Introduction

Dans le cadre de la formation d'ingénieurs à l'Ecole Supérieur des Sciences Appliquées et de technologie privée de Gabes, nous sommes appelés à consolider notre formation théorique par des connaissances et des acquis pratiques à travers des projets dont le plus évaluatif en terme de contenu de la formation est celui de fin d'études, c'est dans ce cadre que s'articule notre application que nous sommes appelés à développer (concevoir et réaliser).

La compétitivité des entreprises repose essentiellement sur leur capacité à utiliser efficacement les ressources dont elles disposent afin de satisfaire au mieux la demande de leurs clients. Selon le contexte, le terme « ressource » peut renvoyer à des notions très différentes telles que les matières premières, l'outil de production, les canaux de distribution, ou le personnel. Nous nous intéressons ici à la gestion efficace du personnel et de document qui est primordiale pour de nombreuses entreprises.

Les objectifs d'une bonne organisation administrative sont de faire vite, bien et pas cher. Donc d'être productif en maintenant un excellent niveau de qualité dans les tâches exécutées. En effets, ces tâches ne produisent pas de valeur ajoutée, mais contribuent à ce que l'ensemble soit bien huilé.

La gestion de tâches, ça peut être aussi simple qu'une liste de tâches notées sur une feuille, que vous cochez au fur et à mesure (c'est ce que font un grand nombre de gens). Bien sûr, des applications permettent aujourd'hui de faire la même chose sur votre ordinateur ou votre téléphone, avec plus ou moins d'options superflues. Mais, Quand vous devez gérer plusieurs tâches et plusieurs membres de votre équipe, les choses peuvent rapidement devenir ingérables avec une application de gestion de tâches basique.

Aussi comme la gestion de tâches, un bon système de gestion documentaire, correctement communiqué à l'ensemble des employés est donc un outil précieux, voire essentiel au bon fonctionnement d'une entreprise. Ce n'est plus à l'utilisateur d'aller à la recherche du document, mais c'est le document qui vient directement vers l'utilisateur.

C'est dans cette optique que se situe notre projet de fin d'études. Il s'agit de développer une application inter service multiplateforme modulaire.

Le présent rapport trace les phases du déroulement du projet. Il sera présenté en trois chapitres :

Le premier chapitre intitulé Etude de l'existant. Dans ce chapitre, nous décrivons l'analyse et les critiques des applications existantes.

Le chapitre suivant intitulé conception est dédié à la présentation de l'architecture ainsi que la conception des patrons utilisés et des modèles de données.

Le dernier chapitre intitulé implémentation de la solution présente les étapes de la réalisation du projet.

Le rapport s'achève par une Conclusion Générale dans laquelle nous exposerons les perspectives du projet.

Chapitre 1

Etude Préalable

L'étude préalable constitue une étape préliminaire pour la réalisation d'une application.

En effet, elle permet d'analyser, d'évaluer et de critiquer le fonctionnement habituel, tout en élaborant la liste des solutions possibles.

Ce chapitre sera réservé pour présenter l'étude préalable de notre projet. Nous commençons par la description de l'existant. Ensuite l'analyse et le critique de l'existant nous ont permis de cerner nos objectifs afin de développer un système de qualité dans le futur. Enfin, nous proposons les différentes solutions aux problèmes soulevés.

Description de l'existant

Les gestionnaires de tâches sont plus élaborés que les to-do lists. Ces dernières se résument à des listes de choses à faire que l'on peut classer par ordre d'importance par exemple. Elles permettent souvent d'ajouter des notes au format texte enrichi (mise en gras, couleurs, images). Les to-do lists sont là pour noter les bonnes idées quand elles nous viennent à l'esprit : elles doivent donc être facilement accessibles.

Cependant les "todo" ne sont pas suffisantes pour s'organiser et se mettre dans une dynamique de recherche de productivité. Pourvoir faire plus avec moins, c'est justement le créneau des outils de gestion de tâches.

Le logiciel de gestion des tâches aide les collaborateurs et leurs équipes à gérer les tâches individuelles et collectives en organisant le flux de travail quotidien. Les outils de gestion des tâches adressent des problématiques de productivité et de collaboration en créant des listes de tâches partagées qui indiquent les dates de début et de fin, décrivent les éléments importants à réaliser pour chaque tâche.

Les individus utilisent généralement un logiciel de gestion de tâches pour suivre l'avancement de leur travail de manière opérationnelle en opposition aux grands objectifs de l'équipe projet. Ces outils peuvent être utilisés dans pratiquement n'importe quelle industrie, dès lors que les individus ont besoin de centraliser et clarifier leur organisation et d'accomplir leur travail quotidien avec efficacité.

Les fonctionnalités classiques :

- Créer et affecter des tâches à des collaborateurs
- Définir une échéance
- Permettre aux individus de gérer leurs tâches personnelles
- Modifier les statuts d'une tâche (à faire, en cours, terminé)
- Focaliser le travail sur une mission précise et non sur un projet

D'un point de vue purement personnel, le gestionnaire de tâche permet :

- De moins procrastinée (remettre à plus tard les choses à faire aujourd'hui)
- De prioriser son travail personnel
- D'être plus efficace dans sa vie professionnelle
- De faire plus de choses en stressant moins (productivité)

Critique de l'existant

Une mauvaise organisation dans la gestion des tâches peut mener à des résultats chaotiques qui sont catastrophiques pour notre productivité.

Que l'on travaille seul ou en équipe, il est donc primordial d'utiliser les outils à notre disposition à leur plein potentiel.

Inconvénients

Il y a peu d'inconvénients à un gestionnaire de tâches : ils sont presque indispensables dans tous les domaines.

Pour autant on peut noter quelques points qui peuvent être décevants :

- Ils ne gèrent pas les workflows ou les étapes de travail.
- Il peut devenir complexe et illisible.
- Développer sur mesure selon la taille et le nombre des personnes qui utilise l'application.
- La suppression des tâches est parfois trop facile : une mauvaise manipulation peut entrainer une suppression sans s'en rendre compte.

Solution proposée

Fini les petits papiers collés à l'ordinateur pour ne plus oublier vos différentes missions de la journée ! Il vous faut un support approprié pour vous assister à la gestion de vos tâches.

Sur votre smartphone, votre tablette ou votre ordinateur, Mac ou Pc, votre nouvel outil doit être disponible et consultable de partout. Pour cela, une application web répondra parfaitement à vos besoins. Sans rien installer, vous pourrez l'utiliser via votre navigateur. Vous êtes mobiles, elle le sera avec vous et vous permettra de gagner du temps et d'être plus productif.

- De plus, vous pourrez en suivre la progression en temps réel.
- Dématérialisation, Partage et synchronisation des documents.

- Vous pourrez aisément assigner les différentes tâches à vos collaborateurs avec une date butoir, la description du travail, des commentaires, des pièces jointes…
- Plus de problème pour trouver la dernière version d'un document, l'avancée d'une tâche, le responsable d'une mission… Tout sera centralisé et simple d'utilisation, avec une interface claire et ergonomique.
- Assuré la sécurité de l'information et l'accès au document.
- Une solution qui englobe à la fois la gestion de tache et la gestion de document.
- Un espace sociaux collaboratif qui permet de crée des évènements.
- Moins d'emails, optimisation du temps, moins de réunions, plus de temps pour la production et les idées.

Conclusion

Dans ce chapitre, nous avions pu insérer notre projet dans son contexte en présentant la solution adoptée pour résoudre les problèmes et qui répond à nos besoins. Dans le chapitre suivant, nous allons présenter la conception qui a été mise en œuvre tout au long de la réalisation de ce projet.

Chapitre 2

Conception

Lors de ce chapitre, nous allons identifier les diagrammes de classes, de cas d'utilisation et de séquence réalisés pour mettre en œuvre l'architecture de notre application. La motivation fondamentale de la modélisation est de fournir une démarche antérieure afin de réduire la complexité du système étudié lors de la conception et d'organiser la réalisation du projet en définissant les modules et les étapes de la réalisation. Plusieurs démarches de modélisation sont utilisées.

Nous adoptons dans notre travail une approche objet basée sur un outil de modélisation UML.

Choix de la méthodologie de conception :

Dans la cadre de notre projet, nous avons opté pour le langage UML comme une approche de conception. Ci-dessous, nous présentons ce langage puis nous justifions notre choix.

Présentation d'UML :

UML (Unified Modeling Language) est un langage formel et normalisé en termes de modélisation objet. Son indépendance par rapport aux langages de programmation, aux domaines de l'application et aux processus, son caractère polyvalent et sa souplesse ont fait lui un langage universel. En plus UML est essentiellement un support de communication, qui facilite la représentation et la compréhension de solution objet. Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation des solutions. L'aspect de sa notation, limite l'ambigüité et les incompréhensions.

