Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Dédicace

Nous dédions cette présente mémoire :

A nos parents

A nos frères et sœurs

A tous nos amis et camarades de promotion

A tous ceux qui ont contribué à la réussite de ce travail

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Remerciements

Nos remerciements vont d’abord à l’endroit de Mme ZAKOU Fatima-Zara qui est notre encadreur de terrain. De plus, son sérieux et sa perspicacité dans le travail bien fait imposent le respect et l’admiration de tous.

Nous tenons à remercier également :

* Mr. Rabiou Issoufa : Notre maitre de stage pour tout le temps qu’il nous a octroyé et pour tous les conseils qu’il nous a prodigué.
* Professeur Soumana HAMMA BEIDI : Représentant Résidant pour son humilité et sa courtoisie, à faire de l’IAI NIGER une référence nationale.
* Mr. Abdoulaye Sofiani : Directeur des Etudes de l’IAI-Niger pour le suivi et la patience dont il a fait preuve tout au long de notre cycle.

Nous remercions également l’ensemble des employés de L’Inspection Général des Services du Ministère des Finances pour le bon accueil, la sympathie, la disponibilité et la patience dont ils ont fait preuve à notre égard tout au long de notre séjour en leur compagnie.

Nous ne saurions oublier l’ensemble des enseignants de l’IAI-NIGER qui nous ont dispensé le savoir durant toutes les années de notre formation.

Enfin, nous tenons à exprimer nos sincères reconnaissances et notre profonde gratitude à tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Table des figures

Table des tableaux

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Table des matières

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

# Introduction

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Les systèmes d’information jouent un rôle primordial dans l’activité des organisations du fait qu’ils permettent de fournir l’information nécessaire à la prise de décision selon leur finalité principale.

De nos jours l’informatique couvre à peu près toutes les branches de l’activité humaine. Elle contribue à l’évolution de plusieurs domaines grâce à ses applications variées en industrie, communication, enseignement… etc. A son tour, la qualité a joui du progrès de l’informatique et de ses apports vu que la qualité représente aujourd’hui le moyen le plus sûr de gagner la concurrence.

Le responsable de l’Inspection General des Services du Ministère des Finances, ayant pris conscience de l’apport de l’informatique sur l’amélioration de la procédure du suivi et de qualité a été convaincu par la Direction de l’Informatique Financière (DIF), qui est le siège de notre stage, de La mise en place d’un système informatisé de gestion des Programmations ainsi que le Suivi des missions de contrôle. C’est dans cette optique qu’il nous a été proposé comme thème de projet de fin d’études « La mise en place d’un système informatisé de gestion des Programmations et Suivis des missions d’une Inspection Générale des Services : cas du Ministère des Finances ».

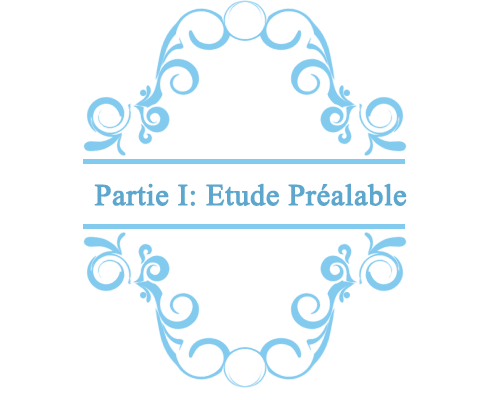
Le présent rapport reflète et décrie la démarche suivie pour mettre en place ce système informatisé demandé :

-La première partie décrie de façon globale la présentation du centre d’accueil ainsi que du projet ;

-La deuxième partie présente de manière globale l’analyse et la conception du projet ;

-La troisième partie est consacrée à la réalisation où nous présentons l’environnement de développement et quelques captures d’écrans.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018



Chapitre I : Présentation du centre d’accueil

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

## Présentation du Ministère des Finances :

**L**e Ministère des Finances est organisé conformément au décret [n°2018-479/PRN/MF du 20 juillet 201](http://www.finances.gouv.ne/images/decrets/Decret_2016_387_PRN_MF.pdf)8 .

1.1 Organisation du Ministère des Finances

Le Ministère des Finances est organisé ainsi qu’il suit :

* L’administration centrale qui comprend :
* le cabinet du Ministre des Finances ;
* le cabinet du Ministre Délégué auprès du Ministre des Finances, chargé du Budget ;
* le Secrétariat Général ;
* l’Inspection Générale des Finances ;
* l’Inspection Générale des Services ;
* les Directions Générales ;
* les Directions Techniques Nationales et les Directions Nationales d’Appui ;
* les Organes Consultatifs ;
* les Administrations de Mission ;
* les Services Techniques Déconcentrés ;
* Les Services Techniques Déconcentrés et Les Services Décentralisés  qui sont composés :
* Des Services Extérieurs dont:
* des Directions Régionales, des Recettes et des Centres des Impôts ;
* des Directions Régionales, des Recettes, des Bureaux et des Brigades des Douanes ;
* des Directions Régionales du Budget ;
* des Trésoreries Régionales et Départementales ;
* des Trésoreries des Représentations Diplomatiques et Consulaires du Niger à l’étranger ;
* des Régies des Administrations Financières ;
* des Postes des Contrôleurs Régionaux des Marchés Publics et Engagements Financières ;
* des Postes des Receveurs et Municipaux ;
* des Services Régionaux de la Conservation Foncière et des Hypothèques ;
* des Services Régionaux du Patrimoine de l’Etat.
* les Services Rattachés et des Etablissements Publics comprennent :
* la Cellule Nationale de Traitement des Informations Financières (CENTIF) ;
* l’Agence Judiciaire de l’Etat(AJE) ;
* la Caisse de Dépôts et Consignations (CDC).
* l’Agence de Régulation du Secteur de la Micro-finance(ARSM) ;
* le Secrétariat Exécutif de la Stratégie Nationale de Finance Inclusive ;
* le Bureau Nationale de la Carte Brune CEDEAO(BNCB) ;
* la Cellule Union Européenne (CUE) ;
* la Cellule chargée du Suivi de l’Intégration UEMOA-CEDEAO (CSI/ UEMOA-CEDEAO).
* la Commission des Calcules Fiscaux (CCF) ;
* le Comité Nationale de Politique Economique (CNPE);
* le Comité Arbitral des Recours Fiscaux (CARFI) ;
* le Loterie Nationale du Niger (LONANI) ;
* le Fonds de Garantie Automobile (FGA) .

## 2. Présentation de la Direction de l’informatique financière :

La Direction de l’Informatique Financière (DIF) est organisée conformément à l’[arrêté n° 254/MF/CAB](http://www.finances.gouv.ne/images/Textes/0254_MF_CAB_du_27_Juin_2014.pdf) du 27 Juillet 2014.et comprend les services suivants:

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

* Le secrétariat de la direction.
* La division exploitation et production informatiques.
* La division études, développements informatiques et formation.
* La division équipements, réseaux informatiques et télécommunications.
* La division administrative et ressources.
* La cellule des architectes de systèmes d'information et des administrateurs.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

## Présentation de l’inspection générale de service :

L’Inspection Générale des Services (IGS) du Ministère des Finances a été créé en 2003 pour être en conformité avec le décret N° 2013-070|PRN|PM du 22/02 /2013 portant attributions, organisation et modalités des fonctionnements de l’inspection générale des services au sein de chaque département ministériel. L’IGS est organisée conformément aux dispositions de l’Arrêté N° 259MF/IGS du 07 juillet 2014.

L’Inspection Générale des Services (IGS) est un organe de veille, de contrôle interne et d’appui-conseil. Elle assiste le Ministre des Finances dans la gestion administrative et technique des services et contrôle l’utilisation rationnelle et optimale des ressources humaines, matérielles et financières mise à la disposition du Ministère des Finances et des organismes qui en dépendent.

D’une manière générale l’IGS exerce les missions suivantes :

* Audit;
* Inspection ou vérification;
* Les passations de service.

### 3.1 Organisation de l’inspection générale de service :

L’Inspection Générale des Services est Organisée ainsi qu’il suit et comprend :

-Une Inspecteur Général des Services ;

-Des Inspecteurs de Services ;

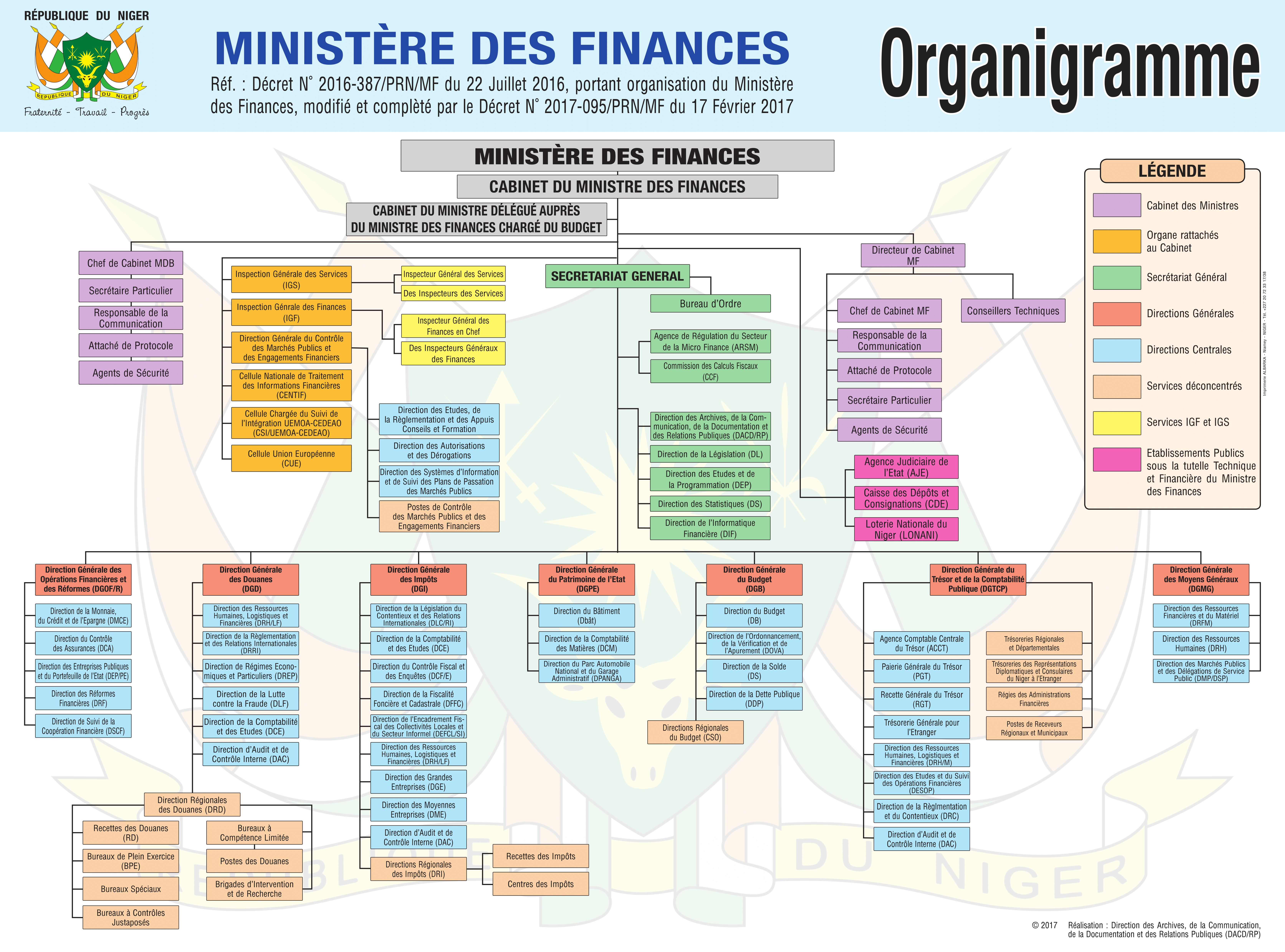
-Un Secrétariat.

Entant que besoin, un personnel d’appui est affecté à l’Inspection Générale des Services (IGS).L’Inspection Générale des services est dirigée par un Inspecteur Général des Services. Les Inspecteurs des Services sont nommés parmi les cadres expérimentés de la catégories A1 ayant au moins quinze (15)ans d’ancienneté dont cinq (5) ans au moins à un poste de responsabilité.

.

## Organigramme

Mémoire de fin de cycle : Analyste Programmeur 2017-2018



# Chapitre 2 : Présentation du projet

## Contexte du projet :

**L**a complexité croissante des systèmes informatiques a conduit les concepteurs à s’intéresser aux méthodes de développement. Ainsi, L’utilisation de l’outil informatique est une réalité de plus en plus présente dans les structures.

