolution des risques
* Résultats
* Plans
* Implication

**Intérêt**

* Validation concrète et non sur document
* Limitation du risque à chaque itération
* Concentre l’attention sur les options de réutilisation
* Client partenaire : retour rapide sur attente.
* Progressivité : pas d’explosion des besoins à l’approche de la livraison.
* Flexibilité :
* Modification des spécifications = nouvelle itération
* Maintenance = forme d’itération.
* Planification renforcée.

**Limitations**

* Ce n’est pas le processus parfait
* Cycle itératif : planification très attentive et rigoureuse, mais peut dérouter au premier abord.
* Cycle en « V » : Processus éprouvé le plus rependu, surtout pour le système connus.
* Nécessite des expertises sur l’évaluation des risques.
* Processus adapté à la modélisation objet :
* Modèle objet : se prête parfaitement à une démarche incrémentale.

#### Le modèle de cycle de vie par incrément

Face aux dérives « bureaucratiques » de certains gros développements, et à l’impossibilité de procéder de manière linéaire

Le développement incrémental est une programmation et une stratégie par relais dans laquelle les différentes parties du système sont développés à des différents moments et intégrés lorsqu’elles sont terminées [Processus cyclique].

Le modèle de cycle de vie par incrément consiste développer à la fois un seul ensemble de composant, contrairement aux modèles précédents. Il s’agit d’une intégration des incréments à un noyau de logiciel développé au préalable. Ce modèle a pour principe de diviser le projet en incréments (sous-partie fonctionnelle cohérente du produit final), chaque incrément ajoute de nouvelles fonctions et est testé comme un produit final. Ce modèle a pour avantages de moins complexités de chaque développement, des progressivités des intégrations, de la possibilité de livraison et de mise en service pour chaque incrément. Les risques de ce type de modèle sont la remise en cause des incréments précédents ou le noyau, et le non intégration de nouveaux incréments.

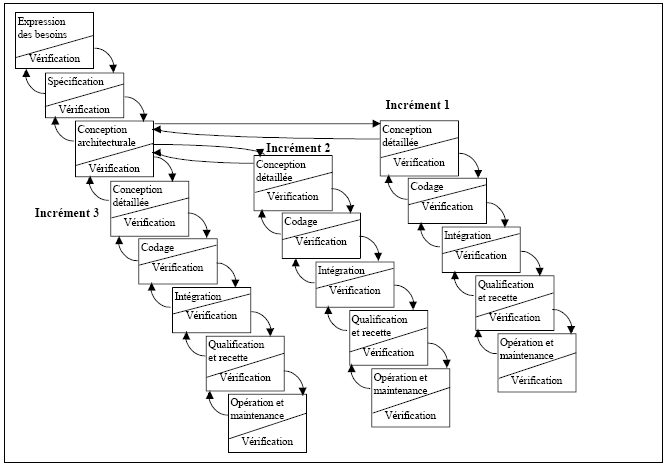


Figure 4.3‑3:Cycle de vie par incrément

### Choix du modèle de cycle de vie

Chacun des modèles présentés ci-dessus a ses avantages et ses inconvénients, c’est-à-dire qu’il y a des différences entre ces types de modèle. En comparant ces différents modèles, nous avons fait un choix pour mieux adapter.

| **Modèle de cycle de vie** | **Avantages** | **Inconvénients** |
| --- | --- | --- |
| En cascade | * Facile à comprendre et à utiliser ; * Adapté pour une équipe inexpérimentée ; * Les limites de chaque étape sont visibles ; * La définition des besoins est non évolutive. | * Tous les besoins doivent être bien définis au départ ; * Pas d’interaction entre les phases de développement ; * L’intégration n’a lieu qu’à la fin du cycle ; * Le client peut se retrouver non satisfait ; * Pas de retour en arrière d’une étape à l’autre. |
| En V | * Facile à utiliser ; * Les tests sont effectués à chaque étape ; * Le contrôle se fait progressivement à chaque étape ; * Favorise la décomposition hiérarchique fonctionnelle ; * Les phases de validation sont prises en main très tôt dans le processus de développement. | * Une mauvaise prise en compte des évènements concurrents ; * Le processus n’est pas itératif ; * Une mauvaise prise en compte des changements de la spécification des besoins ; * Ne contient pas des activités d’analyse de risque. |
| En spirale | * Les fonctions critiques à haut risques sont développées en premier lieu ; * Validation concrète et non sur documents ; * Limitation du risque à chaque itération ; * La conception ne doit pas forcement terminée ; * Les utilisateurs finaux sont intimement associés à toutes les étapes de développement ; * Le développement se fait avec interaction avec les clients ; * Les utilisateurs ont dès le départ une vue globale du système. | * Requiert la définition complète des fonctionnalités du système pour une définition des différents incréments ; * Le temps consacré à l’évaluation des risques est très élevé pour des petits projets ; * Ce modèle est complexe ; * Une expertise en évaluation des risques est nécessaire ; * La spirale peut être infinie ; * Il est difficile de définir les objectifs et les points de validation intermédiaire entre les différentes étapes. |
| Par incrément | * Le client peut valider chaque étape de processus ; * Utilisation de la méthode « diviser pour régner » r ; * Le coût du lancement est moindre ; * Un produit exploitable peut être délivré à tout moment ; * Le risque du changement des besoins est minimal. | * Requiert une bonne conception ; * Le coût total du développement du système n’est pas négligeable ; * Les différentes interfaces doivent être bien définies. |

Figure 4.3‑4. Choix du modèle de cycle de vie

Chaque modèle de cycle de vie a ses propres points forts et ses ponts faibles, illustrés dans le tableau ci-dessus. A partir de ce tableau comparatif, on peut choisir à utiliser le modèle de cycle de vie en V, pour les raisons suivantes : ce modèle est un modèle éprouvé par calque sur la production industrielle classique, il permet l’organisation du travail et des équipes, favorise la décomposition hiérarchique fonctionnelle, il propose des étapes clés (documentation, revues) qui entraine un bon suivi du projet, il permet de garantir une certaine de qualité, et enfin il est adapté à de grand projet.

Le choix revient au fait que ce cycle de vie est le plus efficace avec son principe de travail qui nécessite la vérification de chaque étape et la possibilité de corriger les fautes avant de se lancer vers l’étape suivante.

## Présentation du langage UML

UML (Unified Modeling Language), littéralement « Langage de Modélisation Unifié » est un langage de modélisation graphique et textuel. Il ne s'agit pas d'une simple notation graphique, car les concepts transmis par un diagramme ont une sémantique précise et sont porteurs d'un sens au même titre que les mots d'un langage.

UML n’est pas à l’origine des concepts objets, mais il en donne une définition plus formelle et apporte la dimension méthodologique qui faisait défaut à l’approche objet. Maitriser la notation graphique d’UML n’est pas une fin en soi. Ce qui est primordial, c’est d’utiliser les concepts et d’appliquer la démarche d’analyse correspondante [Olivier B., 2004].

UML se définit comme un langage de modélisation graphique et textuelle destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue [Pascal R., Franck V., 2003].

UML a été construit afin de standardiser les artéfacts de développement (modèles, notations, diagrammes), sans standardiser le processus de développement.

Nous avons proposé d’utiliser le langage UML, pour la modélisation de notre projet, qui sert à représenter abstraitement le monde réel.