UML fournit un moyen astucieux permettant de représenter diverses projections d'une même représentation grâce aux vues.

Une vue est constituée d'un ou plusieurs diagrammes. On distingue deux types de vues :

- **La vue statique**, permettant de représenter le système physiquement :
  - Diagrammes de classes : représentent des collections d'éléments de modélisation statiques (classes, paquetages…), qui montrent la structure d'un modèle.

- Diagrammes de cas d'utilisation : identifient les utilisateurs du système (acteurs) et leurs interactions avec le système.
- **La vue dynamique**, montrant le fonctionnement du système :
- Diagrammes de séquence : permettent de représenter des collaborations eu objets selon un point de vue temporel, on y met l'accent sur la chronologie (envois de messages).

Diagramme des cas d'utilisation :

Les cas d'utilisation décrient un ensemble d'actions réalisées par le système, en réponse à une action d'un acteur.

Identification des acteurs :

L'identification des acteurs sert à délimiter le contour du système. Par définition : 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(Bourque,2014).

- Administrateur : : cet acteur possède tous les droits d'accès qui lui permettent d'administrer le système.sa fonction principale est la gestion des manager.
- Manager : cet acteur est la gestionnaire de l'application .sa fonction principale est la gestion des taches et des documents.
- Utilisateur : cet acteur est en interaction avec le manager par la tache assignée.

Identification des cas d'utilisation :

Nous décrivons pour chaque acteur les cas d'utilisation. On distingue les cas d'utilisation suivants :

- Utilisateur :

  - S'inscrire.
  - S'authentifier.
  - Gere le profile (mettre à jour ses informations personnel).
  - Consulter la liste des taches.
  - Modifier une tache.
  - Ajouter un document
  - Ajouter un évènement.
  - Modifier un évènement.
  - Télécharger un document.

- Administrateur :

  - S'authentifier.
  - Consulter la liste des utilisateurs
  - Consulter la liste des managers.
  - Consulter la liste des taches.
  - Consulter la liste des évènements.
  - Consulter la liste des documents
  - Ajouter un manager, document, évènement, tache.
  - Supprimer un manager, document, évènement, tache.
  - Modifier un document, manager, évènement, tache.
  - Valider un évènement.
  - Télécharger un document.

- Manager :

  - S'authentifier.
  - Consulter la liste des utilisateurs.
  - Consulter la liste des taches.
  - Consulter la liste des évènements.
  - Consulter la liste des documents
  - Ajouter un document, évènement, tache.
  - Supprimer un document, évènement, tache.
  - Modifier un document, évènement, tache.
  - Valider un évènement.
  - Télécharger un document.

Spécifications fonctionnelles détaillées :

La spécification des besoins sert à associer chaque acteur réactif du système à l'ensemble d'actions avec lesquelles il intervient. Nous utilisons les diagrammes des cas d'utilisations pour modéliser les besoins fonctionnels.

Cas d'utilisation globale :

Le diagramme des cas d'utilisation dans la figure donne une vue globale sur le système :

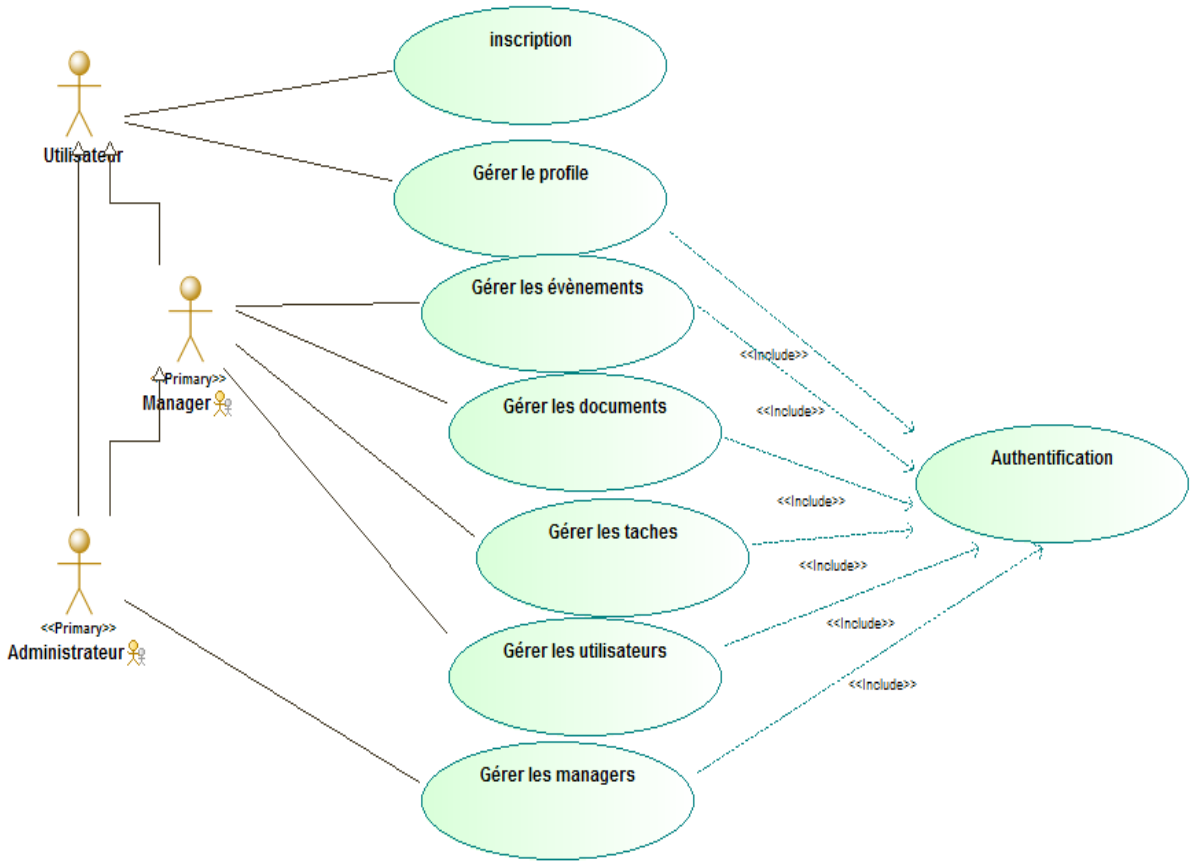

Figure 1:Diagramme de cas d'utilisation global

Le tableau 1 présente les différents cas d'utilisation avec les messages reçus et émis :

Cas d'utilisation	Acteur	Message reçus / émis
Gérer les utilisateurs	Manager	**Reçus** : Listes des utilisateurs **Emis** : Activer ou désactiver les Comptes des Utilisateurs, chercher et modifier
Gérer le profile	Utilisateur	**Reçus** : information sur le profile **Emis** : modifier, ajouter
Gérer les Taches	Manager	**Reçus** : Listes des Taches **Emis** : assigné, ajouter, supprimer, modifier, chercher
Gérer les Evènements	Manager	**Reçus** : Listes des Evènements **Emis** : ajouter , modifier, supprimer ,chercher
Gérer les manager	Administrateur	**Reçus** : Listes des Managers **Emis** : ajouter , modifier, supprimer

Inscription	Utilisateur	**Reçus** : Formulaire d'inscription **Emis** : nouvel utilisateur inscrit.

