

Réf : PFE-2021

Rapport de Projet de Fin d'Etude

Présenté et soutenu publiquement le .././2021

Par

Prénom NOM

Intitulé du projet

Composition du jury

Monsieur

Président

Monsieur

Encadrant

Année universitaire : 2018-2019

Dédicaces

Remerciements

Table des matières

Chapitre 2. Phase de Planification.....	1
Introduction	1
1. Présentation de l'équipe	1
2. Capture des besoins	1
2.1. Identification des acteurs.....	1
2.2. Backlog Produit.....	2
2.3. Besoins fonctionnels	5
2.4. Besoins non fonctionnels	7
3. Analyse et spécifications des besoins.....	7
3.1. Diagramme des cas d'utilisations globale	7
3.2. Architecture de la solution	8
3.2.1. Architecture physique	8
3.2.2. Architecture Logique.....	9
3.2.2.1. Modèle architectural du Front-End	9
3.2.2.2. Modèle architectural du Back-End.....	10
4. Environnement de travail	12
4.1. Environnement matériel	12
4.2. Environnement Logiciel.....	13
4.2.1. Outils de développement et modélisation	13
4.2.2. Les outils de test	15
4.2.3. Langages de programmation	15
4.2.4. Technologies utilisés	16
Technologies	16
Logo	16
Description	16
Spring Boot 2	16
.....	16
Angular 8.....	17
Chart.js	17
.....	17
5. Planification des sprints	17
Conclusion.....	19

Table des figures

Figure 1 : Diagramme des cas d'utilisation globale.....	8
Figure 2 : Architecture physique du système	9
Figure 3 : Architecture Front-End	10
Figure 4 : Diagramme de package de l'application	11

Table des tableaux

Tableau 1 : Identification des acteurs.....	1
Tableau 2 : Backlog du produit	3
Tableau 3 : Environnement Matériel.....	12
Tableau 4 : Outils de développement et modélisation	13
Tableau 5 : Outils de test.....	15
Tableau 6 : Langages de programmation	16
Tableau 7 : Technologies utilisées	16
Tableau 8 : Déroulement de stage	18
Tableau 9 : Planification des releases	18

Chapitre 2. Phase de Planification

Introduction

Après avoir défini le cadre général de notre projet, nous allons nous concentrer dans ce chapitre sur l'identification des besoins fonctionnels et non fonctionnels, nous allons définir, par la suite, les acteurs de notre système tout en présentant le diagramme des cas d'utilisation globale. Dans une deuxième partie, nous allons présenter l'environnement du travail. Finalement, nous allons exposer Le backlog produit du projet et les sprints panifiés.

1. Présentation de l'équipe

- Client : monsieur Heni Ouehezi est le fondateur de billcom Consulting
- Développeurs : Nous somme deux développeurs au moment de la rédaction de ce rapport, un salarié développeur monsieur Radhouan Ghribi et un stagiaire développeur.
- Le reste de l'équipe sont dans le service consulting

2. Capture des besoins

La capture des besoins consiste à identifier les acteurs, exposer le backlog produit du projet, ensuite nous allons décrire les besoins fonctionnels et non fonctionnels que les utilisateurs s'attendent à voir par le système.

2.1. Identification des acteurs

Un acteur est une personne, un matériel ou un logiciel qui interagit avec le système dans le but de réaliser un plus ou une amélioration. Notre étude fonctionnelle sera organisée selon les rôles attribué à chaque acteur qui sont en interaction avec notre système. Le tableau numéro 1 ci-dessous présente les acteurs de notre projet :

Tableau 1 : Identification des acteurs

Acteur	Fonctionnalités
Manager	C'est un utilisateur final de l'application. Il va jouer le rôle d'un administrateur et il aura l'accès total à toutes les fonctionnalités de notre système tel que La gestion des utilisateurs, la gestion des

	<p>rôles, la gestion des tâches et des équipes, la gestion des notifications et messagerie, la consultation des statistiques, des tableaux de suivi.</p> <p>Dans notre cas, le Manger sera monsieur Heni Ouelhezi.</p>
Leader	C'est l'acteur responsable de suivi de son équipe, ses fonctionnalités consistent à Suivre le travail de son équipe, accorder des tâches aux membres d'équipes, gérer les notifications.
Member	C'est l'acteur responsable de la gestion des tâches.