### Historique du langage UML

Né de la fusion des méthodes objets dominantes (OMT, Booch et OOSE), puis normalisé par l’OMG en 1997, UML ou « Unified Modeling Language » est rapidement devenu un standard incontournable. Passons à une courte histoire de ces trois (03) méthodes dominantes :

* **Objet Modeling Technique (OMT) :** il a été développé par James RUMBAUGH dans le centre de Recherche et Développement de la société Général Electric, à la fin des années 80. Cette méthode connaît un certain succès en France et en Europe. OMT propose un triple de perception du système d’information : une dimension statique qui décrit le schéma objet du système, une dimension dynamique qui décrit le changement d’états des objets en fonction d’évènements, une dimension fonctionnelle qui décrit les processus de transformation des informations.
* **Object Oriented Development (OOD) :** Grady BOOCH publie en 1981 un article intitulé « Object Oriented Development » qui présente une méthode de modélisation objet. Cette méthode applique le modèle objet pour synthétiser la structure logique, physique, statique et dynamique d’une application. Pour cela, elle utilise une notion expressive et clairement définie permettant de : capturer des points de vue différents. La méthode Booch2 propose des diagrammes de classe, diagramme d’instancié et diagramme d’état-transitions. Le niveau physique contient les diagrammes de modules et diagramme de processus.
* **Object Oriented Software Engineering (OOSE) :** la méthode OOSE de Ivar JACOBSON couvre tout le cycle de développement du logiciel, crée dans un centre de recherche d’Ericson (Suède), elle repose essentiellement sur l’analyse des besoins des utilisateurs à l’aide des diagrammes de cas d’utilisation formalisés par Ivar JACOBSON dans sa méthode puis introduits dans UML. Ils décrivent, sous la forme d’actions et de réactions, le comportement d’un système du point de vue de l’utilisateur, définissent les limites du système et les relations entre le système et l’environnement. Ils permettent d’introduire une manière spécifique d’utiliser un système. C’est l’image d’une fonctionnalité du système, déclenchée en réponse à la stimulation d’un acteur externe.

### Les avantages et inconvénients de l’UML

Ce langage de modélisation n’est pas parfaitement satisfaisant complètement, il possède ses points faibles. Tandis qu’il adopte aussi ses points forts

|  |  |
| --- | --- |
| **Avantages** | **Inconvénients** |
| * Langage formel et normalisé ; * Permet un gain de précision, et de stabilité ; * Assure la pérennité ; * Facilite l’utilisation d’outils ; * Facilite la communication ; * Propose un cadre d’analyse ; * Permet la représentation d’éléments abstraits et complexes ; * Support de communication performant ; * Permet de bien définir les besoins clients, et ainsi d’éviter des surcoûts liés à la livraison d’un logiciel qui ne satisfait pas le client (Selon une étude du Standish Group en 1994, pour 53% des logiciels crées, le taux de délai de livraison non respecté est de 120%, on avait 90 % de budgets non tenus, et 60% de non disponibilité de certaines fonctionnalités) ; * Apporte une compréhension rapide du programme à d’autres développeurs externes en cas de reprise du logiciel et facilite sa maintenance. * Vulgarise les aspects liés à la conception et à l’architecture, propres au logiciel, au client. | * Son utilisation nécessite une formation préalable pour connaître ses normes standards (pour le client qui n’a pas de compétences dans ce domaine) ; * Il peut y avoir une mauvaise correspondance entre l’UML et le projet finalisé ;  |  | | --- | | * Pour fonctionner correctement, les diagrammes UML doivent synchronisés avec le code du logiciel qui nécessite du temps pour mettre en place et à entretenir, et ajoute du travail à un projet de développement logiciel. Les petites entreprises et les développeurs indépendants pourraient ne pas être en mesure de gérer la quantité supplémentaire de travail nécessaire pour synchroniser le code ; * UML n’est pas une méthode ; * Il nécessite un apprentissage et de l’expérience. | |

Figure 4.4‑1 Les avantages et inconvénients de l’UML

### Les grandes lignes essentielles

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

UML 2.0 comporte treize types de diagrammes pour représenter des concepts particuliers du système d’information. Ils sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie [Laurent A., 2007-2008]. Ils se répartissent en deux grands groupes :

Les diagrammes structurels ou diagrammes statiques rassemblent cinq diagrammes :

* Le diagramme de classes qui représente les [classes](http://fr.wikipedia.org/wiki/Classe_%28informatique%29) intervenant dans le système ;
* Le diagramme d’objets qui sert à représenter les instances de classes (objets) utilisées dans le système ;
* Le diagramme de composants qui permet de montrer les composants du système d'un point de vue physique, tels qu'ils sont mis en œuvre ([fichiers,](http://fr.wikipedia.org/wiki/Fichier_%28informatique%29) [bibliothèques,](http://fr.wikipedia.org/wiki/Biblioth%C3%A8que_logicielle) [bases de données,](http://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es) …) ;
* Le diagramme de déploiement qui sert à représenter les éléments matériels [(ordinateurs,](http://fr.wikipedia.org/wiki/Ordinateur) [périphériques,](http://fr.wikipedia.org/wiki/P%C3%A9riph%C3%A9rique_%28informatique%29) [réseaux,](http://fr.wikipedia.org/wiki/R%C3%A9seau_informatique) systèmes de stockage, …) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux ;
* Le diagramme de structures composites qui permet de décrire sous forme de [boîte blanche](http://fr.wikipedia.org/wiki/Bo%C3%AEte_blanche) les relations entre composants d'une classe.

Les diagrammes comportementaux ou diagrammes dynamiques rassemblent huit diagrammes :

* Le diagramme de cas d’utilisation qui permet d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système ;
* Le diagramme d’activités qui permet de décrire sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants ;
* Le diagramme d’états-transitions qui permet de décrire sous forme de machine à états finis le comportement du système ou de ses composants ;
* Le diagramme d’interaction qui permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences ;
* Le diagramme de séquence : représentation séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs ;
* Le diagramme de communication : représentation simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets ;
* Le diagramme global d’interaction qui permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences ;
* Le diagramme de temps qui permet de décrire les variations d'une donnée au cours du temps.

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

# Analyse du projet

Quel que soit le type de projet que vous pouvez être amené à gérer, il est indispensable d’être méthodique et rigoureux, sous peine de rapidement perdre la maîtrise des événements. Une absence de méthode est généralement synonyme d’échec. Un projet engage des moyens financiers, des personnes et du matériel, il ne peut y avoir de place pour l’improvisation. Toutes les méthodes ont deux points communs, une démarche de projet, et des outils. Dans ce billet, nous allons définir ce qu’englobe la démarche de projet et ce qu’elle peut vous apporter.

## Analyse des besoins

**L'analyse des besoins vise à identifier les exigences du projet et à faire le point sur les éléments** attendus. Il s'agit de contextualiser le projet et d'analyser les attentes pour **donner un cadre au projet.**

### Besoins de l’utilisateur

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

* Enregistrement des agents, leur affectation, leur avancement, leur retraite et leur congé dans une base de données.
* Partage des données, des informations entre les responsables ;
* Affichage de toutes les informations sur chaque agent, les affectations et les avancements.
* Savoir qui sont les agents devant être affectés.
* De savoir qui sont les agents, il faut augmenter le grade.

### Besoins fonctionnels

Les besoins fonctionnels sont les fonctionnalités de l’application qui va répondre aux besoins des utilisateurs :

* Recherche des informations selon un critère défini ;
* De gérer (ajouter, modifier, supprimer) les agents, les affectations et les avancements.
* Impression des informations des agents ;
* Graphe de données concernant les effectifs des affectations et les avancements par années.
* Liste complète des agents avec leurs affectations et leurs avancements.

### Besoins non fonctionnels

Les besoins non fonctionnels sont des besoins en matière de performance, de type matériel ou le type de conception.

* Dans cadre de ce travail, l’application devra être extensible, c’est-à-dire qu’il pourra y avoir une possibilité d’ajouter ou de modifier de nouvelles fonctionnalités. ;
* L’application devra être compatible avec n’importe quel système d’exploitation ;
* L’application devra être hautement sécurisée.

## Etude de faisabilité

L’étude de faisabilité concerne l’initialisation du projet en fonction des ressources du service qui devrait en disposition de l’application.

Concernant le développement de notre application, l’utilisation des outils informatiques suivant sont importants :

## Analyse de l’existant

L’étude de l’existant est basée sur le dialogue entre les responsables et l’interviewer. Les informations sont obtenues à l’aide des interviews sur la responsable de la ressource humaine et les membres du personnel des différents postes de travail. Pour cela, il faut que les dialogue entre les informaticiens et les utilisateurs décideurs soit dans le bon terme, afin de pouvoir aboutir à une proposition et/ou une suggestion dans l’analyse et l’étude des données recueillies. Donc l’étude de l’existant est le point de passage obligatoire qui matérialise le premier contact du concepteur avec les utilisateurs.