Tableau 1 : Les principaux cas d'utilisation et les acteurs correspondants

Module Gestion des Managers :

Le diagramme des cas d'utilisation suivant donne une vue globale sur le module « Gestion des managers » :

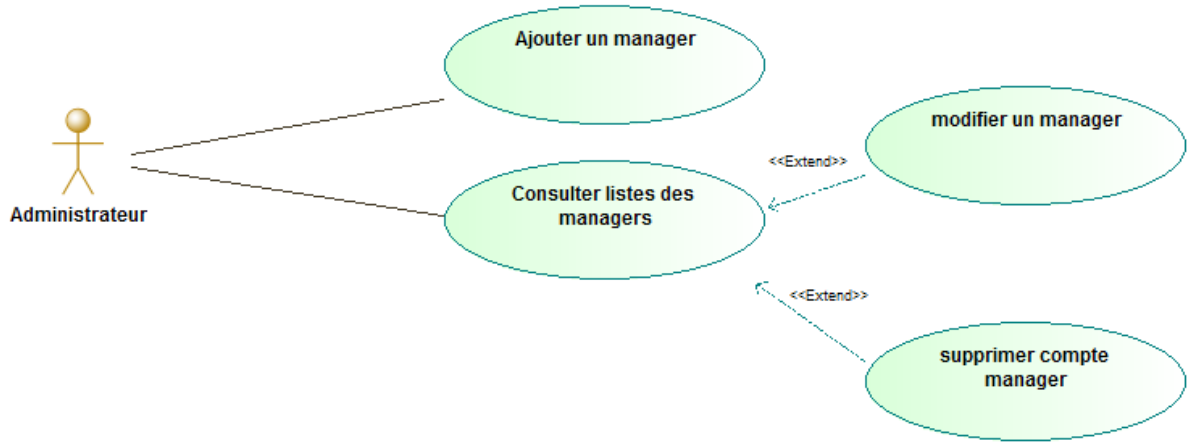

Figure 2: Diagramme de cas d'utilisation « Gestion des managers »

Le tableau 2 représente les différents cas d'utilisation « Gestion des managers » avec les messages reçus et émis :

Cas d'utilisation	Acteur	Message reçus / émis
Ajouter un manager	Administrateur	Reçus : Formulaire d'ajout Emis : nouvel manager créer
Consulter listes des managers	Administrateur	**Reçus** : Listes des managers **Emis** : consulter, chercher
Modifier un manager	Administrateur	**Reçus** : informations de manager **Emis** : manager modifié
Supprimer compte manager	Administrateur	**Reçus** : message de confirmation **Emis** : manager supprimer

Tableau 2: description de cas d'utilisation « Gestion des managers »

Module Gestion des Utilisateurs :

Le diagramme des cas d'utilisation suivant donne une vue globale sur le module « Gestion des utilisateurs » :

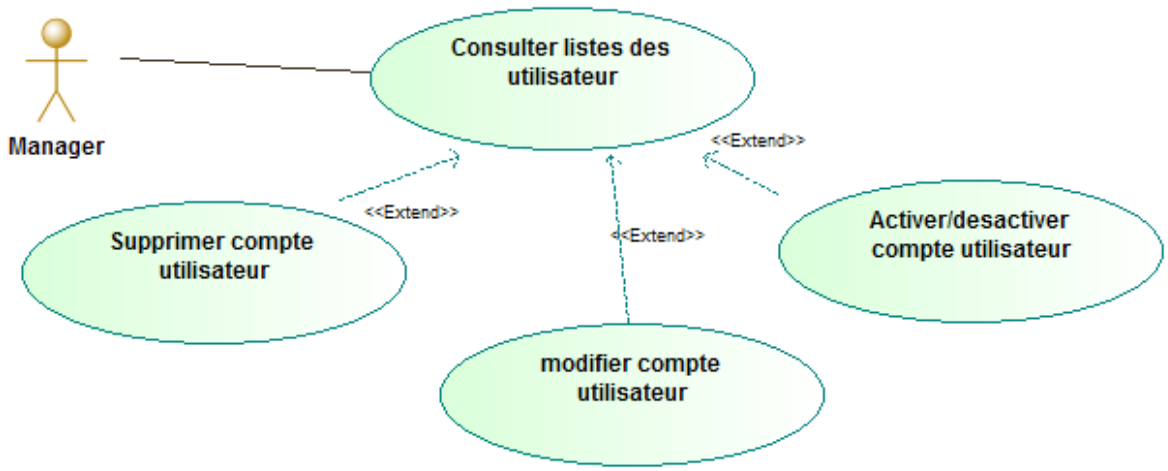

Figure 3: cas d'utilisation Gestion des utilisateurs

Le tableau 3 représente les différents cas d'utilisation « Gestion des utilisateur » avec les messages reçus et émis :

Cas d'utilisation	Acteur	Message reçus / émis
Consulter listes des utilisateurs	Manager	**Reçus** : Listes des utilisateurs **Emis** : consulter, chercher
Modifier un utilisateur	Manager	**Reçus** : informations de manager **Emis** : utilisateur modifié
Supprimer compte utilisateur	Manager	**Reçus** : message de confirmation **Emis** : utilisateur supprimer
Activer/désactiver compte utilisateur	Manager	**Reçus** : Listes des utilisateurs **Emis** : email de confirmation

Tableau 3:description de cas d'utilisation « Gestion des utilisateurs »

Module Gestion des Taches :

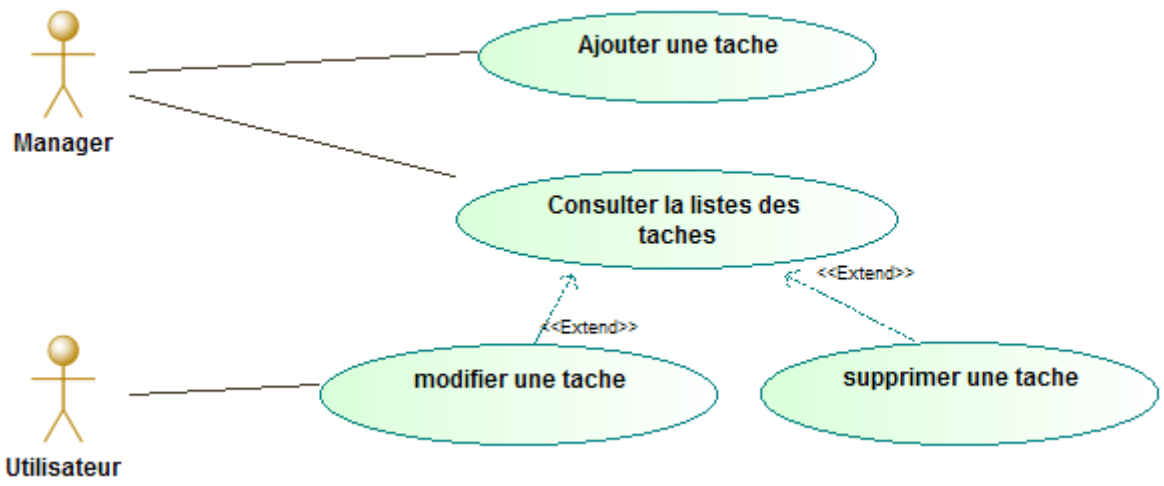

Figure 4:cas d'utilisation Gestion des taches

Le tableau 4 représente les différents cas d'utilisation « Gestion des taches » avec les messages reçus et émis :

Cas d'utilisation	Acteur	Message reçus / émis
Consulter listes des taches	Manager, Utilisateur	**Reçus** : Listes des taches **Emis** : consulter, chercher
Modifier une tache	Manager, Utilisateur	**Reçus** : informations sur une tache **Emis** : tache modifié
Supprimer une tache	Manager	**Reçus** : message de confirmation **Emis** : tache supprimer
Ajouter une tache	Manager	**Reçus** : Formulaire d'ajout **Emis** : nouvel tache créer

Tableau 4:description de cas d'utilisation « Gestion des taches »

Module Gestion des Évènements :

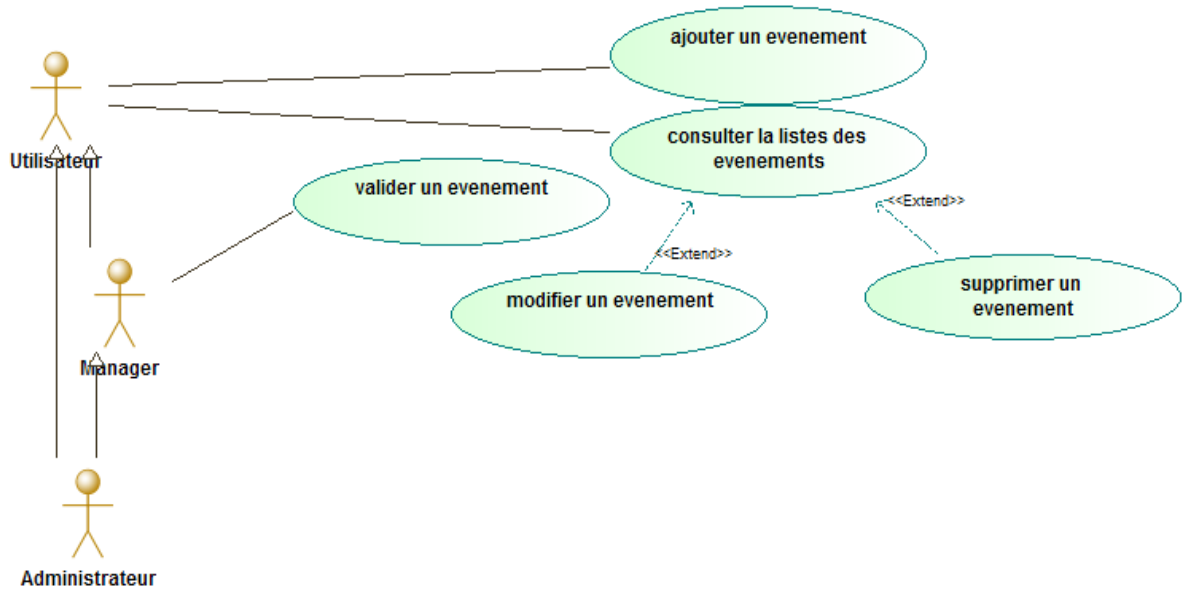

Figure 5: cas d'utilisation Gestion des évènements

Le tableau 5 représente les différents cas d'utilisation « Gestion des taches » avec les messages reçus et émis :