Cependant, à ce jour l’IGS ne dispose pas d’outils informatiques de gestion de ses activités administratives. Face à cette situation où la gestion du quotidien se fait de façon quasi manuelle, la Direction de l’Informatique Financière (DIF) a jugé utile d’informatiser les tâches quotidiennes de L’IGS. Elle souhaiterait en effet rendre la gestion de l’IGS, plus efficiente et accélérer la planification des activités de contrôle et l’assignation des ressources.

C'est dans ce cadre que la DIF nous a proposé pour thème de mémoire « la mise en place d'un système informatisé de gestion de programmations et suivis des missions d’une IGS : cas du Ministère des Finances ». Ce document ne concerne que ces aspects.

# Description de l’existant :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Après les interviews que nous avons eues avec les acteurs intervenant dans le cadre de notre étude :

Une programmation annuelle se fait principalement :

* Sur la base des rapports des missions des années précédentes et/ou des rapports d’activités des différentes structures du ministère;
* Sur proposition des Inspecteurs des Services
* D’une décision du Ministre des Finances;

Suite à cette programmation, il est établi un ordre de mission à un//ou un groupe d’Inspecteurs de Service.

Une fois les ordres de mission rédigée, l’IGS établit un suivi des missions pour s’assurer, à travers un tableau, de la bonne marche de ces missions. Les Inspecteurs des Services exécutent les missions selon les étapes suivantes :

# la collecte de l’information ;

* l’exécution de la mission ;
* la rédaction du rapport ;
* la validation du rapport .

Les rapports définitifs sont généralement assortis des recommandations et d’un plan d’actions.

Ensuite sur instruction du Ministre, commanditaire de la mission l’IGS envoie les recommandations et plan d’actions à la structure qui a fait l'objet du contrôle .Aussi l'Inspection Générale des Services peut désigner un Inspecteur de Service pour s’assurer que ces recommandations sont suivies et respectées.

Il est à noter que les missions de passation ne font pas partie de la programmation annuelle. Par ailleurs, en cas de besoin, le Ministre peut ordonner des missions non contenues dans le programme annuel.

## Problématique soulevée :

Du fait que la gestion de tous les documents de l’IGS se fait manuellement, elle se confronte à des difficultés de :

* Remplissage correct des fiches de programmations annuel et des ordres de mission ;
* Recherche des fiches de programmation annuelle ainsi que des ordres de missions ;
* Sécurisation des rapports, et des suivis de mission, des recommandations, ce qui peut engendrer le manque de confidentialité.
* Gestion des suivis des missions et des recommandations ;

La lenteur dans l’élaboration et la transmission des ordres de mission signés par le Ministre ;

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

## Travail attendu :

Ce projet a pour objectif principal de proposer une solution à un problème concret, et ceci en partant d’une définition des besoins. Le principal objectif du futur système est de permettre la transmission des informations concernant la programmation annuelle, les ordres de de missions transmis, ainsi que le suivi de ces missions entre les différents professionnels intervenant dans celle-ci et de conserver ces informations pour consultation ultérieure pour une prise en charge

Continue, coordonnée et sécurisée des documents.

Le système permettra globalement de :

* Assurer que toutes les missions de l’IGS soient communiquées ;
* Assurer que toutes les situations nécessitant une action soient prises en charge, immédiatement et sans délai, afin d'éviter qu'une situation demeure dans l’oubli, et que les conséquences se matérialisent ou augmentent ;
* Permettre d’avoir un suivi plus efficace des missions des Inspecteurs des Services d’une part et des recommandations d’autre part ;
* Économiser du temps lors de la préparation des rapports de mission ;
* Assurer une traçabilité complète des évènements sans effort, facilite la gestion des preuves (essentiel en cas d'incident et de poursuite) ;
* Fournir une information de qualité, précise, en temps réel, centralisée, à portée des mains ;
* Se débarrasser des formulaires papiers ;
* Assurer une planification annuelle de qualité.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

* 1. Planning prévisionnel

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

## E:\memoireIGS\intro\test_2.png

## 

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

\*

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Chapitre 1 :Analyse

## I Méthode d’analyse

Une méthodologie de développement est un cadre utilisé pour structurer, planifier et contrôler le développement d’une application .En effets, nous disposons de nos jours d’un ensemble de méthodologie chacune respectant un certain nombre de principe pour aboutir à la réalisation d’un projet. Ainsi nous allons parler principalement de quatre types de méthodes de conception.

## Pourquoi utiliser une méthode d’analyse:

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

On appelle « gestion de projet »éventuellement « conduite de projet » l’organisation méthodologique mise en œuvre pour faire en sorte que l’ouvrage réalisé par le maitre d’œuvre réponds aux attentes du maitre d’ouvrage et qu’il soit livré dans les conditions de coût et de délais prévus initialement.

En effet, les projets sont gérés avec la méthode dite « classique » qui se caractérise pour recueillir les besoins, définir le produit, le développer et le tester avant de le livrer. On parle alors ici d’une approche prédictive. Comme son nom l’indique, il s’agit ici de prévoir des phases séquentielles où il faut valider l’étape précédente pour passer à la suivante. Le chef de projet doit alors  
s’engager sur un planning précis de réalisation du projet en prévoyant des  
jalons de débuts et fins de phases ainsi que les tâches à effectuer. Il faut tout  
faire bien du premier coup car elle ne peut pas permettre de retours en arrière.  
Une décision ou un problème rencontré dans une phase peuvent remettre en  
cause partiellement ou totalement les phases précédentes validées. Dans un  
cycle « en cascade » les risques sont détectés tardivement puisqu’il faut  
attendre la fin du développement pour effectuer la phase de test. Plus le projet  
avance, plus l’impact des risques augmente : il sera toujours plus difficile et  
coûteux de revenir en arrière lorsqu’on découvre une anomalie tardivement.  
Par conséquent, comment peut-on augmenter la satisfaction du client en  
facilitant la gestion de projet et améliorant la qualité de développement ? Pour  
ce faire, nous allons voir d’autres méthodes modernes plus adéquates.

## Classification des méthodologies :

### 2.1 Méthodes fonctionnelles :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Les méthodes fonctionnelles ou cartésiennes sont des démarches consistant à rechercher et à caractériser les fonctions offertes par un produit pour faire les besoin de son utilisateur. Il est aussi définit comme étant une méthode permettant à décomposer hiérarchiquement une application en un ensemble de sous applications. Parmi les fonctions de chacune de celles-ci sont affinées alternativement en sous fonctions simples à coder dans un langage de programmation donné. L’une de ces méthodes, on peut noter FAST (Fonction Analyse System Technique), SADT (Structurel-Analyse-Design-Technique) Warnier …

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Forces de ces méthodes :

* Simplicité du processus de conception préconisée;
* Adéquations à capturer les besoins de l’utilisateur;
* Capacités de produire des solutions à plusieurs niveaux d’abstraction.

Faiblesses de ces méthodes

* Effort d’analyse concentré sur les fonctions (négligeant la cohérence des données) ;
* Règles de décomposition non explicites;
* Difficulté à tenir compte des interactions non hiérarchiques dans les systèmes complexes.

### 2.2 Méthodes classiques :

Souvent spécialisées pour la conception d’un certain type de systèmes, les méthodes systémiques définissent différents niveaux de préoccupation ou d’abstraction et proposent de nombreux modèles complémentaires. Autrement dit ces méthodes proposent une double démarche de modélisation  , celles des données et des traitements .Elles sont influencées par les systèmes de gestion de bases de données SGBD. On a comme exemple MERISE, AXIAL RACINE...

### 

Forces de ces méthodes :

* Les concepts sont peu nombreux et simples
* Elle est assez indépendante vis-à-vis de la technologie,
* Bonne adaptation à la modélisation des données et à la conception des bases de données,

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Faiblesses de ces méthodes

* Double démarche de conception celles des données et des traitements,
* Impossible de fusionner les deux aspects (données et traitements)

### 2.3 Méthodes unifiées :

Les méthodes unifiées sont des méthodes de développement pour les logiciels orientés objets. Plus exactement, ce sont les meilleures pratiques du  
développement objet suivies pour la réalisation d’un système. Donc le Processus Unifié (UP) est un processus de développement logiciel «Itératif et incrémental, centré sur l’architecture, conduit par les cas d’utilisation et piloté par les risques » :

* **Itératif et incrémental** : le projet est découpé en itérations de courte durée (environ 1 mois) Qui aident à mieux suivre l’avancement global. A la fin de chaque itération, une partie exécutable du système final est produite, de façon Incrémentale.
* **Centré sur l’architecture** : tout système complexe doit être décomposé en parties Modulaires afin de garantir une maintenance et une évolution facilitées. Cette architecture (Fonctionnelle, logique, matérielle, etc.) doit être modélisée en UML et pas seulement Documentée en texte.
* **Piloté par les risques** : les risques majeurs du projet doivent être identifiés au plus tôt, mais surtout levés le plus rapidement possible. Les mesures à prendre dans ce cadre déterminent l’ordre des itérations.
* **Conduit par les cas d’utilisation** : le projet est mené en tenant compte des besoins et des exigences des utilisateurs. Les cas d’utilisation du futur système sont identifiés, décrits avec précision et priorisés.

#### 2.3.1 Méthodes UP :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Le Processus Unifié (PU ou UP en anglais pour Unified Process) est une méthode de développement logiciel construite sur UML ; elle est itérative et incrémentale, centrée sur l’architecture, conduite par les cas d’utilisation et pilotée par les risques.

itérative et incrémentale : la méthode est itérative dans le sens où elle propose de faire des itérations lors de ses différentes phases, ceci garantit que le modèle construit à chaque phase ou étape soit

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

* affiné et amélioré. Chaque itération peut servir aussi à ajouter de nouveaux incréments.
* conduite par les cas d’utilisation : elle est orientée « utilisateur » pour répondre aux besoins de celui-ci.
* centrée sur l’architecture : les modèles définit tout au long du processus de développement vont contribuer à établir une architecture cohérente et solide.

### pilotée par les risques : en définissant des priorités pour chaque fonctionnalité, on peut minimiser les risques d’échec du projet.

### 2.3.2 Méthode 2TUP :

**2TUP** est un processus de développement logiciel qui implémente le Processus Unifié. Le **2TUP** propose un cycle de développement en Y, qui dissocie les aspects techniques des aspects fonctionnels. Il commence par une étude préliminaire qui consiste essentiellement à :

* identifier les acteurs qui vont interagir avec le système à construire ;
* identifier les messages qu'échangent les acteurs et le système ;
* produire le cahier des charges ;
* modéliser le contexte (le système est une boîte noire, les acteurs l'entourent et sont reliés à lui, sur l'axe qui lie un acteur au système on met les messages que les deux s'échangent avec le sens).

Le processus s'articule ensuite autour de 3 phases essentielles:

* Contraintes fonctionnelles,
* Système d’information de l’entreprise,
* Contraintes techniques.
* **a branche gauche** (fonctionnelle) comporte : la capture des besoins fonctionnels, qui produit un modèle des besoins focalisé sur le métier des utilisateurs. Elle qualifie au plus tôt le risque de produire un système inadapté aux utilisateurs. De son côté, la maîtrise d’œuvre consolide les spécifications et en vérifie la cohérence et l'exhaustivité.
* L'analyse, qui consiste à étudier précisément la spécification fonctionnelle de manière à obtenir une idée de ce que va réaliser le système en termes de métier. Les résultats de l'analyse ne dépendent d'aucune technologie particulière.

### 

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

**La branche droite** (architecture technique} comporte :

La capture des besoins techniques, qui recense toutes les contraintes et les choix dimensionnant la conception du système. Les outils et les matériels sélectionnés ainsi que la prise en compte de contraintes d'intégration avec l'existant conditionnent généralement des prés requis d'architecture technique.

