

# **Architecture physique et environnement de travail**

## Table des matières

1. Architecture de la solution .....	3
1.1. Architecture physique .....	3
1.2. Architecture Logicielle.....	4
1.2.1. Design Pattern .....	4
1.2.2 Modèle architectural du Front-End .....	4
2. Environnement de travail .....	7
1.2. Environnement Logiciel.....	7
1.2.1. Outils de développement et modélisation .....	7
1.2.2. Les outils de test.....	9
1.2.3. Langages de programmation .....	10
1.2.4. Technologies utilisés .....	11

## Table des figures

Figure 1 : Architecture physique du système .....	3
Figure 2 : Design Pattern MVVM.....	4
Figure 3 : Architecture Front-End.....	6
Figure 4 : Diagramme de package de l'application .....	6

## 1. 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.

### 1.1. Architecture physique

Dans ce contexte nous allons détailler l'architecture physique de notre application, qui présente l'ensemble des composants matériels qui supportent l'application. Partant du fait que notre projet englobera une application web, nous allons adopter une architecture de type client / serveur , un backend qui expose des services web, et deux projets front-end (Clients) , web , qui communiquent avec le backend via le protocole HTTP

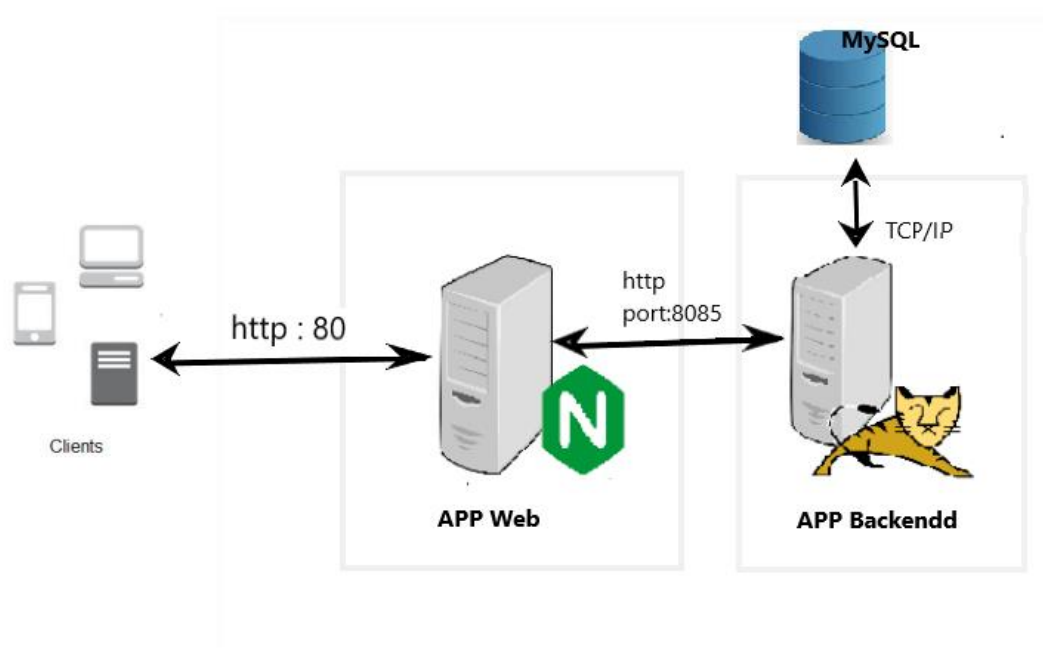


Figure 1 : Architecture physique du système

Notre application sera composée essentiellement de :

- Un serveur web Nginx sur lequel est déployée la partie FrontEnd de l'application,
- Un serveur web Apache Tomcat sur lequel est déployée la partie Backend de l'application,
- Une base de données MySQL
- Client web : Représente le navigateur web.

## 1.2. Architecture Logicielle

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.