2.2. Backlog Produit

Nous allons présenter dans cette section le backlog du produit qui est considéré comme élément fondamental de la méthodologie Scrum. Il s'agit d'une liste de tâches priorisées définissant les besoin métiers de l'utilisateur et un outil de travail principal qui se charge de recueillir les besoins auprès des parties prenantes et de les transformer en liste de fonctionnalités prêtes à être développées par l'équipe de développement.

Le Backlog de produit présenté dans le tableau numéro 2 comprend les champs suivants :

- ID : Identifiant unique auto-incrémenté des modules.
- Module : Un nom clé pour décrire un ensemble de fonctionnalités regroupées.
- User story : C'est la définition précise et claire de la fonctionnalité souhaitée par l'utilisateur.
- Priorité : L'attribution de la priorité de chaque tâche qui est classée comme suit : "Élevée" ou "Moyenne" ou "Faible".

Tableau 2 : Backlog du produit

ID	Module	User Story	Priorité
1	Authentification et gestion d'accès	En tant qu'utilisateur de l'application, je veux me connecter à l'application.	Elevée
		En tant qu'utilisateur je veux récupérer l'accès à l'application avec l'option mot de passe oublié.	Elevée
		En tant que manager, je veux activer/désactiver un compte employé.	Elevée
		En tant que manager, je veux créer des comptes employés.	Elevée
2	Gestion des rôles des utilisateurs	En tant que manager, je veux ajouter un ou plusieurs droits d'accès pour les utilisateurs.	Moyenne
		En tant que manager, je veux supprimer un ou plusieurs droits d'accès pour les utilisateurs.	Moyenne
3	Gestion des équipes	En tant que chef d'équipe/Manager, je veux créer une équipe.	Elevée
		En tant que chef d'équipe/Manager, je veux ajouter/modifier/supprimer /Sélectionner des employée à l'équipe.	Elevée
		En tant que Chef équipe/Manager, je veux consulter les détails de chaque équipe.	Moyenne
		En tant que Chef équipe/Manger je veux appliquer des recherches filtrés.	Moyenne

4	Gestion tâche	En tant qu'utilisateur de l'application je veux créer des tâches.	Elevée
		En tant que chef d'équipe, je veux ajouter/modifier/supprimer des tâches.	Elevée
		En tant qu'utilisateur de l'application, je veux consulter mes tâches.	Moyenne
		En tant qu'utilisateur de l'application je veux consulter l'état d'avancement de la tâche.	Moyenne
		En tant qu'utilisateur de l'application je veux uploader des fichiers.	Moyenne
5	Statistiques, historique et calcul des KPI	En tant que Manager/Chef équipe, je veux Consulter le tableau de suivi de travail pour chaque équipe.	Elevée
		En tant que Manager, je veux consulter les historiques des visites de l'application.	Moyenne
		En tant que Manager, je veux consulter le tableau de suivi global de travail (par jour/ mois / année).	Elevée
		En tant que Manager je veux consulter les KPI.	Elevée
6	Gestion des notifications et	En tant qu'utilisateur de l'application, je veux envoyer des e-mails via l'application en temps réel.	Elevée

	messagerie en temps réels	En tant qu'utilisateur, je veux supprimer des e-mails.	Faible
		En tant qu'utilisateur de l'application, je veux recevoir des e-mails en cas de retard de livraison de tâche.	Elevée
		En tant qu'utilisateur de l'application je veux recevoir des rappels sous forme des e-mails.	Moyenne
		En tant qu'utilisateur de l'application je veux envoyer des messages privés en temps réel à un autre utilisateur de l'application.	Moyenne

2.3. Besoins fonctionnels

Les besoins fonctionnels sont les principales exigences que doit fournir notre application à ses utilisateurs, nous allons citer dans ce qui suit l'ensemble des fonctionnalités du système