### Matériels existants

Les matériels informatiques existants dans le service de la ressource humaine sont :

## Critique de l’existant

## Solutions proposées

En prenant conscience de l’objectif principal de ce projet qui est l’amélioration du contrôle des agents, les affectations et les avancements, la solution la plus adéquate et praticable est l’informatisation de cette gestion.

* La première solution est d’acheter un logiciel pour gérer ce contrôle.
* La deuxième solution est de concevoir une application capable de faire toutes les tâches de cette gestion du personnel.

## Solution retenue

D’après la comparaison de ces deux solutions, la deuxième solution est la plus convenable pour le cas de la Direction de La Ressource Humaine parce que concevoir une application répondrait au mieux aux exigences et aux particularités de l’application souhaité par la DRH puisque la conception sera visée sur les règles de gestion tiré de l’interview effectué avec le responsable du Ressource Humaine

# Conception du projet

## Conception d’un logiciel

La conception du logiciel est la mise en œuvre d’un ensemble d’activités à partir d’une demande d’informatisation du système permettant la conception, l’écriture et la mise au point d’un logiciel jusqu’à sa livraison au maitre d’ouvrage.

### Présentation de l’outil de conception

Il existe plusieurs outils de conception pour concevoir un système d’information, mais nous avons adopté un outil de conception qui est le Visual Paradigm.

#### Le Visual Paradigm

Visual paradigm permet la création des diagrammes UML et des modèles qui en sont à l’origine. Ceux-ci peuvent alors générer du code dans un langage de programmation déterminé. Il propose également la création d’autres types de diagrammes, comme celui qui permet la modélisation des bases de données pouvant, lui aussi, générer des canevas d’applications basés sur des Framework et Pattern mais en plus, générer du code SQL qu’il peut ensuite déployer automatiquement dans différents environnements. Visual paradigm assure aussi la gestion des exigences, l’analyse d’impacts, qui permet de connaître à l’avance les conséquences d’un changement et la création de rapports automatisés. Il dispose des outils de paramétrage permettant de définir et de mettre en forme les types d’objets utilisés dans les modélisations.

## Dictionnaire de donnée

Avant tout, il convient de faire une synthèse des données décrites dans les messages, pour s’assurer qu’il n’y a aucun problème de signification des informations en suspens.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NOM | DESCRIPTION | TYPE | LONG | ETAT | FORMAT |
| Im | Immatriculation | N | 7 | E |  |
| Nom | Nom de l’agent | A | 40 | E |  |
| Prénom | Prénom de l’agent | A | 15 | E |  |
| Date\_nais | Date de naissance | D | 50 | E |  |
| Situation | Situation familiale | A | 15 | E |  |
| Lieu\_nais | Lieu de naissance | A | 25 | E |  |
| Ref\_aff | Refference affectation | AN | 15 | E |  |
| Gr\_indice | Grille indiciaire | A | 20 | E |  |
| Nouv\_poste | Nouveau poste | A | 20 | E |  |
| Design\_obj | Désignation objet | A | 50 | E |  |
| Nature\_obj | Nature de l’objet | A | 50 | E |  |
| Num\_obj | Numero de l’objet | N | 5 | E | JJ/MM/AAAA |
| Sigle\_obj | Sigle de l’objet | A | 50 | E |  |
| Date\_obj | Date de l’objet | D | 10 | E |  |
| Effet\_obj | Date d’effet | D | 10 | E |  |
| Ref\_ex\_aff | Ancienne référence d’affectation | AN | 15 | E |  |
| Logt | logement | A | 100 | E |  |
|  |  |  |  |  |  |

Tableau ‑: Dictionnaire de données

## Règle de gestion

C’est une règle pour bien appliquer la gestion et la création de la base de données.

Les règles sont listées ci-dessous :

RG1 : Un agent est caractérisé par son immatriculation, nom et son prénom.

RG2 : Tous les agents peut être affecté.

## Modélisation

### Les diagrammes UML

**Cas d’utilisation et description des séquences**

Avant de se lancer dans la réalisation d’un logiciel, il faut comprendre, clarifier et structurer les attentes et les besoins du client.

Le diagramme des cas d’utilisations constitue la première étape de l’analyse UML en :

Modélisant les besoins des utilisateurs

Identifiants les grandes fonctionnalités et les limites du système.

Représentant les interactions entre le système et ses utilisateurs.

Le diagramme de cas d’utilisation apporte une vision utilisateur et absolument pas une vision informatique. Il ne nécessite aucun connaissance informatique et l’idéal serait qu’il soit réalisé par le client.

Le diagramme des cas d’utilisation n’est pas un inventaire exhaustif de toutes les fonctions du système. Il ne liste que des fonctions générales essentielles et principales sans rentrer dans les détails.

**Les éléments d’un diagramme des cas d’utilisation**

**Acteurs**

Un acteur représente l’abstraction d’un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système) qui interagissent directement avec le système étudié. Il peut consulter et/ou modifier directement l’état du système, en émettant et/ou en recevant des messages éventuellement porteurs de données.

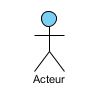


Figure 6.4‑1: Image qui représente l’acteur

**Cas d’utilisation**

Le ca0s d’utilisation décrit les exigences fonctionnelles du système. C’est l’étape qui devrait être conçue avant tout. C’est l’unité cohérente qui représente la fonctionnalité visible de l’extérieur. Dans cette étape, le recensement des acteurs et la description des actions liées au système sont les plus reconnus de sa description soit en diagramme soit textuellement.



Figure 6.4‑2: Image qui représente le cas d’utilisation

**Relation entre cas d’utilisation**

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

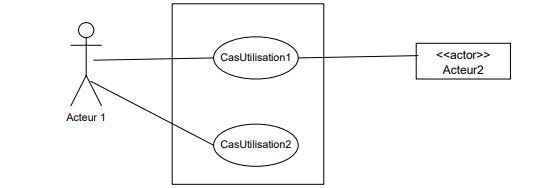
**Formalisme du diagramme**

Figure 6.4‑3: Formalisme d’un cas d’utilisation

#### Le diagramme de classe

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

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

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

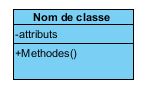
**Voici la notation pour les classes**

Figure 6.4‑4: Formalisme d’une classe

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

Visibilité nom: type=valeur initiale

Syntaxe des attributs :

Tableau 6.4‑1: Les différents types de visibilité

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

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

Tableau 6.4‑2: Tableau montrant les types de relation entre les Classes

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

**Notion de cardinalité**

Ces relations peuvent aussi être additionnées par des cardinalités, qui sont :

Une cardinalité est le nombre de fois minimum et maximum qu’une instance d’une classe participe à l’association. Ou encore c’est le nombre d’instances de l’association pour une instance de la classe. On l’appelle également la multiplicité

Tableau 6.4‑3: Tableau montrant les différentes multiplicités

|  |  |
| --- | --- |
| 1 | Un et un seul |
| 1..1 | Zéro ou un |
| m.. n | De m à n (entier) |
| \* | Plusieurs |
| 0.. \* | De zéro à plusieurs |
| 1.. \* | D’un à plusieurs |

**Héritage**

La généralisation : définit une relation de classification entre une classe plus générale (superclasse0 et une classe plus spécifique(sous-classe). Il s’agit de prendre des classes existantes (déjà) mise en évidence et de créer de nouvelles classes qui regroupent leurs parties communes, il faut aller du plus spécifiques au plus général, c’est une démarche ascendante.

La spécialisation : il s’agit de sélectionner des classes existantes (déjà) identifiées et d’en dériver de nouvelles classes plus spécialisées en spécifiant simplement les différences. Il s’agit d’une démarche descendante Une classe abstraite est une classe qui ne s’instancie pas directement, mais qui représente une simple abstraction afin de factoriser les propriétés communes des sous-classes. Elle se note en italique.

Formalisme :

Sa formalité est représentée par la figure ci-dessous

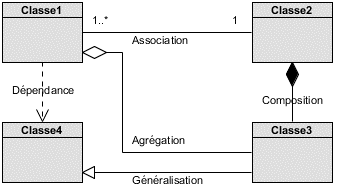


Figure 6.4‑5: Formalisme du diagramme de classe

**Agrégation**

L’agrégation est un type d’association, elle est symbolisée par



Figure 6.4‑6: Image qui représente l’agrégation

**Composition**

Une composition est une agrégation forte, elle est représentée par le symbole



Figure 6.4‑7: Image qui représente l’agrégation

#### Le diagramme de séquence

**Rôle du diagramme de séquence**

Le diagramme de séquence fait partie des diagramme comportementaux (dynamique) et plus précisément des diagrammes d’interactions.