La conception générique, qui définit ensuite les composants nécessaires à la construction de l'architecture technique. Cette conception est complètement indépendante des aspects fonctionnels. Elle a pour objectif d'uniformiser et de réutiliser les mêmes mécanismes pour tout un système. L'architecture technique

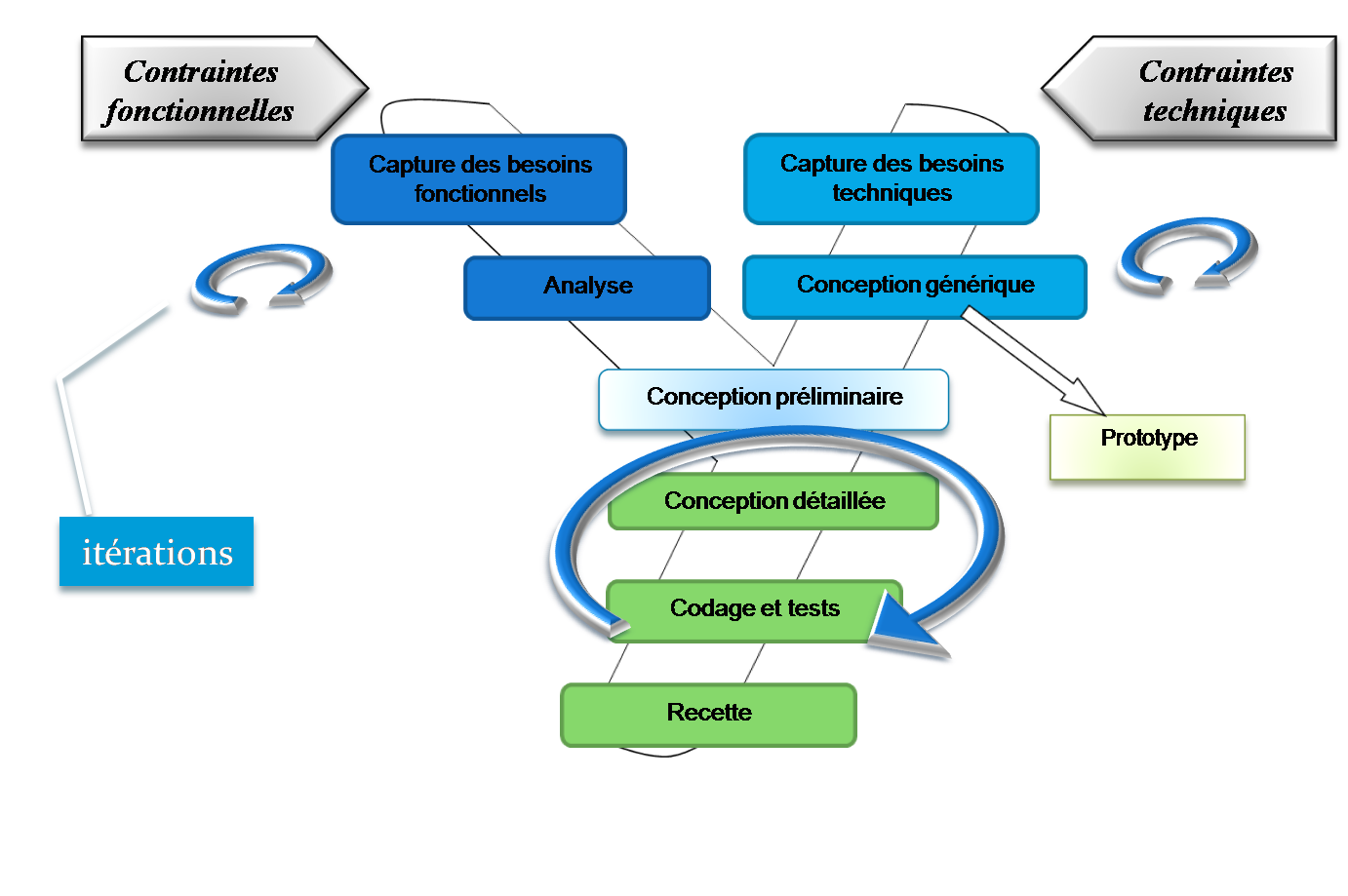
Construit le squelette du système informatique et écarte la plupart des risques de niveau technique. L'importance de sa réussite est telle qu'il est conseillé de réaliser un prototype pour assurer sa validité.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

**La branche du milieu** comporte :

* **la conception préliminaire**, qui représente une étape délicate, car elle intègre le modèle d'analyse dans l'architecture technique de manière à tracer la cartographie des composants du système à développer, la conception détaillée, qui étudie ensuite comment réaliser chaque composant ; l'étape de codage, qui produit ces composants et teste au fur et à mesure les unités de code réalisées
* **la conception détaillée**, qui étudie ensuite comment réaliser chaque composant;
* **le codage et tests**, qui produisent ces composants et teste au fur et à mesure les unités de code réalisées ;
* **l'étape de recette**, qui consiste enfin à valider les fonctions du système développé.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018



### Figure 2.3.1 la 2TUP

### 2.4 Méthodes agiles :

Ces méthodes constituent un gain en productivité ainsi qu'un avantage  
compétitif tant du côté client que du côté du fournisseur. En effet ,ces méthodes de développement dites « méthodes agiles » visent à réduire le cycle de vie du logiciel (donc accélérer son développement) en développant une version minimale, puis en intégrant les fonctionnalités par un processus itératif basé sur une écoute client et des tests tout au long du cycle de développement. Parmi ces méthodes nous pouvons citer : Scrum, XP (eXtreme Programming),   
RAD (Rapid Application Development), etc.…

#### 2.4.1 XP :

Extrême Programming, ou XP, est une méthode agile de gestion de projet particulièrement bien adaptée aux projets de développement informatique. Elle a été conçue par Kent Beck pour accélérer les développements alors qu’il travaillait pour la société Chrysler.L’idée lui est venue alors qu’il devrait intervenir sur un logiciel de paie écrit en langage « Small talk » ayant accumulé une dette technique considérable, le rendant particulièrement complexe à maintenir et à faire évoluer.

Le principe fondamental de la méthode XP est de faire collaborer étroitement tous les acteurs du projet et d’opter pour des itérations de développement très courtes. La planification des tâches reste très souple et l’estimation des charges simplifiée par des projections à très court terme. Ainsi la correspondance entre ce qu’attendent le client et les réalisations est garantie. Les fonctionnalités sont livrées régulièrement, afin d’être testées et validée au travers de prototypes opérationnels. L’Extrême Programming préconise également le travail en binôme des développeurs, facilitant ainsi la production d’un code simple, facilement lisible et maintenable.

#### 2.4.2 Scrum :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Scrum est considéré comme un cadre ou « Framework » de gestion de projet.  
Ce cadre est constitué d'une définition des rôles, de réunions et d'artefacts. Le mot Scrum est un terme emprunté au rugby qui signifie « mêlée » encore appelée « morning **»**. Dans cette méthode, on focalise l'équipe de façon itérative sur un ensemble de fonctionnalités. En effet, Scrum définit 3 rôles :

* Le propriétaire du produit « Product Owner » qui porte la vision du produit à réaliser (représentant généralement le client);
* Le directeur du produit « Scrum Master »garant de l'application de la  
  méthodologie Scrum ;
* L'équipe de développement qui réalise le produit.

La vie d'un projet Scrum est rythmée par un ensemble de réunions clairement  
définies et strictement limitées dans le temps (timeboxing):Planification du Sprint (Sprint = itération) : au cours de cette réunion, l'équipe de développement sélectionne les éléments prioritaires du « Product Backlog » (liste ordonnancée des exigences fonctionnelles et non fonctionnelles du projet) qu'elle pense pouvoir réaliser au cours du sprint (en accord avec le « Product Owner »).Revue de Sprint : au cours de cette réunion qui a lieu à la fin du sprint, l'équipe de développement présente les fonctionnalités terminées au cours du sprint et recueille les feedbacks du Product Owner et des utilisateurs finaux. C'est également le moment d'anticiper le périmètre des prochains sprints et d'ajuster au besoin la planification de release (nombre de sprints restants).  
Rétrospective de Sprint : la rétrospective qui a généralement lieu après la revue de sprint est l'occasion de s'améliorer (productivité, qualité, efficacité, conditions de travail, etc.) à la lueur du "vécu" sur le sprint écoulé (principe d'amélioration continue).Mêlée quotidienne : il s'agit d'une réunion de synchronisation de l'équipe de développement qui se fait debout (elle est aussi appelée "stand up meeting") en 15 minutes maximum au cours de laquelle chacun répond principalement à 3 questions :« Qu'est-ce que j'ai terminé depuis la dernière mêlée ? Qu'est-ce que j'aurai terminé d'ici la prochaine mêlée ? Quels obstacles me retardent ? »

C:\Users\MINA\Desktop\momo IGS\Untitled-Diagram.png

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Figure 2.4.2.1 la méthode Scrum

3. Choix de la méthodologie :

### 3.1 Méthodologie de conception :

Après l’étude des différentes méthodologies, nous avons opté pour Scrum et UP pour les raisons qui suivent :

* Scrum convient aux équipes ayant un nombre de développeurs réduits .Ceci est le cas de notre projet ;
* la progression des tâches s’effectue pendant une durée de développement courte ;
* UP s’applique à tous les types de projet informatique pour les suggestions et remarques ;
* UP est une méthode itérative et incrémental ;
* UP est essentiellement basée sur la satisfaction des besoin set des exigences du clients et des utilisateurs ;
* UP se base sur le diagramme des cas d’utilisation qui illustre au mieux les besoins fonctionnels et opérationnels auxquels doivent répondre le produit final.

### 3.2 Présentation du langage UML :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Les auteurs des méthodes (OOSE de Jacobson, OMT de Rumbaugh, BOOCH deGrady Booch) se sont fixé des objectifs dont entre autre la représentation des systèmes entiers par des concepts objet ; l’établissement d’un couplage explicite entre les concepts et les interfaces exécutables et la création d’un langage de modélisation utilisable à la fois par les humains et les machines et adaptés aux systèmes simples et complexes. D’où la naissance d’UML qui est donc une norme du langage de modélisation objet qui a été publiée, dans sa première version en septembre 1997 par l’OMG (Object Management Group), instance de normalisation internationale du domaine de l'objet.UML est un langage de modélisation graphique et textuel destiné à comprendre et à décrire des besoins, spécifier et documenter des systèmes, esquisser architectures logicielles, concevoir des solutions et communiquer des points des vues.  De plus UML comble une lacune importante des technologies objet, il permet d’exprimer, d’élaborer et de modéliser au sens de la théorie des langages. De ce fait il contient les éléments constitutifs de ces derniers : concepts, une syntaxe et une sémantique.

#### 3.2.1Les diagramme UML :

UML définit 13 types de diagrammes divisés en deux catégories : les diagrammes structurels ou statiques rassemblent :

* diagramme de classe (Class Diagram) ;
* diagramme d’Objet (Object Diagram) ;
* diagramme de composant (Composent Diagram) ;
* diagramme de déploiement (Deployment Diagram) ;
* diagramme de paquetage (Package Diagram)
* diagramme de structure composite (Composite Structure Diagram) .

Les diagrammes comportementaux qui rassemblent :

* diagramme des cas d’utilisation (Use Case Diagram).
* diagramme d’état-transitions (State Machine Diagram) ;
* diagramme d’activité (Activity Diagram) ;
* diagramme de séquence (Sequence Diagram) ;
* diagramme de communication (Communication Diagram) ;
* diagramme global d’interaction (Interaction Overview Diagram) ;
* diagramme de temps (Timing Diagram) ;

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

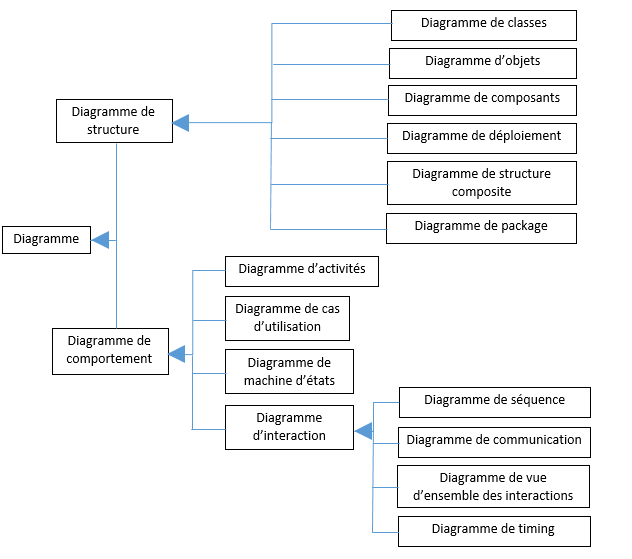


Figure 3 .2.1 Diagramme UML

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

II. Identification des besoins

## Les besoins fonctionnels

Au cours de cette étape, nous allons définir les différentes fonctionnalités qu’offrira notre application :

* Planification et gestion de toutes les missions :
  + - Conformité légale,
    - Prévention des risques (Environnement / Sécurité / Etc.),
    - Observation de tâches,
    - Vérification d’une compétence,
* Planification des calendriers des missions,
* préparation de la programmation, assignation des inspecteurs, enregistrement des résultats et des rapports des missions,
* création d’un formulaire électronique,
* possibilité de joindre des documents, fichiers et liens URL partout et à tout moment,
* suivi de l’exécution et l’identification rapide des retards grâce aux tableaux de bord,
* archivage et protection des informations inactives.

## Les besoins non fonctionnels :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Les besoins non fonctionnels sont importants car ils agissent de façon indirecte sur le résultat et sur le rendement de l’utilisateur, ce qui fait qu’ils ne doivent pas être négligés. Pour cela, il faut répondre aux exigences suivantes :

1. **F IABILITE**

L’application doit fonctionner de façon cohérente, sans erreurs et doit être satisfaisante.