Gérer les utilisateurs : Le système doit permettre à l'administrateur (Le manager) de gérer les utilisateurs

- Ajouter un utilisateur
- Modifier les détails d'un utilisateur
- Consulter la liste des utilisateurs
- Appliquer des recherches filtrées et triés sur la liste des utilisateurs
- Activer/ désactiver un utilisateur

Gérer les rôles : Le système doit permettre à l'administrateur (Le manager) de gérer les rôles

- Ajouter un ou plusieurs rôles à un utilisateur
- Supprimer un ou plusieurs rôle d'un utilisateur

Gérer les équipes : Le système doit permettre au manager ou au chef d'équipe de gérer les équipes

- Ajouter une équipe
- Ajouter des utilisateurs à l'équipe
- Consulter les détails des équipes
- Supprimer une équipe

Gérer les tâches : Le système doit permettre aux utilisateurs de gérer les tâches

- Ajouter une tâche
- Accorder une tâche à un utilisateur
- Modifier l'état d'avancement d'une tâche
- Ajouter des commentaires à la tâche
- Afficher les détails de la tâche
- Supprimer une tâche

Gérer les Statistiques, historique et le calcul des KPI :

- Consulter les statistiques
- Consulter les historiques
- Appliquer des recherches filtrées sur les historiques
- Consulter les KPI

Gérer les notifications:

- Envoyer des e-mails en temps réel.
- Planifier des rappels en temps réel
- Envoyer des messages en temps réel

2.4. Besoins non fonctionnels

Afin d'assurer un bon fonctionnement de l'application et de garantir la satisfaction de l'utilisateur, des contraintes techniques et ergonomiques doivent être prises en compte tout au long du développement du projet :

- **La maintenabilité** : Le code doit être modulaire, bien commenté et doit respecter les règles standards et universelles de codage.
- **L'ergonomie et la convivialité** : L'application doit fournir des interfaces simples pour l'utilisateur afin de faciliter l'exploitation et la manipulation des services
- **L'évolutivité** : Le code doit être extensible et capable d'accueillir les modifications susceptibles d'être ajoutées selon les besoins.
- **La sécurité** : L'accès à l'application ainsi qu'aux données doit être sécurisé. Elle doit donc tenir compte de confidentialité des données des utilisateurs en cryptant tous les mots de passe de leurs comptes, au niveau de la base de données, pour éviter tout accès non autorisé. Il faut aussi sécuriser l'action d'authentification en utilisant l'approche de connexion à base d'échange des jetons (tokens)
- **Le temps réel** : Les différentes fonctionnalités de l'application doivent être gérées en temps réel.

3. Analyse et spécifications des besoins

Dans cette partie nous allons présenter le diagramme des cas d'utilisation globale et l'architecture globale du projet.

3.1. Diagramme des cas d'utilisations globale

Les diagrammes de cas d'utilisation ont pour rôles de recueillir, d'analyser et d'organiser les besoins, ainsi que de recenser les grandes fonctionnalités d'un système. Il s'agit donc de la première étape de modélisation UML pour la conception d'un système. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs (actors), ils interagissent avec les cas d'utilisation (use cases).[1]

La figure numéro 1 ci-dessous présente le diagramme des cas d'utilisation globale

