



# Mémoire de Projet de Fin d'Études

Présenté en vue de l'obtention du titre

D'Ingénieur d'État

Spécialité : Intelligence Artificielle et Génie Informatique

✿✿ **NAINIA Youness** ✿✿

**Titre :**

Application Desktop d'une solution RH

**Encadrant Sapress:**  
*Mr. Zakrani Abdelali*

**Encadrant ENSAM:**  
*Mr. Ahmed Abraray*



***Société d'Accueil :***  
***Sapress***



---

**Soutenu le : ??/07/2022 devant le jury composé de :**

- |       |                                      |                  |
|-------|--------------------------------------|------------------|
| • Pr. | : Professeur à l'ENSAM de Casablanca | Président        |
| • Pr. | : Professeur à l'ENSAM de Casablanca | Examineur        |
| • Pr. | : Professeur à l'ENSAM de Casablanca | Encadrant        |
| • Mr. | : Chef de Projet                     | Parrain de stage |





## Dédicaces

**A ma mère**

**A mon père**

**A mes très chers frères et sœurs**

**A toute ma famille**

**A tous mes chers amis**

**Au staff du Groupe Edito**

Je vous remercie pour votre soutien

## Remerciements

Je commencerais d'abord par remercier Allah le tout puissant. Puis j'aimerais avant toute chose, prendre le temps d'adresser toutes mes sincères remerciements aux personnes qui ont contribué de près ou de loin au bon déroulement de ce projet.

Je saisis cette occasion pour remercier notre professeur Mr Zakrani Abdelali pour sa disponibilité, son œil attentif et ces conseils précieux qui ont su m'orienter tout au long de ce projet.

Je tiens ensuite à remercier tout particulièrement les membres du service informatique de Sapress. Mon encadrant de stage Mr. Ahmed ABRARAY (Senior Business Solutions Manager), Et Md. Chaymaa Alouani (chargé de mission) car ce projet n'aurait pas pris jours sans eux et leurs aides. Sans oublier le Directeur des opérations, Mr Yassine Boukhanfar.

Je suis reconnaissant aussi envers Le directeur Rh Mr Mouad Bensouda et toute l'équipe RH pour leurs soutiens et investissement considérable dans ce projet.

Aux finales, Je tiens à remercier Mon institut et tous le corps professoral de L'Ecole Nationale Supérieure d'Arts et Métiers pour la qualité de la formation reçu.

## Résumé

Ma mission durant Ces mois était de développer une application Desktop dédié à l'équipe RH de la société SAPRESS. Son but est de les permettre de gérer efficacement et de manière sécurisés les employés du groupe. En plus de faciliter et automatiser les différents calculs effectués avec ces données sensibles. Cette application viendra remplacer le stockage et traitements faits avec des fichiers Excels qui est vite devenu obsolète et pas pratique. Notre projet suit une approche agile SCRUM et arrive à l'aide d'une conception réussi à gérer ces employés dans une base de données et à effectuer ces calculs qui seront affiché fidèlement à l'utilisateur final.

### Mots clés :

SCRUM, MySQL, UML, base de données, REST API, formules, employés, Angular, Django

## Abstract

My mission during these months was to develop a desktop application dedicated to the SAPRESS HR team. Its purpose is to allow them to manage the entire group employees. In addition to facilitating and automating the various calculations performed with this sensitive data. This application will replace the storage and processing done with Excel files which quickly became obsolete and impractical.

Our project follows an agile SCRUM approach and was able with the help of a successful design to manage the company employees in a database and to perform these calculations. Which then will be displayed faithfully to the end user.

### Key words:

SCRUM, MySQL, UML, database, REST API, formulas, employees, Angular, Django

## ملخص

كانت مهمتي خلال هذه الأشهر تطوير تطبيق سطح المكتب مخصص لفريق الموارد البشرية في شركة صابريس. والغرض منه هو تمكينهم من إدارة موظفي المجموعة بشكل فعال وآمن. بالإضافة إلى تسهيل وأتمتة العمليات الحسابية المختلفة التي يتم إجراؤها باستخدام هذه البيانات الحساسة. سيحل هذا التطبيق محل التخزين والمعالجة التي كان تم إجراؤها باستخدام ملفات بخجل التي سرعان ما أصبحت غير قابلة للإدارة وغير عملية ويصل بمساعدة تصميم ناجح لإدارة هؤلاء الموظفين في قاعدة بيانات وتنفيذ CRUM يتبع مشروعنا نهجاً رشيقاً لص هذه الحسابات التي سيتم عرضها بأمانة للمستخدم النهائي.

## مفاتيح كلمات:

, SCRUM, MySQL, UML, REST API, الموظفين, الصيغ, Angular, Django

## Liste des abréviations

<b>SGBD</b>	Système de gestion de base de données
<b>RH</b>	Ressources humaines
<b>SI</b>	Système d'information
<b>BDD</b>	Base de données
<b>SQL</b>	Structured query langage
<b>JWT</b>	Json web token
<b>UI</b>	User interface
<b>CLI</b>	Command line interface
<b>ORM</b>	Object relationship mapping
<b>SIRH</b>	Système d'information des Ressources Humaines
<b>TS</b>	Typescript



## Liste des figures

Figure 1: Logo du groupe Edito .....	18
Figure 2: Les trois entités du groupe Edito .....	19
Figure 3: Logo de l'entreprise Sochepress.....	19
Figure 4: Logo de l'entreprise Warak .....	20
Figure 5: Logo de l'entreprise Sapress .....	20
Figure 6: Représentants et distributeurs locaux SAPRESS.....	20
Figure 7: Organigramme de la direction des opérations SAPRESS.....	22
Figure 8: Organigramme de la direction système d'information SAPRESS .....	22
Figure 9: Diagramme de Gantt du planning du projet.....	32
Figure 10: Diagramme de cas d'utilisation générale .....	37
Figure 11: Architecture trois tiers .....	41
Figure 12: Logo UML .....	42
Figure 13: Diagramme de classe générale.....	45
Figure 14: Diagramme de séquence authentification .....	51
Figure 15: Vs Code logo .....	54
Figure 16: Logo Postman .....	54
Figure 17: Logo Github.....	55
Figure 18: Logo Typescript.....	55
Figure 19: Logo Python.....	56
Figure 20: Sass Logo.....	56
Figure 21: Logo Django .....	57
Figure 22: Pourcentage des langages de programmations .....	61
Figure 23: Interface Authentification .....	63
Figure 24: Interface authentification Erreurs .....	64
Figure 25: Header état non connecté.....	64
Figure 26: Header état connecté.....	64
Figure 27: Header Utilisateurs options.....	65
Figure 28: Interface changer d'entité .....	65
Figure 29: Interface Menu Gestion .....	66
Figure 30: Header Menu Gestion .....	66
Figure 31: Interface Gestion Employé .....	67
Figure 32: Interface Gestion Employés Filtrés.....	68
Figure 33: Interface Liste Employé.....	68
Figure 34: Liste Employé attributs normaux.....	69
Figure 35: Liste Employé attribut one to many .....	69
Figure 36: Liste Employé many to many .....	69
Figure 37: Footer .....	70
Figure 38: Interface modifier Employé.....	70
Figure 39: Modifier Employé attribut one to many.....	70
Figure 40: Modifier Employé attribut many to many .....	71
Figure 41: Structure de fichiers 2 de Angular .....	72
Figure 42: structure de fichiers 1 de Angular .....	72
Figure 43: models dans Angular .....	74
Figure 44: Code Angular du Modèle Affectation.....	74
Figure 45: cycle de vie NGRX.....	76

Figure 46: Header avec taille d'écran petite .....	78
Figure 47: Interface Liste Employé avec taille d'écran petite .....	78
Figure 48: Structure de fichiers Django .....	83
Figure 49: composition en arbre d'une expression .....	86
Figure 50: Exemple de metadonnées.....	88
Figure 51: Exemple d'un access token .....	90
Figure 52: Erreur non authentifié backend.....	91
Figure 53: Diagramme de classe d'autorisation.....	93
Figure 54: Diagramme de déploiement de l'application.....	95

## Liste des tableaux

Tableau 1:Fiche technique de l'entité SAPRESS .....	21
Tableau 2:Formules de traitement.....	25
Tableau 3:Formules dans la base de données.....	28
Tableau 4: Product Backlog .....	38
Tableau 5: Table Employé Rubrique.....	48
Tableau 6:Table Affectation.....	49
Tableau 7:Environnement Matériel.....	53
Tableau 8: Comparaison popularité des SGBDs 2022(source: <a href="https://db-engines.com">https://db-engines.com</a> ) .....	60
Tableau 9:Nombre de lignes de codes Django .....	61
Tableau 10:Nombre de lignes de code Angular .....	62
Tableau 11: Comparaison de stockages dans le navigateur .....	80

## Table des matières

<i>Dédicaces .....</i>	<b>3</b>
<i>Remerciements .....</i>	<b>4</b>
<i>Résumé.....</i>	<b>5</b>
<i>Liste des abréviations .....</i>	<b>8</b>
<i>Liste des figures .....</i>	<b>9</b>
<i>Liste des tableaux.....</i>	<b>11</b>
<i>Table des matières.....</i>	<b>12</b>
<i>Introduction générale.....</i>	<b>15</b>
<b>CHAPITRE 1 : « CADRE GENERALE DU PROJET».....</b>	<b>17</b>
<b>I. STRUCTURE ET ORGANISATION DU GROUPE EDITO .....</b>	<b>18</b>
1. Présentation de l'organisme d'accueil Groupe Edito .....	18
2. Organisme du Groupe.....	18
3. L'entité Sapress .....	21
<b>II. PRESENTATION GENERALE DU PROJET .....</b>	<b>23</b>
1. Définitions Des Ressources humaines.....	23
2. Etude de l'existant .....	23
3. Le but à atteindre .....	23
4. Problématiques .....	24
5. Valeur du projet .....	24
6. Traitement et formules explication.....	25
7. Solution proposée .....	26
<b>III. METHODOLOGIE DE TRAVAIL.....</b>	<b>29</b>
1. Méthodologie Agile.....	29
2. Pourquoi Scrum ?.....	29
3. Scrum.....	30
4. Scrum individuel.....	30
<b>IV. PLANIFICATION DU PROJET .....</b>	<b>31</b>
1. Diagramme de Gantt.....	31
2. Résumé des taches accomplis .....	32
<b>CHAPITRE 2 : « ANALYSE DE L'APPLICATION ET.....</b>	<b>34</b>
<b>SPECIFICATION DES BESOINS » .....</b>	<b>34</b>

<b>I.</b>	<b>SPECIFICATIONS DES BESOINS .....</b>	<b>35</b>
1.	Les besoins fonctionnels .....	35
2.	Les besoins non fonctionnels .....	36
<b>II.</b>	<b>Analyse des cas d'utilisations .....</b>	<b>36</b>
1.	Présentation des acteurs : .....	36
2.	Diagramme de cas d'utilisation.....	36
<b>III.</b>	<b>Le Backlog du produit .....</b>	<b>38</b>
	<b>CHAPITRE 3 : « Conception » .....</b>	<b>39</b>
<b>I.</b>	<b>Conception Générale .....</b>	<b>40</b>
3.	Architecture trois tiers .....	40
<b>II.</b>	<b>Conception détaillé .....</b>	<b>42</b>
1.	Langage UML .....	42
2.	Diagrammes de classes .....	44
3.	Diagrammes de séquences .....	49
	<b>CHAPITRE 4 : « Réalisation » .....</b>	<b>52</b>
<b>I.</b>	<b>Environnement de travail .....</b>	<b>53</b>
1.	Environnement Matériel.....	53
2.	Environnement Logiciel.....	53
3.	Langages de Programmation .....	55
4.	Frameworks utilisés .....	57
5.	Justification des choix .....	58
6.	Analyse des lignes de codes.....	61
<b>II.</b>	<b>Réalisation du Frontend .....</b>	<b>62</b>
1.	Description des interfaces graphiques .....	62
2.	Structure des fichiers .....	71
3.	Gestion des états .....	75
4.	Réactivité .....	77
5.	Concepts utilisés .....	79
<b>III.</b>	<b>Réalisation Backend .....</b>	<b>81</b>
1.	Django Rest Framework.....	81
2.	Description des Endpoints .....	82
3.	Structure des fichiers .....	83
4.	Rest Api .....	84
5.	Filtrage, Pagination et tri.....	85
6.	Authentification .....	88
	<b>CHAPITRE 5 : « Perspectives d'amélioration » .....</b>	<b>92</b>
<b>I.</b>	<b>Autorisation .....</b>	<b>93</b>

<b>II.</b>	<b>Tests et documentations .....</b>	<b>94</b>
<b>III.</b>	<b>Déploiement .....</b>	<b>95</b>
<b>IV.</b>	<b>Développement.....</b>	<b>96</b>
	<i>Problèmes rencontrés .....</i>	<i>97</i>
	<i>Conclusion générale et perspectives .....</i>	<i>99</i>
	<i>Bibliographie .....</i>	<i>101</i>

## Introduction générale

Au cours des dernières décennies, le domaine des ressources humaines s'est élargi vers bien plus que le recrutement et l'administration des salariés. Donc L'informatisation du domaine est devenue d'une grande importance Surtout depuis la pandémie. Et D'après une Enquête en 2021 réalisée sur un panel de 825 professionnels des Ressources Humaines en France par les Éditions Tissot et PayFit. 83% des RH utilisent des outils digitaux.

Dans le cadre de l'obtention du diplôme d'ingénieur d'état en Génie Informatique et intelligence artificiel, j'ai effectué un stage Du 01 février 2022 au 01 Aout 2022 au sein du siège de l'entreprise SAPRESS à Casablanca qui mise beaucoup sur la révolution d'informatique.

En plus d'améliorer mon esprit d'analyse et mes compétences techniques, Ce stage s'est montré comme l'opportunité optimale d'améliorer mes soft skills et approfondir mes connaissances métier logistique ou autre. Une étape sans doute importante dans mon parcours d'ingénieur.

L'entreprise SAPRESS est la filiale dédiée à la logistique du groupe Edito. Possédant un réseau logistique performant dans plusieurs villes du royaume, elle propose donc plusieurs services logistiques notamment la distribution de la presse.

Dans cet environnement, j'ai été amené à développer une application Desktop qui aide les agents du service RH dans divers opérations qu'ils font au quotidien. Elle consiste essentiellement en une application qui gère efficacement les employés de tout le groupe. Premièrement, en stockant de manière sécurisé toutes les données relatives à ces derniers. Puis de permettre à l'utilisateur d'ajouter, supprimer et modifier ces données. Ensuite, en se basant sur ça, l'équipe RH doit pouvoir faire des calculs et des traitements sur demande en utilisant des formules qu'ils définissent.

Une méthodologie SCRUM a été adopté tout au long du projet pour répondre aux besoins instables de ce projet.

Le défi présent est de gérer de manière sécurisé ces données autrefois gérer dans des fichiers Excel tout en donnant à l'utilisateur une possibilité de tout configurer et de faire les traitements quand il veut. Le tout dans une application Desktop conviviale et facile à utiliser.

Le présent rapport se divise en cinq parties. Premièrement, la partie cadre générale du projet ou on commence par présenter la structure et organisation du groupe Edito. S'en suivra une présentation générale du projet ou le but et la problématique du projet vont être discutés. L'étape suivante est la méthodologie, la planification et la spécification des besoins. Deuxièmement, c'est la partie conception qui regroupe l'architecture générale et des diagrammes UML. Ensuite la partie réalisation du frontend et backend. Puis on finira par les perspectives d'amélioration.



## CHAPITRE 1 : « CADRE GENERALE DU PROJET »

## Introduction

Ce chapitre a pour objectif de présenter en détails des généralités sur l'environnement de l'entreprise SAPRESS et ensuite de présenter le cadre général de notre projet.

## I. STRUCTURE ET ORGANISATION DU GROUPE EDITO

### 1. Présentation de l'organisme d'accueil Groupe Edito

Le groupe EDITO VENTURES est le fruit d'une fusion depuis l'an 2019 de trois entreprises, SAPRESS créé en 1977 (Société Arabo-Africaine de distribution et d'éditions de presse), SOCHEPRESS crée le 1924 (Société Chérifienne de distribution et de presse) et de Warakpress. Chacune d'entre elles effectue des tâches bien précises et contribue ainsi au succès du groupe.

Ce groupe possède un réseau de plus de 5000 points de ventes répartie dans un total de 23 villes du royaume. Ce large réseau leurs permet de soutenir le développement de ses activités dans les services de la logistique pour le marché marocain et d'instaurer la culture et l'éducation avec leurs stratégie de distributions de livres et manuelles scolaires.

Ces trois Filiales de la société mère édito partage une vocation essentielle d'investir dans le secteur de la distribution, de la culture et de l'éducation et ont pour principal objectif de maintenir la disponibilité de la presse partout dans le royaume auprès des lecteurs dans des conditions économiques équilibrées, tout en donnant aux éditeurs l'opportunité de bénéficier de la synergie des deux réseaux de distribution en termes de couverture et de distribution numérique.