Cas d'utilisation	Acteur	Message reçus / émis
Consulter listes des évènements	Manager, Utilisateur, Administrateur	**Reçus** : Listes des **évènements** **Emis** : consulter, chercher
Modifier un évènements	Manager, Utilisateur, Administrateur	**Reçus** : informations sur un évènement **Emis** : évènements modifié
Supprimer un évènements	Manager, Administrateur	**Reçus** : message de confirmation **Emis** : évènement supprimer
Ajouter un évènements	Manager, Utilisateur, Administrateur	**Reçus** : Formulaire d'ajout **Emis** : nouvel évènement créer
Valider un évènements	Manager, Administrateur	**Reçus** : informations sur un évènement **Emis** : évènements validé

Tableau 5:description de cas d'utilisation « Gestion des évènements »

Module Gestion des Documents :

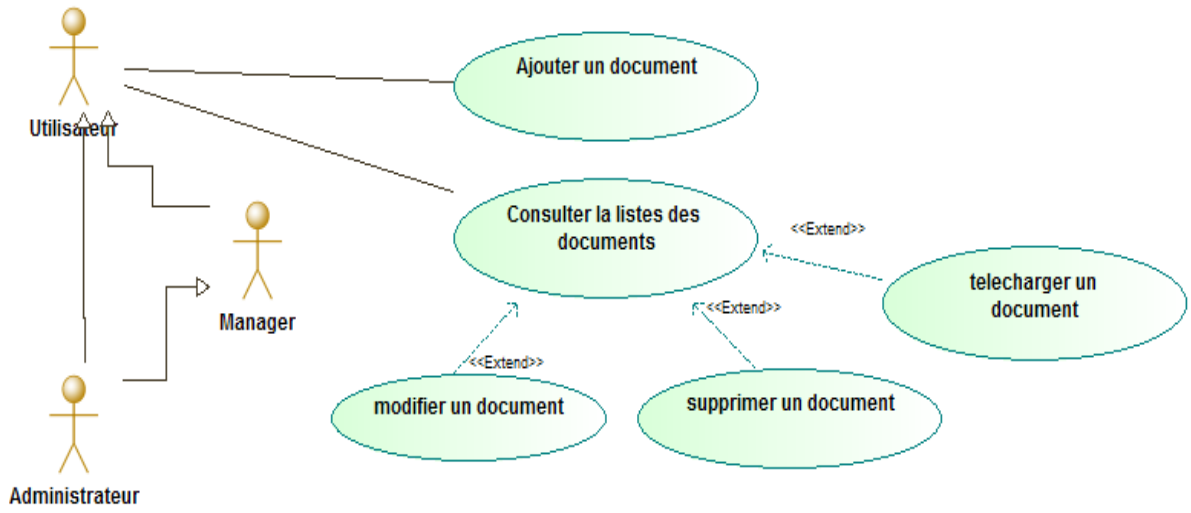

Figure 6:cas d'utilisation Gestion des documents

Le tableau 6 représente les différents cas d'utilisation « Gestion des documents » avec les messages reçus et émis :

Cas d'utilisation	Acteur	Message reçus / émis
Consulter listes des documents	Manager, Utilisateur, Administrateur	**Reçus** : Listes des documents **Emis** : consulter, chercher
Modifier un documents	Manager, Utilisateur, Administrateur	**Reçus** : informations sur un document **Emis** : document modifié
Supprimer un documents	Manager, Administrateur	**Reçus** : message de confirmation **Emis** : document archivé
Ajouter un documents	Manager, Utilisateur, Administrateur	**Reçus** : Formulaire d'ajout **Emis** : nouveaux documents créer
Télécharger un document	Manager, Administrateur, Utilisateur	**Reçus** : informations sur un document **Emis** : document téléchargé

Tableau 6:description de cas d'utilisation « Gestion des documents »

Module Gestion de profile :

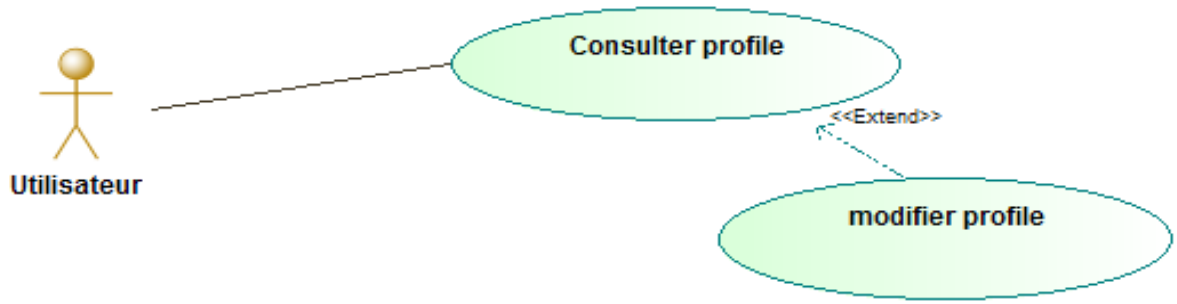

Figure 7:cas d'utilisation Gestion de profile

Le tableau 7 représente les différents cas d'utilisation « Gestion de profile » avec les messages reçus et émis :

Cas d'utilisation	Acteur	Message reçus / émis
Consulter profile	Utilisateur	**Reçus** : information sur le profile **Emis** : consulter
Modifier profile	Utilisateur	**Reçus** : informations sur le profile **Emis** : profile modifié

Tableau 7:description de cas d'utilisation « Gestion de profile »

Diagramme de classes

La figure ci-dessous présente un diagramme de classes qui Contient toutes les informations telles que les classes, les méthodes, les associations et les propriétés.