Il permet de représenter des échanges entre les différents objets et acteurs du système en fonction du temps.

A moins que le système à modéliser soit extrêmement simple, nous ne pouvons pas modéliser la dynamique globale de système dans un seul diagramme. Nous ferons donc appel à un ensemble de diagramme de séquence chacun correspondant à une sous fonction du système, généralement d’ailleurs pour illustrer un cas d’utilisation.

Représentation du diagramme de séquence

Délimitation du diagramme de séquence

Le diagramme de séquence est placé dans un rectangle qui dispose d’une étiquette sd en haut à gauche (qui signifie sequence diagram) suivi du nom du diagramme.

**Formalisme du diagramme**

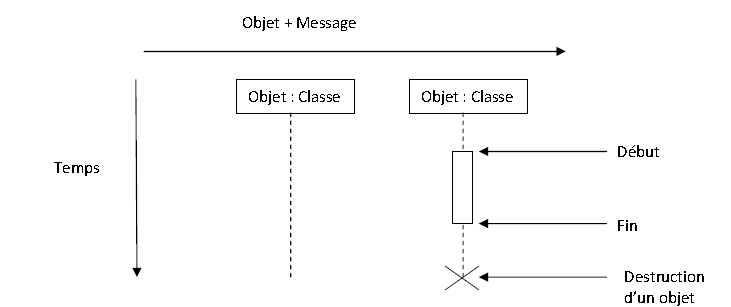


Figure 6.4‑8: Formalisme du diagramme de séquence

#### Le diagramme d’activité

Le diagramme d’activité fait partie des diagrammes comportements. Il est utilisé pour modéliser les aspects dynamiques d’un système. Il s’agit de représenter les opérations d’un processus et leurs conséquences sur les objets (logiciels ou matériels). La modélisation peut être utilisée pour décrire le déroulement d’un cas d’utilisation ou d’une méthode.

Les diagrammes d’activité affichent le flux de travail d’un point de départ à un point d’arrivé en détaillant les nombreux chemins de décision existant dans la progression des évènements contenus dans l’activité.

Ils peuvent être utilisés pour détailler des situations dans lesquelles un traitement parallèle peut avoir lieu lors de l’exécution de certaines activités. Les diagrammes d’activités sont utiles pour la modélisation métier car ils sont utilisés pour détaillés les processus impliqués dans les activités métier.

**Formalisme**

Le diagramme d’activité est sémantiquement proche des diagrammes de communication, ou d’états-transition, ces derniers offrant une vision microscopique des objets du système.

Le diagramme d’activité présente une vision macroscopique et temporelle du système modélisé :

* Action ;
* Action structurée ;
* Historique ;
* Fusion ;
* Décision ;

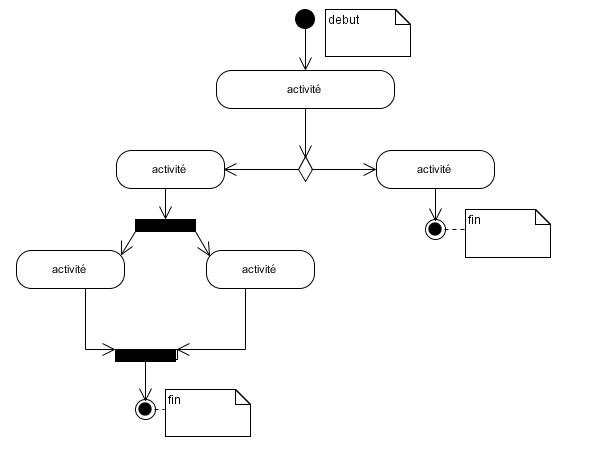


Figure 6.4‑9: Formalisme du diagramme d’activité

#### Diagramme des composants

**Notion de composant**

En UML, un composant est un élément logicielle remplaçable et réutilisable qui fournit ou reçoit un service bien précis. Il peut être vu comme une pièce détachée du logiciel. Les plugins, les drivers, les codecs, les bibliothèques sont des composants.

La notion de composant est proche de celle l’objet, dans le sens de la modularité et de réutilisation avec toutefois une granularité qui peut être différente. Le composant est à l’architecture du logiciel ce que l’objet est à l’architecture du code.

Les composants fournissent des services via des interfaces. Un composant peut être remplacé par n’importe quel autre composant compatible c’est-à-dire ayant les mêmes interfaces. Un composant peut évoluer indépendamment des applications ou des autres composants qui l’utilise à partir du moment où les interfaces sont respectées.

**Notion de port**

Un port est un point de connexion entre le composant et son environnement, il est la matérialisation de l’interface. Un port est représenté par un petit carré à la périphérie du composant.

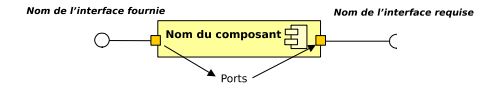


Figure 6.4‑10: Image d’un port

**Formalisme de diagramme de composant**

Pour représenter notre diagramme, nous pouvons choisir entre différents formalismes. Nous avons alors utilisé le formalisme montrant la dépendance entre composants. Ce formalisme est représenté par la figure ci-dessous.

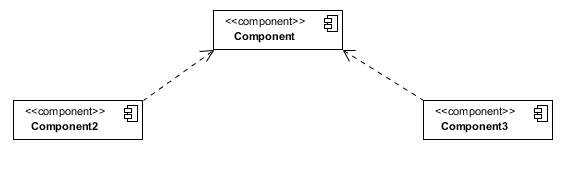


Figure 6.4‑11: Formalisme d'un diagramme de composant

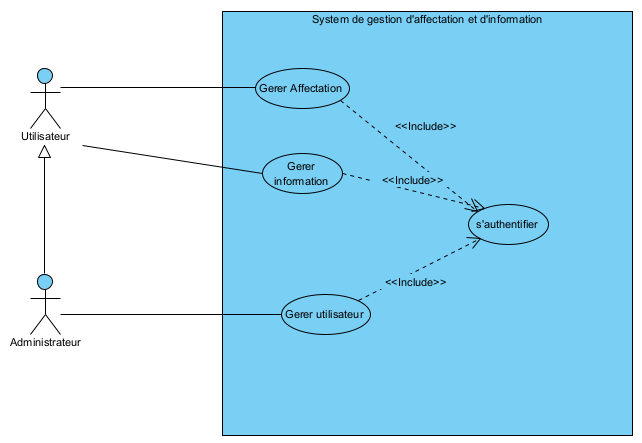
**Représentation du diagramme de composant de notre projet**

Selon le formalisme du diagramme, nous allons représenter le diagramme de composant de notre application par la figure ci-dessous.

Figure 6.4‑12: Diagramme de composant du système

### Description des diagrammes UML du projet

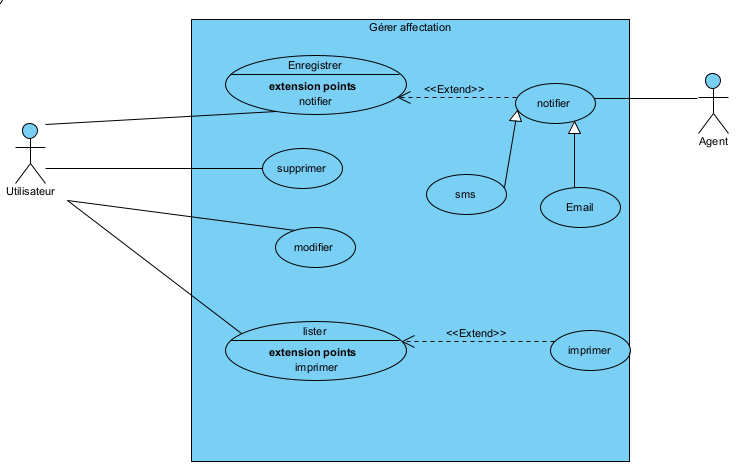
Diagramme de cas d’utilisation globale du système



Le diagramme de cas d’utilisation global résume toutes les fonctionnalités possibles de notre système. En partant des formalismes cités ci-dessus des acteurs et des cas d’utilisations.