*Figure 1: Logo du groupe Edito*

### 2. Organisme du Groupe

Le Groupe EDITO est constitué de trois entreprises, SAPRESS, SOCHEPRESS, et WARAKPRESS (warak trading)



Figure 2: Les trois entités du groupe Edito

## a. Entité de distribution des livres

Créée en 1924, SochePress est l'entité qui s'occupe de la distribution des livres et fournitures scolaires ainsi que la presse nationale et internationale. Elle a commencé aussi ses premières éditions des Manuels scolaires. Lorsqu'on parle de SochePress, on fait référence à la partie qui administre les produits, les fournisseurs et les éditeurs.



Figure 3: Logo de l'entreprise Sochepress

## b. Entité de commercialisation

C'est l'entité qui commercialise les fournitures scolaires (BTS), les fournitures de bureau, papiers A4, toners, les cahiers et les manuels scolaires. Elle offre de plus un service de paiement mobile.



Figure 4:  
Logo de  
l'entreprise Warak

### c. Entité logistique

Avec une expérience de plus de 95 ans et une présence Nationale avec 23 Agences sur les 12 régions du Royaume., Sapress se positionne comme étant le leader sur son domaine.

Son activité reste est principalement la distribution de la presse nationale et internationale Et la logistique en général.



Figure 5: Logo  
de l'entreprise Sapress

### Couverture logistique :

10 millions de colis / an

Distribution avant 8h

5000 points de livraison / jour

50 relais couvrant les 12 régions du Royaume

161 villes couvertes + de 300 collaborateurs



Figure 6: Représentants et  
distributeurs locaux SAPRESS

Sapress est l'entreprise au sein du groupe Edito ou ce stage s'est déroulé.

### 3. L'entité Sapress

#### a. Fiche technique de l'entité SAPRESS

Tableau 1: Fiche technique de l'entité SAPRESS

Raison sociale	SAPRESS Logistique & messagerie
Forme juridique	Société anonyme
Nom du dirigeant	M. Amine BENCHEKRI
Effectif	Entre 200 et 500
Date de création	1977
Activité	Distribution de presse & messagerie
Siège social	Zone industrielle Sidi Maarouf, rue 80 n° 20 20280 Casablanca
Capital	30,000,000 DH
Chiffre d'affaires	De 100,000,000 à 500,000,000 DH
Site web	<a href="http://www.sapress.ma">www.sapress.ma</a>

#### b. L'organigrammes de SAPRESS

Vous trouvez ci-dessous Deux organigrammes simplifiés Ayant pour but de montrer les liens de hiérarchies entre les personnes qui sont intervenues dans le projet.

Ceci dit, Ce projet c'est passer principalement dans la direction des opérations.

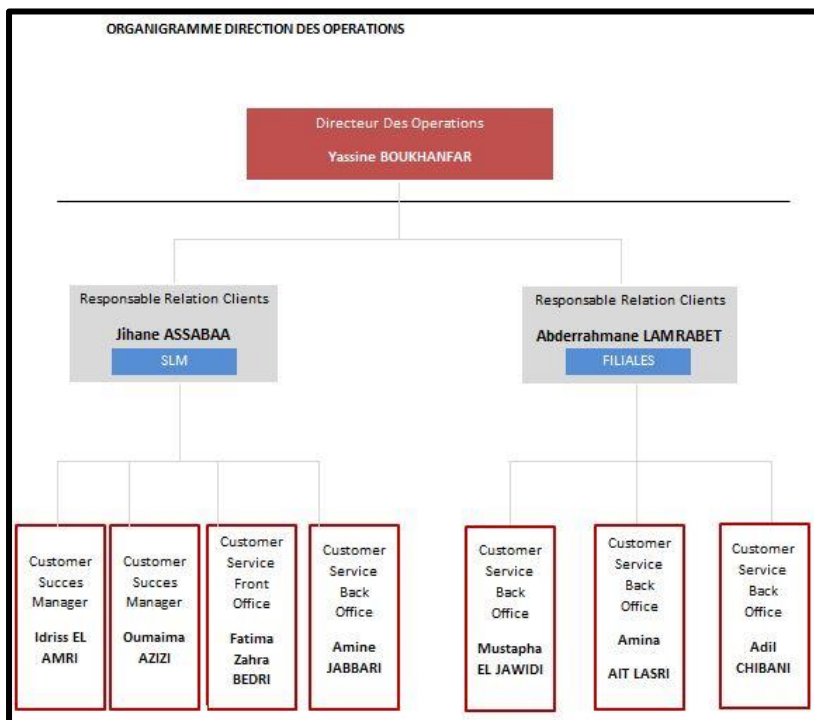


Figure 7: Organigramme de la direction des opérations SAPRESS

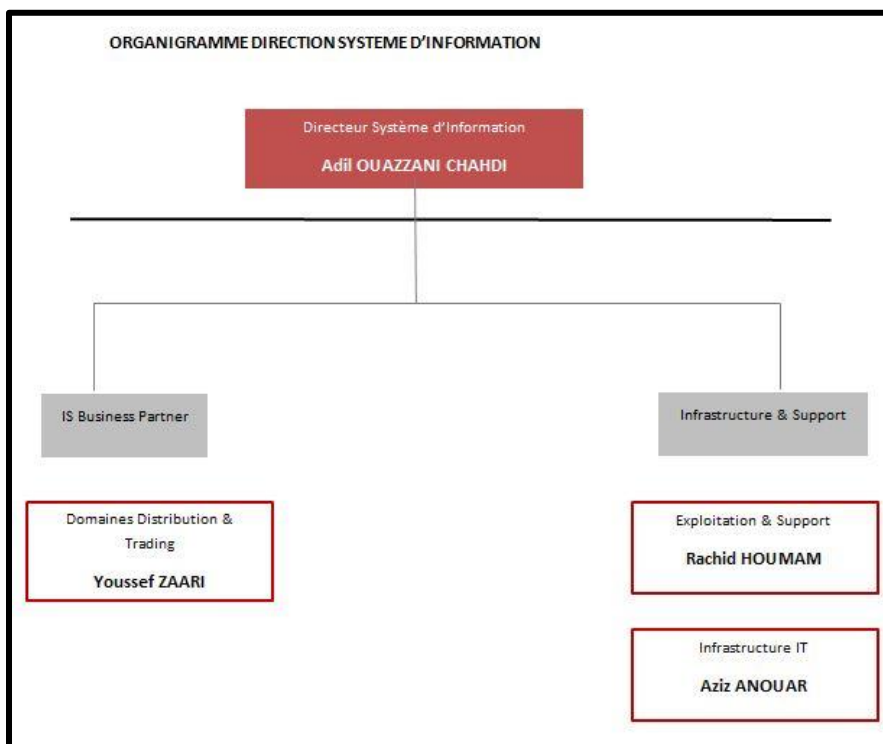


Figure 8: Organigramme de la direction système d'information SAPRESS

## II. PRESENTATION GENERALE DU PROJET

### 1. Définitions Des Ressources humaines

Les ressources humaines, souvent connues sous leur acronyme RH, correspondent en général à un département de l'entreprise qui est en charge des relations entre employeurs et salariés.

En effet, c'est ce département qui s'occupe d'une partie importante du management des collaborateurs au sein d'une organisation. Un bon management des ressources humaines est impératif si une entreprise ou toute autre entité administrative veut correctement fonctionner.

### 2. Etude de l'existant

Au moment où ce stage se déroule, le groupe Edito n'utilise aucun SIRH (système de gestion de ressources humaine).

Toute la gestion des ressources humaines est faite dans des fichiers Excel allant du recrutement au départ passant par la gestion des stagiaires, le suivi et reporting Rh.

Au sein du groupe Edito, toute la gestion paie est prise en charge par un prestataire externe. Le groupe reçoit ensuite toutes les informations sur les employés regroupées dans des fichiers Excel qui doivent être maintenus et mis à jour régulièrement.

On utilise ensuite des formules mathématiques spécifiques, pour retrouver un tableau de traitement qui doit être généré à la fin de chaque mois.

Tout ce travail est exclusivement fait dans des fichiers Excel ce qui rend la tâche longue et fastidieuse.

### 3. Le but à atteindre

D'après ce qui précède, l'équipe Rh (en plus de leurs tâches quotidiennes) s'occupe de faire les deux tâches suivantes très importantes :

- Gérer (ajouter, supprimer, modifier) et maintenir les données de tous les employés en temps réel

- Générer un tableau de Traitement à la fin de chaque mois

Notre but est donc de faciliter et organiser ce travail. Ce qui aura comme résultat l'accélération considérable de l'efficacité et de la performance de l'entreprise.

## 4. Problématiques

Plusieurs problèmes se pose dans l'état présent des choses :

La gestion des employés dans des fichiers excès est non seulement pas pratique mais surtout présente un risque sérieux de sécurité car toutes les informations sensibles des employés (comme leurs salaire) sont gardées sur un ordinateur portable, au sein d'un fichier plat.

De plus, comme déjà expliqué, La présence d'un système de gestion des ressources humaines efficient et efficace permet de considérablement augmenter la productivité des salariés au sein d'une entreprise. En effet, ces taches répétitif et calculs laborieux qui peuvent prendre jusqu'à une semaine de travail empêche l'équipe RH de faire leurs travaux et taches quotidiennes qui sont d'une grande importance a l'entreprise.

Donc comment arriver à gérer de manière sécurisée non seulement les employés mais aussi toutes les informations relatives à eux dans une seule application conviviale ? Et permettre à l'utilisateur de faire des traitements divers ?

## 5. Valeur du projet

Premièrement, Le résultat finale de notre application est le tableau de traitement qui doit être calculé à la fin de chaque mois. Ce tableau sert à faire des prédictions faites en calculant l'évolution de la masse salariale au fils du temps. En d'autres termes, grâce à la masse salariale actuelle qu'on peut déduire du tableau de traitement et grâce quelques hypothèses, on arrive à estimer la masse salariale potentiel jusqu'au 5 prochaines années. Ce qui représente une information très importante pour la Trésorerie de l'entreprise car ça leurs permet de savoir approximativement combien d'argent l'entreprise doit mettre de côté pour assurer cette somme dans les années futures. Chose qui peut très bien influencer les décisions stratégiques de l'entreprise.



## 6. Traitement et formules explication

Quand on parle de traitement, on fait allusion à plusieurs formules mathématiques qui doivent être appliquées sur chaque employé à la fin de chaque mois.

On notera ci-dessous les plus importantes sous forme de fonctions Excel modifiées pour qu'elles soient lisibles :

Tableau 2: Formules de traitement

Nom de formule	Formule
<b>Ancienneté1</b>	(aujourd'hui – date entrée)/365
<b>Taux d'ancienneté</b>	SI (Ancienneté >=25 ;25 ; SI (Ancienneté >=20 ;20 ; SI(Ancienneté >=12;15;SI( Ancienneté >=5;10;SI( Ancienneté >=2;5;0))))
<b>Ancienneté2</b>	(salaire de base * Taux d'ancienneté) / 100
<b>Brut imposable</b>	$\Sigma(\text{primes} + \text{indemnités})$ + participation
<b>Base Cnss plafonnée</b>	SI( Brut imposable >6000;6000; Brut imposable )
<b>Brut global</b>	$\Sigma(\text{primes} + \text{indemnités})$ +total primes V soumises
<b>MUTUELLE PP</b>	(5,156* Brut imposable )/100
<b>CNSS PP</b>	( Brut imposable *9,85%)+( Base Cnss plafonnée *8,98%)
<b>CIMR PP</b>	Brut imposable *7,8%
<b>Coût salarial</b>	CIMR PP + CNSS PP + MUTUELLE PP + Brut global
<b>MUTUELLE PS</b>	Brut imposable *(4,176)/100
<b>CNSS PS</b>	SI( Brut imposable >6000;268,8; Brut imposable *4,48%)
<b>CIMR PS</b>	Brut imposable *0,06
<b>Brut global - MUTUELLE PS - CNSS PS - CIMR PS - IR</b>	
<b>Ancienneté3</b>	ARRONDI.SUP( Ancienneté1;0)

<b>cc annuel</b>	Coût salarial*12,85
<b>tx horaire brut</b>	Brut global /191
<b>PREAVIS MOIS</b>	SI( Ancienneté1<1;1;SI(AGE<5;2;3))
<b>Préavis net</b>	PREAVIS MOIS * SALAIRE NET
<b>Domage &amp; intérêts net</b>	SI(SALAIRE NET *1,5* Ancienneté3> SALAIRE NET*36; SALAIRE NET*36; SALAIRE NET*1,5* Ancienneté3)
<b>ID LEGAL</b>	domage & intérêts net+ préavis net+ LICENCIEMENT 2
<b>Date retraite</b>	Date naissance+(60*365,25)
<b>Mois restants</b>	( Date retraite - AGE)/365,25*12
<b>ARR</b>	ARRONDI.SUP( Mois restants;0)
<b>Net restant à 100%</b>	SALAIRE NET* prime de nuits

## 7. Solution proposée

### a. Gestion des employés

Pour la partie gestion, On propose ici de permettre à nos clients (équipe Rh) d'abord de pouvoir gérer leurs Employés au sein même de notre application desktop. Toutes les données critiques seront stockées dans une base de données qu'on pourra accéder via une api. Ce qui nous permettra éventuellement d'ajouter, supprimer ou modifier n'importe quelle table.

### b. Formules à calculer

Ensuite, En ce qui concerne les formules, plusieurs solutions sont présentes :

La première proposition était de les définir comme fonctions dans la partie frontend. Ce qui nous octroi un avantage considérable niveau temps de réponse (le serveur traite plusieurs requêtes). Mais il s'est vite avéré impossible d'utiliser cette méthode car le client veut absolument la possibilité de créer de nouvelles formules depuis l'interface graphique.

Chose impossible à faire car les fonctions seraient codées à dur.

Donc la solution qui nous reste est de stocker ces formules dans notre base de données. Sous forme de deux tables. Une table Formule où les formules seront stockées et une autre pour les variables (constantes que l'utilisateur peut modifier) utilisées par les formules. Mais si l'enregistrement des variables est facile, comment faire pour stocker une formule dans le champ d'une table ?

La solution qui se présente est de les stocker sous forme texte (string) dans un champ. Puis de les compiler une fois on a besoin de les exécuter. Pour ce faire, il va falloir les écrire dans un langage de programmation dont on a l'accès au compilateur (Lexer, Parser et compilateur) Car le créer du zéro nous-même demandera beaucoup de temps précieux qui devrait être investi dans la conception et développement de notre application. Une autre question serait donc encore une fois si ces formules doivent être exécutées niveau backend ou frontend ?

Face à ça, On s'est arrêté sur le langage frontend Javascript. Javascript propose une fonction « eval() » qui prend n'importe quel code JS sous forme de chaîne de caractères et l'exécute.

Il est clair que cette approche présente tout de même un sérieux problème de sécurité car n'importe qui peut ajouter un extrait de code (texte) dans la base de données et ce dernier sera exécuté dans l'application desktop.

Pour parer ce problème, avant de l'exécuter, une série de tests aura lieu pour s'assurer de sa validité. (Ces tests seront aussi faits avant de l'ajouter (POST) à la base de données) :

On divise d'abord le texte en tokens (séparant par virgule, parenthèses et accolades) puis si on trouve un token qui n'existe pas dans la liste suivante, on affiche une erreur :

- If, else
- Les nombres (1,2 ect..) et opérations (+,-,\* .. ect)
- Les variables (comme nombre\_jour\_annee=365)
- Les données de l'employés (salaire par exemple) et ses rubriques (primes et indemnités)
- Des fonctions qu'on donne à disposition du client (comme round() ou today() ) pour qu'il n'utilise pas les fonctions javascript

Si une erreur de syntaxe existe, on demande de modifier la formule bien sûr.

Voici un exemple concret de comment une formule sera stocké dans le champ de la table Formule:

Tableau 3:Formules dans la base de données

Formule	Définition	Code JS dans la BDD
<b>Taux d'ancienneté</b>	SI (Ancienneté >=25 ;25 ; SI (Ancienneté >=20 ;20 ; SI(Ancienneté >=12;15;SI( Ancienneté >=5;10;SI( Ancienneté >=2;5;0))))))	<pre> if(anciennete&gt;=25) {   25 } else if(anciennete&gt;=20) {   20 } else if(anciennete&gt;=12) {   15 } else if(anciennete&gt;=5) {   10 } else if(anciennete&gt;=2) {   5 } else{   0 } </pre>
<b>Ancienneté1</b>	(aujourd'hui – date entrée)/365	(today()– date_entree)/ nombre_jour_annee

### **III. METHODOLOGIE DE TRAVAIL**

Pour garantir la satisfaction du client et l'accomplissement de ses exigences avec un cout et délais minimum, nous avons opté pour une méthode agile tout au long du projet.

#### **1. Méthodologie Agile**

Une méthode agile est une approche itérative et incrémentale, qui, au lieu de prévoir la planification totale du projet avant la phase de développement, On fixe des objectifs à court terme qu'on traite séparément.