2) **LES ERREURS**

Les ambigüités doivent être signalées par des messages d’erreurs bien

organisés pour bien guider l’utilisateur et le familiariser avec notre plateforme.

3) **ERGONOMIE ET BONNE I NTERFACE**

µ

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

L’application doit être adaptée à l’utilisateur sans qu’il ne fournisse

aucun effort (utilisation claire et facile) du point de vue de la navigation entre les différentes pages, couleurs et mise en texte utilisées.

4)  **SECURITE**

Notre solution doit respecter surtout la confidentialité des données

Personnelles , de même Les différents comptes créés doivent être sécurisés et vérifiés pour éviter les faux comptes et les fausses informations.

5) **Aptitude à la maintenance et la réutilisation**

Le système doit être conforme à une architecture standard et claire permettant sa maintenance et sa réutilisation.

III. Analyse des besoins

## 1. Les acteurs intervenants dans le projet :

## 1.1 Le comité de pilotage :

### Le comité de pilotage est un groupe d'encadreurs chargés de veiller au bon fonctionnement du projet. Il a pour rôle de guider le groupe de travail, de valider les choix méthodologiques et les orientations générales, de définir les moyens à mettre en place pour la réalisation du projet, de coordonner les activités et de donner la validation finale au projet. Il est composé de  la Direction de l’Informatique Financière (DIF) et de l’Inspection Générale des Services.

### 1.2 Les chargés du projet :

La mission essentielle d’un chargé de projet est de conduire le projet à savoir : son étude, sa conception et sa réalisation. Il est en collaboration étroite avec le comité de Pilotage. Il s’agit pour le présent projet des étudiants suivants :

* Mr Hamadou Seydou Abdoul Salam,
* Mr Hassane Wankoye Abdourahamane,
* Mme Mohamed Tourad Moulay Sidi Manitou.

1.3 Le groupe des utilisateurs

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

La mission essentielle d’un chargé de projet est de conduire le projet à savoir : son étude, sa conception et sa réalisation. Il est en collaboration étroite avec le comité de Pilotage. Il s’agit pour le présent projet des étudiants suivants :

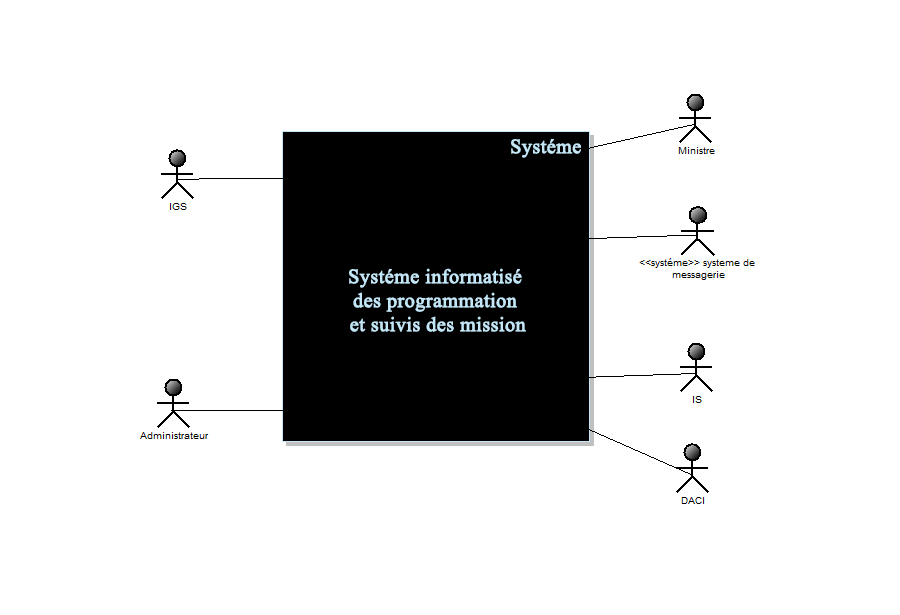
* Mr Hamadou Seydou Abdoul Salam,
* Mr Hassane Wankoye Abdourahamane,
* Mme Mohamed Tourad Moulay Sidi Manitou.
* La Direction d’Audit et de Contrôle Interne des regrées financières (DACI).
  1. Diagramme de contexte statique :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Le iagramme de contexte statique délimite le domaine d’étude en précisant ce qui est à la charge du système et en identifiant l’environnement extérieur au système étudié avec lequel ce dernier communique. Ses composants sont :

* Les acteurs externes. Un acteur externe est un entité externe au système étudié qui interagit avec le système.
* Un processus unique symbolisant le Système Information étudié :

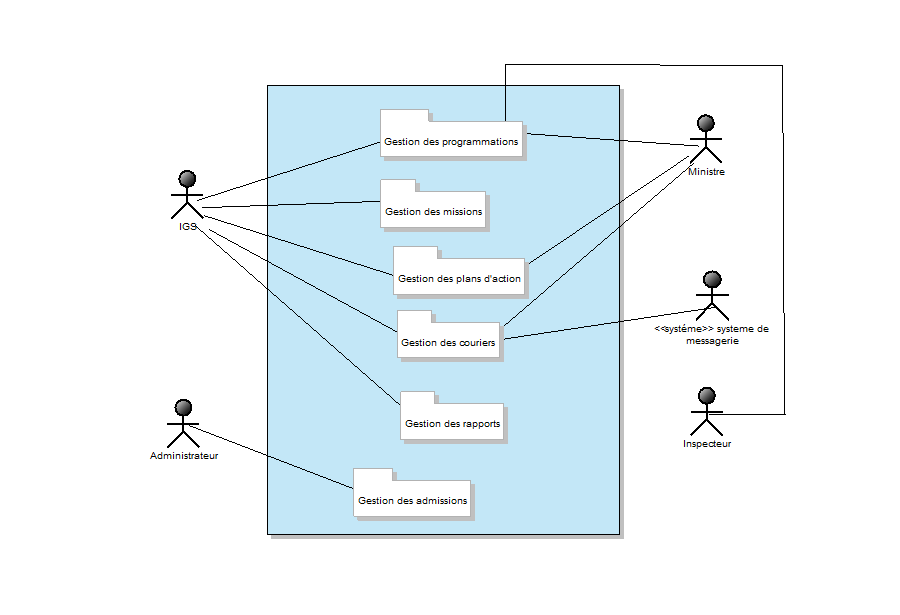
Nom du SI

* Echange entre le système étudié et son environnement

# Figure 2.1 Diagramme de contexte

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

## Diagramme de package :

 Les besoins très différents des acteurs et le nombre de fonctionnalités dont le Futur logiciel devra disposer nous semble assez souvent compliqué. Pour y voir clair et pour nous faciliter la tâche, on peut découper le futur logiciel en parties distinctes, en fonction des « familles » de fonctionnalités et de façon à pouvoir les analyser séparément. Chacune de ces parties correspond à un domaine fonctionnel ou **package** .Il ne nous reste plus qu’à réaliser le **diagramme de packages,** en mettant en évidence les acteurs qui interviennent dans chacun de ces packages. Pour cela, on représente au centre le Système, qui comprend les packages que nous avons a eu à trouvés : La gestion de programmation, la gestion des missions, la gestion des suivis des missions, la gestion des plans d’action, la gestion des suivis des plans d’action, la gestion des courriers, la gestion des rapports et la gestion des admissions. Autour de cela, nous devons donc relier les acteurs principaux et secondaires au package correspondant !

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Figure 3.1 Diagramme de package :

# Chapitre 2 : Conception

* 1. Diagramme des cas d’utilisations :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Ils permettent de décrire l'interaction entre l'acteur et le système. L'idée forte est de dire que l'utilisateur d'un système logiciel a un objectif quand il utilise le système ! Le cas d'utilisation est une description des interactions qui vont permettre à l'acteur d'atteindre son objectif en utilisant le système. Les *use case* (cas d'utilisation) sont représentés par une ellipse sous-titrée par le nom du cas d'utilisation (éventuellement le nom est placé dans l'ellipse). Un acteur « représenté  par un bonhomme dont le nom est indiqué en bas » et un cas d'utilisation sont mis en relation par une association représentée par une ligne. On distingue ainsi les inclusions  ou « include » qui, représentés de A vers B, signifient que le cas d’utilisation A fait toujours appel au cas d’utilisation B pour s’exécuter ; les extensions  ou « extend » qui, représentés de B vers A, signifient que le cas d’utilisation A fait appel au cas d’utilisation B sous certaines conditions.

Le plus souvent, le diagramme des cas est établi par la [maîtrise d’ouvrage](https://fr.wikipedia.org/wiki/Ma%C3%AEtrise_d%27ouvrage) (MOA) d'un projet lors de la rédaction du cahier des charges afin de transmettre les besoins des utilisateurs et les fonctionnalités attendues associées à la [maîtrise d'œuvre](https://fr.wikipedia.org/wiki/Ma%C3%AEtrise_d%27%C5%93uvre) (MOE).

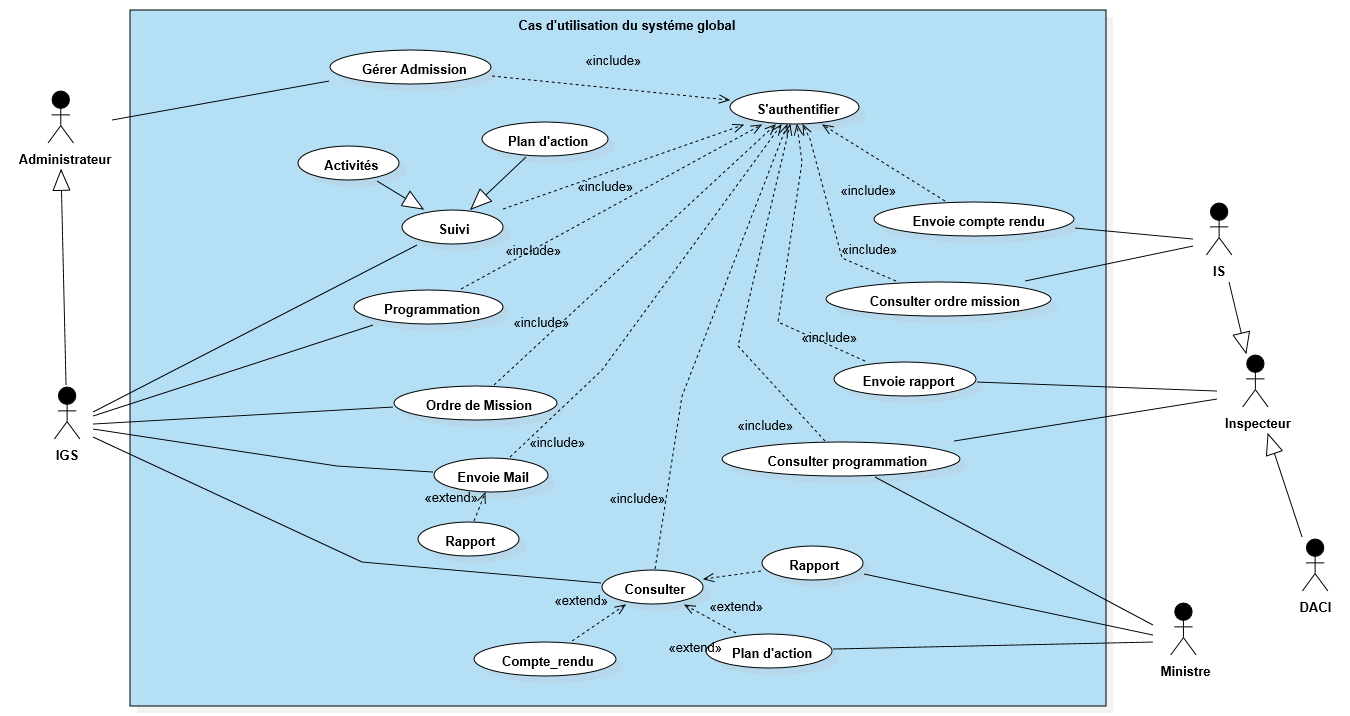


Figure 1.1 Diagramme de cas d’utilisation global :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

En s’appuyant sur la figure 3.1 nous avons les cas d’utilisation suivants :   
1. gestion des admissions : Permet de géré les compte utilisateur ainsi que leur rôle d’accès ;

2. l’authentification : Le système doit bien vérifier les informations de l’utilisateur pour savoir que le compte existe;

3. programmation annuelle : établir le programme annuel ;

4. ordre de mission : établir et soumettre l’ordre de mission non signé au Ministre puis une fois signé il sera transmis aux Inspecteurs de Service ;

5. suivi : assure-le suivi des différentes missions, et plan d’action;

6. envoie mail : permet d’envoyer les ordres de mission non signés, ainsi que les rapports ;

7. consulter : permet d’avoir une vue sur tous les rapports et les comptes rendus ;

8. consulter : permet au Ministre de consulter les rapports disponibles sur le système ainsi que les plans d’actions ;