**Etude de cas d’utilisation « Gérer affectation »**

Etudions le cas « Enregistrer affectation »



Dans ce cas d’utilisation, le responsable de la ressource humaine peut ajouter, modifier et supprimer une affectation

**Description textuelle du cas « Enregistrer Affectation »**

|  |  |
| --- | --- |
| Cas d’utilisation | Enregistrer Affectation |
| Acteur principal | Responsable de la ressource humaine |
| But | Pour conserver les informations concernant l’affectation des agents |

**-Scénario nominal**

Scénario nominal du cas d’utilisation « Gérer Affectation »

|  |  |
| --- | --- |
| Numéro d’enchainement | Action |
| 1 | Le responsable demande au système le formulaire de saisie du nouvel affectation |
| 2 | Le système émet un formulaire |
| 3 | Le responsable remplie le formulaire |
| 3.1 | Validation du formulaire |
| 3.1.1 | Le système vérifie le formulaire |
| 4 | Vérification des informations dans le système |
| 6 | Vérification des informations dans le système |
| 6.1 | Envoyer une notification |

**-Scénario alternatif**

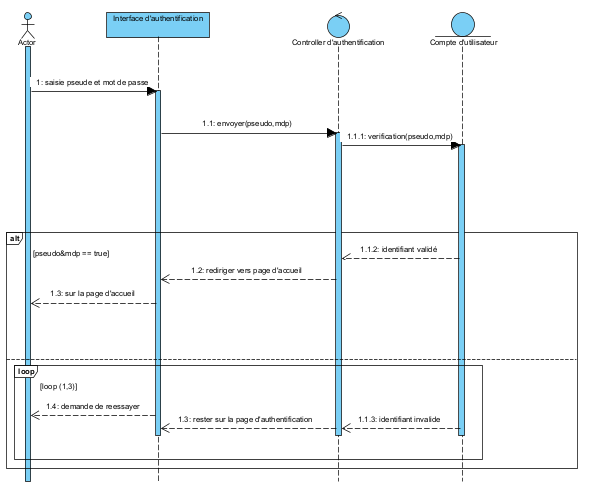
Scénario alternatif du cas d’utilisation « Gérer adhérent »

|  |  |
| --- | --- |
| Numéro d’enchainement | Action |
| 3.2 | Affectation déjà enregistrer |
| 3.2.1 | Enregistrement refusé |
| 5 | Erreur lors de la saisie des données ou des informations |
| 5.1 | Le système demande de ressaisir les données |

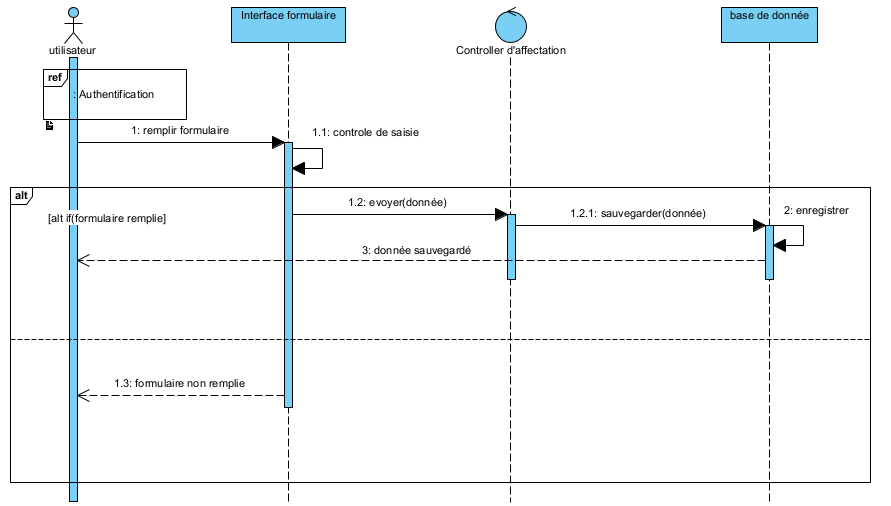
**Diagramme de séquence de cas « Authentification »**

Les diagrammes de séquences sont la représentation graphique des interactions entre l’acteurs et le système selon un ordre chronologique dans la formulation UML.

Le diagramme de séquence de notre cas ici est représenté par la figure ci-contre.



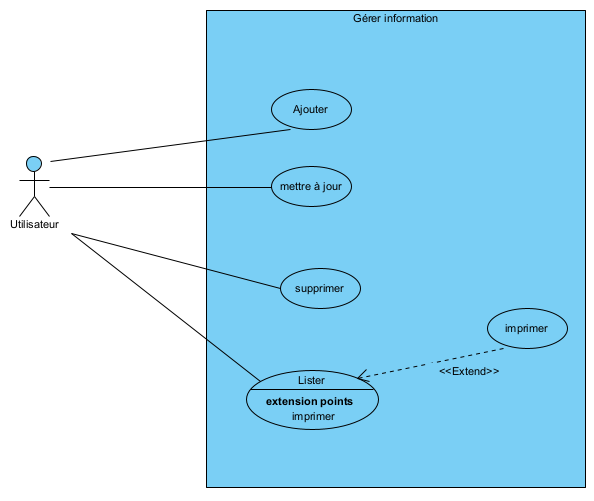
**Diagramme de séquence de cas « Enregistrer Affectation »**

****

**Diagramme de classe du cas « Gérer affectation »**

**Etude de cas d’utilisation « Gérer information »**

Dans ce cas d’utilisation, le responsable peur ajouter, mettre à jour ou supprimer des informations.



**-Généralité**

Description textuelle du cas « Enregistrer information »

|  |  |
| --- | --- |
| Cas d’utilisation | Enregistrer information |
| Acteur principal | Responsable de la ressource humaine |
| But | Pour conserver les informations concernant les agents |

**-Scénario nominal**

Scénario nominal du cas d’utilisation « Gérer information »

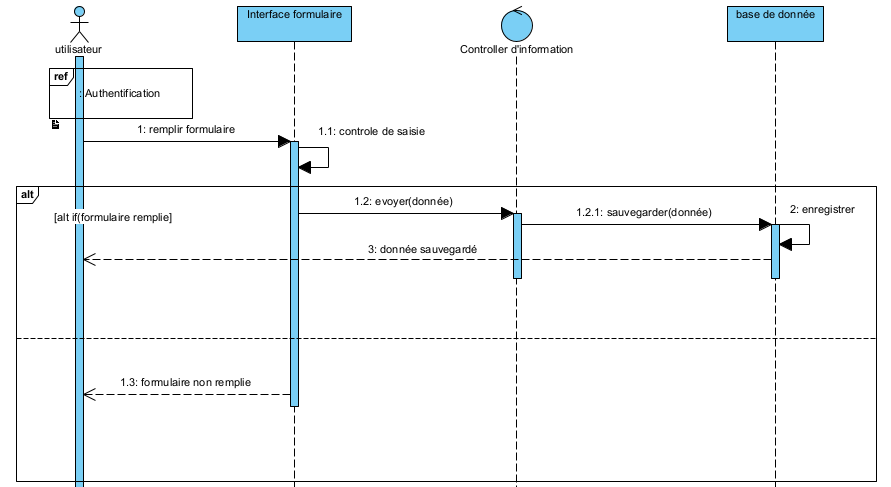
|  |  |
| --- | --- |
| Numéro d’enchainement | Action |
| 1 | Le responsable demande au système le formulaire de saisie d’une nouvelle information |
| 2 | Le système émet un formulaire |
| 3 | Le responsable remplie le formulaire |
| 3.1 | Validation du formulaire |
| 3.1.1 | Le système vérifie le formulaire |
| 4 | Vérification des informations dans le système |
| 6 | Ajout des informations avec succès |

**-Scénario alternatif**

Scénario alternatif du cas d’utilisation « Gérer information »