### 1.2.1. Design Pattern

Le design pattern MVVM est un dérivé du modèle de conception MVC qui a pour objectif de diviser le développement en 3 couches, ces dernières sont présentées dans

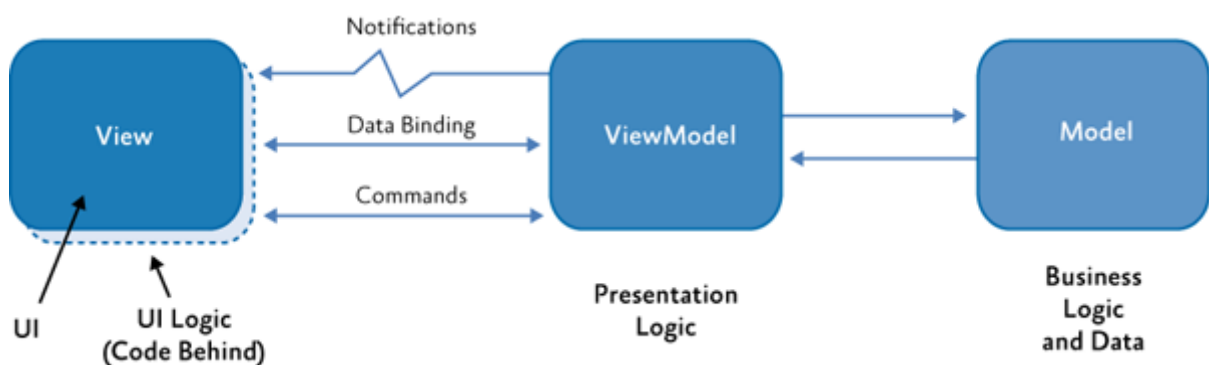


Figure 2 : Design Pattern MVVM

- Model : Cette partie représente le contenu en mode réel (une approche orientée objet), ou à la couche d'accès aux données (une approche centrée sur la donnée).
- View : Cette partie représente l'interface apparente à l'utilisateur à travers laquelle l'utilisateur peut interagir avec des événements.
- ViewModel : Cette couche va jouer le rôle d'un intermédiaire entre la vue et le modèle, elle se base sur la gestion de la logique de la vue