9. consulter  « Programmation » : permet au Ministre de consulter la programmation annuelle et aux Inspecteurs de voir les programmations annuelles à caractère visible ;

10. envoie rapport : permet aux Inspecteurs d’envoyer leurs rapports ;

11. consulter ordre de mission : permet aux Inspecteurs de Service de consulter leurs ordres de mission ;

12. transmet compte rendu : permet aux Inspecteurs de Service de transmettre leurs comptes rendus.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

A l’issue de l’expression des besoins à l’aide du diagramme des cas d’utilisation globale, dans ce qui suit, nous détaillons chacun des cas d’utilisations présenté dans les figures (1.2) jusqu’à la figure (1.13), en donnant sa description textuelle.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

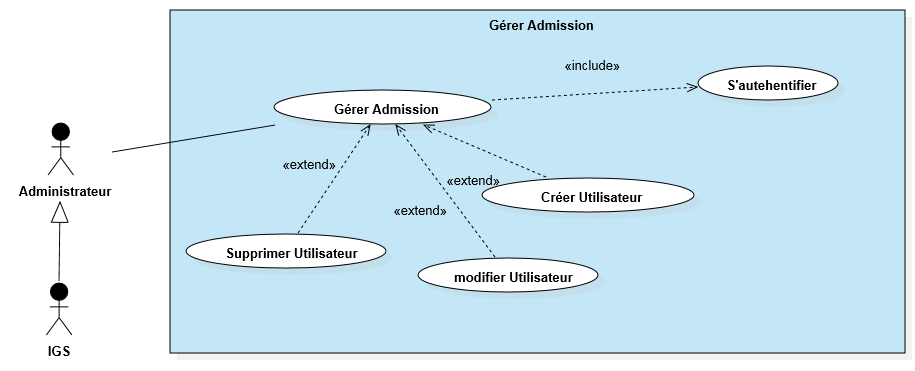


Figure 1.2 Diagramme de cas d’utilisation géré Les admissions

Acteur principal : IGS, Administrateur

Objectif : Pouvoir ajouter, modifier, supprimer un utilisateur

Scenario nominal

Cas1 : Créer un utilisateur

1. l’administrateur choisi d’ajouter un utilisateur,
2. le système affiche le formulaire d’ajout,
3. l’administrateur remplie et valide le formulaire,
4. le système ajoute les informations dans la base,
5. le système actualise la liste des utilisateurs et l’affiche.

Cas 2 : Modifier un utilisateur

1. l’administrateur choisie l’utilisateur à modifier,
2. le système affiche le formulaire de modification,
3. il modifie les champs voulus,
4. Le système met à jour les informations dans la base,
5. Le système actualise la liste des utilisateurs et l’affiche.

Cas 3 : Supprimer l’utilisateur.

1. l’administrateur choisit l’utilisateur à supprimer,
2. le système demande une confirmation,
3. l’administrateur confirme ou annule la suppression,
4. le système supprime l’utilisateur dans la base,
5. Le système actualise la liste des utilisateurs et l’affiche.

Scenario alternatif

Cas 1 :

Utilisateur existe déjà ou champs non conforme aux types, formulaire vide : un message d’erreur sera affiché ,

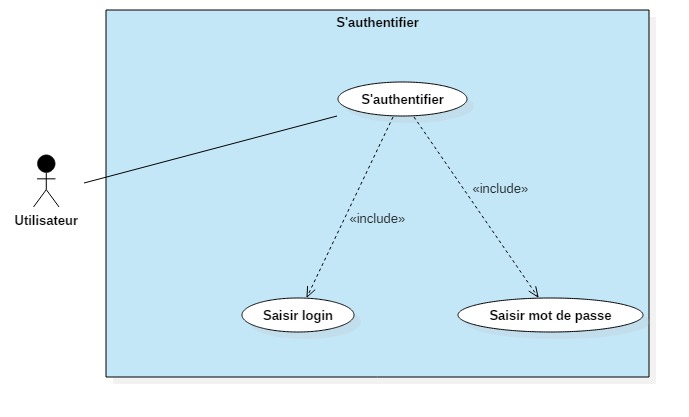
**Cas 2 :**

Modification avec des champs vides, champs non conforme aux types : un message d’erreur sera affiché,

**Cas 3 :**

Utilisateur inexistant : un message d’erreur sera affiché

Mémoire de fin de cycle : Analyste Programmeur 2017-2018



Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Figure 1.3) Diagramme de cas d’utilisation s’authentifier

-Acteur principal : utilisateurs (IGS, administrateur, ministre, IS, DACI)

-Objectif : s’assurer que l’utilisateur est bien celui qui prêtent y être

Scenario nominal :

1. l’utilisateur saisit son nom d’utilisateur et son mot de passe,
2. le système vérifie le nom d’utilisateur et le mot de passe dans la base de données,
3. . le système affiche l’´espace approprié pour chaque utilisateur.

Scenario alternatif

1. Login et mot de passe sont incorrects, un retour vers la page d’authentification sera effectué avec un message d’erreur.

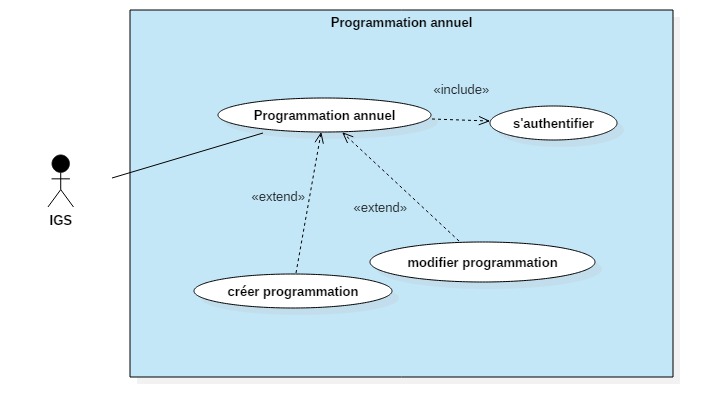


Figure 1.4Diagramme de cas d’utilisation programmation annuel

Acteur principal : IGS.

Objectif : Pouvoir ajouter, modifier une programmation annuel

Scenario nominal

Cas1 : Créer une programmation annuelle

1. l’IGS choisit d’ajouter une programmation annuelle ;
2. le système affiche le formulaire d’ajout ;
3. l’IGS remplie et valide le formulaire ;
4. le système ajoute les informations dans la base ;
5. le système actualise la liste des activités et l’affiche.

Cas 2 : Modifier une programmation annuelle.

1. L’IGS choisie la programmation annuelle. à modifier.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

1. le système affiche le formulaire de modification;
2. il modifie les champs voulus ;
3. le système met à jour les informations dans la base ;
4. le système actualise la liste des activitéss.et l’affiche.
5. Scenario alternatif

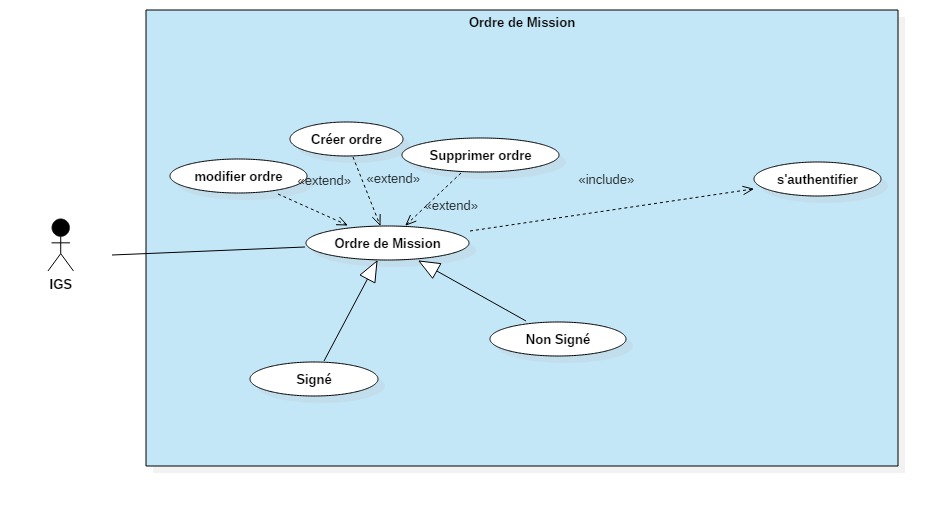
Scenario alternatif

Cas 1 :

Une programmation existe déjà ou champs non conforme aux types, formulaire vide : un message d’erreur sera affiché

Cas 2 :

Modification avec des champs vides, champs non conforme aux types : un message d’erreur sera affiché.



Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Figure 1.5 Diagramme de cas d’utilisation ordre de missions

Acteur principal : IGS.

Objectif : Pouvoir ajouter, modifier, supprimer un ordre de mission non signé

Scenario nominal

Cas1 : Créer un ordre de mission non signé

1. l’IGS choisi d’ajouter un ordre de mission ;
2. le système affiche le formulaire d’ajout ;
3. l’IGS remplie et valide le formulaire ;
4. le système ajoute les informations dans la base ;
5. le système actualise la liste des ordres de missions et l’affiche.

Cas 2 : Modifier un ordre de mission non signé

1. l’IGS choisi l’ordre de mission à modifier ;
2. le système affiche le formulaire de modification ;
3. il modifie les champs voulus ;
4. le système met à jour les informations dans la base ;
5. le système actualise la liste des ordres de missions et l’affiche.

Cas 3 : Supprimer l’ordre de mission non signé

1. l’IGS choisit l’ordre de mission à supprimer ;
2. le système demande une confirmation ;
3. l’IGS confirme ou annule la suppression ;
4. le système supprime l’ordre de mission dans la base ;
5. le système actualise la liste des ordres de missions et l’affiche.

Scenario alternatif

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Cas 1 :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Ordre de mission existe déjà ou champs non conforme aux types, formulaire vide : un message d’erreur sera affiché

Cas 2 :

Modification avec des champs vides, champs non conforme aux types : un message d’erreur sera affiché

Cas 3 :

Ordre de mission inexistant : un message d’erreur sera affiché

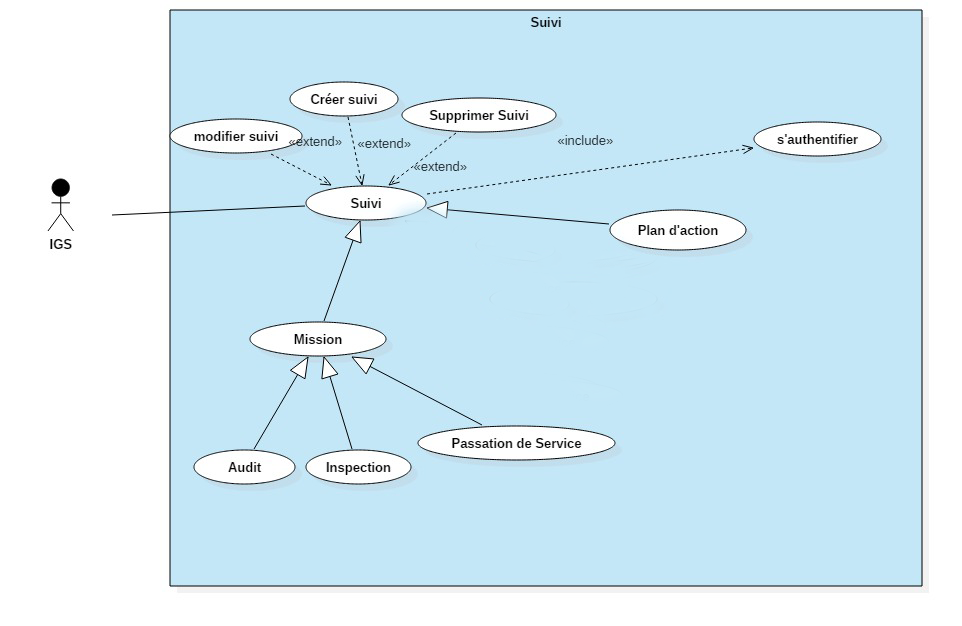


Figure1.6 Diagramme de cas d’utilisation suivi

Acteur principal : IGS.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Objectif : Pouvoir ajouter, modifier, supprimer un suivi

Scenario nominal

Cas1 : Créer un suivit

1. l’IGS choisit d’ajouter un suivi.
2. le système affiche le formulaire d’ajout et la listes des options.
3. l’IGS remplie en faisant son choix (mission, plan d’action) et valide le formulaire ;
4. le système ajoute les informations dans la bas ;
5. le système actualise la liste des suivis et l’affiche.

Cas 2 : Modifier un suivi

1. l’IGS choisit le suivi à modifier ;
2. le système affiche le formulaire de modification ;
3. il modifie les champs voulus ;
4. le système met à jour les informations dans la base ;
5. le système actualise la liste des suivis et l’affiche.