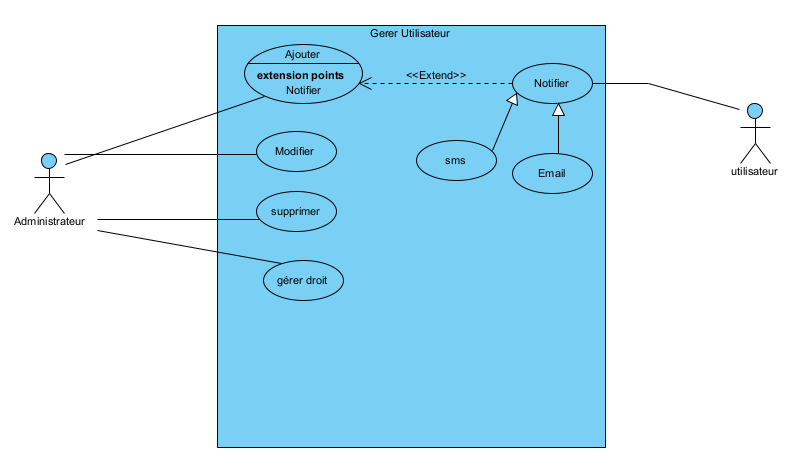
|  |  |
| --- | --- |
| Numéro d’enchainement | Action |
| 3.2 | Information déjà existée. |
| 3.2.1 | Ajout refusé |
| 5 | Erreur(s) lors de la saisie des données |

**Diagramme de séquence de cas « Gérer information »**



**Etude de cas d’utilisation « Gérer Utilisateur »**

Dans ce cas d’utilisation, le responsable peur ajouter, mettre à jour ou supprimer un utilisateur et gérer les droits d’utilisateur.



-Généralité

Description textuelle de cas « Ajouter utilisateur »

|  |  |
| --- | --- |
| Cas d’utilisation | Ajout utilisateur |
| Acteur principal | Administrateur au responsable de la ressource humaine |
| But | Pour conserver les informations concernant les utilisateurs |

-Scénario nominal

Scénario nominal du cas d’utilisation « ajout utilisateur »

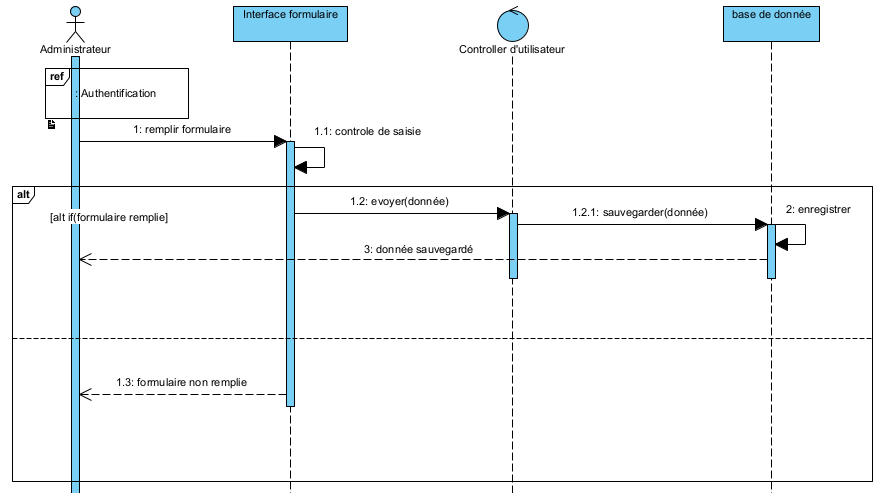
|  |  |
| --- | --- |
| Numéro d’enchainement | Action |
| 1 | L’Administrateur demande au système le formulaire de saisie d’une nouvelle utilisateur |
| 2 | Le système émet un formulaire |
| 3 | L’administrateur remplie le formulaire |
| 3.1 | Validation du formulaire |
| 3.1.1 | Le système vérifie le formulaire |
| 4 | Vérification des informations dans le système |
| 6 | Ajout d’utilisateur avec succès |

-Scénario alternatif

Scénario alternatif du cas d’utilisation « ajout utilisateur »

|  |  |
| --- | --- |
| Numéro d’enchainement | Action |
| 3.2 | Utilisateur déjà existée. |
| 3.2.1 | Ajout refusé |
| 5 | Erreur(s) lors de la saisie des données |

**Diagramme de séquence du cas « Ajouter utilisateur »**



PARTIE III : REALISATION DU PROJET

# Spécification des outils de réalisation

## Choix technique

Pour le développement et la réalisation de notre application, nous choisissons la technologie java spring et Angular

### Java spring

Spring est un socle pour le développement d’applications, principalement d’entreprises mais pas obligatoirement. Il fournit de nombreuses fonctionnalités parfois redondantes ou qui peuvent être configurées ou utilisées de plusieurs manières : ceci laisse choix au développeur d’utiliser la solution qui lui convient le mieux et/ou qui répond aux besoins.

Spring est ainsi un des frameworks les plus répandus dans le monde java : sa popularité a grandi au profit de la complexité de java EE notamment pour ses versions antérieures à la version 5 mais aussi grâce à la qualité et la richesse des fonctionnalités qu’il propose :

* Son cœur reposant sur un conteneur de type loC assure la gestion du cycle de vies des beans et l’injection des dépendances
* L’utilisation de l’AOP
* Des projets pour faciliter l’intégration avec de nombreux projets open source ou API de java EE

#### Le but de Spring

Le but de Spring est de faciliter et de rendre productif le développement d’applications, particulièrement les applications d’entreprises.

#### Les fonctionnalités proposées par Spring

Spring propose de nombreuses fonctionnalités de base pour le développement d’applications :

* Un conteneur léger implémentant le design pattern MVC pour la gestion des objets et de leurs dépencances en offrant des fonctionnalités avancées concernant la configuration et l’infection automatique. Un de ses points forts est d’être non intrusif dans le code de l’application tout en permettant l’assemblage d’objets faiblement couplés.
* Une gestion des transactions par déclaration offrant une abstraction du gestionnaire de transactions sous-jacent
* Faciliter le développement des DAO de ça couche de persistance en utilisant JDBC, JPA, JDO ou une solution open source comme Hibernate, iBatis et une hiérarchie d’exceptions
* Un support pour un usage interne à Spring (notamment dans les transactions) ou personnalité de l’AOP qui peut être mis en œuvre avec Spring AOP pour les objets gérés par le conteneur et/ou avec AspectJ.
* Faciliter la testabilité de l’application

#### Les avantages de Spring

* Spring propose une très bonne intégration avec des frameworks open source (Struts, Hibernate, …) ou des standards de Java (Servlets, JMS, JDO, ...)
* La documentation de Spring est complète et régulièrement mise à jour lors de la diffusion de chaque nouvelle version.
* Les fonctionnalités offertes par Spring sont très nombreuses et les sujets couverts ne cessent d’augmenter au fur et mesure des nouvelles versions.

### Angular

Angular Js est un framework Javascript (bibliothèque de fonctions pré-écrites en JavaScript) libre et open source formée par le géant Google en 2009. Ses deux fondateurs avaient au départ un projet qui leur posait problème, contenu du manque de structure des codes qu'ils avaient élaborés et qui ne pouvaient être exploitables. En dehors de ce projet, Misko Hevesy un des fondateurs, avait commencé à travailler sur un framework **Javascript**. Cela lui a permis d'avancer sur son projet commun et de rendre plus malléable les lignes de codes. Ils sont donc passé de 17 000 lignes à 1 500 lignes. Cette solution plutôt convaincante a donné à son créateur une raison de finaliser ce fameux framework et de le présenter sous le nom d'Angular JS.

#### En quoi consiste Angular JS ?

AngularJS est un framework, c’est-à-dire qu’il va structurer le code et y imposer une architecture. C’est un framework de type MVC (Modèle-vue-contrôleur). En effet, il utilise de nombreux langages de programmation pour apporter une certaine architecture à l’application. Une application conforme au motif MVC comporte trois types de modèles, les vues et les contrôleurs.

* Un modèle qui contient les données. Il représente le milieu dans lequel adhère l’application en question.
* Une vue qui contient la présentation de l’interface graphique. La vue se sert du modèle, et peut se présenter sous différentes formes comme un diagramme ou encore un formulaire. Elle contient différentes balises HTML.
* Un contrôleur qui va traiter les actions effectuées par l’utilisateur, mais également modifier les données du modèle et de la vue.