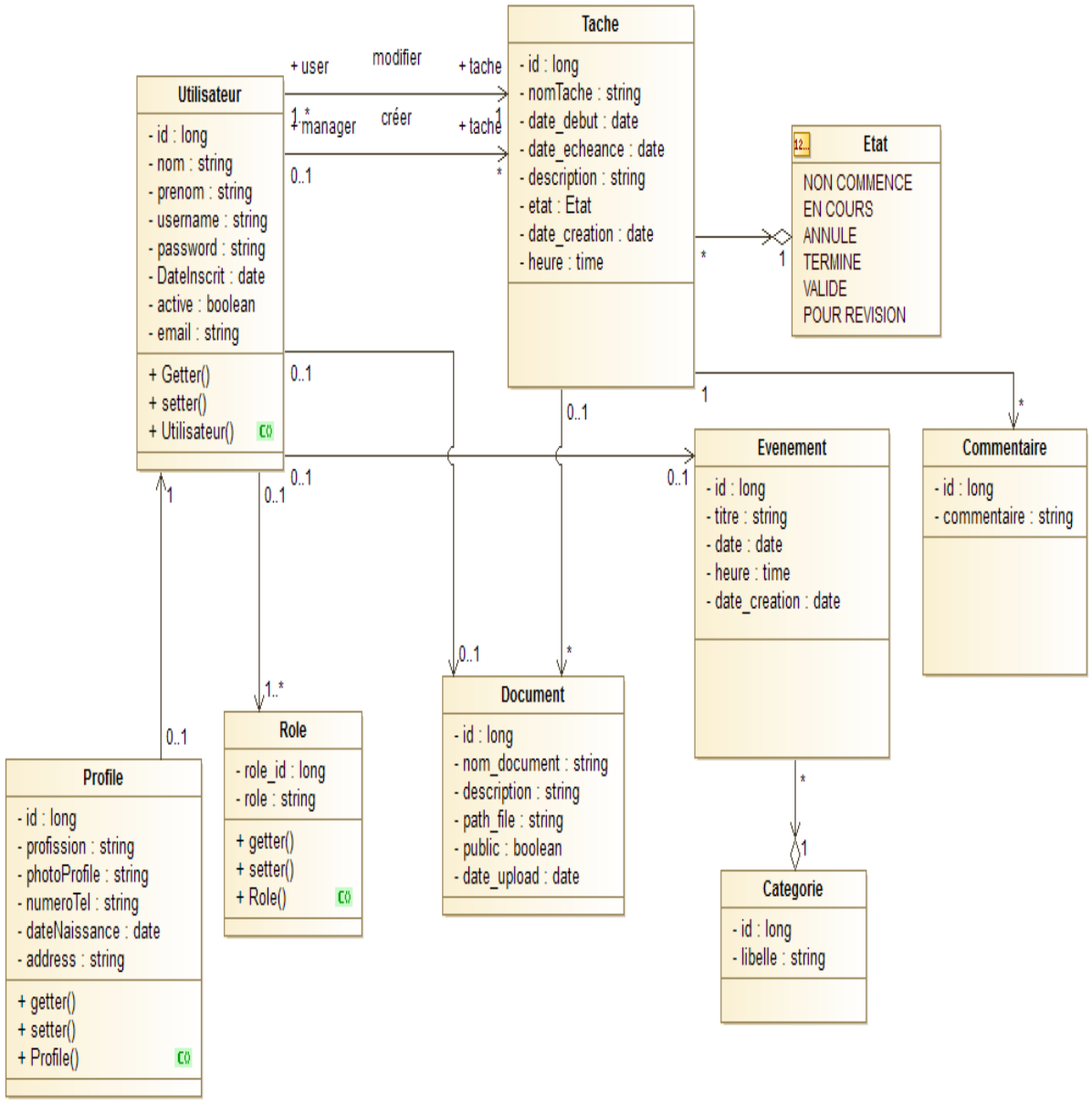

Figure 8:Représentation du diagramme de classe

Diagrammes de séquences

Les diagrammes de séquences représentent les interactions entre les objets en indiquant la chronologie des séquences. Les diagrammes de séquences ajoutent une dimension temporelle aux diagrammes de classe.

Diagramme de séquence : « Authentification et Autorisation »

JWT (JSON Web Token) est une authentification stateless basée sur l'échange d'un token entre l'utilisateur et le serveur. Le token contient les informations suffisantes :

- Pour identifier l'utilisateur (le token est bien celui qui a été remis lors de l'authentification)
- Pour autoriser l'utilisateur (le token contient les droits du client)

Le diagramme de séquence ci-dessous présente une vue globale sur le séquencement des Interactions entre utilisateur, Authentication Controller et la base de donnée.

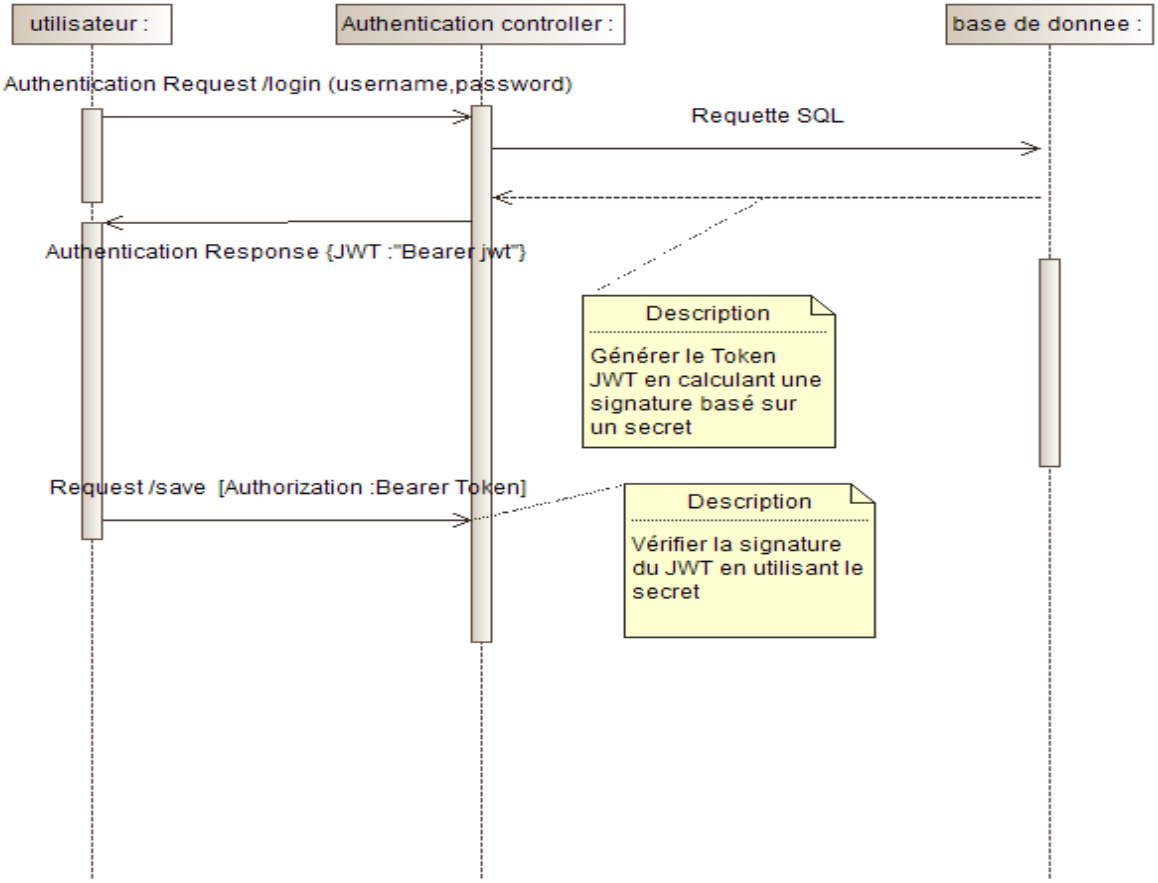

Figure 9:Digramme de séquence Authentification et Autorisation basé sur JWT

Le token est envoyé avec chaque requête que le client fera auprès de l'application, qui autorisera, ou non, le client à accéder à ses services, suivant la validité du token. Ce type d'authentification, ne stocke pas les sessions utilisateurs dans le contexte de l'application.
Pour utiliser notre token, il faut tout d'abord le créer. Pour cela, il est nécessaire de s'authentifier avec son username et son mot de passe auprès de l'application afin que celle-ci nous renvoie le token. Une fois le token obtenu, on peut faire appel à nos URL sécurisées en envoyant le token avec notre requête. La méthode la plus courante pour envoyer le token est de l'envoyer à travers l'en-tête HTTP Authorization en tant que Bearer token :

Authorization : Bearer 'token'

Le diagramme de séquence « Authentification » présente le séquencement détaillé des Interactions entre utilisateur, Spring Security (FilterChain, JWTAuthenticationFilter), l'api JWT et la base de donnée.