Cas 3 : Supprimer le suivi.

1. l’IGS choisit le suivi à supprimer ;
2. le système demande une confirmation ;
3. l’IGS confirme ou annule la suppression ;
4. le système supprime le suivi dans la base ;
5. le système actualise la liste des suivis et l’affiche.

Scenario alternatif

Cas 1 :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Suivi existe déjà ou champs non conforme aux types, formulaire vide : un message d’erreur sera affiché

Cas 2 :

Modification avec des champs vides, champs non conforme aux types : un message d’erreur sera affiché

Cas 3 :

Suivi inexistant : un message d’erreur sera affiché.

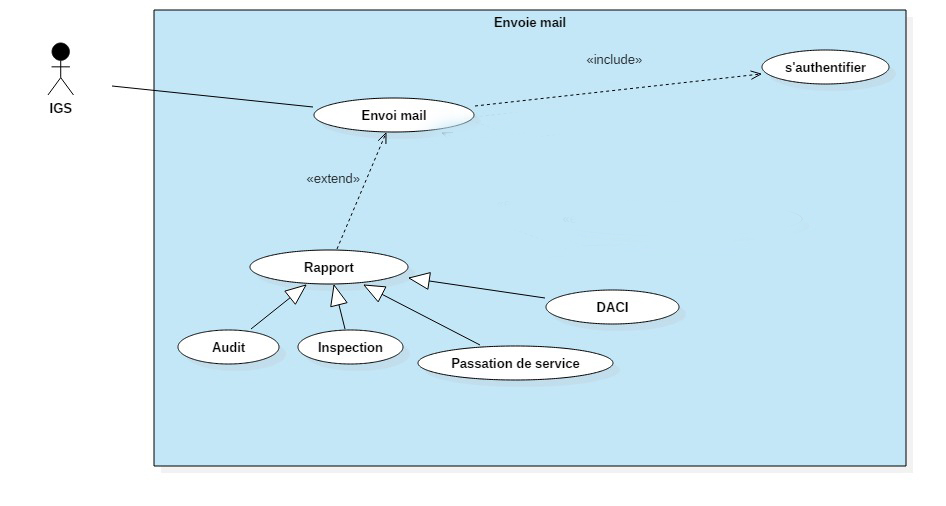


Figure 1.7 Diagramme de cas d’utilisation envois mail

Acteur principal : IGS.

Objectif : Pouvoir envoyer un mail au Ministre

Scenario nominal

L’IGS demande d’envoyé un Email.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

1. le système affiche le formulaire et la liste d’option ;
2. il fait son choix (rapport, ordre de mission non signé) et valide ;
3. le système envoi le mail et affiche un message de confirmation d’envoi mail.

Scenario alternatif

Mail incorrecte, un retour à la page d’envoi mail sera effectué avec un message d’erreur.

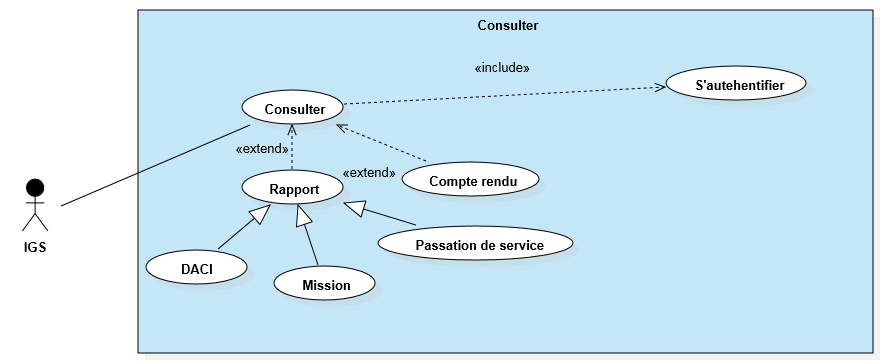


Figure 1.8 Diagramme de cas d’utilisation consulté

Acteur principal : IGS.

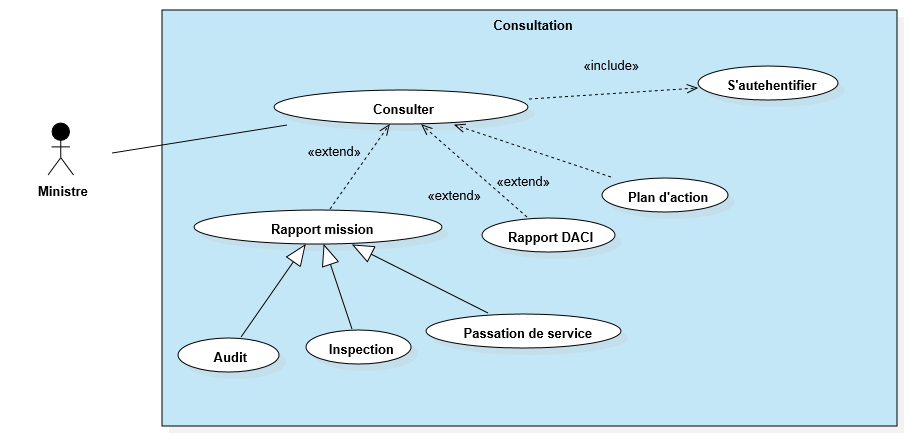
Objectif : Pouvoir consulter les rapports et comptes rendus

Scenario nominal

1. l’IGS demande de consulter les rapports ou comptes rendus ;
2. le système affiche la liste des rapports et comptes rendus ;
3. il fait son choix ;
4. le système lui affiche son choix.

Scenario alternatif

Aucun rapport ou compte rendu existant dans la base de données, le système affiche une alerte.

 Figure 1.9 Diagramme de cas d’utilisation consulté

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Acteur principal : Ministre.

Objectif : Pouvoir consulter les rapports et plan d’action.

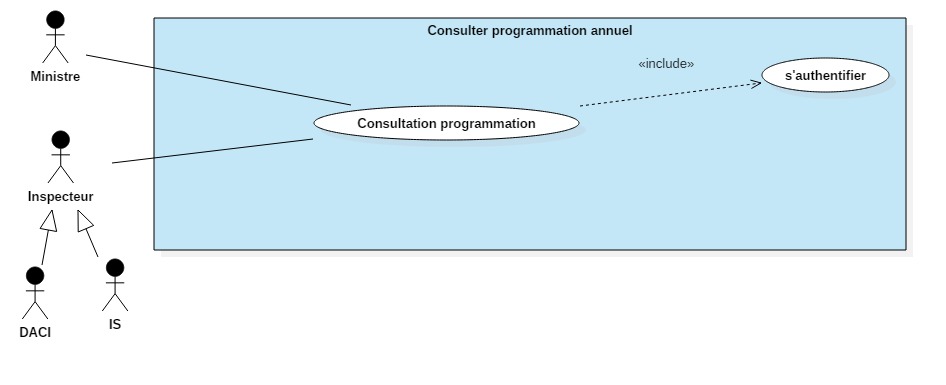
Scenario nominal

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

1. le Ministre demande la liste des rapports (mission, DACI), ou plan d’action ;
2. le système affiche la liste des rapports, ou plans d’actions ;
3. le Ministre fait son choix ;
4. le système affiche l’option choisi.

Scenario alternatif

Aucun rapport ou plan d’action existant dans la base de données le système affiche une alerte.



Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Figure 1.10 Diagramme de cas d’utilisation consulter programmation

Acteur principal : Ministre, Inspecteur (DACI, IS).

Objectif : Pouvoir consulter la programmation annuelle à caractère visible.

Scenario nominal

1. l’utilisateur demande la programmation annuelle ;
2. le système affiche la liste des programmations par an ;
3. l’utilisateur effectue son choix ;
4. le système affiche les détails du choix.

Scenario alternatif

Aucune programmation annuelle concernant existant dans la base de données le système affiche une alerte.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

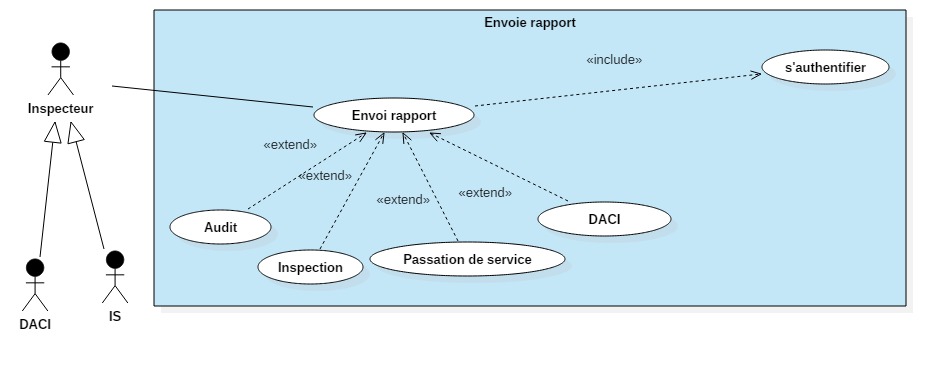


Figure 1.11Diagramme de cas d’utilisation envoi rapport

Acteur principal : Inspecteur (DACI, IS).

Objectif : Pouvoir envoyer les rapports validés

Scenario nominal

1. l’inspecteur demande d’envoyer un rapport ;
2. le système affiche le formulaire ;
3. il remplit le formulaire ;
4. le système envoi le rapport avec un message de confirmation.

Scenario alternatif

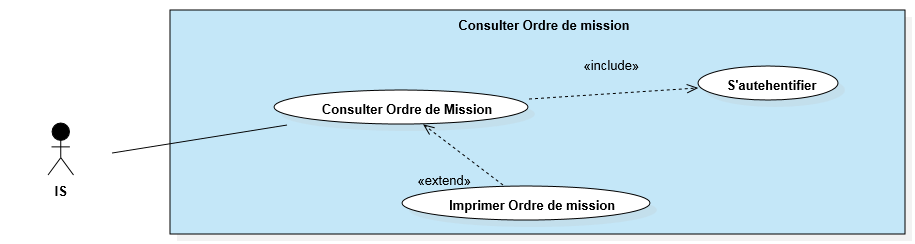
Formulaire incorrect, un retour à la page d’envoi rapport sera effectué avec un message d’erreur. 

Figure 1.12Diagramme de cas d’utilisation consulté ordre de mission

Acteur principal : IS.

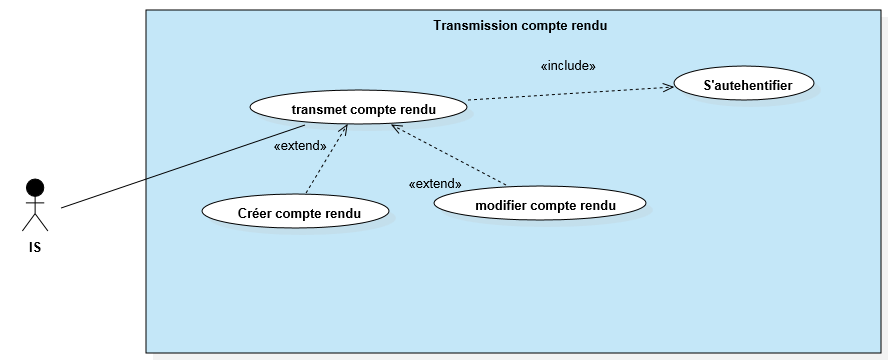
Objectif : Pouvoir envoyer consulter l’ordre de mission et l’imprimer.

Scenario nominal

1. l’IS demande de consulter l’ordre de mission ;
2. le système affiche l’ordre ;
3. il demande de l’imprimer l’ordre de mission ;
4. le système imprime l’ordre de mission.

Scenario alternatif

Aucun ordre de mission le concernant n’existe dans la base de données le système affiche une alerte.



Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Figure 1.13 Diagramme de cas d’utilisation transmission compte rendu

Acteur principal : IS**.**

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Objectif : Pouvoir ajouter, modifier et transmettre un compte rendu

Scenario nominal

Cas1 : Créer un compte rendu

1. l’IS choisit d’ajouter un compte rendu ;
2. le système affiche le formulaire d’ajout ;
3. l’IS remplie et valide le formulaire ;
4. le système ajoute les informations dans la base ;
5. le système actualise la liste des comptes rendus et l’affiche.

Cas 2 : Modifier un compte rendu

1. l’IS choisit le compte rendu à modifier ;
2. le système affiche le formulaire de modification ;.
3. il modifie les champs voulus ;
4. le système met à jour les informations dans la base ;
5. le système actualise la liste des comptes rendus et l’affiche.

Ces 3 : Transmettre un compte rendu

1. l’IS demande la transmission de compte rendu ;
2. le système l’affiche un message de confirmation ou annulation de transmission ;
3. l’IS confirme la transmission ;
4. le système transmet le compte rendu.