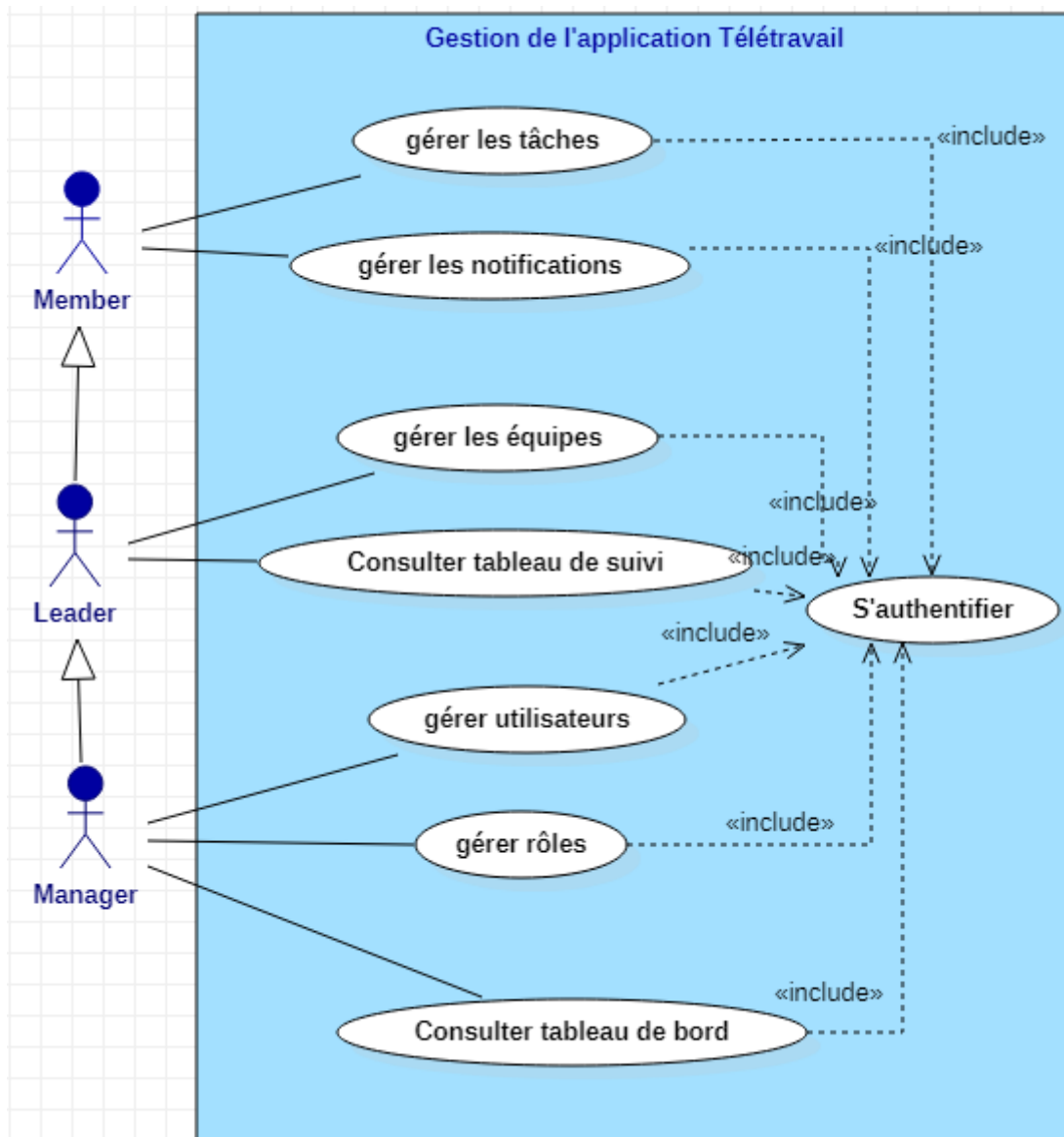


Figure 1 : Diagramme des cas d'utilisation globale

3.2. Architecture de la solution

Pour chaque système informatique, nous avons besoin de choisir l'architecture de la solution adéquate pour sa réalisation et qui peut assurer un bon fonctionnement et une haute performance. Nous allons présenter dans cette section l'architecture physique et l'architecture de déploiement de ce système.

3.2.1. Architecture physique

Dans ce contexte nous allons détailler l'architecture physique de notre application, qui présente l'ensemble des composants matériels que supportent l'application, nous allons adopter une architecture à trois niveaux.

Les trois niveaux de l'architecture sont :

Un client léger : c'est le navigateur web permettant à l'utilisateur d'accéder au site via internet.

Un middle tiers : c'est le serveur d'application qui héberge toutes les couches de l'application à développer.