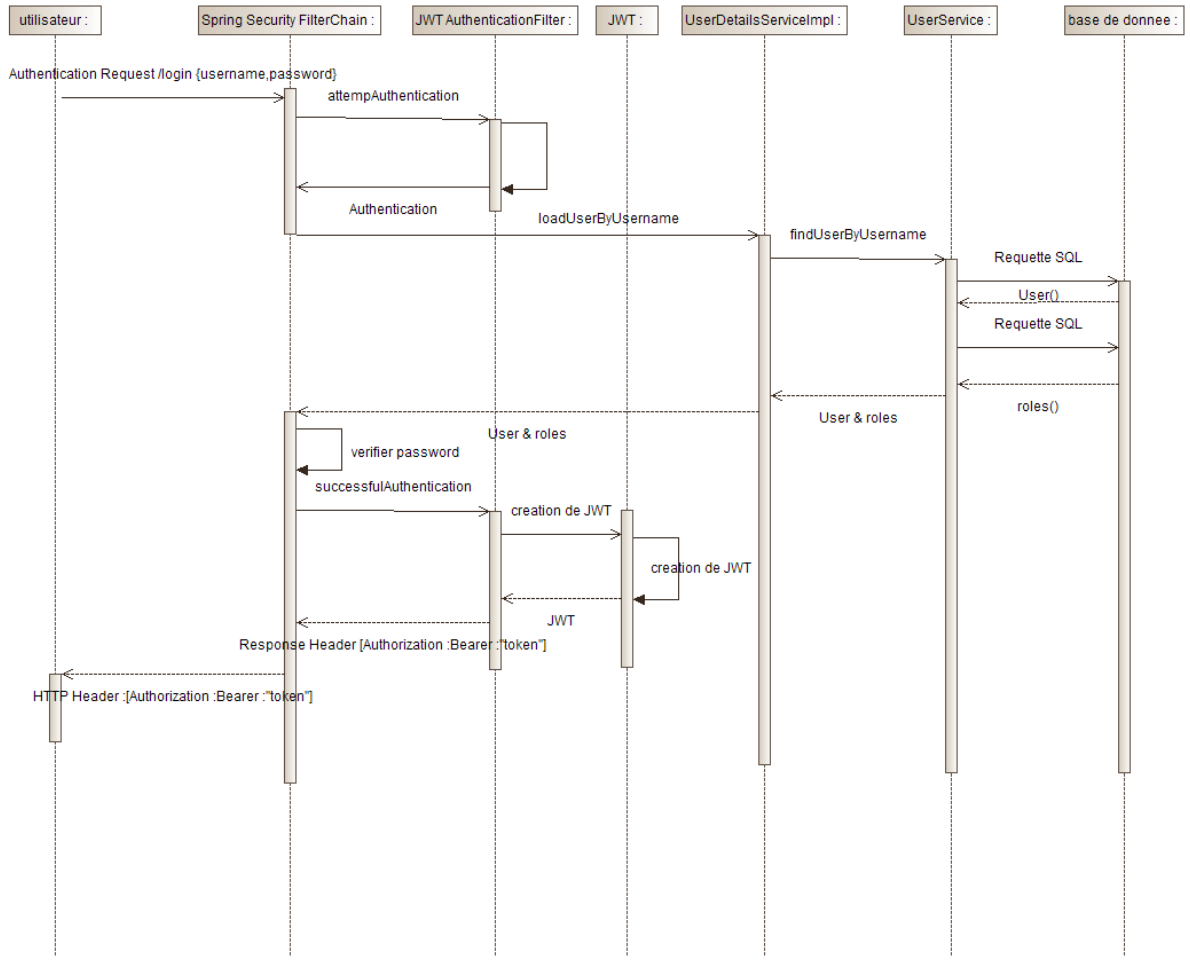

Figure 10:Digramme de séquence : «Authentification et Autorisation »

Diagramme de séquence : « Consulter la liste des taches » :

Pour traiter le token, on utilise un filter qui va l'extraire du header, le valider puis ajouter au contexte de Spring une authentication correspondant à l'utilisateur pour lequel le token a été émis : notre utilisateur est authentifié pour le reste de sa requête.
Le filter est sans doute la partie essentielle de notre chaîne d'authentification. Il va intercepter toutes les requêtes et vérifier la présence d'un JWT, et va ensuite valider le token et récupérer l'objet "authentication" pour l'ajouter au contexte de spring-security. Notre client sera donc authentifié pour la suite de l'exécution de sa requête. À la fin du processus, on supprime l'authentification du contexte.
Le diagramme de séquence « Consulter la liste des taches » présente le séquencement détaillé des Interactions entre utilisateur, Spring Security (FilterChain, JWTAuthenticationFilter), l'api JWT et la base de donnée.

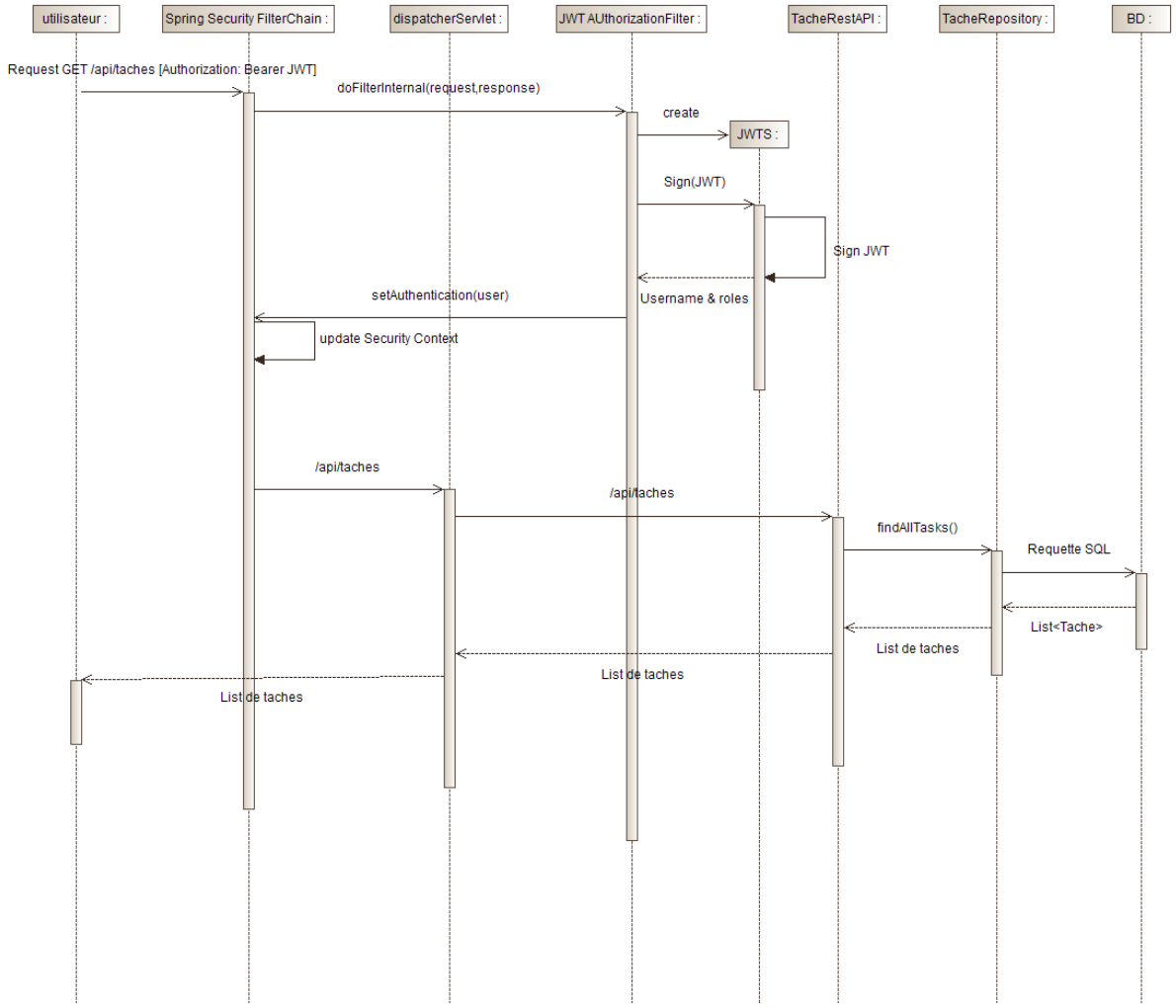

Figure 11:Diagramme de séquence : « Consulter la liste des taches »

Spécification non fonctionnelle :

Ce sont des exigences qui ne concernent pas spécifiquement le comportement du système mais plutôt identifient des contraintes internes et externes du système.

Les principaux besoins non fonctionnels de notre application se résument dans les points suivants :

- Sécurité : l'application doit respecter la confidentialité des données.
- Performance : l'application devra être performante c'est-à-dire que le système doit réagir dans un délai précis quel que soit l'action de l'utilisateur.
- Extensibilité : l'application doit être fermer à la modification ouverte à l'extension c'est-à-dire qu'il pourra y avoir une possibilité d'ajouter de nouvelles fonctionnalité.

Conclusion :

Ce chapitre a été consacré à la conception de la solution et le modèle d'analyse à partir de l'approche statique à travers le diagramme de cas d'utilisation et de classes et l'approche dynamique à travers les diagrammes de séquence.
Dans le chapitre suivant nous détaillons les étapes de la réalisation de l'application.

Chapitre 3

Réalisation

Après avoir élaboré la conception de notre application, nous abordons dans ce chapitre le dernier volet de ce rapport, qui a pour objectif d'exposer la phase de réalisation.
La phase de réalisation est considérée comme étant la concrétisation finale de toute la méthode de conception.
Nous menons tout d'abord une étude technique où nous décrivons les ressources logicielles utilisées dans le développement de notre projet. Nous présentons en premier lieu notre choix de l'environnement de travail, où nous spécifions l'environnement matériel et logiciel qu'on a utilisé pour réaliser notre application puis nous détaillons l'architecture, aussi nous présentons quelques interfaces réalisées pour illustrer le fonctionnement de quelques activités du système.

Environnement de travail :

L'implémentation consiste la phase d'achèvement et d'aboutissement du projet. Pour accomplir cette tâche, il faut utiliser des outils adéquats à la réalisation du projet. Ces outils représentent l'environnement de travail.