On citera comme méthodes de développement Agile :

- L'extrême Programming (XP)
- Scrum
- Feature Driven Development (FDD)
- Lean Software Development
- Agile Unified Process (Agile UP ou AUP)
- Crystal
- Dynamic Systems Development Method (DSDM)

#### **2. Pourquoi Scrum ?**

Dans le cadre de notre projet, le processus Scrum s'adapte parfaitement à la décomposition du notre projet de fin d'étude.

Voici quelques raisons qui justifie ce choix :

- Le besoin du client n'étant pas très clair et définit au début, une méthode en cascade (whaterfall) classique n'aurait pas servit
- Plus de souplesse et de réactivité

- Un gain de temps et une meilleure réactivité grâce aux réunions fréquentes avec le client
- S'assurer de la satisfaction du client car il est intégré tous au long du processus de développement
- Facile à utiliser

### 3. Scrum

La méthode Scrum est une méthode agile de gestion de projets informatiques qui est conforme aux principes des méthodes agiles. Son but est d'arriver rapidement à une première itération de livrable (produit ou service) utilisable et fonctionnel En le validant au fur et à mesure par le client a la fin de chaque sprint.

Les sprints constitut un mode itératif et incrémental pour pouvoir proposer, à chaque étape, un livrable et de s'adapter graduellement jusqu'à arriver au besoin réel et final du client

Donc Scrum est une méthodologie simple pour assurer une bonne collaboration de l'équipe sur des projets complexes.

Mais ce projet étant un projet individuel, Peut-on appliquer cette méthodologie ?

### 4. Scrum individuel

Une autre raison de choisir Scrum, serait qu'il peut-être appliquer pour des projets individuels sans équipe.

Une Equipe Scrum est normalement constituer d'un Product Owner, Scrum Master et d'une équipe de développement. Pour le bon Déroulement Du projet, Tout ces rôles doivent être assurer par une seule personne,

Voici comment ceci s'est manifesté :

- **Product owner** : en tant que product owner, Le produit final doit être bien compris et ses exigences et caractéristiques bien définies. Il est aussi primordial de garder une vue globale sur le produit sans se perdre dans le développement.
- **Scrum master** : Le rôle de Scrum master consiste en identifiant les problèmes puis à chercher à les résoudre. Il faut donc pouvoir identifier les obstacles qui se présentent et s'assurer que le travail finisse durant le time box prédéfinie.
- **Equipe de développement** : Faire le développement demandé tout en restant motivé et organisé

En ce qui concerne les événements de Scrum, voici comment ils ont été intégrés :

- **Sprint Planning** : Avant chaque sprint on choisit quel Product Backlog Items va être inclus dans le sprint
- **Daily Scrum** : Pas appliqué
- **Sprint Review** : A la fin de chaque Sprint, une réunion avec le client est organisée pour montrer le livrable et voir s'il y'a des changements à faire dans le prochain sprint
- **Sprint Rétrospective** : Pas appliqué

## IV. PLANIFICATION DU PROJET

### 1. Diagramme de Gantt

Pour s'assurer de la bonne réalisation du travail demandé dans les délais établis par la convention de stage, il est primordial de définir toutes les étapes essentielles et d'estimer le temps à consacrer pour chacune.

Pour faire cela, nous allons utiliser Le diagramme de Gantt qui est l'un des outils les plus efficaces pour représenter visuellement l'état d'avancement des différentes activités (tâches) qui constituent un projet.

En résumé, un diagramme de Gantt serre à regrouper toutes les tâches à accomplir pour que le projet finit dans le temps imparti et indique la date à laquelle ces tâches doivent être effectuées (le planning).

**teamgantt**  
Created with Free Edition

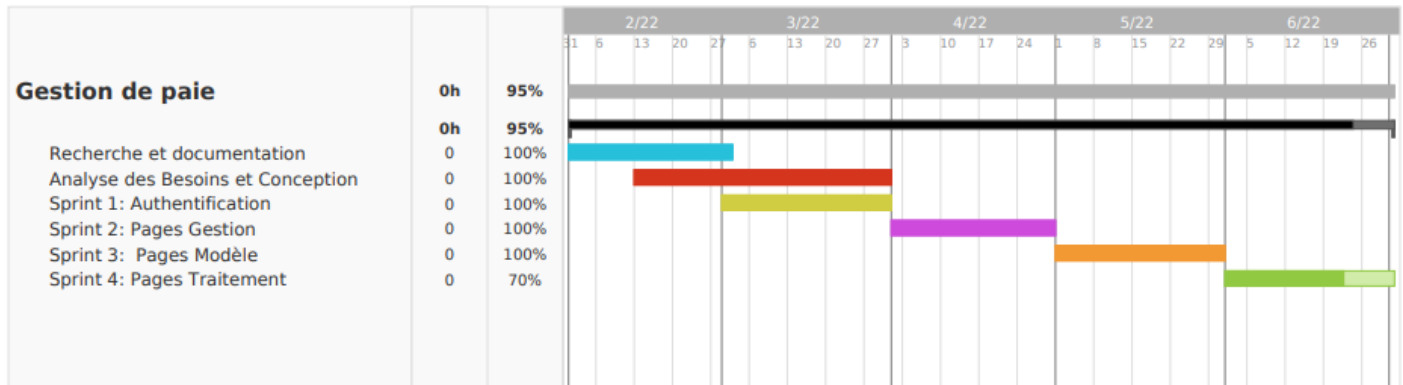


Figure 9: Diagramme de Gantt du planning du projet

## 2. Résumé des taches accomplies

Cette section a pour objectif de détailler les opérations effectuées durant les tâches du planning réel énoncées dans la section précédente.

- **Recherche et documentation**

Cette période a été en premier lieu consacré aux choix des technologies qui seront utilisés tout aux longs du projet (Angular, Typescript, Django). Ensuite Pour les apprendre en s'appuyant sur les documentations et des cours en ligne.

Le but n'étant pas de les maitriser mais plutôt d'avoir un niveau acceptable qui nous permettra d'entamer le développement ou la plupart de l'apprentissage aura lieu.

- **Analyse des besoins et Conception**

Durant cette période, il est question de comprendre le besoin en programmant des réunions avec l'équipe RH tout en faisant des recherches sur le sujet (pour mieux comprendre leurs jargon). En plus de concevoir une première version de la base de données.

- **Sprint 1**



Le premier Sprint consiste à s'occuper de l'authentification en créant une page login qui communique avec un backend api.

- **Sprint 2**

Ce Sprint est où les tables essentielles ont été créées dans la base de données. Puis la Page Gestion où on peut voir tous les enregistrements des tables a été créée avec des filtres.

- **Sprint 3**

Le troisième Sprint est celui où les pages Modèle où on peut Ajouter, Supprimer et Modifier des enregistrements dans les tables ont été développés.

- **Sprint 4**

Durant ce sprint, on a développé la page Traitement où on calcule les formules

La rédaction du rapport est répartie tout au long des sprints.

Tout au long de ce projet, on a travaillé sur le système d'information l'entreprise Odoo en répondant aux requêtes internes et externes des clients. Et en faisant avec l'équipe SI la gestion de plusieurs projets importants.

## Conclusion :

Ce Chapitre a mis l'accent sur le cadre général du projet. Nous avons commencé par une description générale du groupe « EDITO » et plus particulièrement de la société « Sapress ». Puis une présentation générale du projet avec une brève présentation, le but à atteindre et le travail demandé en finissant avec l'environnement de développement et la planification temporelle.

## **CHAPITRE 2 : « ANALYSE DE L'APPLICATION ET SPECIFICATION DES BESOINS »**

## Introduction :

Ce chapitre mettra le point sur toutes les fonctionnalités importantes de cette application et les besoins fonctionnels et non fonctionnels qui les alimentés.

## I. SPECIFICATIONS DES BESOINS

### 1. Les besoins fonctionnels

L'application permet de :

- S'authentifier
- Gérer les Employés : Ajouter, modifier, supprimer, rechercher, afficher et imprimer la liste.
- Gérer les Fonctions : Ajouter, modifier, supprimer, rechercher, afficher et imprimer la liste.
- Gérer les centres Couts : Ajouter, modifier, supprimer, rechercher, afficher et imprimer la liste.
- Gérer les Directions : Ajouter, modifier, supprimer, rechercher, afficher et imprimer la liste.
- Gérer les Villes : Ajouter, modifier, supprimer, rechercher, afficher et imprimer la liste.
- Gérer les Contrats : Ajouter, modifier, supprimer, rechercher, afficher et imprimer la liste.
- Gérer les Affectations : Ajouter, modifier, supprimer, rechercher, afficher et imprimer la liste.
- Gérer les Formules : Ajouter, modifier, supprimer, rechercher, afficher et imprimer la liste.
- Gérer les Variables : Ajouter, modifier, supprimer, rechercher, afficher et imprimer la liste.
- Gérer les Rubriques : Ajouter, modifier, supprimer, rechercher, afficher et imprimer la liste.
- Faire des traitements

Le but étant de Créer un tableau de traitement ou pour chaque employé, on calcule les formules basées sur les variables et les informations sur l'employé.

## 2. Les besoins non fonctionnels

L'application doit être :

- Convivial
- Simple, facile à utiliser et intuitif
- Rapide
- Sécurisé
- Performance : Un minimum de délai d'attente.

## II. Analyse des cas d'utilisations

### 1. Présentation des acteurs :

Un acteur est la personne ou le matériel qui interagit avec notre système afin de réaliser une valeur ajoutée. Notre Plateforme ne fait intervenir qu'une seule personne de l'équipe Rh.

Donc nous disposons d'un seul Acteur qu'on appellera utilisateur.

### 2. Diagramme de cas d'utilisation

La figure ci-dessous représente le diagramme de cas d'utilisation générale du projet :

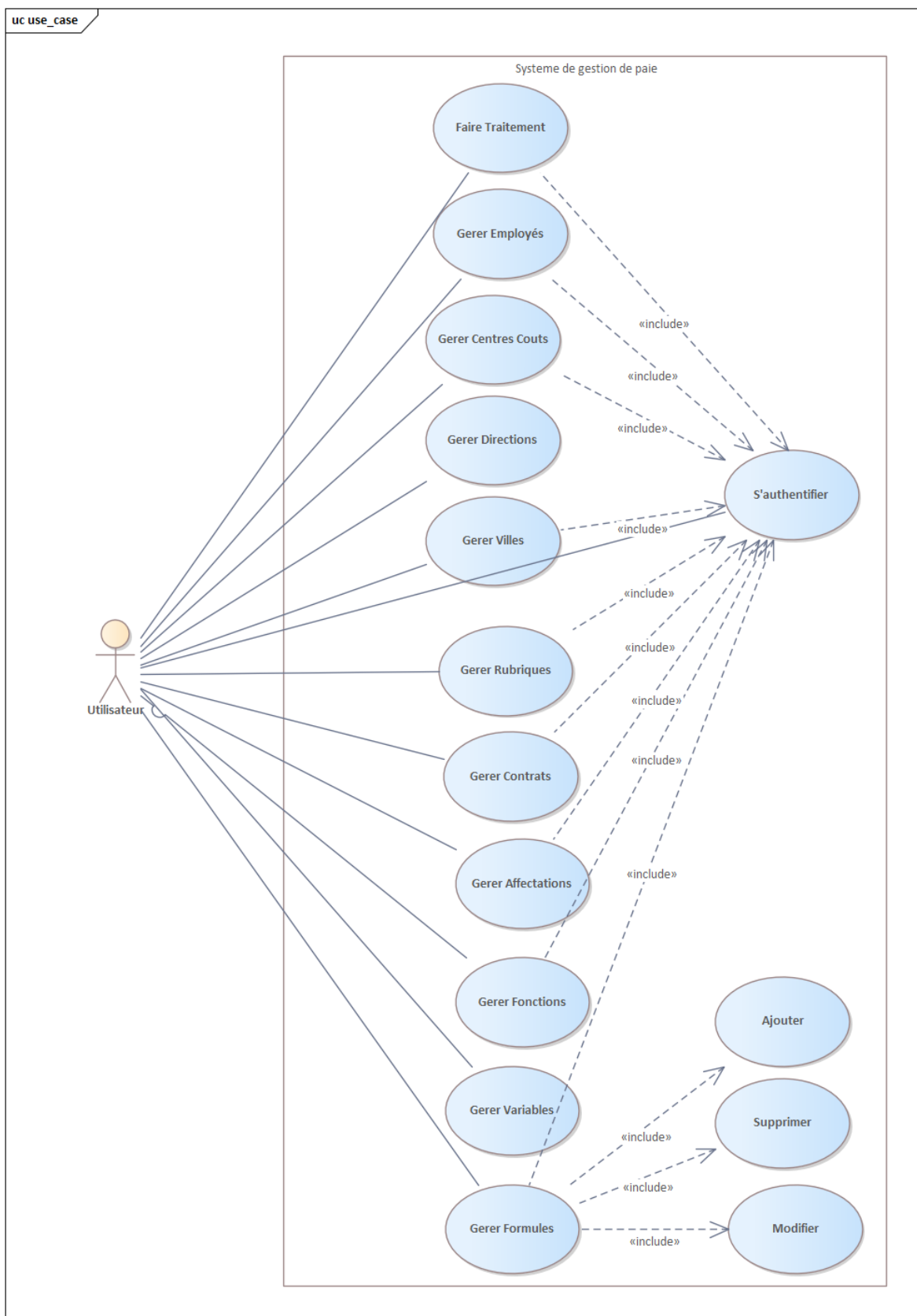


Figure 10: Diagramme de cas d'utilisation générale

Pour résumer ce diagramme de cas d'utilisation. Notre utilisateur doit pouvoir gérer toutes les tables de la base de données et faire des traitements.

### III. Le Backlog du produit

Le product backlog est défini par le guide scrum comme étant “une liste ordonnée et émergente de ce qui est nécessaire pour améliorer le produit”. Xavier Koma dit que c’est une liste ordonnée de toutes les fonctionnalités/items, peu importe la forme ou le format de l’item, qui peut être des user story, des spécifications, un brief, une maquette ou un powerpoint, tant que l’équipe se met d’accord sur le format, de façon à conserver la même communication et la même compréhension.

En tant que Product Owner, cette responsabilité doit être assurée :

Tableau 4: Product Backlog

User Story	Points du Story	Priorité
<b>Je peux me connecter en insérant mon email et mot de passe</b>	2	1
<b>Je peux voir la liste des enregistrements de toutes les tables et les filtrer</b>	3	2
<b>Je peux Consulter un enregistrement en particulier. Le supprimer, Modifier et Ajouter.</b>	3	3
<b>Je peux faire des traitements</b>	2	4

### Conclusion :

Ce chapitre a traité les besoins fonctionnels et non fonctionnels de notre application et a présenté ses cas d'utilisation.

## CHAPITRE 3 : « Conception »

## Introduction :

La conception d'un système informatique est une étape très importante qui va Influencer la qualité et la fiabilité de toute application. Ce chapitre détaillera les différents éléments de la conception (générale et détaillée).

### I. Conception Générale

Le but de cette partie est d'aborder l'architecture technique utilisé et de montrer les choix des technologies les plus adaptés aux besoins et aux contraintes de l'organisation d'accueil.

#### 3. Architecture trois tiers

##### a. Définition

L'architecture à trois tier ou architecture à trois niveaux est une architecture d'application logicielle bien établie qui divise les applications en trois couches logiciel logiques et physiques

- **Niveau Présentation** : correspondant à l'affichage, Son objectif principal est d'afficher des informations et de collecter des informations auprès de l'utilisateur.
- **Niveau Application** : Ce niveau est le cœur de l'application ou toutes les informations collectées dans le niveau de présentation sont traitées
- **Niveau Données** : aussi appelé la couche base de données, elle correspond à l'endroit où les informations traitées par l'application sont stockées et gérées.



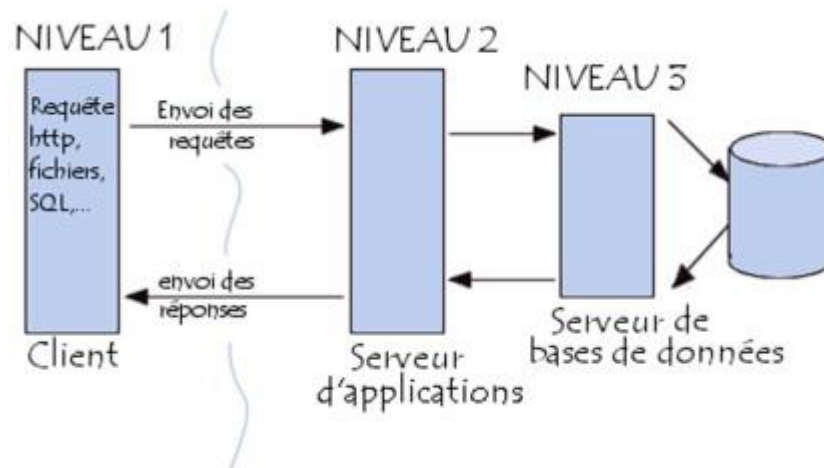


Figure 11: Architecture trois tiers

Dans notre cas, La couche présentation correspond à la partie frontend de notre application, la couche Application est notre api et la couche données est notre base de données

En d'autres termes, Votre application (dans le serveur Applications) ne communique pas directement avec la base de données mais communique plutôt via le Protocol http avec une Rest api déployée dans un serveur web.