Un tiers de données : c'est le serveur qui met les données à la disposition des opérateurs d'obsèques tout en assurant des droits accordés à ces derniers.[2]

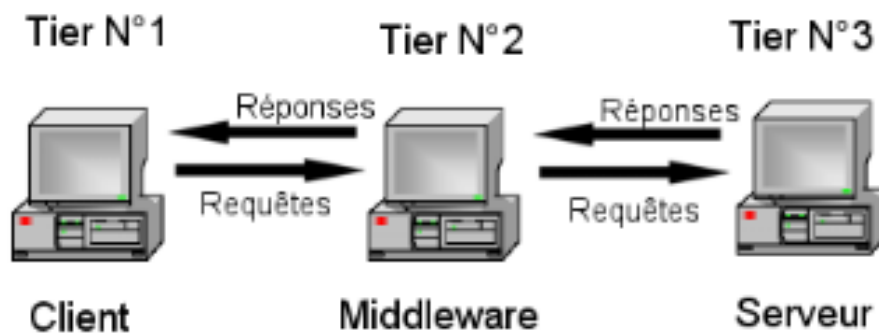


Figure 2 : Architecture physique du système

3.2.2. Architecture Logique

Par complémentarité à l'architecture physique qui permet de distinguer les différents niveaux physiques de l'application, l'architecture logique s'intéresse plutôt au découpage logique de l'application et la façon de regrouper les composants selon les traitements qu'ils effectuent.

3.2.2.1. Modèle architectural du Front-End

Pour la partie web du Front-End, nous utilisons Angular8 qui est doté d'une architecture robuste et évolutive.

Cette architecture est basée sur le composant et son Template associé. Voici ses différents éléments :

- **Modules** : un module déclare un contexte de compilation pour un ensemble de composants dédiés à un domaine d'application.
- **Components** : un composant définit une classe qui contient une logique et des données d'application (peuvent être séparés dans des classes ou interfaces modèles), et est associé à un modèle HTML (Template) qui définit une vue à afficher avec un style d'affichage.

- **Template** : un Template est un modèle qui combine du HTML et du balisage Angular qui peut influencer sur la vue avant l’affichage de cette dernière. Une communication entre un composant et sa Template est possible grâce à la liaison de données (data binding).
- **Service** : c’est une classe injectable (Injection de dépendance) destinée à exécuter les opérations, dont les données, ou la logique n’est pas associée à un Template spécifique et qui est principalement partagé entre plusieurs composants.
- **Routing (ou routage)** : le routage fournit un service qui permet de définir un chemin de navigation.

La figure numéro 3 ci-dessous présente l’architecture du front-end du projet[3]