Scenario alternatif

Cas 1 :

Compte rendu existe déjà ou champs non conforme aux types, formulaire vide : un message d’erreur sera affiché

**Cas 2 :**

Modification avec des champs vides, champs non conforme aux types : un message d’erreur sera affiché

Cas 3 :

Transmission incorrecte, un retour à la page de compte rendu sera effectué avec un message d’erreur.

## Diagramme de séquences :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation Unified Modeling Language (UML).On .montre ces interactions dans le cadre d’un scénario du diagramme des cas d’utilisation. Dans le souci de simplification, on représente l’acteur principal à gauche du diagramme et les acteurs secondaires éventuels à droite du système. Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets. Les périodes d’activité des classes sont symbolisées par des rectangles. Plusieurs types de messages (actions) peuvent transiter entre les acteurs et objets :

* **Message simple:** le message n'a pas de spécificité particulière d'envoi et  
  de réception.
* **Message avec durée de vie**: l'expéditeur attend une réponse du  
  récepteur pendant un certain temps et reprend ses activités si aucune  
  réponse n'a lieu dans un délai prévu.
* **Message synchrone:** l'expéditeur est bloqué jusqu'au signal de prise en  
  compte par le destinataire.
* **Message asynchrone:** le message est envoyé, l'expéditeur continue son  
  activité que le message soit parvenu, pris en compte ou non. Les  
  messages asynchrones sont symbolisés par des demi-flèches.
* **Message dérobant :** le message est mis en attente dans une liste  
  d'attente de traitement chez le récepteur.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

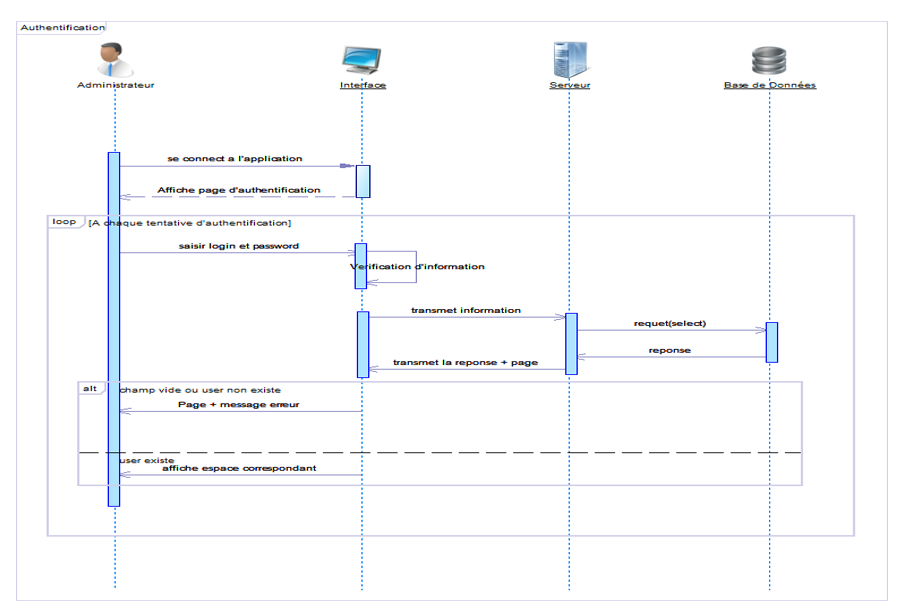


Figure 2.1 **Diagramme de séquence d’authentification**

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

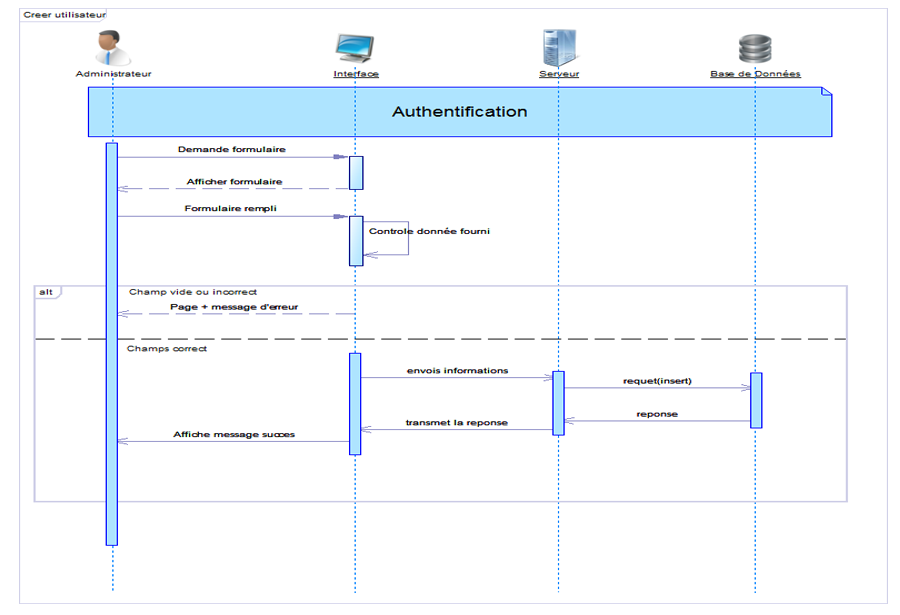
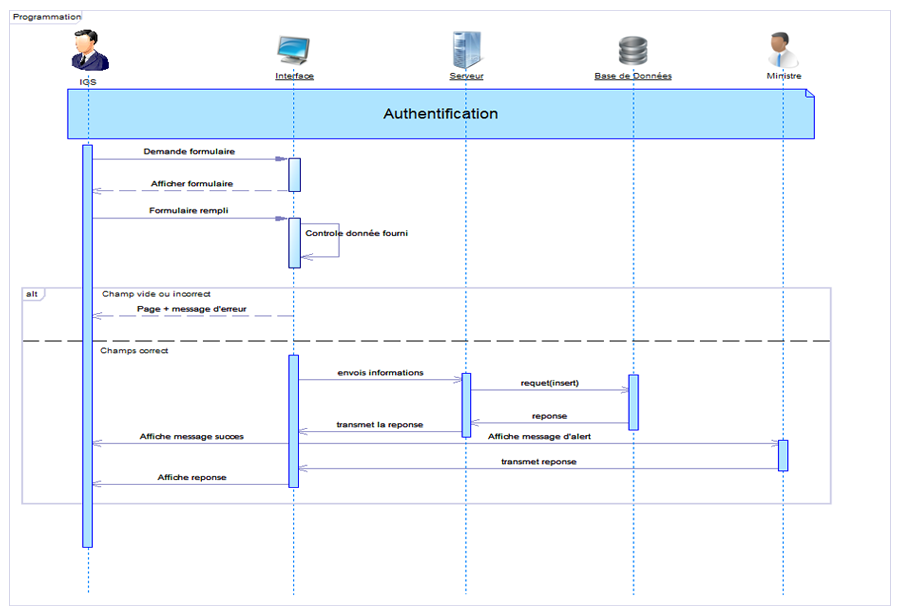


Figure 2.2**Diagramme de séquence créer u**

Mémoire de fin de cycle : Analyste Programmeur 2017-2018



**Figure 2.3 Diagramme de séquence de programmation annuel**

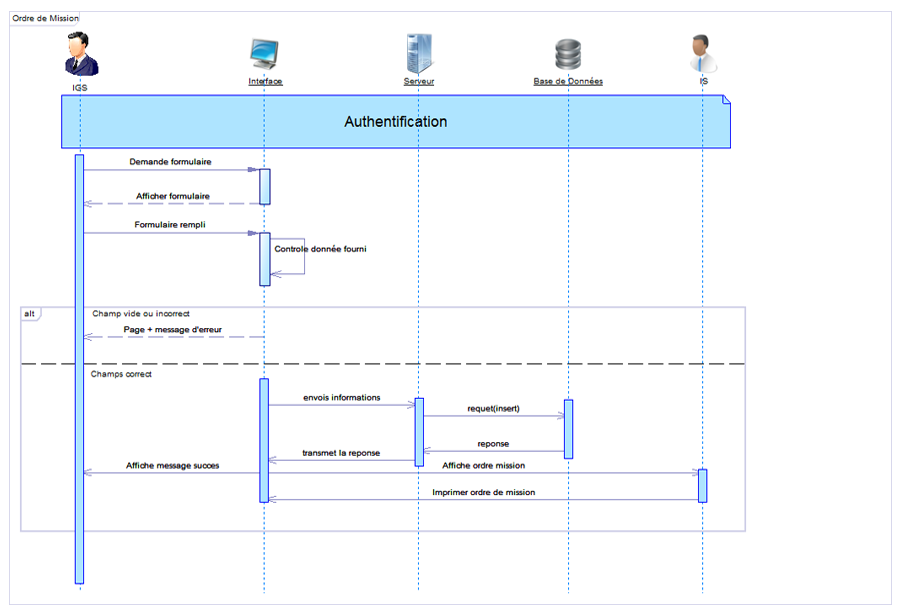


Figure 2.4 **Diagramme de séquence d’ordre de mission**

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

## Diagramme d’activité :

Les diagrammes d’activités décrit le comportement d’une méthodes ,le déroulement d’un cas d’utilisation, les enchainements d’activités .Une activité désigne une suite d’action .Le passage d’une action à une autre est matérialisé par une transition .Les transitions sont déclenchées par la fin d’une action et provoquent le début immédiat d’une autre(elles sont automatiques ).



Figure 3.1 Diagramme d’activité :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

1. Diagramme de classe :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Le diagramme de classes exprime la structure statique du système en termes des classes et de relations entre ces classes. L’intérêt du diagramme de classe est de modéliser les entités du système d’information. Le diagramme de classe permet de représenter l’ensemble des informations finalisées qui sont gérées par le domaine. Le diagramme met en évidence d’éventuelles relations entre ces classes. Le diagramme de classes comporte 5 concepts :

* la notion de classe : Elle définit leur structure, leur comportement et leurs relations. Les classes sont représentées par des rectangles compartimentés en trois (3) parties. Une première partie contenant le nom de la classe, une deuxième représentant les attributs de la classe et enfin la troisième partie représentant les opérations de la classe ;
* la notion d’attribut : C’est un ensemble d’informations élémentaires d’une classe. Un attribut représente la modélisation d’une information élémentaire représentée par son nom et son format. UML définit 3 niveaux de visibilité pour les attributs : les attributs publics notés (+) ; les attributs privé notés (-) et les attributs protégés notés (#).
* la notion d’identifiant : c’est un attribut particulier qui permet de repérer de façon unique chaque objet, instance de la classe. ;
* la notion d’opération : l’opération représente un élément de comportement des objets, défini de manière globale dans la classe. Une opération est une fonctionnalité assurée par une classe. La description des opérations peut préciser les paramètres d’entrée et de sortie ainsi que les actions élémentaires à exécuter. Tout comme les attributs, les opérations ont des niveaux de visibilité : la visibilité des opérations publiques est notée (+) ; la visibilité des opérations privées est notées (-) et enfin (#) pour la visibilité des opérations protégées ;

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

* la notion de relation : S’il existe des liens entre objets, cela se traduit nécessairement par des relations qui existent entre leurs classes respectifs .Les liens entre les objets doivent être considérer comme des instances de relation entre classes .

Mémoire de fin de cycle : Analyste Programmeur 2017-2018



Mémoire de fin de cycle : Analyste Programmeur 2017-2018

# Figure 4 .1 Diagramme de classe

## Diagramme de déploiement :

Il indique l'organisation matérielle de l'application à concevoir et spécifie-les

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Composants physiques nécessaires pour l'application



Figure 5.1 Diagramme de déploiement



Mémoire de fin de cycle : Analyste Programmeur 2017-2018

# Chapitre1:Architecture et description des outils et technologies utilisés

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

## Architecture interne de l’application :

Une architecture est un modèle générique et conceptuel qui se rapporte à un sujet et qui représente la fonctionnalité, la structure, le positionnement, l’interrelation des différentes types d’élément qui la composent.

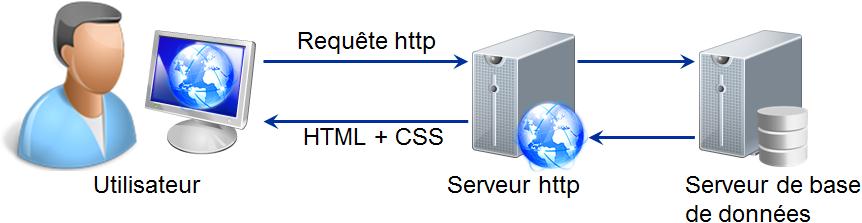
En règle générale, une application est découpée 3 couches d’abstraction :

* la couche présentation ; c’est la partie de l’application visible par les utilisateurs (nous parlons de l’interface utilisateur).Dans notre cas cette couche est un navigateur web, qui se présente sous forme de page HTML, composé de formulaire et de bouton ;
* la couche métier : correspond à la partie fonctionnelle de l’application, celle qui implémente la logique, et qui décrit les opérations que l’application opère sur les données en fonction des requêtes d’un utilisateur effectuées à travers la couche présentation :
* la couche accès aux données : c’est la partie gérant l’accès à la base des données du système.