### b. Justification du choix

La communication directe entre l'application et la base de données est appelé architecture 2 tier ou client-serveur et elle présente plusieurs inconvénients :

- **Un coût élevé** : Le coût impliqué dans la configuration et la maintenance du serveur est généralement élevé dans le réseau client-serveur
- **Un maillon faible (single point of Failure)** : le serveur est le seul maillon faible du réseau client/serveur, Si par malheur il tombe en panne, le client ne peut plus accéder aux données car tout le réseau est architecturé autour de lui.
- **Congestion du trafic** : si beaucoup de clients lancent des requêtes à au même serveur, ceci peut entrainer un problème d'accès aux informations. Ainsi que des plantages ou un ralentissement de la connexion.

## II. Conception détaillé

Dans cette partie nous présentons le diagramme de classes ainsi de séquences constituant le système et les associations entre elles afin de mieux structurer les différentes classes prise en compte dans notre application.

### 1. Langage UML

#### a. Présentation du langage UML

UML (en anglais Unified Modeling Language, « langage de modélisation unifié ») est un langage graphique de modélisation objet unifié est une démarche orientée objet .il a été créé pour forger un langage de modélisation visuel normalisée, sémantiquement et syntaxiquement riche pour l'architecture, la conception et la mise en œuvre de systèmes logiciels complexes.

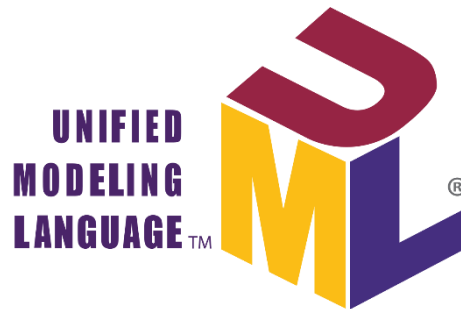


Figure 12: Logo UML

#### b. Raisons de la modélisation informatique

La modélisation d'un système d'information permet d'obtenir des représentations consolidées sous différents domaines : processus métiers, capacités fonctionnelles, couches applicatives et échanges de données, infrastructures et technologies... Chaque domaine étant interconnecté avec les autres.

Un modèle est donc comparable aux cartes géographiques, ils peuvent être déclinés à différentes échelles qui permettront de zoomer sur une partie du domaine traité pour en connaître les détails les plus subtils. Ce qui nous permet avant tout d'avoir une vue macro sur notre système d'information.

### *c. Avantages de l'UML*

Le langage Uml présente plusieurs avantages. Parmi eux, on cite :

- Anticipation de problèmes qui peuvent se manifester avant même qu'ils n'arrivent
- Langage formel et normalisé
- Support de communication performant
- L'analyse devient plus facile
- Facilite la compréhension de représentations abstraites complexes

Un diagramme UML est une représentation graphique, et à chaque vue correspondent des diagrammes qui sont répartis selon leurs aspects statiques ou dynamiques :

Modéliser les aspects statiques

- Diagramme de cas d'utilisation (déjà présenté)
- Diagramme d'objets
- Diagramme de classes
- Diagramme de composants
- Diagramme de déploiement

Modéliser les aspects dynamiques

- Diagramme de collaboration
- Diagramme de séquences
- Diagramme d'états-transitions
- Diagramme d'activités

## 2. Diagrammes de classes

### a. Définition

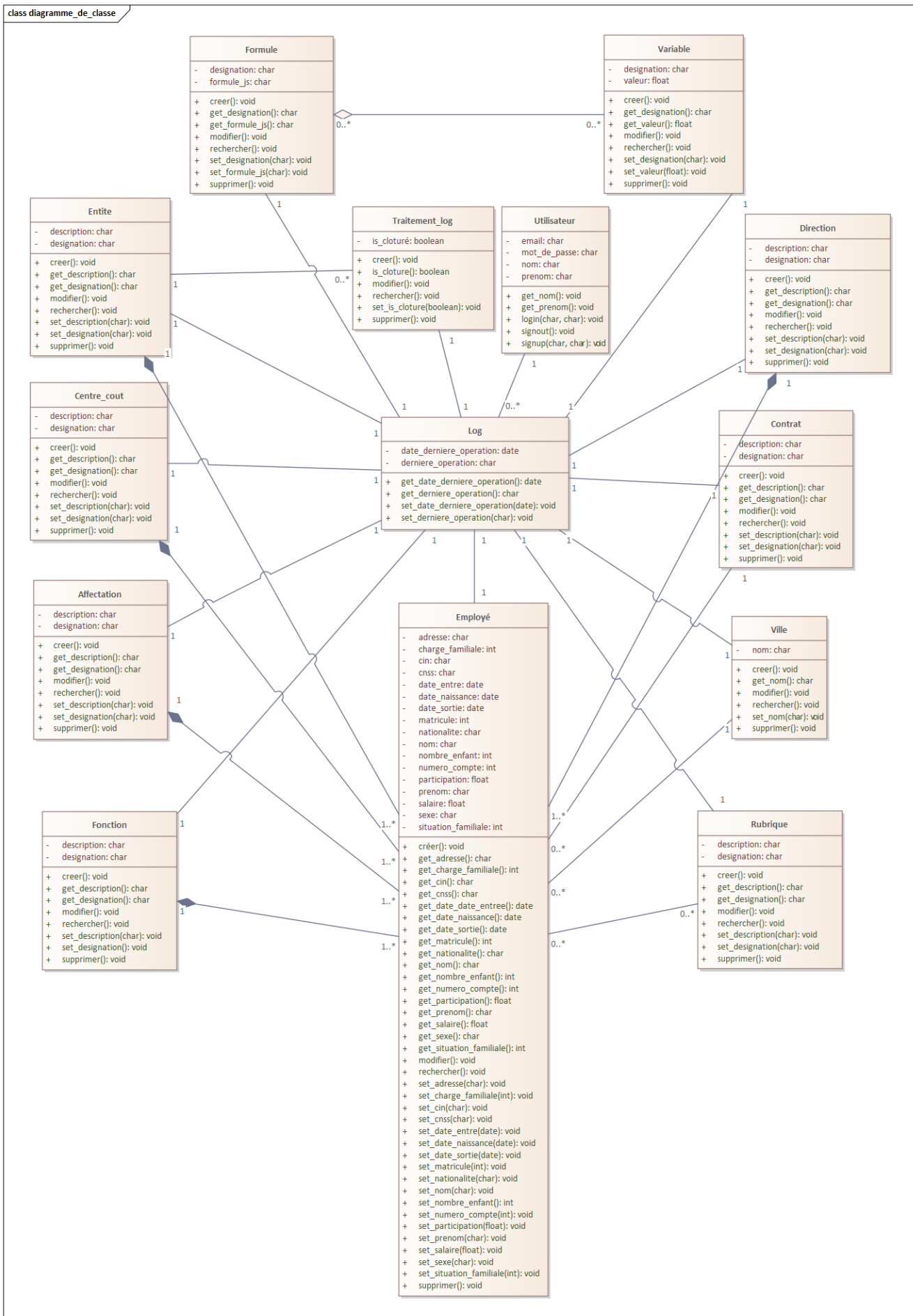
Le diagramme de classe représente les entités utilisées dans l'application, et plus précisément leurs attributs ainsi que les méthodes qu'ils utilisent pour traiter les données. Ils expriment aussi de manière générale la structure statique d'un système, en termes de classes et de relations entre elles.

### b. Diagramme

Nous disposons dans notre application de deux bases de données, une principales ou toutes les tables importantes de notre application existent Et une autre des utilisateurs. Le but de ce séparation est de rester flexible et garder une certaine marge de scalabilité si le nombre d'utilisateur augmente drastiquement.

La nomination des classes est conforme au termes et jargons RH.

Dans la figure suivante nous présentons les classes associées à notre projet :



### *c. Description des classes*

Ici on fera une description de chaque classe

- **Entité :**

Une Entité est une entreprise. Donc comme déjà expliqué, le groupe Edito est un regroupement de Troie Entités : Sapress, SochePress et Warak trading.

- **Employé :**

L'employé est bien sur tout salarié de l'entreprise.

- **Affectation :**

On appelle affectation l'agence ou l'employé travaille actuellement. Par exemple : Agence Marrakech, Agence Tétouan etc.

- **Centre Cout :**

Au sein d'une même Affectation, Le fonctionnement de l'entreprise s'articule autour différents centres Cout, ou processus. Parmi eux, on peut citer : Logistique, Support, commercial

- **Fonction :**

C'est une classe qui contient les fonctions principales que les employés effectuent au sein de l'entreprise. C'est-à-dire son titre officiel au sein de l'entreprise.

Exemple : Responsable de la caisse

- **Direction :**

Un employé, en plus d'appartenir à un centre cout, appartient réellement à une direction spécifique comme : La Direction logistique et développement ou encore la direction réseau et proximité

- **Contrat :**

Ceci correspond aux différents types de contrat de travail qui existent et varie selon la

durée Les plus fréquents sont le contrat de travail à durée indéterminée (CDI) et le contrat à durée déterminée (CDD)

- **Ville :**

Dans cette table, on stocke les villes où réside les salariés

- **Rubrique :**

Les rubriques incluent toutes les primes, indemnité et avantages que l'employé possède. Exemple : Prime de Transport, Avantage nature carburant

- **Log :**

Une table où on garde une trace de l'utilisateur qui a dernièrement modifier l'enregistrement, ainsi que la nature et la date de cette modification

- **Traitement log :**

Cette classe contient tous les traitements qui ont été fait et s'ils ont été clôturé ou pas

- **Formule :**

Les formules qu'on utilisera pour créer notre tableau. En guise d'exemple voici une des formules les plus simples :

Ancienneté = (date d'aujourd'hui – date d'entrée) / 365

- **Variable :**

Les variables utilisées par toutes ces formules et qui peuvent être modifier par notre utilisateur.

Dans la formule de l'ancienneté, on peut considérer 365 comme une variable.

- **Utilisateur :**

Cette classe contient les informations de tous les utilisateurs qui peuvent accéder à notre application

## d. Règles Et Commentaires

- Plusieurs Employés peuvent appartenir à la même entité, centre cout, affectation, direction et ville
- Plusieurs Employés peuvent avoir la même fonction et le même contrat de travail
- Un Employé doit avoir une combinaison matricule/entité unique. Car le matricule est l'identifiant unique de l'employé dans l'entreprise. Mais il se peut que 2 employés dans différentes Entités ait le même matricule.
- Une formule peut avoir une ou plusieurs variables
- Pour chaque Entité, on peut faire un traitement des employés qui y existent
- Chaque Employé peut avoir une ou plusieurs rubriques (primes ou indemnités).
- Plusieurs Employés peuvent avoir la même prime par exemple mais avec un montant qui peut changer. Dans la base de données, ceci se traduit en ajoutant une table « Employée\_rubrique » qui enregistre les montants des rubriques.

Tableau 5: Table Employé Rubrique

Employé_rubrique
FK : employé_id FK : rubrique_id montant

Un exemple d'enregistrement dans cette table serait qu'un employé X a une Indemnité de transport de 200dh.

- Chaque objet de toutes les classes contient un objet de la classe Log pour garder son historique Si on traduit ça en tables dans la base de données, chaque table doit avoir une clé étrangère vers un enregistrement dans la table Log.

Exemple pour la base Affectation :



Tableau 6: Table Affectation

Affectation
PK : id designation description FK : log_id

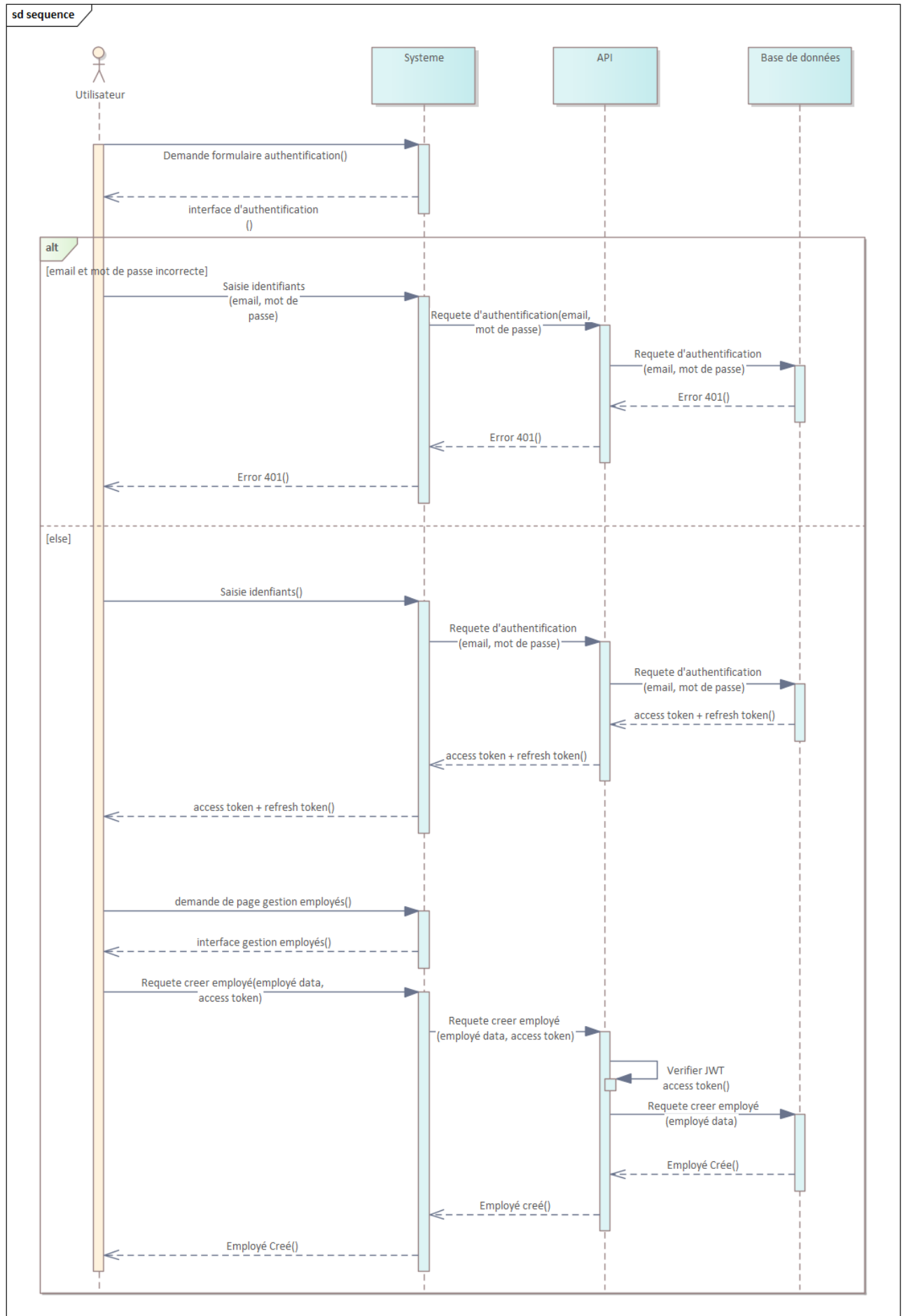
### 3. Diagrammes de séquences

#### a. Définition

Un diagramme de séquence est un diagramme UML qui représente des interactions entre les acteurs et le système. Il décrit les scénarios d'un cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets.

#### b. Diagramme

Dans la figure suivante nous présentons le diagramme de séquence de l'authentification :



*Figure 14: Diagramme de séquence authentification*

Ce diagramme cela expliqué dans la partie de l'authentification.

## Conclusion :

Avant de commencer la partie de réalisation, il est primordial de conceptualiser notre système. Cette conception ensuite se perfectionne et change au fur et à mesure qu'on avance dans les sprints.

## CHAPITRE 4 : « Réalisation »

## Introduction :

Nous arrivons maintenant à la phase ultime. Après avoir détaillé la conception de notre application, on passe maintenant à l'étape de la réalisation. On commencera par une présentation de l'environnement matériel et logiciel de développement. Puis On passera a la réalisation du frontend et backend.

## I. Environnement de travail

### 1. Environnement Matériel

Pour la réalisation de ce projet nous avons utilisé le matériel suivant :

<b>PC portable</b>	<b>Lenovo</b>
<b>RAM</b>	8 Go
<b>Microprocesseur</b>	Intel(R) Core (TM) i5-9300H CPU @ 2.40GHz 2.40 GHz
<b>Carte graphique</b>	NVIDIA GeForce GTX 1050
<b>Disque dur</b>	1,81To
<b>Système d'exploitation</b>	Windows 10

Tableau 7:Environnement Matériel

### 2. Environnement Logiciel

- **Visual code studio :**

Visual Studio Code (connu sous le nom de VS Code) est un éditeur de code gratuit et open-source développé par Microsoft supportant un très grand nombre de langages grâce à des extensions. Il est disponible pour Windows, Linux et MacOS et supporte l'autocomplétions, la coloration syntaxique, le débogage, et les commandes git. Ainsi que beaucoup d'autre fonctionnalités puissantes qui ont fait de VS Code l'un des outils

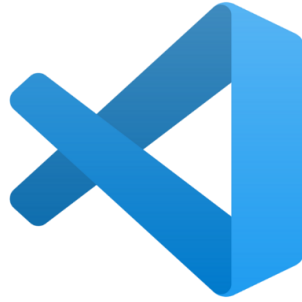


Figure 15: Vs Code logo

d'environnement de développement les plus populaires ces derniers temps.