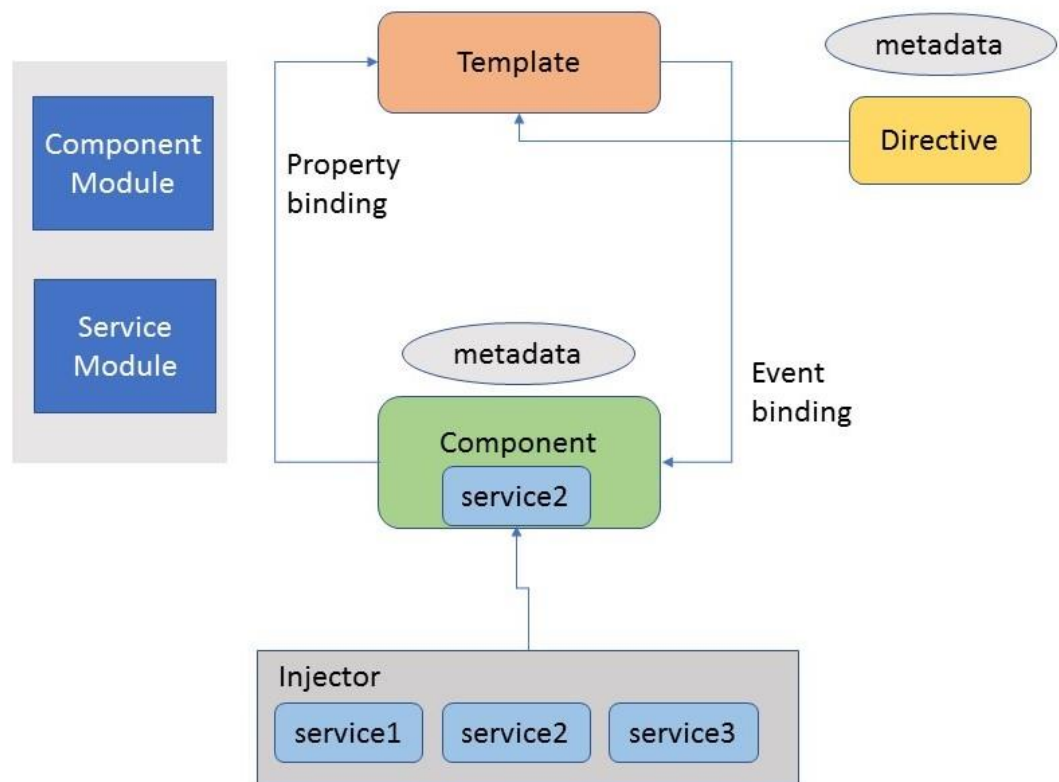


Figure 3 : Architecture Front-End

3.2.2.2. Modèle architectural du Back-End

La conception de notre partie back-end est modélisée comme suit dans le diagramme de package. Ce diagramme illustre une représentation graphique de l’organisation de l’application qui identifie les liens de dépendance entre les paquets de l’application.

La figure numéro 4 montre les packages utilisés pour le développement de notre application.