### GitHub

GitHub est considéré comme un service d’hébergement de projets en cours de développement.

Il propose également un espace collaboratif. Grâce à cela, plusieurs développeurs peuvent travailler sur un même code plus facilement.

#### Avantages de git

* Service gratuit, bien qu’il ait également des services payants.
* Recherche très rapide dans la structure des dépôts.
* Grande communauté et aide facile à trouver.
* Il propose des outils pratiques de coopération et une bonne intégration avec git.
* Facile à intégrer avec d’autres services tiers.

#### Pourquoi j’ai utilisé GitHub ?

Parce que git permet de faciliter la collaboration en permettant de revenir sur les anciennes versions. La plateforme GitHub va contenir quant à elle les dépots dans le cloud afin qu’on arrive à travailler sur un même projet et distinguent en temps réel les modifications apportées par les autres développeurs.

#### Les commandes fréquentes pour vérsionner en local

* Git init ;

Cette commande permet de convertir un projet existant sans version en un dépôt git ou d’initialiser un nouveau dépôt vide. Il s’agit donc généralement de la première commance que nous exécuterons dans un nouveau projet.

* Git add :

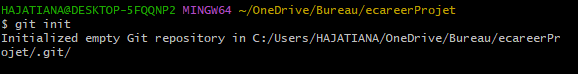
Cette commande ajoute un changement dans le répertoire de travail à la zone de staging. Elle informe Git que vous voulez inclure les mises à jour dans un fichier particulier du commit suivant. Cependant, « git add » n’impacte pas le dépôt de manière significative. Les changements ne sont pas réellement enregistrés jusqu’à ce que vous exécutiez « git commit ».

* Git commit –m

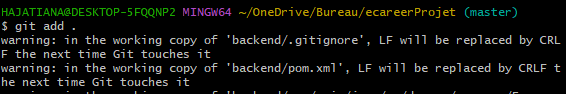
Cette commande capture un instantané des chagements actuellement stagés du projet. Les instantanés peuvent être considéres comme des versions « sûre » d’un projet.

**Application**

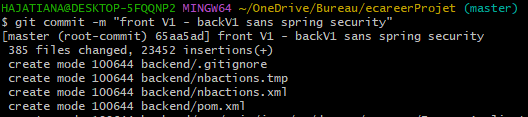
* Git innit :



* Git add . :



* Git commit –m



#### Versionner sur un dépôt distant

* Git clone :

Cette commande est un utilitaire de ligne de commande Git utiliser pour cibler un dépôt existant et créer un clone ou une copie du dépôt

* Git push –u origin main

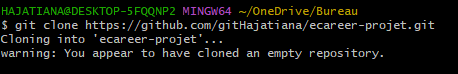
Cette commande est utilisée pour charger le contenu d’un dépôt local vers un dépôt distant. Le push vous permet de transférer les commits de votre dépôt local vers un dépôt distant.

#### Commit son projet sur GitHub

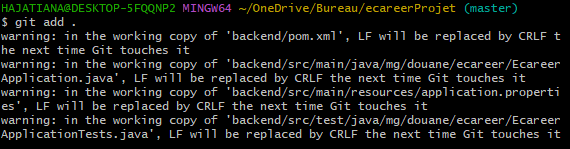
* Git clone
* Git add .
* Git commit –m « message »
* Git push –u origin main

**Application**

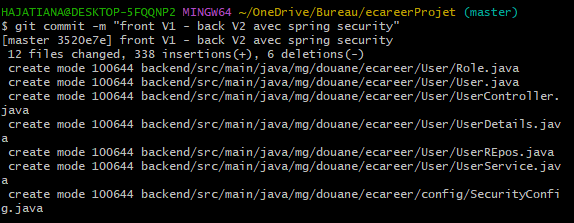
* git clone :

****

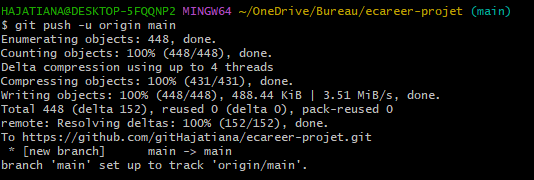
* git add . :

****

* git commit –m « description » :

****

* git push –u origin main :

****

## Présentation des outils de développement utilisés

### Le Visual Studio

Visual Studio Code est un éditeur de code open-source développé par Microsoft supportant un très grand nombre de langages grâce à des extensions. Il supporte l’autocomplétion, la coloration syntaxique, le débogage, et les commandes git.



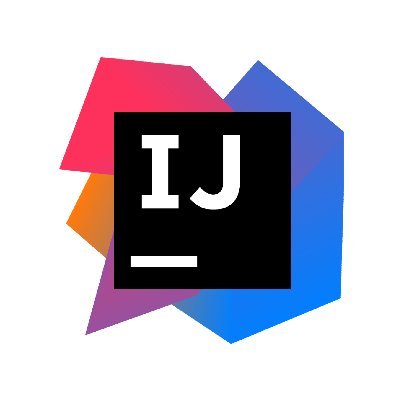
### Historique

Visual Studio Code a été annoncé le 29 avril 2015 par Microsoft lors de la conférence Build 2015 Une version préliminaire a été publiée peu de temps après.

Le 18 novembre 2015, Visual Studio Code a été publié sous la licence MIT et son code source publié sur GitHub. Le support d’extensions a également été annoncé.

Le 14 avril 2016, Visual Studio Code est sorti de phase bêta et a été publié sur le Web

### IntelliJ IDEA

IntelliJ IDEA est un IDE intelligent et tenant compte du contexte qui permet de travailler sur toutes sortes d’applications en Java et dans d’autres langages de la JVM tels que Kotlin,Scala et Groovy. De plus, IntelliJ IDEA Ultimate vous aide à développer des applications web full-stack grâce à ses puissants outils intégrés, à la prise en charge de JavaScript et de ses technologies connexes et à la prise en charge avancée de frameworks populaires tels que Spring, Spring Boot, Jakarta RR, Micronaut, Quarkus et Helidon. IntelliJ IDEA peut être complété par des plugins gratuits développés par JetBrains afin de pouvoir travaliller avec d’autres langages de programmation, parmi lesquels Go, Python, SQL, Ruby et PHP.

## Le système de gestion de base de données

Une base de données est un ensemble structuré de données enregistrées dans un ordinateur et accessible de façon sélective par plusieurs utilisateurs. Un logiciel qui permet d’interagir avec une base de données s’appelle un Système de Gestion de Base de Données (SGBD) [Richard Grin, 2000].

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

### Utilité d’une base de données

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

### Caractéristique d’un SGBD

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

**Indépendance physique :** le niveau physique peut être modifié indépendant du niveau conception. Cela signifie que tous les aspects matériels de la base de données n’apparaissent pas pour l’utilisateur, il s’agit simplement d’une structure transparente de représentation des informations.

**Indépendance logique :** le niveau conceptuel doit pouvoir être modifié sans remettre en cause le niveau physique, c’est-à-dire que l’administrateur de la base doit pouvoir la faire évoluer sans que cela gêne les utilisateurs.

**Rapidité d’accès :** le système doit pouvoir fournir les réponses à la requête le plus vite possible, cela implique des algorithmes de recherche rapides.

**Administration centralisée :** le SGBD doit pouvoir éviter dans la mesure du possible des informations redondantes, afin d’éviter d’une part un gaspillage d’espace mémoire mais aussi des erreurs.

**Sécurisation des données :** le SGBD doit représenter des mécanismes permettant de gérer les droits d’accès aux données selon les utilisateurs.

Considérons quatre SGBD le plus souvent, tel que : MS Access, MySQL, PostgreSQL, Oracle Database. Les avantages et les inconvénients se chaque SGBD est figuré dans le table.