- **Postman :**

Postman est un excellent outil lorsque vous essayez de disséquer des API RESTful créées par d'autres ou de tester celles que vous avez créées vous-même. C'est une application utilisée pour les tests d'API. Il s'agit d'un client HTTP qui teste les requêtes HTTP, en utilisant une interface utilisateur graphique, à travers laquelle nous obtenons différents types de réponses qui doivent ensuite être validées.



Figure 16: Logo  
Postman

- **GitHub :**

GitHub est un service d'hébergement Web open source pour les projets de développement de logiciels qui utilisent le système de contrôle de version Git. Qui a été créé par Linus Torvalds dans le but d'accélérer le développement logiciel. Livrée en tant que logiciel à la demande (SaaS, Software as a Service), Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités, la gestion de tâches et un wiki pour chaque projet.



*Figure  
17: Logo Github*

### 3. Langages de Programmation

- **Typescript :**

C'est un langage de programmation libre et open source développée par Microsoft en 2012. Pour être plus précis, c'est un sur ensemble de Javascript qui a pour but d'améliorer et de sécuriser la production de code JavaScript. Son ambition principale est d'améliorer la productivité de développement d'applications complexes. Le langage introduit des fonctionnalités optionnelles comme le typage ou la programmation orientée objet. Et ceci, sans l'utilisation d'aucune librairie. Il suffit d'utiliser l'outil de compilation de TypeScript pour le transcompiler en Javascript.



*Figure 18: Logo  
Typescript*

- **Python :**

Le langage Python est un langage de programmation open source multi-plateformes et orienté objet puissant et facile à apprendre. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet. Parce que sa syntaxe est élégante, que son typage est dynamique et qu'il est interprété, Python est un langage idéal pour l'écriture de scripts et le développement rapide d'applications dans de nombreux domaines et sur la plupart des plateformes.



*Figure 19: Logo Python*

- **Html 5 :**

HTML est le langage de balisage conçu pour représenter les pages web. Html 5 n'est rien d'autre qu'une révision de ce dernier développé pour résoudre les problèmes de compatibilité qui se posait



- **SASS :**

Sass est un langage de script préprocesseur qui est compilé ou interprété en CSS (Cascading Style Sheets). . Il vous permet d'utiliser des variables, des règles imbriquées, des mixins, des fonctions, etc., le tout avec une syntaxe entièrement compatible CSS. Sass propose deux syntaxes et donc deux extensions, .SASS et .SCSS. Dans Ce projet on utilisera SCSS.



*Figure 20: Sass Logo*



#### 4. Frameworks utilisés

- **Angular 13 :**

Angular est une plate-forme et un Framework écrit en TypeScript permettant de créer des applications Web et plus particulièrement de ce qu'on appelle des « Single Page Applications » à l'aide de HTML et TypeScript . Il implémente les fonctionnalités de base et facultatives sous la forme d'un ensemble de bibliothèques TypeScript que vous importez dans vos applications



- **Django :**

Django est un Framework Python de haut niveau, gratuit et open source, permettant un développement rapide de sites internet, sécurisés, et maintenables. Il est donc clairement orienté pour les développeurs ayant comme besoin de produire un projet solide rapidement et sans surprise. C'est pourquoi, le projet a pour slogan « Le Framework pour les perfectionnistes avec des deadlines ». En Effet, Django Te permet de ne plus réinventer la roue mais plutôt de vous concentrer sur l'écriture de votre application alors qu'il prend en charge la plupart des tracas du développement web. Le tout couronné par une bonne documentation et une communauté active.



*Figure 21: Logo Django*

- **Electron :**

Electron.js est un Framework qui permet à l'utilisateur de créer des applications desktop avec les technologies web comme HTML5, CSS et JavaScript. Electron s'occupe de la partie technique pour que vous puissiez vous concentrer sur le cœur de votre application. C'est un projet open source lancé par Cheng Zhao, ingénieur chez GitHub.



- **MySQL :**

MySQL est un système de gestion de base de données relationnelle SQL open source développé et pris en charge par Oracle. Il améliore la rapidité et la souplesse de l'ensemble en séparant les données dans des tables reliées par des relations définies. Le SQL dans "MySQL" signifie "Structured Query Language" : le langage standard pour les traitements de bases de données.

- **Material UI :**

Angular Material est une bibliothèque de composants d'interface utilisateur. Il est spécialement conçu pour les développeurs AngularJS. Cela aide à concevoir l'application de manière structurée. Les composants Angular Material aident à créer des pages Web et des applications Web attrayantes, cohérentes et fonctionnelles tout en respectant les principes de conception Web modernes tels que la portabilité du navigateur, l'indépendance des appareils et la dégradation gracieuse.

## 5. Justification des choix

### a. *Electron :*

Il existe plusieurs technologies pour créer des applications desktops.

Le choix se dirige d'abord vers du développement natif comme :

- Cocoa software framework pour MacOS
- NET framework pour windows

Mais vu que notre utilisateur peut très bien utiliser différents systèmes d'exploitation, on a opté pour une approche cross plateformes. Là aussi on trouve plusieurs choix comme :

- Le Framework java Swing
- Java fx
- Electron.js

Mais le choix d'Electron nous offre plusieurs avantages comme :

- **Facile à apprendre :**

Vu qu'il utilise seulement JavaScript, HTML et CSS, tout développeurs avec des compétences en développement web peut commencer sans effort. Ceci résout aussi un problème de coût puisqu'en générale, un développeur d'applications desktop coûte plus cher.

- **Vitesse de développement accrue**

Le langage javascript est un avantage qu'on ne peut pas nier car contrairement aux autres langages de programmation bas niveau qu'on utilisait, on n'a pas besoin de tout développer de zéro (from scratch). Typescript rend le développement encore plus sûr et efficace.

Plusieurs applications desktop très connues utilisent Electron comme : Web Torrent desktop app, WordPress desktop app, WhatsApp Desktop app ou encore Discord.

## **b. Angular**

Maintenant qu'on a choisi d'utiliser Electron, développer une application seulement avec Javascript, HTML et CSS n'est pas pratique pour un projet important comme le nôtre. Il faut à présent choisir un Framework convenable. Et bien qu'il y ait plusieurs frameworks, le choix reste entre les deux plus utilisés React.js et Angular.

On s'est arrêté sur Angular car il nous offre une grande communauté active, l'efficacité incontestable de Typescript avec une expérience de développement plus propre et organisée, une séparation claire entre l'interface UI et la partie métier (fichiers html et fichiers ts) et une façon très simple de faire des changements, qui est l'Angular CLI. De plus, Angular est généralement utilisé pour des projets d'entreprise.

Material UI est une librairie de composants qui remplace Bootstrap et qui s'est

montré particulièrement efficace.

## c. Django

Devant la complexité de Angular, python se montre comme un bon choix pour la partie backend avec un code facile à lire et écrire. Django est un Framework crée par les développeurs pour les développeurs, ce qui veut dire que tous les problèmes communs sont déjà résolus avec plusieurs fonctionnalités. Le tout avec une sécurité indéniable.

## d. MySQL

Toujours dans l'optique d'une optimisation de l'outil, pour notre SGBD, il est important de savoir d'abord quel type de base de données est le plus adapté à notre projet entre SQL ou NoSQL. Une fois on choisit SQL car on a besoin d'un modèle relationnel clair, MySQL devient un choix raisonnable pour assurer une rapidité de traitement optimale. Chose qui peut être parfois compromise pour une plus grande scalabilité et intégrité des données comme dans le SGBD PostgreSQL.

De plus, on trouve ci-dessous un classement des 10 SGBD les plus utilisés actualisé chaque mois ou on peut clairement voir que MySQL prend la deuxième place avant oracle en termes de popularité

Tableau 8: Comparaison popularité des SGBDs 2022(source: <https://db-engines.com>)

Rank			DBMS	Database Model	Score		
Jun 2022	May 2022	Jun 2021			Jun 2022	May 2022	Jun 2021
1.	1.	1.	Oracle +	Relational, Multi-model i	1287.74	+24.92	+16.80
2.	2.	2.	MySQL +	Relational, Multi-model i	1189.21	-12.89	-38.65
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model i	933.83	-7.37	-57.25
4.	4.	4.	PostgreSQL +	Relational, Multi-model i	620.84	+5.55	+52.32
5.	5.	5.	MongoDB +	Document, Multi-model i	480.73	+2.49	-7.49
6.	6.	↑ 7.	Redis +	Key-value, Multi-model i	175.31	-3.71	+10.06
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model i	159.19	-1.14	-7.85
8.	8.	8.	Elasticsearch	Search engine, Multi-model i	156.00	-1.70	+1.29
9.	9.	↑ 10.	Microsoft Access	Relational	141.82	-1.62	+26.88
10.	10.	↓ 9.	SQLite +	Relational	135.44	+0.70	+4.90

## 6. Analyse des lignes de codes

Comme déjà discuté, les langages utilisés sont Typescript, Python, Javascript, HTML, SCSS.

Ils ont bien sûr été utilisés avec des pourcentages différents :

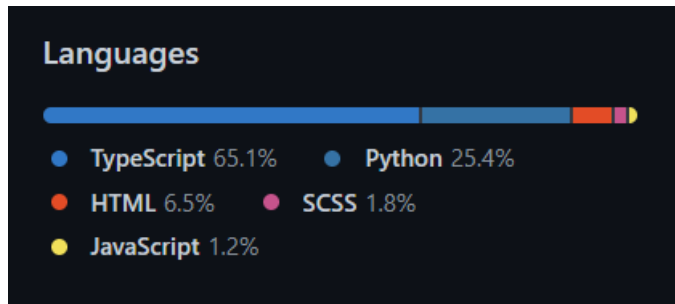


Figure 22: Pourcentage des langages de programmation

La partie frontend étant la plus consistante, TypeScript a un pourcentage de 65.1%

Le nombre de lignes varie aussi considérablement

Pour l'API Django :

Tableau 9: Nombre de lignes de codes Django

Languages					
language	files	code	comment	blank	total
Python	142	3,141	369	869	4,379
pip requirements	1	25	0	0	25
toml	1	8	0	4	12

On a un nombre de lignes totales de 4379 et sachant qu'une nouvelle application Django contient déjà 201 lignes. En conclusion, le nombre réel de lignes est 4178 dans un total de 142 fichiers.

Pour le frontend Angular :

Tableau 10: Nombre de lignes de code Angular

Languages					
language	files	code	comment	blank	total
TypeScript	214	10,104	155	1,733	11,992
JSON	5	9,797	2	5	9,804
HTML	43	714	36	124	874
SCSS	44	443	27	137	607
JavaScript	2	69	9	11	89
JSON with Comments	1	31	1	1	33
Markdown	1	14	0	14	28

Si on additionne tous les fichiers Typescript, html, scss et javascript on se retrouve avec 13.562 lignes de code. Et sachant qu'un nouveau projet Angular a 721 lignes. Le nombre de lignes totale est : 12.841 éparpillé dans 301 fichiers.

Ces diagrammes ont été réalisés à l'aide d'une extension de vs code appelé « VS Code Counter »

## II. Réalisation du Frontend

### 1. Description des interfaces graphiques

L'interface graphique est une partie très importante pour la réalisation d'une application convenable offrant un certain plaisir à l'utilisateur lors de sa navigation. Nous allons consacrer cette partie pour la présentation des principales interfaces de l'application :

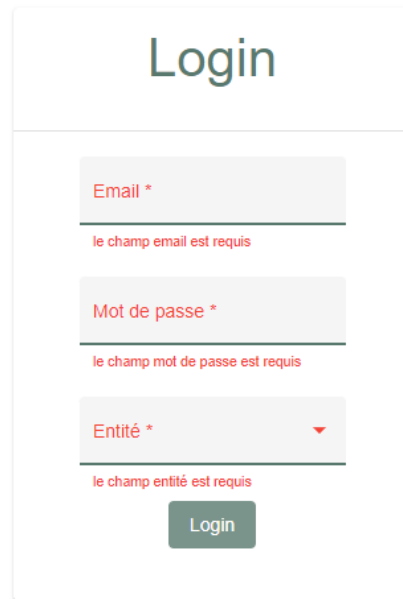
#### a. Interface authentication

L'interface authentification a une importance primordiale, c'est la phase d'identification pour accéder à l'application.

Elle s'affiche dès la connexion et l'utilisateur est amené à saisir son email, son mot de passe, et l'entité à laquelle il veut accéder (l'entité peut être modifier après) pour qu'il puisse accéder à l'application.

*Figure 23:Interface Authentification*

Une gestion des erreurs est bien sûr effectuée si les identifiants ne sont pas saisis. Si par exemple l'input n'est pas un email ou s'ils ne sont pas valides. Ou encore si l'email et le mot de passe n'existe pas.



The image shows a login form titled 'Login'. It contains three input fields, each with a red error message below it: 'Email \*' with 'le champ email est requis', 'Mot de passe \*' with 'le champ mot de passe est requis', and 'Entité \*' with 'le champ entité est requis'. A 'Login' button is at the bottom.

*Figure 24: Interface authentification Erreurs*

Nous ne disposons pas de page signUp car personne ne devrait pouvoir créer son propre compte. Pour l'instant quelques comptes ont été créés dans la base de données mais dans des versions futures un administrateur aura le droit de le faire dans l'application elle-même dans une interface d'administration.

Voici donc les deux états du header de notre application

- Si l'utilisateur n'est pas connecté



*Figure 25: Header état non connecté*

- Si l'utilisateur est connecté



*Figure 26: Header état connecté*

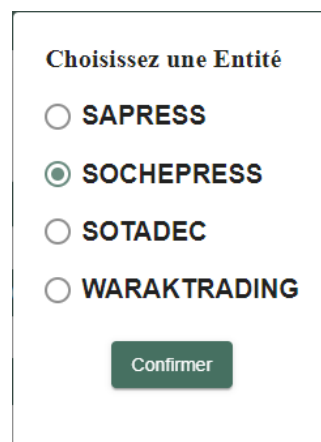
Si vous appuyez sur votre email, vous pouvez soit vous déconnecter soit changer d'entité.





*Figure 27: Header Utilisateurs options*

Si vous appuyer sur changer d'entité, le dialogue suivant s'affiche :



*Figure 28: Interface changer d'entité*

## b. Interface Menu Gestion

Cette page est un menu ou vous pouvez choisir quelle table vous souhaitez gérer.

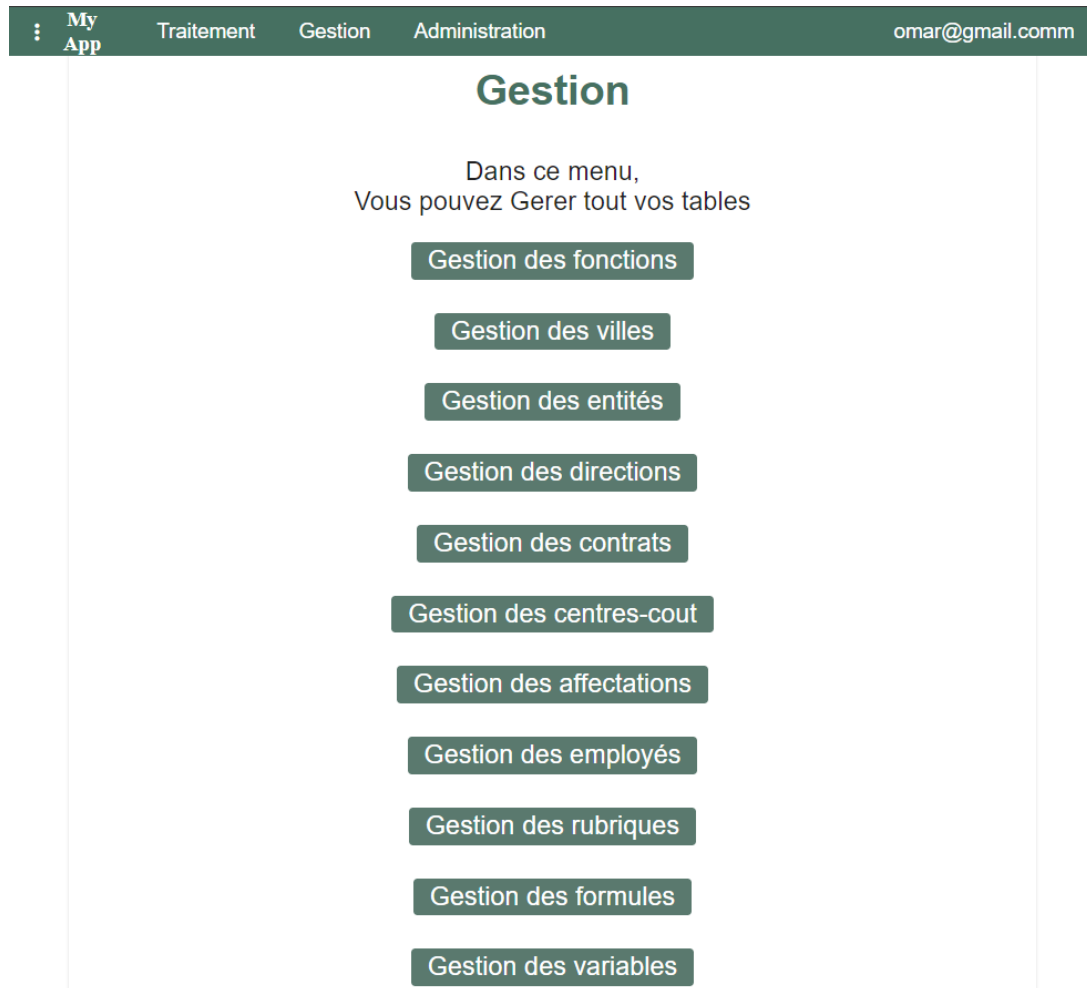


Figure 29: Interface Menu Gestion

Si vous n'êtes pas dans cette page et vous voulez changer de page de gestion sans passer par cette interface. Le menu apparait dans le header en appuyant sur



Figure 30:

Header Menu Gestion

## c. Interface Gestion Employés

Une fois que vous appuyez sur l'un des boutons de gestion dans le menu, vous êtes automatiquement redirigée vers la page gestion de la table correspondante. On considère par la suite que vous avez appuyé sur le bouton « Gestion Employés » car les mêmes interfaces apparaîtront pour les autres tables. Voici l'interface de gestion employés :

ID	Matricule	Nom	Prenom	Date naissance	Sexe	Cin	date entrée	Situation familiale	Charge familiale	Adresse	Nationalite	Cnss	Salaire	Numero compte	Participation	Date sortie	fonction
1	484	Hafid	Mohamed	1994-01-02	M	BK179415	2017-03-01	Marié(e)	2	RES El mehdi IMM B N°13 Sidi Maarouf	MAR	8000	1907802	56665	4346	2022-03-18	Responsable d...
2	4849	Salma	Hafidi	1990-08-02	F	BK179420	2016-02-09	Célibataire		RES El gadir IMM D N°18 Oasis	MAR	80098	10000	56687	2000	2022-08-09	Responsable d...
3	4848	Nainia	Youness	1995-07-01	M	BK179418	2018-04-05	Marié(e)	2	RES El safa IMM D N°13 Maarif	MAR	80090	9000	56657	43846	2048-04-05	Responsable d...