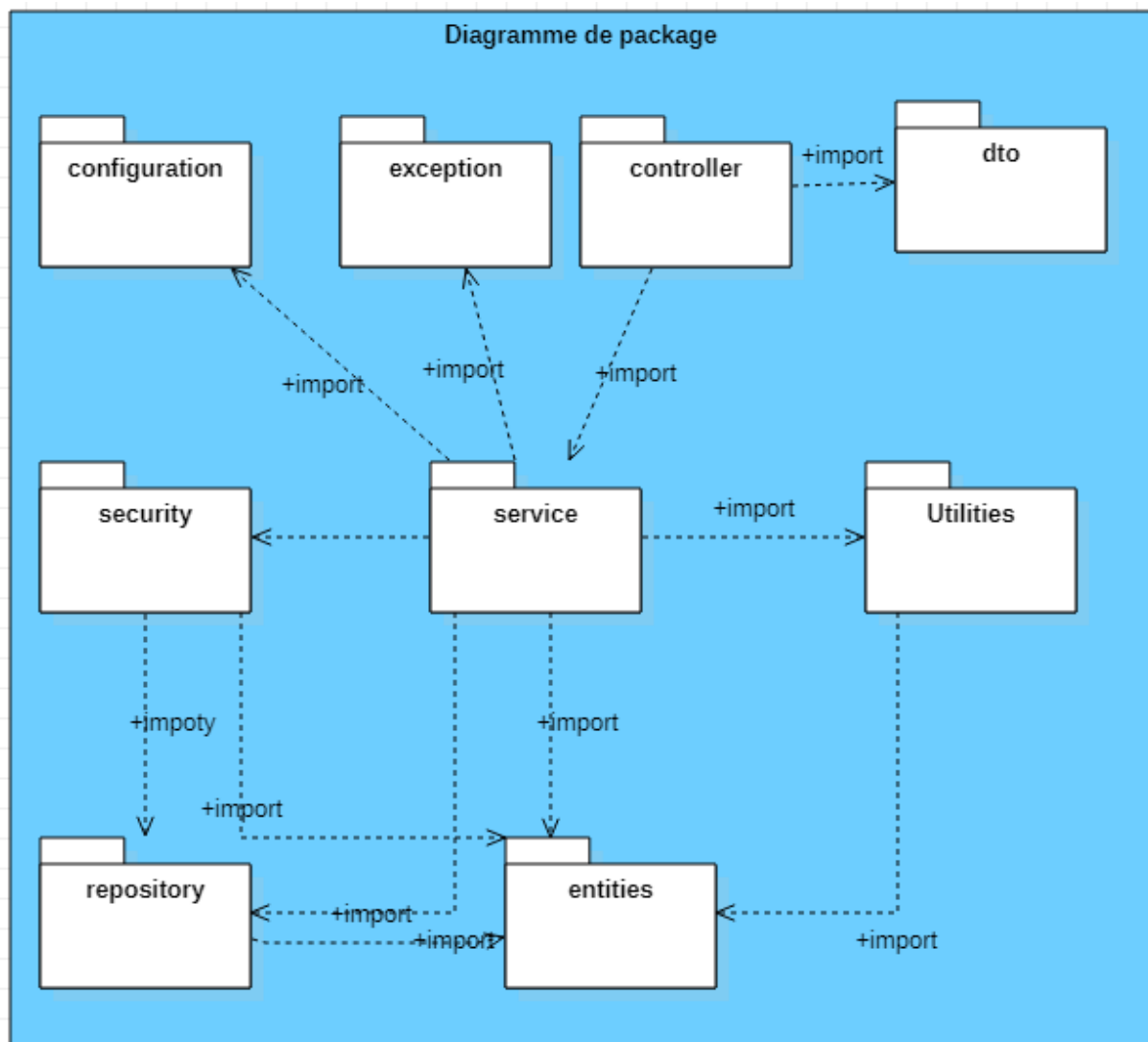


Figure 4 : Diagramme de package de l'application

- **Entities** : Contient les classes entités de l'application.
- **DTO** : Data Transfer Object. Contient les classes qui vont être utilisées pour encapsuler les données et les envoyer vers l'application Front-end, et aussi comme types de retour.
- **Repository** : C'est l'ensemble des interfaces héritant de l'interface JpaRepository. L'objectif de ces interfaces consiste à rendre la création de la couche d'accès aux données plus rapide et fluide.
- **Service** : C'est l'ensemble des classes qui implémentent la logique métier de l'application.
- **Controller** : Contient les Contrôleurs Rest.
- **Security** : C'est le packet où nous avons implémenté le module de sécurité de l'application.
- **Exceptions** : Contient les déclarations des exceptions des modules.
- **Configuration** : Contient les classes de configuration qui doivent être lancés dès le lancement du projet.
- **Utilities** : C'est le package où se fait l'intégration des apis et fonctionnalités externes tels que l'Email, sms et autres.

4. Environnement de travail

La présentation de l'environnement de travail consiste à présenter l'environnement matériel et logiciel utilisé pour l'implémentation de notre application.

4.1. Environnement matériel

Ce projet a été réalisé en utilisant un ordinateur ayant les caractéristiques suivantes :

Tableau 3 : Environnement Matériel

Caractéristique	PC
Processeur	Intel CORE I5
Ram	8GO
Système d'exploitation	Windows 10 professionnel




4.2. Environnement Logiciel

Dans cette section, nous allons présenter les outils de développement et de modélisation, les langages de programmations et les technologies utilisés.

4.2.1. Outils de développement et modélisation

Les logiciels que nous avons utilisés pour la réalisation de ce projet sont représentés par le tableau numéro 4 ci-après

Tableau 4 : Outils de développement et modélisation



Logiciel	Logo	Description
Spring tools Suite (STS)		Spring Tool Suite est une application basée sur Eclipse facilitant la création de projet Spring. [4]
Visual Studio Code		Visual Studio Code un éditeur de code léger mais puissant qui fonctionne sur toutes les plateformes. Il prend en charge la plupart des langages connus comme JavaScript, TypeScript, Python.[5]
MySQL		MySQL est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde.[6]

StarUML		StarUML est un logiciel de modélisation UML, qui a été "cédé comme open source" par son éditeur, à la fin de son exploitation commerciale (qui visiblement continue ...), sous une licence modifiée de GNU GPL.[7]
XAMPP		XAMPP est un ensemble de logiciels permettant de mettre en place un serveur Web local, un serveur FTP et un électronique.[8]
Gitlab		c'est une plateforme permettant d'héberger et de gérer des projets web de A à Z. Présentée comme la plateforme des développeurs modernes, elle offre la possibilité de gérer ses dépôts Git et ainsi de mieux appréhender la gestion des versions de vos codes sources.[9]