### Avantages et inconvénients des SGBD

Tableau 7.2: Tableau montrant les avantages et inconvénients des SGBD

| Base de données | Avantages | Inconvénients |
| --- | --- | --- |
| MS Access | * Intégré avec Microsoft office ; * Dispose une interface graphique pour saisir la requête ; * Base de données à petite taille, facile à gérer. | * Mono poste (MS Windows) et capacité de stockage limité ; * N’intègre que le langage Visual Basic ; * Pas d’héritage de tables. * Client/serveur mais seulement un SGBD fichier. * Il n'est pas exécutable |
| MySQL | * Multi-utilisateurs ; * Gratuit (sauf si vous commercialisez un service ou un logiciel qui utilise MySQL) ; * Interfaces de programmation (API) : C, Perl, PHP, Python et Java ; * Langage de requête : SQL (langage de requête le plus répandu) ; * Portabilité ; * Multi pilotage. | * Pas d’héritage de tables ; * Support incomplet de triggers et procédure stockées ; * Manque de robustesse avec forte volumétrie. * N’importe pas des références d’intégrité relationnelle. |
| PostgreSQL | * Multi plate-forme ; * Libre ; * Largement reconnu, comportement stable et plus proche de l’Oracle. | * Les tables sont obligatoirement transactionnelles ; * Les commandes INSERT, DELETE et UPDATE sont plus lente. |
| Oracle Database | * Row Level Storage Security (RLSS) : permet de na faire apparaître que certaines lignes des tables pour un utilisateur / une application donnée. * Parallélisme, cache nommés, haute disponibilité.   Assistants performants via Oracle Manager server, possibilité de gérer en interne des tâches et des alarmes.  Gestion centralisé | * Prix élevé, tant au point de vue des licences que composants matériels (RAM, CPU) à fournir pour de bonne performance ; * Administration complexe liée à la richesse de fonctionnelle. |

**L’Oracle**

Oracle est un système de gestion de base de données relationnel (SGBDR) fourni par Oracle Corporation. Il a été développé par Lawrence Ellison, accompagné d’autres personnes telles que Bob Miner et Ed Oates

**Historique**

Softwar Development Laboratories a été créé en 1977. En 1979, SDL change de nom en devenant Relational Sofrware, Inc. (RSI) et introduit son produit Oracle V2 comme base de données relationnelle. La version 2 ne supportait pas les transactions mais implémentait les fonctionnalités SQL basiques de requête et jointure. Il n’y a jamais eu de version 1, pour des raisons de marketing, la première version a été la version 2. Celle-ci fonctionnait uniquement sur les systèmes Gigital VAX/VMS.

**Avantages de l’Oracle**

Oracle db est la BD préférée des entreprises pour de multiples raisons. Sa version gratuite permet de tester ses capacités bien qu’elles soient limitées, mais le détail qui fait pencher la balance vers Oracle db à chaque fois qu’un prospect a le choix entre plusieurs bases de données, c’est bien la réputation de fiabilité. Son unicité, la simplicité de ses outils, logiciels et programmes font de

# Mise en œuvre et implémentation

## Architecture logicielle

L’architecture logicielle décrit d’une manière symbolique et schématique les différents éléments d’un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions. Contrairement aux spécifications produites par l’analyse fonctionnelle, le modèle d’architecture, produit lors de la phase de conception, ne décrit pas ce que doit réaliser un système informatique mais plutôt comment il doit être conçu de manière à répondre aux spécifications. L’analyse décrit le « quoi faire » alors que l’architecture décrit le « comment le faire ».

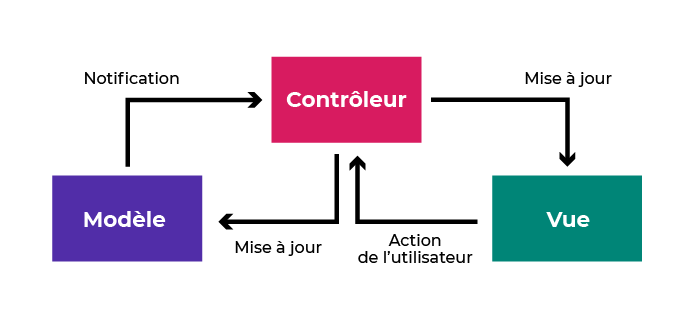


Figure 8.1‑1: Architecture logicielle

## Architecture matérielle

Selon Wikipédia, « *L´architecture matérielle est une vue tournée sur le choix et l'organisation des différents composants électroniques d'un appareil informatique* ». L’architecture matérielle est donc une définition des composants matériels pour la réalisation d’une application.

L’architecture matérielle décrit l’agencement interne de [composants électroniques](https://fr.wikipedia.org/wiki/Composant_%C3%A9lectronique) ainsi que leurs interactions.

Le terme interne employé ici permet de bien faire la différence avec l’[architecture (externe) de processeur](https://fr.wikipedia.org/wiki/Architecture_de_processeur) (ou architecture de [jeu d'instruction](https://fr.wikipedia.org/wiki/Jeu_d%27instructions)), qui s'intéresse à la spécification fonctionnelle d'un [processeur](https://fr.wikipedia.org/wiki/Processeur), du point de vue du [programmeur](https://fr.wikipedia.org/wiki/D%C3%A9veloppeur) en [langage machine](https://fr.wikipedia.org/wiki/Langage_machine). Plusieurs architectures internes peuvent implémenter une même architecture externe.

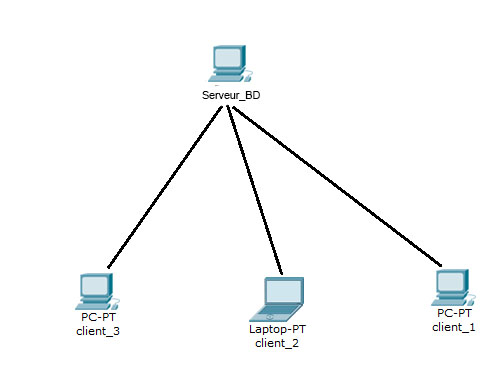


Figure 8.2‑1: Architecture matérielle

## Conception de l’IHM

Il y a trois participants dans l’interface homme-machine, à savoir : l’utilisateur (participant avec choix), la machine (participant avec programme), le concepteur (participant qui anticipe les choix possibles de l’utilisateur et les code dans un programme [Jean-Marc PUJOS].

### Création des menus et interfaces

Actuellement, la majorité des interfaces graphiques sont composées de [fenêtres](https://fr.wikipedia.org/wiki/Fen%C3%AAtre_(informatique)), [icônes](https://fr.wikipedia.org/wiki/Ic%C3%B4ne_(informatique)), [menu](https://fr.wikipedia.org/wiki/Menu_(informatique)) et d'un système de pointage ([souris](https://fr.wikipedia.org/wiki/Souris_(informatique)), trackpad, trackpoint ou [écran tactile](https://fr.wikipedia.org/wiki/%C3%89cran_tactile)) (anglais *Window*, *Icon*, *Mouse*, *Pulldown menu*, abr. *WIMP*), selon le principe lancé par Xerox en 1980[2](https://fr.wikipedia.org/wiki/Interface_graphique#cite_note-conf-2).

Une [fenêtre](https://fr.wikipedia.org/wiki/Fen%C3%AAtre_(informatique)) est une portion d'écran rectangulaire qui contient une vue d'une interface graphique. Les fenêtres peuvent être réduites, agrandies et disposées les unes sur les autres, telles des feuilles de papier. Une [icône](https://fr.wikipedia.org/wiki/Ic%C3%B4ne_(informatique)) est un pictogramme ou un logo d'un objet que l'utilisateur peut manipuler. L'utilisation d'une image est plus innée et « naturelle » que celle d'un texte, les capacités de manipulation visuelles d'un humain apparaissant avant le langage[6](https://fr.wikipedia.org/wiki/Interface_graphique#cite_note-factors-6).

### Fonctionnalité

La fonctionnalité de base de toute IHM (Interface Homme Machine) est de permettre une interaction entre les utilisateurs de l’application et celui-ci. La population visée par cette application est large et son niveau présupposé en informatique est basique.

Il est donc primordial de concevoir une interface :

* Agréable esthétiquement ;
* De prise en main rapide ;
* Intuitive ;
* A l’ergonomie classique, c’est-à-dire non déroulante.

De même, il est absolument inenvisageable de considérer une IHM qui ne permettrait pas le contrôle total de l’application : données, paramètres, …etc.