Figure 31: Interface Gestion Employé

Ici on ne trouve que 3 employées, mais on peut avoir plusieurs pages avec un maximum de 100 Employés par page.

Avec un nombre d'employés importants, il est primordial de pouvoir filtrer ces derniers. Et ceux en se basant sur n'importe quel champ de l'employé.

Pour se faire, On choisit le champ visé dans la première liste déroulante, puis le type du filtrage :

Egal, Contient, Différent, Supérieure Strictement, Inferieur, Supérieure, Compris entre

Puis on saisit la valeur soit manuellement soit dans une liste déroulante en fonction de si on le choix ou pas.

Les filtres peuvent être accumulé et sont affiché sous forme de « chips ». Voici un exemple qui retourne tous les employés avec un salaire supérieur à 9500 d'un sexe masculin.

salair >= "9500" ✕ sexe = "M" ✕

Filtre \* Sexe M Mode du Filtre Egal Confirmer

1 - 1 of 1 < >

ID	Matricule	Nom	Prenom	Date naissance	Sexe	Cin	date entrée	Situation familiale	Charge familiale	Adresse	Nationalite	Cnss	Salaire	Numero compte	Participation	Date sortie	fonction
1	484	Hafid	Mohamed	1994-01-02	M	BK179415	2017-03-01	Marié(e)	2	RES El mehdi IMM B N°13 Sidi Maarouf	MAR	8000	1907802	56665	4346	2022-03-18	Responsable de l

Figure 32: Interface Gestion Employés Filtres

## d. Interface Liste Employé

Une fois on a trouvé l'employé qu'on veut consulter et on appuie dessus, on se retrouve dans la page liste employé ou on peut voir toutes les informations relatives à ce dernier.

My App Traitement Gestion Administration omar@gmail.com

### Employé

Creer Modifier Delete

Id	1	Matricule	484
Nom	Hafid	Prenom	Mohamed
Date naissance	1994-01-02	Cin	BK179415
Date entree	2017-03-01	Nombre enfant	5
Charge familiale	2	Adresse	RES El mehdi IMM B N°13 Sidi Maarouf
Nationalite	MAR	Cnss	8000
Salaire	1907802	Numero compte	56665
Participation	4346	Date sortie	2022-03-18

omar omari 15/06/22 Modifier

Figure 33: Interface Liste Employé

Les propriétés normales comme Adresse, Nombre enfant etc.... sont regroupées en groupe de deux comme suit :


Salaire	1907802	Numero compte	56665
---------	---------	---------------	-------

Figure 34: Liste Employé attributs normaux

Tandis que ceux avec lesquels la table entretient une relation « One to Many » comme centre cout, fonction ou encore direction, ils sont affichés individuellement avec leurs informations respectives. Comme suit :

Centre cout	
Id	1
Description	Logistique
Désignation	Logistique

Figure 35: Liste Employé attribut one to many

En appuyant sur,  On est directement redirigé vers la page liste du centre cout « Logistique »

Pour la table Rubrique avec laquelle employé à une relation « Many to Many », une liste des rubriques est affichée :

Rubriques	
Id	1
Description	Prime de Transport
Désignation	Prime de Transport
Montant	500
Id	2
Description	Avantage nature carburant
Désignation	Avantage nature carburant
Montant	0

Figure 36: Liste Employé many to many

Il est important de préciser que dans toutes les tables, on peut retrouver une barre en bas (footer) qui montre le nom et prénom de l'utilisateur qui a fait la dernière modification sur

l'employé en question. On y retrouve aussi la date et la nature de cette modification.

omar omari	15/06/22	Modifier
------------	----------	----------

Figure 37: Footer

### e. Interface Modifier Employé

Dans l'interface liste employé à droite, on trouve Modifier. On appuie dessus et deux boutons Sauvegarder et Annuler s'affiche d'abord. Puis tous les champs normaux se transforment en input (date ou texte).

Figure 38: Interface modifier Employé

Une liste déroulante apparait pour les tables « One to Many » comme suit :

Figure 39: Modifier Employé attribut one to many

Pour Rubrique avec lequel employé entretient une relation « Many to Many », on peut par exemple ajouter, supprimer ou modifier une prime de l'employé en précisant le montant en DH.

Figure 40: Modifier Employé attribut many to many

Une fois finit, vous pouvez appuyer sur sauvegarder pour garder la modification ou annuler et vous serez rediriger vers l'interface Liste Employé

#### **f. L'interface traitement**

L'interface traitement est en cours de développement. La partie backend elle est déjà réalisé.

#### **g. L'interface administration**

L'interface administration ou l'administrateur peut ajouter de nouveaux utilisateurs est en cours de développement

Comme vous pouvez voir, la couleur omniprésente dans toutes les interfaces est le vert avec des intensités et des valeurs différentes. Ce choix est fait car la couleur verte est une couleur froide associé à la douceur, la tranquillité, la confiance et la contemplation. Rien qu'en faisait ce choix, l'utilisateur est tout de suite plus amené à faire confiance à notre application. Le blanc nous apporte une couleur neutre et un certain contraste avec le vert.

## **2. Structure des fichiers**

Pour des projets importants et consistants, il est essentiel d'avoir une compréhension globale de la manière dont l'application doit être structurée dès le début. Le but est d'arriver à une structure de fichier évolutive et maintenable qui respecte les meilleures pratiques d'Angular et ses lignes directrices.

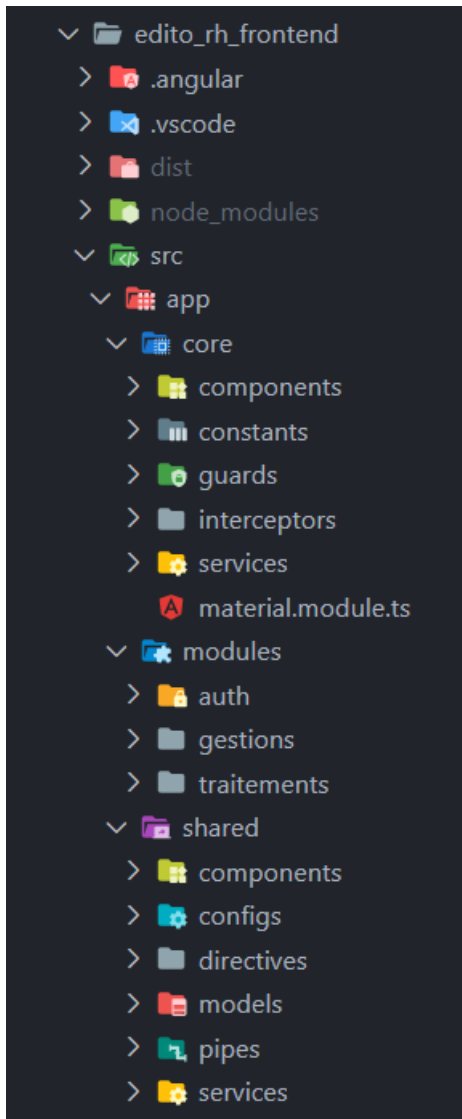


Figure 42: structure de fichiers 1 de Angular

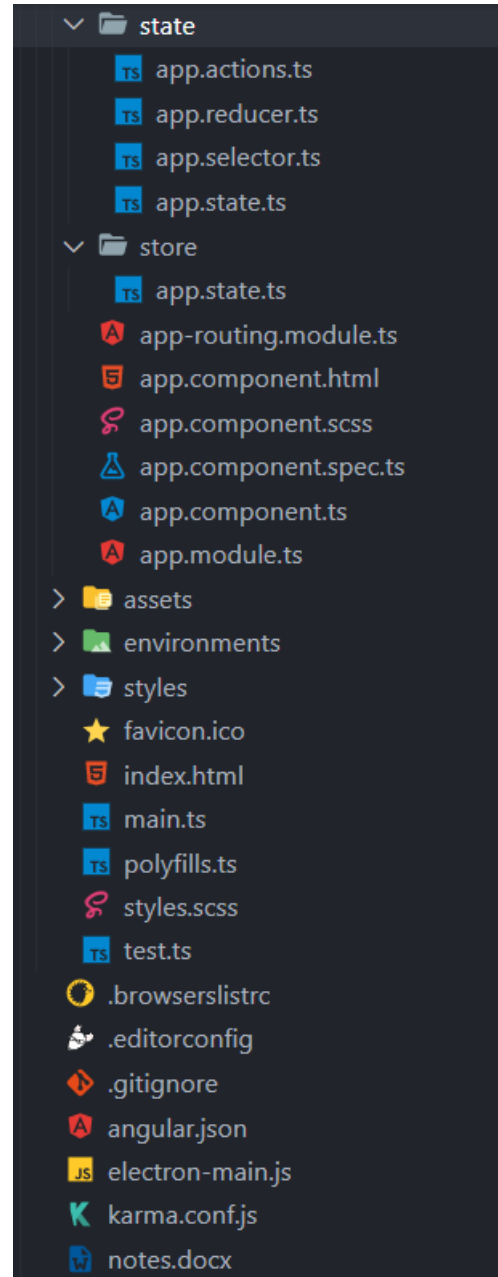


Figure 41: Structure de fichiers 2 de Angular

## Description des Dossiers importants



Angular présente une structure modulaire ou le code est organisé en différents modules donc tous les services et composantes sont divisés en différents groupes.

- **Styles :**

Dans ce dossier, on regroupe tous les fichiers de style qui identifie le style global de l'application.

Et puisqu'on travaille avec Sass, on définit ici aussi les variables globales et les Mixins qui ne sont rien d'autre qu'un extrait de code(style) réutilisable n'importe où.

- **Assets**

Généré par Angular CLI, il sert à garder tous les fichiers médias.

- **Environment**

Utiliser pour préciser l'Endpoint de l'api et séparer en l'environnement prod et dev

Notre application réside au sein du module principale app, voici les dossiers qui s'y trouve :

- **Module Core**

Ce module joue le rôle du module racine de l'application ou on regroupe tout ce qui ne doit être chargé qu'une seule fois. En gros, Il contient tout ce qui est statique et qui ne dépend pas de l'utilisateur comme les gardes (seront expliqués plus tard) et les constantes.

Le module Core contient aussi les fichiers Typescript qui envoient les requêtes à notre API

- **Module Shared**

Ici on regroupe les composants dynamiques qui seront réutilisés souvent dans toute l'application, comme le Header, le Footer, les pipes (fonctions utilisées dans les fichiers html pour transformer les valeurs) et directives (Modifient l'apparence ou le comportement des éléments DOM et des composants angulaires) ect...

On y trouve aussi l'un des dossiers les plus importants qui est « Models » où tous les modèles des classes existent.

Vous trouverez ci-dessous le modèle de la table Affectation :

```
export interface Affectation {
  id:number,
  designation:string,
  description:string,
  path:string,
}

export class AffectationModel implements Affectation{

  private _id
  private _designation
  private _description
  private _path

  constructor(id:number,designation:string,description:string,path:string){
    this._id=id
    this._designation=designation
    this._description=description
    this._path=path
  }

  get id() {
    return this._id;
  }

  get path() {
    return this._path;
  }

  get designation() {
    return this._designation;
  }

  get description(){
    return this._description;
  }
}
```

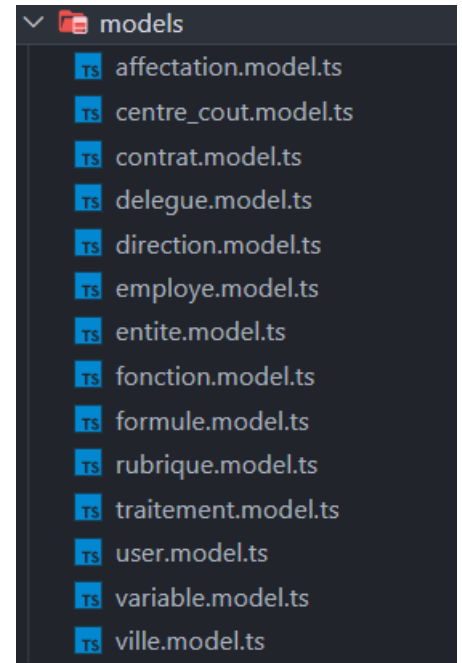


Figure 43: models dans Angular

Figure 44: Code Angular du Modèle Affectation

Vous remarquerez que les attributs sont tous privé. Cette encapsulation nous aide à éviter des erreurs humaines et protège l'objet de tout accès non voulu. Aucun setter n'est créé car aucune modification n'est requise.

- **Module Modules**

Modules contient les modules principaux de notre application. Le module gestion ou les pages de gestions existent, le module auth pour la page login et finalement le module traitement.

### 3. Gestion des états

#### a. Définition

Pour une application monopage (SPA) comme la nôtre avec beaucoup de composants, la gestion des états est un élément clé.

D'abord, l'état représente l'ensemble des données et informations contenu dans le composant, son état actuel. Il est bien sur possible de passer l'état directement du composant parent au fils et vice versa, ceci est la méthode traditionnelle. Mais cette méthode arrive à sa limite si une donnée doit être accessible partout dans l'application par exemple. De plus, de multiples états sont utilisés à travers une application, ce qui explique la complexité inhérente de la gestion de ces états.

C'est là où intervient les Framework de gestion d'états.

#### b. NGRX

NgRx est un Framework de gestions d'états pour créer des applications réactives dans Angular. Il est aussi un groupe de bibliothèques d'Angular pour les extensions réactives et la gestion des états. Il facilite le développement angulaire en simplifiant l'état de l'application dans les objets et en appliquant un flux de données unidirectionnel.