Il existe différentes architecture dont entre autre ;

* l’architecture 1\_tiers ;
* l’architecture 2\_tiers ;
* l’architecture 3\_tiers ;
* l’architecture n\_tiers.

Dans notre application nous avons utilisé l’architecture 3\_tiers qui est illustrée dans la figure qui suit :

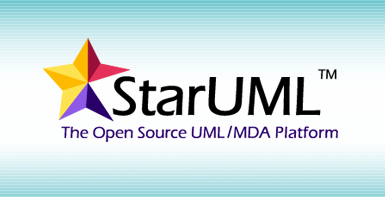


## Outils utilisés :

* PowerAMC :

PowerAMC fait partie des AGL (Atelier de Génie Logiciel), programme qui assiste le développeur au moment de la conception de certaines tâches, les AGL sont très pratique compte tenu de leur souplesse, leur efficacité et permettent aux développeurs de gagner plus de temps dans la réalisation de certain nombre de travaux ;

* StarUML :

 C’est un logiciel de modélisation UML open source sous une licence modifiée de GNU GPL.L'objectif de ce projet est de se substituer à des solutions commerciales comme IBM Rational Rose ou Borland Together. StartUml gère la plupart des diagrammes spécifiés dans la norme Uml 2.0 ;

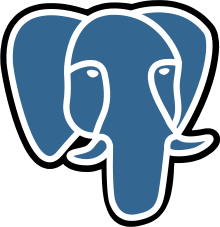
* Drow io :

 Draw.io pro est une application création de diagrammes compatible avec Google Drive(TM) et entièrement gratuite vous permettant de dessiner ( organigrammes, diagramme UML, diagramme ERD, schémas réseaux, Business Process, Models ….) et fonctionnalités( Client HTML5 natif avec support d'IE 6-8,Large bibliothèque d'icones, Interface intuitive en glisser-deposer…).

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

## 1.3 Outils de développement :

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

* **PostgreSQL :**

PostgreSQL est un système de gestion de base de données relationnelle et objet(SGBDRO).C’est un outil libre disponible selon les termes d’une licence de type BSD. Ce système est concurrent d’autre système de gestion de base de données, qu’ils soient libre( comme MariaDB et Firebird), ou propriétaires(comme Oracle, MySql, Sybase,DB2,Informix et Microsoft SQL, Server).Come les projets libres Apache et Linux.PostgreSQL n’est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d’entreprises ;

* **Apache**

 Le logiciel **Apache** est un **serveur** http en Open Source Utilisé principalement sur les hébergements Internet en Linux, bien qu’il soit également utilisable en Windows (concurrent d’Internet et Information Service-IIS), Unix ou OSX. C’est actuellement le plus utilisé sur le WEB ;

* **Xamp**

Xamp est un ensemble de logiciels permettant de mettre en place facilement un serveur Web local, un serveur FTP et un serveur de messagerie électronique. Il s’agit d’une distribution de logiciels libres offrent une bonne souplesse d’utilisation, réputée pour son installation simple et rapide. Ainsi il est à la portée d’un grand nombre de personnes puisqu’il ne requiert pas de connaissance particulières et fonctionne, de plus, sur les systèmes d’exploitation les plus répandus ;

## Outils de conception :

* **Google Chrome**

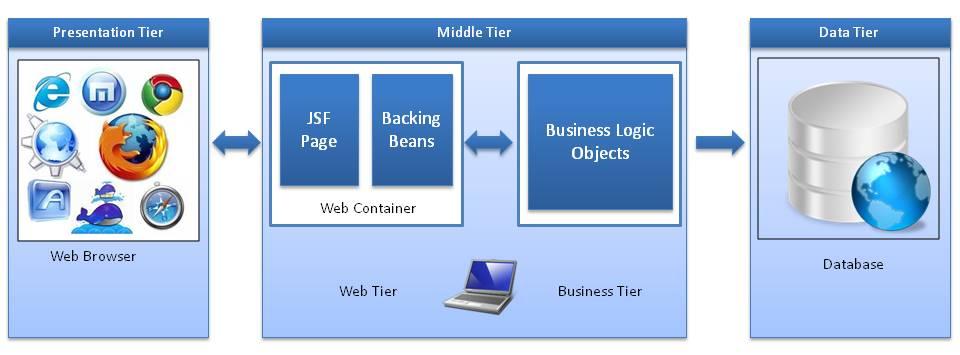
 Google Chrome est un navigateur web créé par Google et basé sur le projet open sourceChromium. L'objectif initial de Google en développant Chrome était de fournir auxinternautes un nouveau navigateur plus rapide et proposant plusieurs innovations par rapportà Mozilla Firefox ou à Microsoft Internet Explorer. En plus des questions de performances,Google Chrome se différencie notamment de sa concurrence en dédiant un espace mémoire àl'utilisateur et un processus unique pour chaque fenêtre ou nouvel onglet, afin d'assurer unemeilleure stabilité et sécurité.

* **Mozilla Firefox**

 Mozilla Firefoxest un navigateur Web libre et gratuit, développé et distribué par la MozillaFoundation avec l'aide de centaines de bénévoles grâce aux méthodes de développementdu logiciel libre/open source et à la liberté du code source.Firefox est à l'origine un programme dérivé du logiciel Mozilla (aujourd'hui connu sous lenom de Sea Monkey),mais reprenant uniquement les fonctions de navigation de ce dernier.Ce logiciel multiplateforme est compatible avec diverses versions de Windows, Mac OSX etGNU/Linux ;

* **MVC**

Mémoire de fin de cycle : Analyste Programmeur 2017-2018



Le paradigne MVC est un schéma de programmation qui propose de sépare une application en trois parties :

* Le modèle : qui contient la logique et l’état de l’application ;
* La vue : qui représente l’interface utilisateur ;
* Le contrôleur qui gère la synchronisation entre la vue et le modèle.
* **Sublime text**

## téléchargement (2)Un environnement de développement intègre (EDI) est un programme regroupant un ensemble d’outils pour le développement de logiciels. En règle générale, un EDI regroupe un éditeur de texte, un compilateur, ou souvent un débogueur. Pour notre cas nous avons utilisé Sublime Text non seulement pour ses couleurs syntaxiques personnalisables mais aussi pour ces fonctions…etc

## Autre outils :

* **Photoshop**

## 

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Pour le design de notre application, nous avons utilisé **Adobe Photoshop** qui est un logiciel de retouche, de traitement et de dessin assisté par ordinateur édité par *Adobe*. Il est principalement utilisé pour le traitement de photographies numériques, mais sert également à la création d’images. *Photoshop* est un logiciel travaillant sur images matricielles (également appelées *bitmap*, à ne pas confondre avec le format d’enregistrement *Windows bitmap*) car les images sont constituées d’une grille de points appelés pixels.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

## Language de programmation utilisé:

1. **PHP**

PHP (abréviation de Personal Home Page Hypertext Préprocesseur )à ce jour est un langage se scripts que s’intègre aux pages Html et qui permet sz réaliser des pages dynamiques .La première version date 1994 et s’appelait PHP FI. Elle n’avait pour ambition que de pouvoir insérer quelques traitement simples dans une page Html, comptage des visites.PHP s’exécute coté serveur, c’est-à-dire que le serveur interprète, analyse et exécute le script PHP avant d’en envoyer le résultat au client (navigateur).De ce fait, le client n’interagit pas directement avec PHP ce qui renforce la sécurité de l’application. De plus PHP est un produit « Open Source » c’est-à-dire que le code est accessible à tout développeur. En plus, il propose un accès facile aux bases de données. Malgré son manque de puissance face à d’autre langage de programmation (Perl, C, Java, JEE, etc …).PHP n’en demeure pas moins un langage solide et accessible à tous.

## 

* **HTML5**

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

HTML (HyperText Markup Language) est un langage de balisage (un langage qui utilise des balises ‘<>’) servant à l’écriture de page du Word Wide Web (WWW). Il permet :

* de structurer sémantiquement et de faire la mise en page du contenu des pages ;
* d’inclure des ressources multimédias dont des images ;
* de réaliser des formulaires de saisie, etc.

Il est souvent utilisé conjointement avec langages de programmation comme PHP, JavaScript et les CSS. HTML5 est l’évolution la plus récente d’HTML. Cette évolution consiste en une multitude des nouvelles fonctionnalités qui ont été apportées au langage HTML ainsi qu’au JavaScript. Si vous savez faire de l’HTML « classique » vous devriez donc apprendre à manipuler ces nouvelles fonctionnalités.

* **CSS**

CSS (Cascading Style Sheets : feuille de style en cascade) est un langage informatique qui sert à décrire la présentation des pages Web CSS s’applique à une ou plusieurs pages du site. Le terme « en cascade » indique que la mise en forme d’une page peut faire appel) plusieurs feuilles de style. La nouvelle version de CSS est le CSS3 qui a apporté au langage de nombreuses nouveautés. Il s’agit par exemple d’un ensemble de nouveaux effets à appliquer sur nos éléments HTML, d’un ensemble d’un ensemble de nouveaux sélecteurs, de nouvelles manières de spécifier les couleurs, une détection des caractéristiques de l'appareil de l'utilisateur, des calculs dans les feuilles de style, des SVG en arrière-

plan... Bref, le CSS3 est un généreux enrichissement des feuilles de style qui, en plus d'être profitable à l'utilisateur, l'est également pour le développeur.

* **Jquery**

Mémoire de fin de cycle : Analyste Programmeur 2017-2018



JQuery est une bibliothèque (c'est-à-dire un ensemble de codes prêts à  
l'emploi) conçue pour simplifier l'écriture de codes JavaScript et AJAX. Créée en 2006 par John Resig, cette bibliothèque est la plus célèbre et la plus utilisée à ce jour.

* **Ajax**

AJAX (acronyme d'Asynchronous JavaScript And XML.) est apparu en 1995. Son utilisation est très intéressante, car elle permet de mettre à jour une partie (et une partie seulement) d'une page Web en demandant les données nécessaires à un serveur. Les échanges client/serveur sont donc limités et les pages Web sont affichées plus rapidement, pour le plus grand plaisir des utilisateurs.

* **JavaScript**

Le JavaScript est un langage de script incorporé dans un document HTML. Historiquement il s’agit même du premier langage de script pour le Web. Ce langage est un lange de programmation qui permet d’apporter des améliorations au langage HTML, en permettant d’exécuter des commandes du coté client, c’est-à-dire au niveau du navigateur et non du coté serveur web. Ainsi le langage JavaScript est fortement dépendant du navigateur appelant la page web dans laquelle le script est incorporé, mais en contrepartie il ne nécessite pas de compilateur, contrairement au langage Java, avec lequel il a longtemps été confondu .Il peut également être utilisé coté serveur comme PHP.

* **bootstrap**

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Bootstrap est un Framework CSS mais pas seulement, puisqu'il embarque également des composants HTML et JavaScript. Il comporte un système de grille simple et efficace pour mettre en ordre l'aspect visuel d'une page web. Il apporte du style pour les boutons, les formulaires, la navigation… Il permet ainsi de concevoir un site web rapidement et avec peu de lignes de code ajoutées.

* **Materialize**

Materialize comme son nom l’indique, c’est un Framework basé sur le Material Designe. L’avantage du Material Designe est que sa permet de créer une interface propre rapidement .Pour les amoureux de Bootstrap, Materialize possède le même système de grid mais il intègre en plus une série de composants JavaScript, comme par exemple des effets de vague sur les boutons.

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

[Tapez le titre du document]

[Tapez le sous-titre du document]

[Sélectionnez la date]

[Tapez le nom de la société]

utilisateur

[Tapez le titre du document]

[Tapez le sous-titre du document]

[Sélectionnez la date]

[Tapez le nom de la société]

utilisateur

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

[Tapez le titre du document]

[Tapez le sous-titre du document]

[Sélectionnez la date]

[Tapez le nom de la société]

utilisateur

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

[Tapez le titre du document]

[Tapez le sous-titre du document]

[Sélectionnez la date]

[Tapez le nom de la société]

utilisateur

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

[Tapez le titre du document]

[Tapez le sous-titre du document]

[Sélectionnez la date]

[Tapez le nom de la société]

utilisateur

Mémoire de fin de cycle : Analyste Programmeur 2017-2018

[Tapez le titre du document]

[Tapez le sous-titre du document]

[Sélectionnez la date]

[Tapez le nom de la société]

utilisateur

[Tapez le titre du document]

[Tapez le sous-titre du document]

[Sélectionnez la date]

[Tapez le nom de la société]

utilisateur