Environnement matériel :

L'environnement matériel, a été mis en place pour le développement qui est un PC DELL.
Ce dernier dispose de la configuration suivante :

- Processeur : Intel® Core™ i5-2540M CPU @ 2.60GHz.
- RAM : DDR4 8Go
- Système d'exploitation : Windows 7 SP1 Edition intégrale.
- Un disque dur SSD de 250 GO.

Environnement logiciel :

Concernant l'environnement logiciel de notre projet, nous avons utilisé parmi les outils et logiciels ce qui suit :

Est un outil d'administration de base de données possédant un éditeur SQL et un constructeur de requête. Il a été développé et optimisé pour être utilisé avec le SGBD relationnel MySQL disponible commercialement ou gratuitement.
Le logiciel est devenu un projet libre en 2006 sous le nom de Heidi SQL Heidi SQL est capable de se connecter à des bases MySQL, SQL Server ainsi que Postgresql2.

Est un outil de modélisation UML disponible sur les plates-formes Windows, Linux et Mac. Il intègre également la modélisation BPMN, et le support de la modélisation des exigences, du dictionnaire, des règles métier et des objectifs.

Est un client REST proposé par Google. Il est disponible sous la forme d'une extension Chrome ou bien d'une application stand-alone. Parmi les nombreuses solutions pour interroger ou tester web services et API, Postman propose de nombreuses fonctionnalités, une prise en main rapide et une interface graphique agréable.

C'est un projet de la Fondation Eclipse visant à développer tout un environnement de développement libre, extensible, universel et polyvalent. Son objectif est de produire et fournir divers outils gravitant autour de la réalisation de logiciel, englobant les activités de codage logiciel proprement dites (avec notamment un environnement de développement intégré) mais aussi de modélisation, de conception, de test, de reporting, etc. Son environnement de développement notamment vise à la généricité pour lui permettre de supporter n'importe quel langage de programmation.

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et OS X4.

Présenté lors de la conférence des développeurs Build d'avril 2015 comme un éditeur de code cross-Platform, open source et gratuit, supportant une dizaine de langages

Outils de développement et SGBD :

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'autres système de gestion de base de données, qu'ils soient libres (comme MySQL), ou propriétaires (Comme Oracle et Microsoft SQL Server).

Java :

Le langage java, développé par Sun, est un langage orienté objet, en effet il possède un mécanisme qui permet de décrire les caractéristiques d'un objet de façon unique et de pouvoir lui faire subir des opérations.
Java Entreprise Edition JEE, qui peut être considéré comme une extension de java, est un ensemble de spécifications destinées aux applications d'entreprises. Ce langage permet la création d'applications performantes et robustes.
JEE s'appuie sur le modèle MVC (Model Vue Contrôler).

JavaScript :

JavaScript est un langage de programmation qui permet d'implémenter des mécanismes complexes sur une page web. À chaque fois qu'une page web fait plus que simplement afficher du contenu statique — afficher du contenu mis à jour à des temps déterminés, des cartes interactives, etc… — JavaScript a de bonnes chances d'être impliqué. C'est la troisième couche des technologies standards du web, les deux premières (HTML et CSS).

Le Framework Angular :

Angular est un Framework JavaScript côté client qui permet de réaliser des applications de type "Single Page Application". Il est basé sur le concept de l'architecture MVC (Model View Controller) qui permet de séparer les données, les vues et les différentes actions que l'on peut effectuer.
Le code source d'Angular est écrit en Type Script. Le Type Script est une couche supérieure au JavaScript développé par Microsoft qui se compile en JavaScript simple. Etant un langage typé, il permet de créer des classes, des variables, des signatures de fonction et l'utilisation de modules. Il est important de noter que l'utilisation du Type Script est facultative, on peut tout à fait utiliser du JavaScript dans un fichier Type Script.

Le Framework Spring :

Est un Framework très riche permettant de structurer, d'améliorer et de simplifier l'écriture d'application Java EE.
Spring est un Framework libre, un conteneur dit « léger », c'est à dire une infrastructure similaire à un serveur d'application Java EE.
Spring s'appuie principalement sur l'intégration de trois concepts clés :

- L'inversion de contrôle est assurée de deux façons différentes : la recherche de dépendances et l'injection de dépendances, cette injection peut être effectuée de trois manières possibles :

    - L'injection de dépendance via le constructeur.
    - L'injection de dépendance via les modificateurs (setters).
    - L'injection de dépendance via une interface.
- Une couche d'abstraction : La couche d'abstraction permet d'intégrer d'autres Framework et bibliothèques avec une plus grande facilité. Cela se fait par l'apport ou non de couches d'abstraction spécifiques à des Framework particuliers. Il est ainsi possible d'intégrer un module d'envoi de mails plus facilement.
- La programmation orientée aspect

➢ **Spring Boot :**

Est un micro-Framework créé par l'équipe de chez Pivotal, conçu pour simplifier le démarrage et le développement de nouvelles applications Spring. Le Framework propose une approche dogmatique de la configuration, qui permet d'éviter aux développeurs de redéfinir la même configuration à plusieurs endroits du code. Dans ce sens, Boot se veut d'être un acteur majeur dans le secteur croissant du développement d'applications rapide.

➢ **Spring Data :**

C'est un projet supplémentaire de Spring créé il y a quelques années pour répondre aux besoins d'écrire plus simplement l'accès aux données et d'avoir une couche d'abstraction commune à de multiples sources de données.
Spring Data s'interface avec plusieurs sources de données parmi lesquelles JPA, Neo4j, Mongo DB, REST et quelques autres.

➢ **Spring Security :**

Est un module incontournable d'une application développée en Spring. Il apporte tout le nécessaire pour sécuriser une application et il a l'avantage d'être vraiment personnalisable. La notion de sécurité informatique n'est pas une mince affaire et sa mise en place est parfois longue et demande d'être constamment adaptée au niveau réseau, serveurs, application... Spring Security n'intervient que sur le domaine applicatif.

➢ **Spring MVC :**

Le Framework Spring offre une implémentation innovante du patron MVC par le biais d'un Framework nommé Spring MVC, qui profite des avantages de l'injection de dépendances et qui, depuis la version 2.5, offre une intéressante flexibilité grâce aux annotations Java 5. Ce module permet dès lors de s'abstraire de l'API Servlet de Java EE, les informations souhaitées étant automatiquement mises à disposition en tant que paramètres des méthodes des contrôleurs.

De plus, à partir de la version 3.0, Spring MVC intègre un support permettant de gérer la technologie REST, les URL possédant la structure décrite par cette dernière étant exploitable en natif.

Le Framework Hibernate :

Est un Framework de persistance utilisé pour gérer la persistance des objets java dans une base de données, il peut être utilisé dans un développement web ou bien un développement client lourd.
Hibernate peut utiliser SQL comme il peut utiliser son propre langage de requête HQL (Hibernate Query Language).

Le Framework Bootstrap :

Est un Framework développé par l'équipe du réseau social Twitter. Proposé en open source (sous licence MIT), ce Framework utilisant les langages HTML, CSS et JavaScript fournit aux développeurs des outils pour créer un site facilement. Ce Framework est pensé pour développer des sites avec un design responsif, qui s'adapte à tout type d'écran, et en priorité pour les smartphones. Il fournit des outils avec des styles déjà en place pour des typographies, des boutons, des interfaces de navigation et bien d'autres encore. On appelle ce type de Framework un "Front-End Framework".

Git et GitHub :

GitHub Est une plateforme open source de gestion de versions et de collaboration destinée aux développeurs de logiciels. la solution GitHub a été lancée en 2008. Elle repose sur Git, qui permet de stocker le code source d'un projet et de suivre l'historique complet de toutes les modifications apportées à ce code.

Architecture :

L'architecture est l'ensemble des aspects techniques et applicatifs qui sont importants pour un logiciel. Les choix architecturaux influent sur la réussite ou l'échec d'un projet. Nous exposons d'abord l'architecture technique cible de la solution ainsi que l'architectures applicative.

Architecture technique :

L'architecture technique est l'environnement technique permettant l'exécution des composants informatiques et les échanges de données. Pour notre application, nous proposons l'architecture cible suivante :

Figure 12:Architecture technique de l'application

L'utilisateur se connecte à l'application à travers un navigateur web. Le navigateur permet d'envoyer des requêtes au serveur REST et d'en interpréter la réponse. Le navigateur et le serveur REST communiquent en utilisant le protocole http. Le serveur REST traite les requêtes HTTP, interprète et exécute le code de l'application, puis génère une réponse qu'il renvoie au navigateur de l'utilisateur. Le système de gestion de bases de données PostgreSQL permet d'interroger les données et de les mettre à jour.