#### c. NGRX cycle de vie

Toute notre application respecte le cycle de vie suivant :

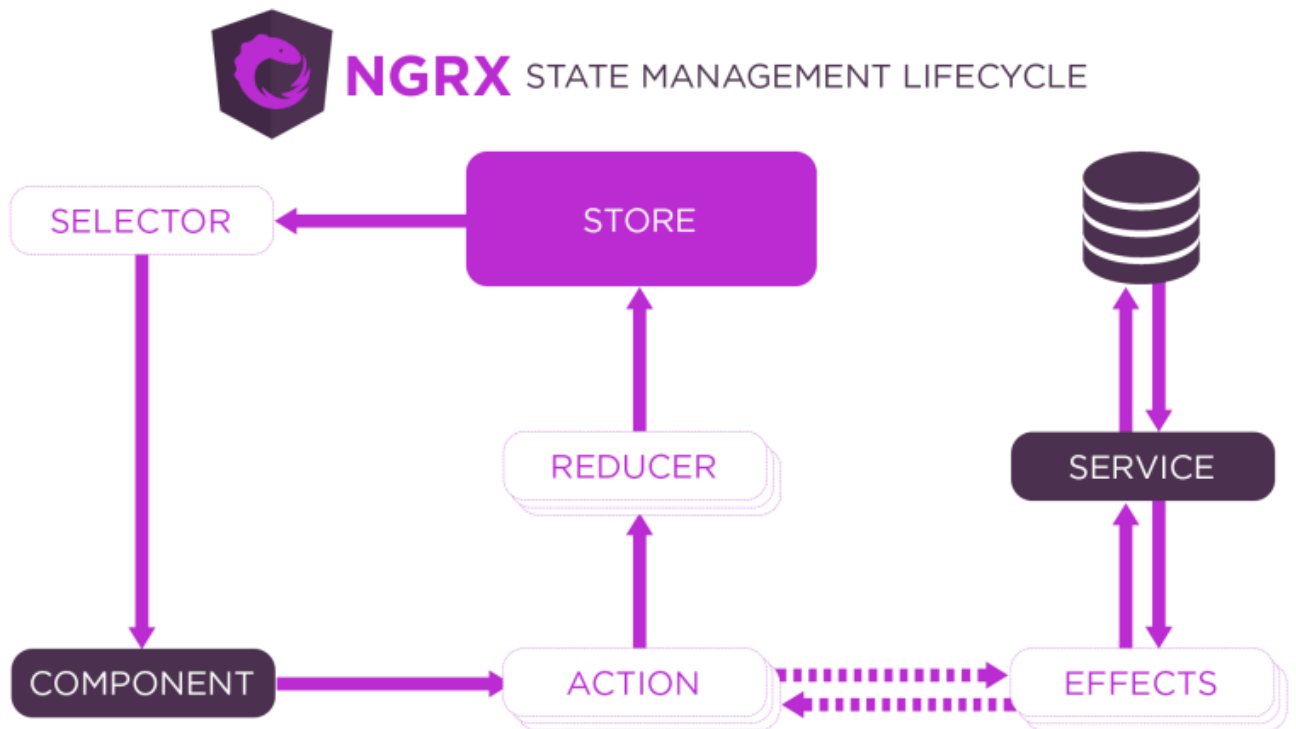


Figure 45: cycle de vie NGRX.( source:<https://medium.com/@knoldus/basics-of-ngrx-719f6360e2bf>)

- **Store :**

Le Store est le composant le plus important de NgRx. Il fournit un magasin unique où toutes les données de l'application sont centralisées et reflète donc l'état global de l'application. C'est une structure de données unique et immuable.

- **Action :**

Les actions expriment des événements uniques qui se produisent dans notre application. Ces événements

Peuvent être entre autres des interactions des utilisateurs ou des requêtes réseau. Les actions sont la façon dont l'application communique avec NgRx pour lui dire quoi faire.

- **Reducer :**

Les réducteurs sont responsables des transitions d'état de votre magasin. Ils réagissent aux actions envoyées et exécutent une fonction pure (fonction qui produit la même sortie pour une entrée donnée) pour mettre à jour le magasin.

- **Selector :**

Les sélecteurs sont essentiellement le mécanisme de détection de changement NgRx. Ils

permettent à notre application d'écouter les changements d'état.

- **Effect :**

Les effets gèrent les effets secondaires de chaque action. Ces effets secondaires sont généralement des communications avec l'API externe via HTTP. Ils isolent les effets secondaires des composants. Cela nous donne des composants "plus purs" qui sélectionnent l'état et exécutent des actions.

On retrouve donc dans chaque module, un dossier nommé « state » qui regroupe l'ensemble des actions, reducers, selectors et effets de tous ses composants. De cette manière, ils restent regroupés et structurés.

#### 4. Réactivité

Comme déjà mentionné, la librairie de composant utilisé est Angular Material UI. Elle propose une fonctionnalité intéressante qui remplace le système Flexbox CSS qui est Flexbox Layout.

Toute l'application est donc extrêmement réactive et s'adapte à une multitude de taille d'écran selon des breakpoints spécifiques : xs, gt-xs, sm, gt-sm, md, gt-md, lg, gt-lg, xl

Voici un exemple de la page gestion :

My App omar@gmail.comm

## Employé

Creer

Modifier

Delete

**Id** 1

**Matricule** 484

**Nom** Hafid

**Prenom** Mohamed

**Date naissance** 1994-01-02

**Cin** BK179415

**omar omari**  
15/06/22  
Modifier

Figure 47: Interface Liste Employé avec taille d'écran petite

My App omar@gmail.comm

Traitement

Gestion

Administration

Figure 46: Header avec taille d'écran petite

Le header s'adapte lui aussi au screen size :

## 5. Concepts utilisés

### a. Chargement paresseux (Lazy loading)

Normalement, une fois l'application démarre, tous les modules sont automatiquement chargés peu importe s'ils sont réellement utilisés ou pas. Pour une application volumineuse comme la nôtre, ceci peut poser quelques problèmes de rapidité et de performances.

Le concept du chargement paresseux nous permet justement de ne charger le module que lorsqu'un itinéraire particulier est appelé. Par exemple une fois l'utilisateur navigue vers l'itinéraire « /gestion/ », le module gestion est chargé.

### b. Réutilisabilité

Comme déjà expliqué, Angular est un Framework Typescript utilisé principalement pour la conception des applications a page unique en se basant sur des composantes. Ceci dit, Il revient au développeur de choisir à quel point il veut que les composantes de son application soient réutilisables. On dispose dans notre cas de 10 pages de gestions. Toutes ces pages ont plus ou moins la même structure, la seule différence sont les données (différents modèles). Il ne serait donc pas sage de créer 10 pages différentes ce qui nous pousserait à dupliquer chaque modification 10 fois. Dans cette optique, Toutes les pages en liaisons avec les modèles suivent un Template développé qui reçoit les données à afficher et fonctions a exécuté spécifique au modèle en question comme input.

Pour arriver à cela, Notre application reçoit des métadonnées de l'API (voir figure 50 )Ces métadonnées comprennent tous les champs du modèle, leurs types et plein d'autre informations qui peuvent être passé au Template pour les affiché comme il se doit.

### c. Stockage local

Le protocole HTTP est l'un des protocoles les plus importants pour une communication fluide entre le serveur et le client. Le principal inconvénient du protocole HTTP, c'est un protocole sans état, ce qui signifie qu'il ne suit aucun type d'information de réponse et de demande du serveur et du client. Ainsi, afin de résoudre ce problème, il existe trois façons de suivre les informations utiles. Stockage locale, Stockage avec session et stockage avec cookies.

Tableau 11: Comparaison de stockages dans le navigateur (source : [https://medium.com/@puneetahuja\\_23950/cookies-vs-localstorage-vs-sessionstorage-9cd77b864f](https://medium.com/@puneetahuja_23950/cookies-vs-localstorage-vs-sessionstorage-9cd77b864f))

## Cookies vs LocalStorage vs SessionStorage

	Cookies	LocalStorage	SessionStorage
Capacity	4kb	5-10 Mbs(Browser Dependent)	5 Mbs
Accessibility	All windows	All windows	Private to tab
Expiration	Manually Set	Never expires	On tab close.
Passed in request	Yes	No	No
Storage	Browser and Server	Browser Only	Browser Only

Le tableau suivant compare ces Trois méthode :

Dans notre application, le choix a été porté vers le stockage locale. Et voici pourquoi :

- Capacité beaucoup plus importante que les autres (5 à 10 Mbs) ce qui est considérablement plus du 4kb stocké dans les cookies
- Accessible et ne s'expire pas, ce qui nous donne plus de contrôle sur quand le crée et le détruire
- il ne passe pas dans les requêtes. (Contrairement aux cookies qui peuvent ralentir la bande passante)

L'utilisation principale du stockage locale dans notre application est d'enregistrer notre access token, refresh token et un objet User qui contient les informations sur l'utilisateur connecté.

Le seul inconvénient rencontré à l'issue de ce choix, c'est le fait qu'on ne peut stocker que des chaînes de caractères avec.

Ce qui fait que pour enregistrer un objet de type User, il faut le sérialiser avant. Puis le désérialiser après pour retrouver l'objet.

### d. Gardes

Les gardes sont un concept d'Angular qui nous permet de limiter les itinéraires que l'utilisateur peut accéder en fonction de s'il est connecté ou pas.

Et sachant que toutes les informations relatives à l'utilisateur sont stockées dans le



stockage local, il suffit de vérifier s'ils existent pour en conclure si l'utilisateur est réellement connecté ou pas.

### III. Réalisation Backend

#### 1. Django Rest Framework

On a déjà précisé que le choix du Framework backend est Django. Cici dit, travailler seulement avec Django risque d'être une expérience fastidieuse et de prendre beaucoup de temps car il va falloir développer beaucoup de chose essentiel pour développer une API Rest. Django Rest Framework se présente justement comme étant une boîte à outils puissante et flexible pour la création d'API Web. Voici quelques-unes de ses avantages notables :

- Propose un API navigables conviviale
- Facilite l'authentification
- Accès au Serializer qui prennent en charge les sources de données ORM et non ORM.
- Une documentation complète et accessible

Django Rest Framework propose aussi plusieurs approches pour développer les views, qui sont :

- Views basé sur les classes
- Utilisation des Mixins
- Utilisation des des Generics

Le développement a été entamés en utilisant les Generics qui facilitent énormément de taches (très peu de ligne de codes). Mais Il s'est avéré difficiles d'ajouter de nouvelle fonctionnalité spécifique à notre application. Au finale, La partie backend a été développé en se basant sur les Mixins. Ce qui nous a permis d'implémenter toutes les fonctionnalités communes utile pour la création d'une API Rest tout en restant flexible.

Comme on a déjà préciser, on dispose de deux bases de données (main\_db et user\_db). Donc notre API se doit de se connecter avec deux bases de données en utilisant le module « MySQL Client ». Django nous permet alors en utilisant son ORM, d'ajouter, modifier ou supprimer des tables et des champs de ces bases sans devoir utiliser des requêtes SQL et sans modifier directement la base à l'aide d'un SGBD.

## 2. Description des Endpoints

Les Endpoints d'une API sont l'emplacement numérique spécifique où les requêtes sont envoyées pour qu'on puisse récupérer les données qui y existe.

On listera ici tous les Endpoints exposés par notre api :

- GET/PUT/DELETE: /api/v1/villes/<id>
- GET/POST: /api/v1/villes/
- GET/PUT/DELETE: /api/v1/variables /<id>
- GET/POST: /api/v1/variables/
- GET/PUT/DELETE: /api/v1/affectations/<id>
- GET/POST: /api/v1/affectations/
- GET/PUT/DELETE: /api/v1/centres-cout/<id>
- GET/POST: /api/v1/centres-cout/
- GET/PUT/DELETE: /api/v1/contrats/<id>
- GET/POST: /api/v1/contrats/
- GET/PUT/DELETE: /api/v1/directions/<id>
- GET/POST: /api/v1/directions/
- GET/PUT/DELETE: /api/v1/entites/<id>
- GET/POST: /api/v1/entites/
- GET/PUT/DELETE: /api/v1/fonctions/<id>
- GET/POST: /api/v1/fonctions/
- GET/PUT/DELETE: /api/v1/formules/<id>
- GET/POST: /api/v1/formules/
- GET/PUT/DELETE: /api/v1/formules/<formule\_id>/variables/<variable\_id>
- GET/PUT/DELETE: /api/v1/employes/<id>
- GET/POST: /api/v1/employes/
- GET/PUT/DELETE: /api/v1/employes/<employe\_id>/rubriques/<rubrique\_id>
- GET/PUT/DELETE: /api/v1/traitements/<id>
- GET/POST: /api/v1/traitements/
- POST : /api/v1/auth/register
- POST : /api/v1/auth/login
- GET: /api/v1/auth/user

- POST: /api/v1/auth/refresh
- POST: /api/v1/auth/logout

### 3. Structure des fichiers

Contrairement à Angular, la structure des fichiers le Django n'a pas besoin d'être autant modifié pour arriver à une structure flexible vues le nombre limité de fichiers et lignes de codes. Ceci dit, Une organisation supplémentaire du projet peut aider à garder le projet encore plus DRY et propre.

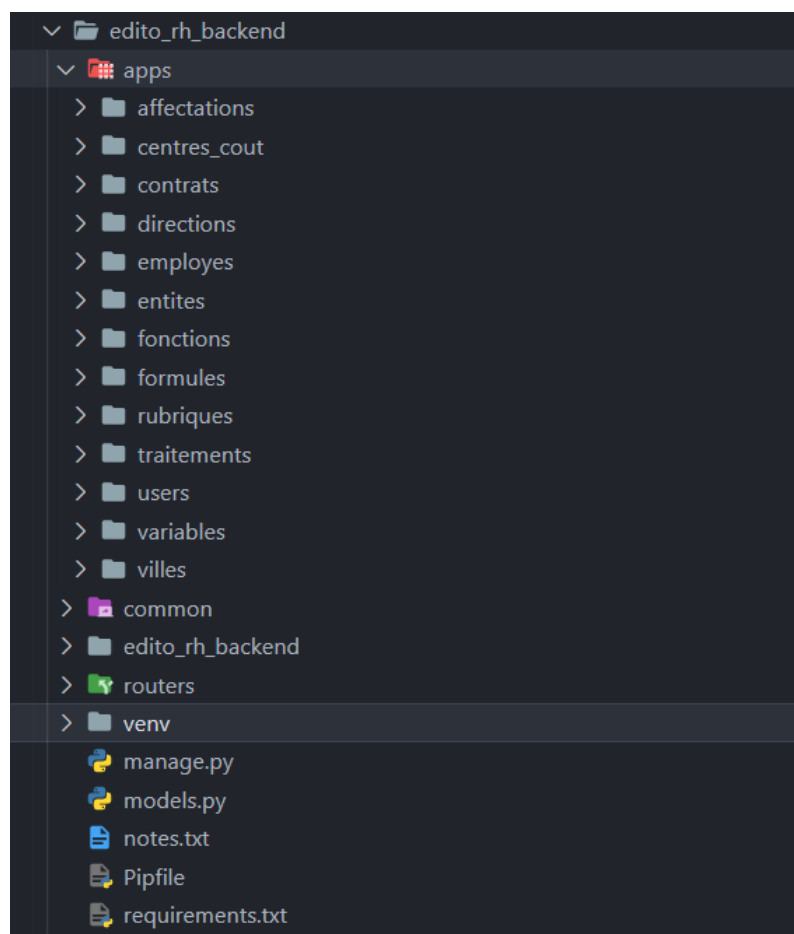


Figure 48: Structure de fichiers Django

## a. Description des Dossiers importants :

On trouvera ici une description de chaque dossier majeur de notre application qui réside dans le dossier racine :

- **Apps** : Contient tous les dossiers concernant de nos modèles. (urls.py, serializer.py, models.py, views.py)
- **Common** : contient toutes les fonctions et classes communes qui peuvent être utilisés par tous les modèles
- **Routers** : le Router d'une base de données limite l'accès à cette dernière. (read, write, allow\_relation, allow\_migrate) et permet de choisir quel modèles peuvent y accéder
- **Venv** : On y trouve notre environnement environnement ou on trouve tous les modules non standard utilisés
- **Edito\_rh\_backend** : Créer par Django, il s'y trouve un seul fichier settings.py et quelques autres fichiers importants tels que urls.py, wsgi.py, etc.

## 4. Rest Api

Une API compatible REST, ou « RESTful », est une interface de programmation d'application qui fait appel à des requêtes HTTP pour obtenir ajouter, supprimer ou modifier des données. C'est aussi un ensemble de contraintes architecturales et meilleurs pratiques et non pas un protocole ou une norme. Il peut donc être implémenté de différente manière dans une API. On citera ici en quoi notre api est-elle REST :

- JSON est le format d'échange de données utilisés
- Utilisation de noms au lieu de verbes dans nos Endpoints car les méthodes http (GET, PUT, POST, DELETE) sont déjà des verbes.
- Utilisation des noms en pluriels comme « /employees » pour que l'utilisateur comprenne qu'il y'a plusieurs enregistrements dans cette Endpoints.
- Utilisation des status code pour la gestion des erreurs
- Utilisation des imbrications logique des Endpoints pour montrer les relations qui existent entre les entités comme pour  
« /employees/<employe\_id>/rubriques/<rubrique\_id> »
- Utilisation du filtrage, pagination et du tri pour recevoir les données

- Précision la version de l'api dans le Endpoints. « /api/v1 ».
- Utilisation des liens hypermédias pour guider l'utilisateur
- Utilisation d'un modèle de requête sans état. Ceci s'explique par l'utilisation d'une authentification en utilisant les tokens jwt.

## 5. Filtrage, Pagination et tri

### a. Filtrage

Le filtrage est l'action d'appliquer une collection de conditions sur la requête en question pour retrouver les sous ensemble désirés

Il existe plusieurs manières d'approcher le problème. On aurait surement pu utiliser un filtrage classique comme montré dans l'exemple suivant :

```
GET /api/v1/employees/ ?nom=youness&prenom=Nainia
```

Mais le problème avec cette méthode et tout autre méthode similaire, c'est qu'elle ne nous donne pas assez de flexibilité et il est pratiquement impossible d'utiliser ou combiner des expressions logiques (or, and) pour arriver à une réponse plus précise de l'API.