### 1.2.2 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]

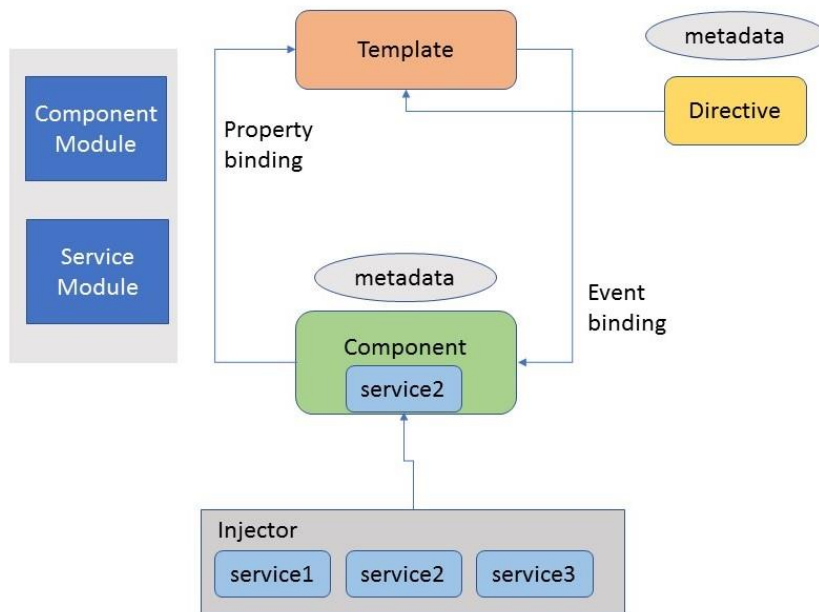
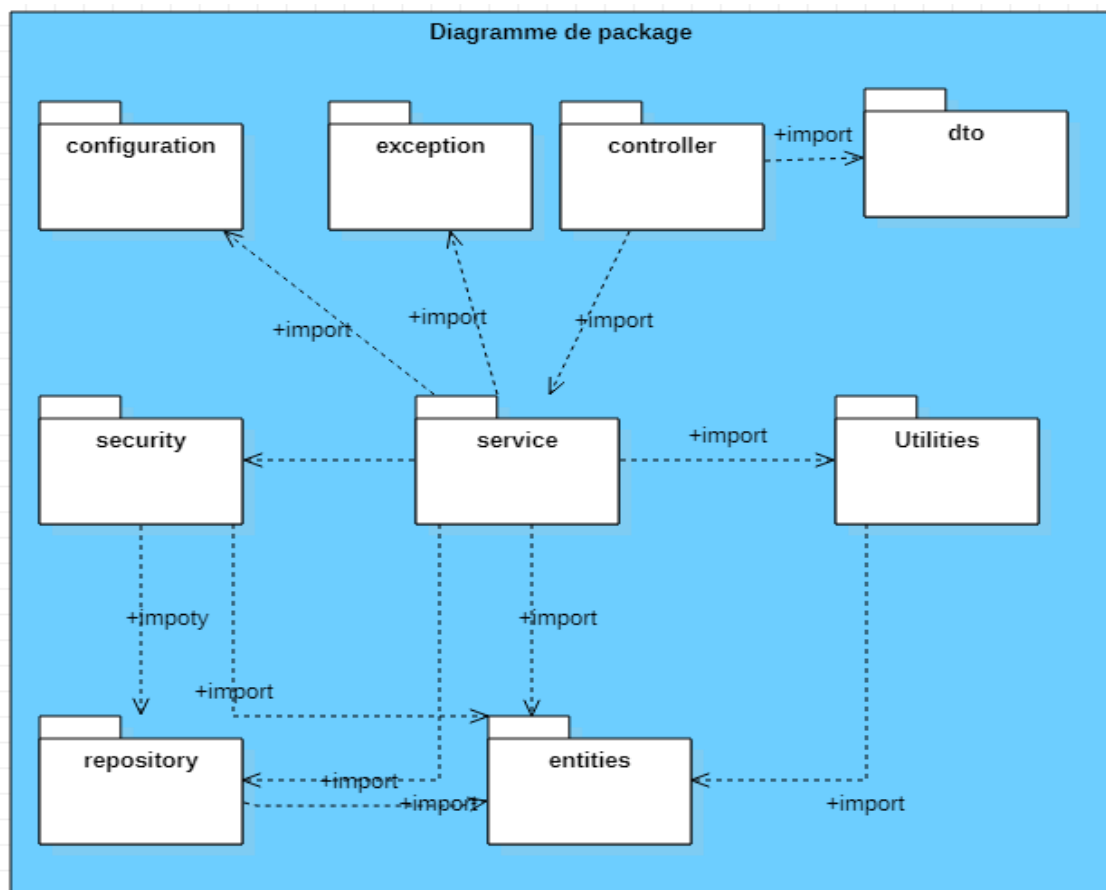


Figure 3 : Architecture Front-End

### 1.2.3. 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.



- **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.

## 2. 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.

### 1.2. Environnement Logiciel

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

#### 1.2.1. 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 1: Outils de développement et modélisation**

Logiciel	Logo	Description
----------	------	-------------

<b>Spring tools Suite (STS)</b>		Spring Tool Suite est une application basée sur Eclipse facilitant la création de projet Spring. [4]
<b>Visual Studio Code</b>		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]
<b>MySQL</b>		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]
<b>StarUML</b>		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]





<b>XAMPP</b>		XAMPP est un ensemble de logiciels permettant de mettre en place un serveur Web local, un serveur FTP et un électronique.[8]
<b>Gitlab</b>		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]

### 1.2.2. 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 2 : Outils de test**



<b>Logiciel</b>	<b>Logo</b>	<b>Description</b>
<b>SwaggerUI</b>		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]
<b>Junit 5</b>		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]

### 1.2.3. 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 3: Langages de programmation**


Langage	Logo	Description
<b>Java</b>		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]
<b>TypeScript</b>		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]
--	--	--

#### 1.2.4. Technologies utilisés

**Tableau 4: Technologies utilisées**

Technologies	Logo	Description
<b>Spring Boot 2</b>		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]
<b>Angular 8</b>		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]
<b>Chart.js</b>		Chartjs est une bibliothèque de code JavaScript open

	 <b>Chart.js</b>	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]
--	--	---