Architecture applicative :

Nous pouvons définir l'architecture applicative comme une organisation des données et des traitements qui mettent en œuvre les fonctions métiers. La figure 13 présente l'architecture applicative de la solution avec les Framework utilisés.

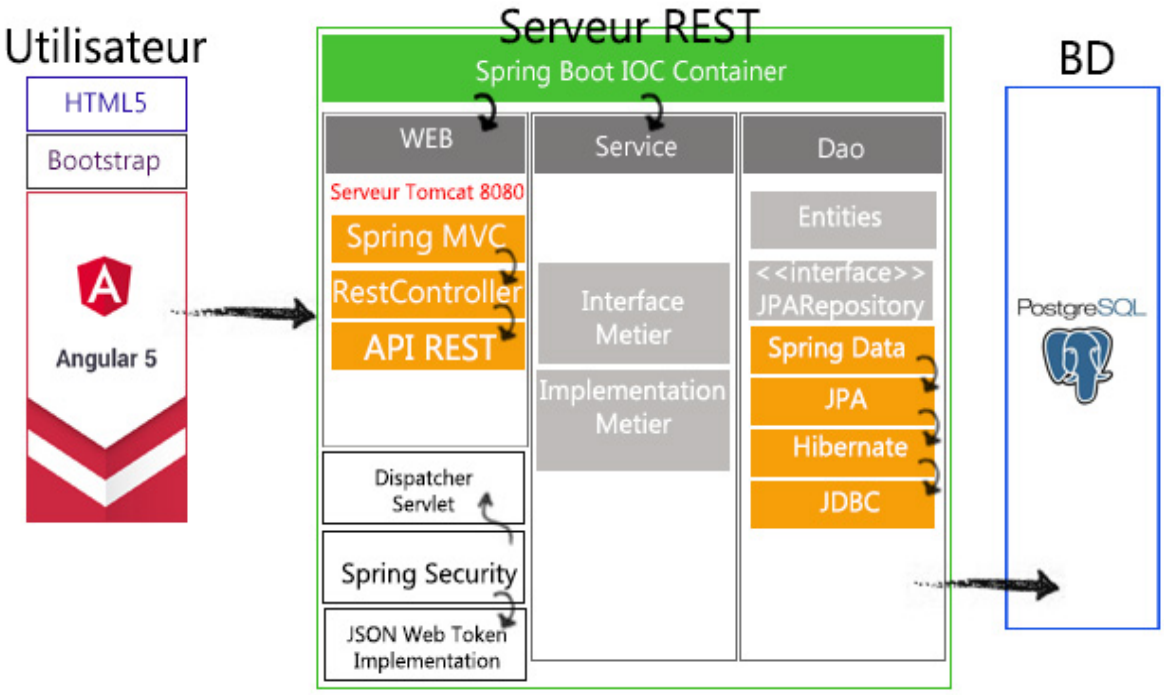

Figure 13.Architecture applicative de l'application

SOA - Architecture Orientée Service :

L'architecture orientée services (SOA, Service-Oriented Architecture) est une approche permettant de créer une architecture qui s'appuie sur l'utilisation de services. Ces services (les services Web RESTful, par exemple) remplissent de petites fonctions, telles que la production de données, la validation d'un client ou la mise à disposition d'analyses simples.

Architecture Client :

La figure 14 présente l'architecture client.

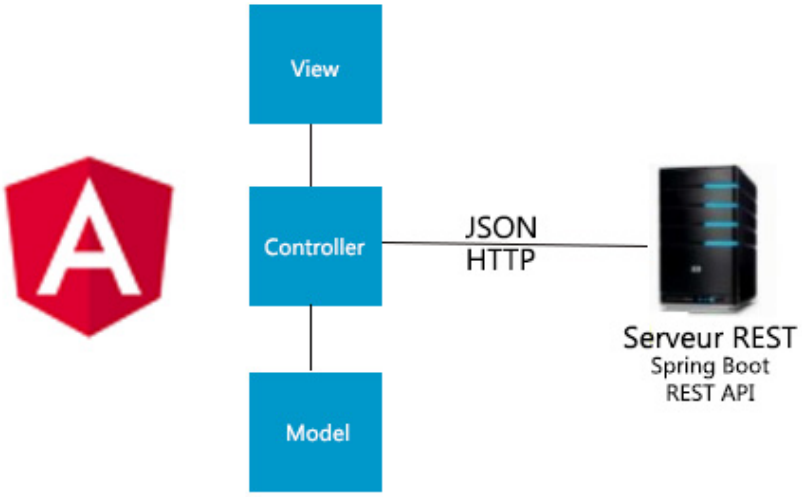

Figure 14: Architecture Client

Le notion service Web :

La technologie des services Web est un moyen rapide de distribution de l'information entre clients et leurs différentes plates-formes. Les services Web sont basés sur le modèle SOA.
La spécificité des Web Services est l'utilisation de HTTP comme support des messages entre clients et serveur.

REST :

Est l'acronyme de Representational State Transfer, REST est un style d'architecture qui repose sur le protocole HTTP : On accède à une ressource (par son URI unique) pour procéder à diverses opérations (GET lecture / POST écriture / PUT modification / DELETE suppression), opérations supportées nativement par HTTP.

Représentation des ressources : JSON :

JSON est l'acronyme de JavaScript Object Notation. C'est un format texte qui permet de représenter des données et de les échanger facilement.
Que ce soit au niveau du serveur ou au niveau des clients, une API RESTful manipule des ressources par une représentation de celles-ci.
Le principe en pratique est de passer de la représentation utilisée en interne, que ce soit sur le client ou le serveur, à la représentation utilisée dans le message HTTP, requête ou réponse.
Deux types de structures sont disponibles pour décrire le modèle objet de JavaScript :

- Objet : une collection de pair nom/valeur
- Tableau : une liste ordonnée de valeurs.

Les valeurs peuvent être des types suivants : booléen, chaîne de caractères, nombre, ou valeur nulle. (boolean, string, number, null).
Voici par exemple la description d'un utilisateur :

{« nom":"g", "age":30, "addresse": { "rue":"avenue l'oasis", "ville":"ghomrassen", "code":3220}, "telephone":[{"type":"maison", "numero" : secret}, {"type":"portable", "numero": secret}] }

(A remplacé par un imprime écran capture POSTMAN)

API REST :

Ci-dessous un tableau qui illustre les principes de création d'une API s'appuyant sur les principes REST dans notre application.

URI	Méthode	Sémantique
http://localhost:8080/api/users	GET	Récupère la liste des utilisateurs
http://localhost:8080/api/users/id	GET	Récupère la représentation de L'utilisateur identifié par id
http://localhost:8080/api/users/id	DELETE	Efface un utilisateur
http://localhost:8080/api/register	POST	Création d'un nouvel utilisateur
http://localhost:8080/api/users/roleUser	GET	Récupère la liste des utilisateurs avec le Rôle user
http://localhost:8080/api/users/roleManager	GET	Récupère la liste des utilisateurs avec le Rôle manager
http://localhost:8080/api/users/manager	POST	Création d'un nouvel manager
http://localhost:8080/api/tasks	GET	Récupère la liste des taches
http://localhost:8080/api/tasks/id	GET	Récupère la représentation de tache identifié par id
http://localhost:8080/api/tasks/id	DELETE	Efface une tache
http://localhost:8080/api/tasks/id	PUT	Modifie une tache
http://localhost:8080/api/tasks/encours	GET	Récupère la liste des taches en cours
http://localhost:8080/api/tasks/nonCommence	GET	Récupère la liste des taches en non commence
http://localhost:8080/api/tasks/termine	GET	Récupère la liste des taches en termine

http://localhost:8080/api/tasks/annule	GET	Récupère la liste des taches en annule
http://localhost:8080/api/events	GET	Récupère la liste des évènements
http://localhost:8080/api/events	POST	Création d'un nouvel évènement
http://localhost:8080/api/events/id	GET	Récupère la représentation d'un évènement identifié par id
http://localhost:8080/api/events/id	PUT	Modifie un évènement
http://localhost:8080/api/events/id	DELETE	Efface un évènement
http://localhost:8080/api/documents/upload	POST	Upload d'un document
http://localhost:8080/api/documents	GET	Récupère la liste des document
http://localhost:8080/api/documents/filename	GET	Récupère un document par nom

Conclusion :

A travers ce chapitre, nous avons présenté la réalisation de l'application en justifiant nos choix technologiques …