4.2.2. Les outils de test

Le test du logiciel fait partie du cycle de vie du développement , son objectif est de s'assurer que le code à déployer est de haute qualité, sans bugs ni erreurs logiques. Afin de faire les tests de l'application nous utilisons les outils suivants



Tableau 5 : Outils de test

Logiciel	Logo	Description
SwaggerUI		C'est une plateforme qui offre des outils permettant de générer la documentation pour son API Web. Il offre également une interface permettant d'explorer et tester les différentes méthodes offertes par le service.[10]
Junit 5		C'est un framework de test unitaire, "JUnit" qui est un framework open source pour le développement et l'exécution de tests unitaires avec le langage Java[11]

4.2.3. Langages de programmation


Le tableau numéro 6 ci-dessous présente les langages de programmation pour la partie Back-end et front-end de l'application.


Tableau 6 : Langages de programmation

Langage	Logo	Description
Java		Java est un langage de programmation orienté objet et une plateforme informatique qui ont été créés par Sun Microsystems en 1995.une plate-forme informatique qui ont été créés par Sun Microsystems en 1995.[12]
TypeScript		C'est un langage de programmation libre et open source développé par Microsoft qui a pour but d'améliorer et de sécuriser la production de code JavaScript. [13]

4.2.4. Technologies utilisés

Tableau 7 : Technologies utilisées

Technologies	Logo	Description
Spring Boot 2		Pour le développement du back end nous avons opté pour Spring boot 2. C'est un Framework java créé par l'équipe Pivotal qui permet de simplifier le démarrage et le

		développement de nouvelles applications Spring en réduisant la complexité de configuration.[14]
Angular 8		Pour la partie front-end web, nous avons opté pour Angular 8. Angular est un Framework orienté composant qui facilite la création d'une application web. Il permet de créer des applications de type SPA (single page application) en se basant sur un système de routage agile (sans rafraîchissement de page).[15]
Chart.js		Chartjs est une bibliothèque de code JavaScript open source simple mais flexible pour les concepteurs et les développeurs. C'est un outil JavaScript de représentation des données sous forme de graphes statistiques.[16]

5. Planification des sprints

Un sprint est une itération de quelques semaines dans laquelle nous travaillons à produire un incrément du produit potentiellement livrable. Après avoir fixé le backlog du produit, nous répartissons l'ensemble des « stories » dans un backlog de sprint, incluant 4 releases.

Nous allons commencer par un tableau descriptif du déroulement général du stage pendant les 6 mois :

Tableau 8 : Déroulement de stage

	5 Avril	5 Mai	5 juin	5 Juillet	5 Aout	5 Septembre
Formation et conception	*					
Développement	*	*	*	*	*	
Test					*	*

Nous présentons ensuite les sprints à réaliser à réaliser tout au long de notre travail par ce présent tableau.

Tableau 9 : Planification des releases

Id	Release	Date de début	Date de fin
1	Gestion d'accès, gestion des utilisateurs et gestion des rôles	20/04/2021	04/05/2021
2	Gestion des équipes et gestion des tâches	05/05/2021	25/05/2021
3	Gestion des Statistiques,historiques et calcul des KPI	26/05/2021	01/07/2021
4	Gestion des notifications et messageries en temps réel	02/07/2021	20/08/2021

Conclusion

Dans ce chapitre, nous avons commencé par la présentation de l'équipe du projet, les besoins fonctionnels, non fonctionnels, le product backlog, le diagramme de cas d'utilisation global,. Après, nous avons illustré l'architecture physique et logique de l'application, le choix des technologies, et la configuration matérielle et logicielle de l'application. Enfin nous avons établi la planification des sprints

Bibliographie

Glossaire / Acronymes

Annexes

Résumé

Mots clés :.

Abstract

Keywords:

الملخص

الكلمات المفتاحية :