C'est là où intervient la méthode du filtrage avec le paramètre query « filter »

Voici un exemple de plusieurs requêtes GET :

```
GET /api/v1/contrats?filter=and(eq(designation,CDI),eq(description,durée indéterminée))
```

```
GET /api/v1/employees?filter= eq(nom,youness)
```

```
GET /api/v1/employees?filter=or(eq(nom,youness),lt(salaire,2000)))
```

```
GET /api/v1/employees?filter=or(eq(nom,youness),lt(salaire,2000)))
```

En utilisant qu'un seul paramètre de requête « filter », on arrive à :

- Combiner plusieurs expressions avec des opérations logiques (et, ou, et, non)
- Faire des comparaisons à l'aide d'opérateurs relationnels (eq, ne, lt, le, gt, ge)
- Et utiliser d'autres fonctions telles que contains, startsWith, in, match, etc.

Voici tous les opérateurs et fonctions supportés par notre api :

eq, ne, lt, lte, gt, gte, iexact, startsWith, icontains, contains, year, month, day

Pour arriver à ce résultat optimal, un Lexer, un Parser et un Compiler ont été développés

« from scratch » car aucune librairie gratuite ne proposait exactement ce service.

- **Lexer** : aussi appelé tokenizer. Son rôle est de prendre l'expression comme input(eq(nom,youness)) et retourne les tokens séparés sans délimiteurs (eq,nom,youness)
- **Parser** : prend la liste de tokens comme input et retourne une représentation logique de l'expression souvent sous forme d'arbre (dans notre cas c'est créé avec des dictionnaires)

Voici la représentation en arbre de l'expression suivante : `and(or(e,not(b)),gt(C,0))`

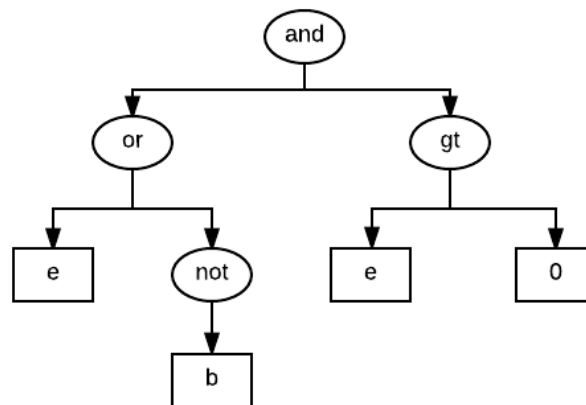


Figure 49: composition en arbre d'une expression  
(source : <https://developer.sas.com/reference/filtering/>)

- **Compiler** : prend le résultat du parser (dictionnaires imbriqués) et applique les filtres sur la requête GET

## b. Pagination

La pagination est faite en utilisant les deux paramètres de requêtes :

- **Limit** : utilisé pour limiter le nombre de lignes renvoyées à partir d'un ensemble de résultats.
- **Offset** : Elle nous permet de spécifier de quelle ligne il faut commencer pour récupérer les données

Chacune de nos pages contient 100 pages, donc en guise d'exemple voici les 2 premières pages de la gestion d'employés :

Page 1: GET /api/v1/employees?limit=100&offset=0

Page 2: GET /api/v1/employees?limit=200&offset=100

### *c. Le tri*

Le tri est implémenté en utilisant le paramètre de requête « filter » suivis d'une liste d'attributs séparé par des virgules qui représentent les champs à trier par dans l'ordre de gauche à droite. On identifie un tri descendant Si l'attribue est précédé par « - ». Sinon, le tri est ascendant.

Exemple :

GET /api/v1/employees?filter=nom,-salaire,date\_naissance

### *d. Métadonnées*

Les métadonnées sont par définition des données sur d'autre données. Dans notre cas il s'agit plutôt de données sur les champs des tables de notre base de données qu'on retourne dans la réponse accompagnée des données demandées.

Django REST Framework (DRF) propose une solution qui génère les Métadonnées Automatiquement pour chaque champ précisé dans le Sérialiser. Mais pour mieux contrôler les données qu'on souhaite partager avec les utilisateurs, on choisit les métadonnées nous-même.

Les métadonnées se trouvent dans le fichier JSON sous l'attribut « metadata ».

```
"metadata": {
  "fields": [
    {
      "type": "number",
      "label": "id"
    },
    {
      "type": "string",
      "label": "description"
    },
    {
      "type": "string",
      "label": "designation",
      "values": [
        {
          "value": "Prime de Transport",
          "id": 1
        },
        {
          "value": "Avantage nature carburant ",
          "id": 2
        }
      ]
    }
  ]
}
```

Figure 50: Exemple de métadonnées

Les métadonnées jouent un rôle crucial dans notre application comme déjà expliqués. On retrouve pour chaque attribut son type et ses valeurs.

## 6. Authentification

L'authentification c'est la procédure visant à certifier l'identité de quelqu'un ou d'un ordinateur afin d'autoriser le sujet d'accéder à des ressources. Il est important aussi de préciser que Django Rest Framework a facilité considérablement le développement de la partie authentification.

### a. JSON Web Token

Aussi appelé JWT, JSON Web Token (JWT) est une norme qui définit un moyen compact et autonome pour transmettre en toute sécurité des informations entre les parties en tant qu'objet JSON.



C'est aussi un mécanisme permettant de vérifier qui est le propriétaire de certaines données JSON. Il s'agit d'une chaîne codée sécurisée pour les URL qui peut contenir une quantité illimitée de données (contrairement à un cookie) et qui est signée de manière cryptographique. Utiliser JWT est l'une des méthodes pour s'assurer que notre application reste sans état (stateless) car toute l'information relative à l'utilisateur sont stocker à l'intérieur même du JWT. En d'autres termes, l'application ne conserve pas l'email et le mot de passe de l'utilisateur.

Il est composé de 3 composantes :

- **Entête :**

Il se compose généralement de deux parties : le type de jeton, qui est JWT, et l'algorithme de signature utilisé. Dans notre cas on a utilisé HS256

- **Charge utile :**

La deuxième partie du jeton est la partie utile qui contient toutes les données importantes qui vont être transféré à l'application. Les informations sont présentées sous forme de clé/valeur. Le standard définit différents types de propriétés (appelées « claims ») réservées, publiques et privées.

Pour les propriétés réservées et publiques, on peut noter • sub : définit le sujet du token, en règle générale un identifiant d'utilisateur ; • exp : définit la date et l'heure (timestamp) à laquelle le token expire (voir expiration plus loin). Les propriétés privées sont ceux qui en générale identifie l'utilisateur et apporte des informations sur lui. Comme :

- **Signature :**

Cette dernière partie permet de s'assurer de l'intégralité des deux dernières composantes. Pour s'assurer que rien n'a été modifier, on calcule la signature en utilisant l'algorithme précisé et en prenant comme input les données de l'entête et la charge utile et une clé privée et on la compare avec la signature qui existe dans le JWT. Si c'est différent, on en conclut qu'elle a été modifiée et on refuse du traité.

Voici un exemple réel d'un access token de notre application :

HEADER: ALGORITHM & TOKEN TYPE
<pre>{   "typ": "JWT",   "alg": "HS256" }</pre>
PAYLOAD: DATA
<pre>{   "user_id": 1,   "user_nom": "omari",   "user_prenom": "omar",   "user_email": "omar@gmail.comm",   "exp": 1655295015,   "iat": 1655291415 }</pre>
VERIFY SIGNATURE
<pre>HMACSHA256(   base64UrlEncode(header) + "." +   base64UrlEncode(payload),   your-256-bit-secret ) <input type="checkbox"/> secret base64 encoded</pre>

Figure 51: Exemple d'un access token(source:<https://jwt.io/>)

### b. Refresh Token

On Dispose maintenant d'un JWT token qui est notre Access token. Son rôle est bien sûr de permettre à notre utilisateur d'accéder à ses recoures en vérifiant son identité à chaque requête reçue.

Le problème que cela peut poser, c'est que si le access token est expiré, le frontend va recevoir une erreur 401 et l'utilisateur sera automatiquement déconnecté. Et vu le fait que la date d'expiration du access token est très petites (maximum 5 minutes), il n'est pas raisonnable de demander à l'utilisateur de se connecter en saisissant son email et mot de passe chaque 5 minutes.

La solution est le refresh token qui est utilisé par le client pour recevoir un nouveau access token en cas d'expiration de son access token en lançant une requête vers le endpoint « /refresh » .

### c. Processus d'authentification

Le principe est le suivant. Une fois connecté, l'utilisateur reçoit son access token et son refresh token (tous les deux sous forme JWT). Il peut maintenant lancer n'importe quelle requête en utilisant son access token comme un bearer token. A chaque fois que la partie backend reçoit l'access token, elle vérifie s'il n'est pas expiré et vérifie son intégrité en calculant la signature et la comparant avec celle dans le token.

Une fois ce dernier est expiré, le frontend reçoit une erreur 401. Il envoie donc automatiquement une autre requête POST à l'Endpoint « /refresh » en envoyant son refresh token pour que l'API lui génère un nouveau access token qu'il pourra ensuite utiliser pendant les 5 prochaines minutes. L'utilisateur n'est donc amené à se déconnecter que lorsque le refresh token lui-même est expiré. Ce qui peut prendre plusieurs jours.

Ce processus est détaillé dans le diagramme de séquence dans le diagramme de séquence (figure 14)

### d. Gestion des erreurs et cors

Comme déjà mentionné, toutes la partie backend respecte les statuts d'erreurs standard. Malgré ça, il est important en cas d'erreur d'envoyer une réponse uniforme avec un format fixe au client frontend pour que le comportement de notre API reste prédictif.

Le modèle qu'on a utilisé inclut des informations détaillées. Il a été fortement inspiré par la manière avec laquelle Facebook retourne ses erreurs pour FACEBOOK Graph API.

Ici un exemple d'un message d'erreur pour une requête non authentifié retourné par notre api :

```
{
  "detail": "Unauthenticated",
  "code": 401,
  "status": "UNAUTHORIZED",
  "isSuccessful": false
}
```

Figure 52: Erreur non authentifié  
backend

Dans « details », on présente une explication de l'erreur lisible par les humains. Le statut d'erreur en plus d'être dans la requête, est visible dans la réponse sous le champ « code ». Ainsi que sa signification dans le champ « statut ». « isSuccessful » est toujours envoyé et

montre si c'est une erreur ou pas.

## CHAPITRE 5 : « Perspectives d'amélioration »

## Introduction :

Vu que Le projet est encore en cours de développement, ils restent plusieurs fonctionnalité a traité. On parlera des améliorations requises dans la partie backend puis frontend.

### I. Autorisation

#### a. Diagramme de classe

Dans la partie Backend, bien que la partie authentification a été bel et bien traité, il est important de traiter l'autorisation qui est tout aussi importante. Il s'agit principalement d'introduire des rôles et fonctions pour limiter l'accès des utilisateurs.

Voici le diagramme de classe qui sera éventuellement respecté durant le développement :

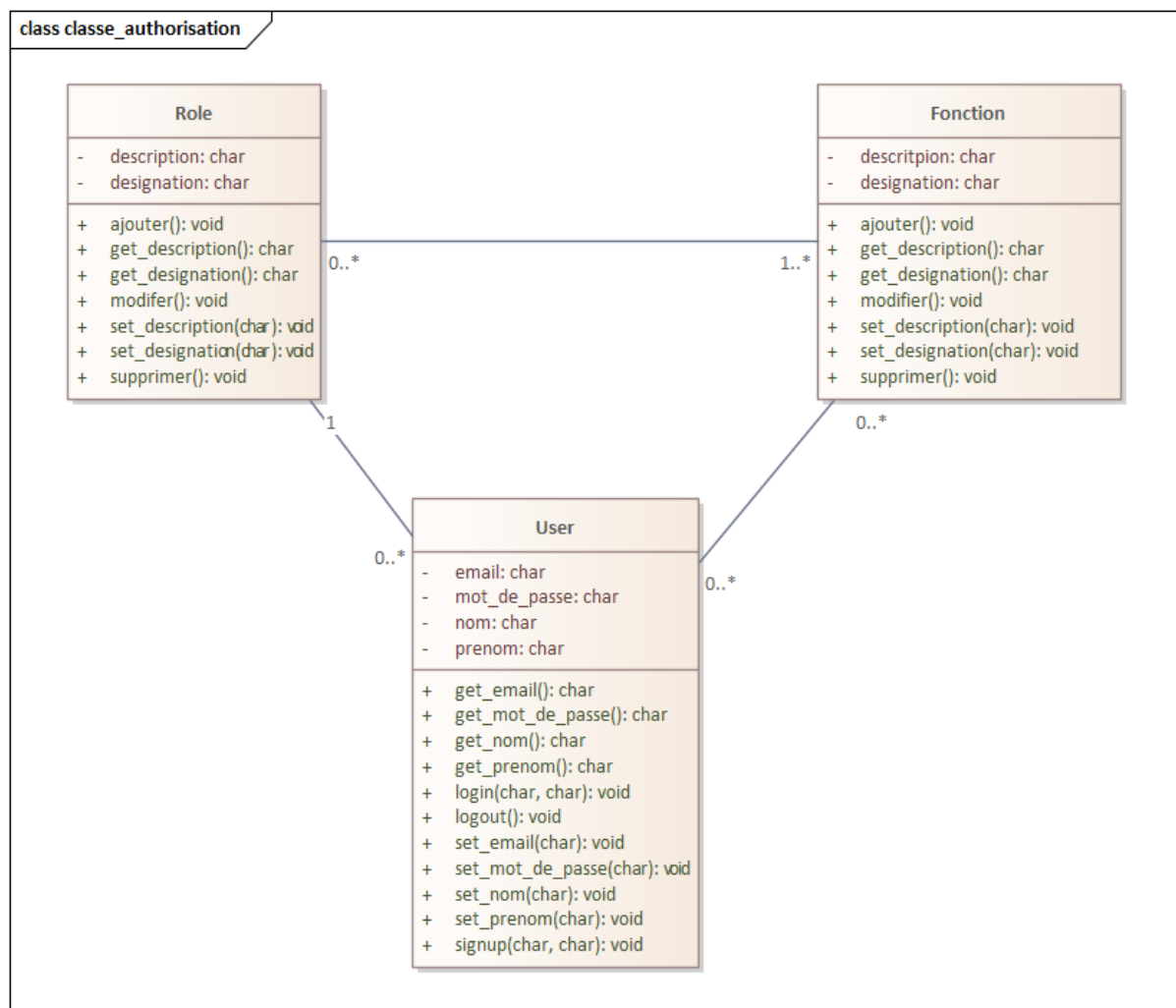


Figure 53: Diagramme de classe d'autorisation

On remarque que l'Utilisateur est à la fois lié à la classe Role et la classe Fonction. L'idée principale est qu'un utilisateur peut appartenir à un rôle et peut quand même bénéficier de Fonctions propre à lui qui peuvent ne pas exister dans le rôle dans lequel il appartient.

### *b. Description des classes*

- **Classe Fonction** : une fonction est une action que l'utilisateur peut faire. Par exemple créer employé, modifier employé, faire traitement
- **Classe Role** : Plusieurs rôles sont regroupés dans un rôle. Par exemple le rôle administrateur doit pouvoir tout faire.

## II. Tests et documentations

### *a. Documentation*

Même Si ce projet est pour l'instant géré par une seule personne, il est possible que d'autres développeurs continuent le développement ou la maintenance de ce dernier. Et bien que le code soit commenté et bien structuré. Ceci n'est pas suffisant pour ces nouveaux développeurs qui viennent tout juste de le découvrir. Sans oublier la possibilité de l'utilisation ou l'intégration de l'API dans d'autre projet. Une documentation de l'API est donc indispensable dans ce cas.

Il existe heureusement plusieurs modules pour Django Rest Framework nous permettant de générer automatiquement des pages de documentation HTML à partir de schémas OpenAPI. Parmi eux, on peut citer les deux les plus populaires Swagger UI et ReDoc.

### *b. Tests*

Ils existent plusieurs types de test. Mais on retrouve deux types de tests qui doivent être implémentés dans le futur pour garantir la stabilité du système et éviter des bugs intentionnels.

- **Test unitaire** : permet de vérifier si le code effectue la fonction attendue de lui. Et ceux en testant chaque entité indépendamment des autres. Peut-être implémenté soit dans le backend soit dans le frontend en utilisant des données mock (données créées qui ne viennent pas de l'API)

- **Test d'intégration** : ils seront développés dans la partie frontend et permette de vérifier si plusieurs unités de code fonctionnent bien ensemble. Dans notre cas on parle de notre application Desktop et de notre API.

### III. Déploiement

Une fois la première version du projet touchera à sa fin, La phase du ploiement commencera. Alors Si la partie frontend restera dans l'ordinateur de l'utilisateur sous forme d'application Desktop (à l'aide d'Electron.js), La partie backend sera déployé dans le serveur locale de l'entreprise vu la sensibilité des données (salaire des employés).

Vous trouverez ci-dessous un diagramme de déploiement qui illustre le process de déploiement physique des informations générées par le logiciel sur des composants matériels.

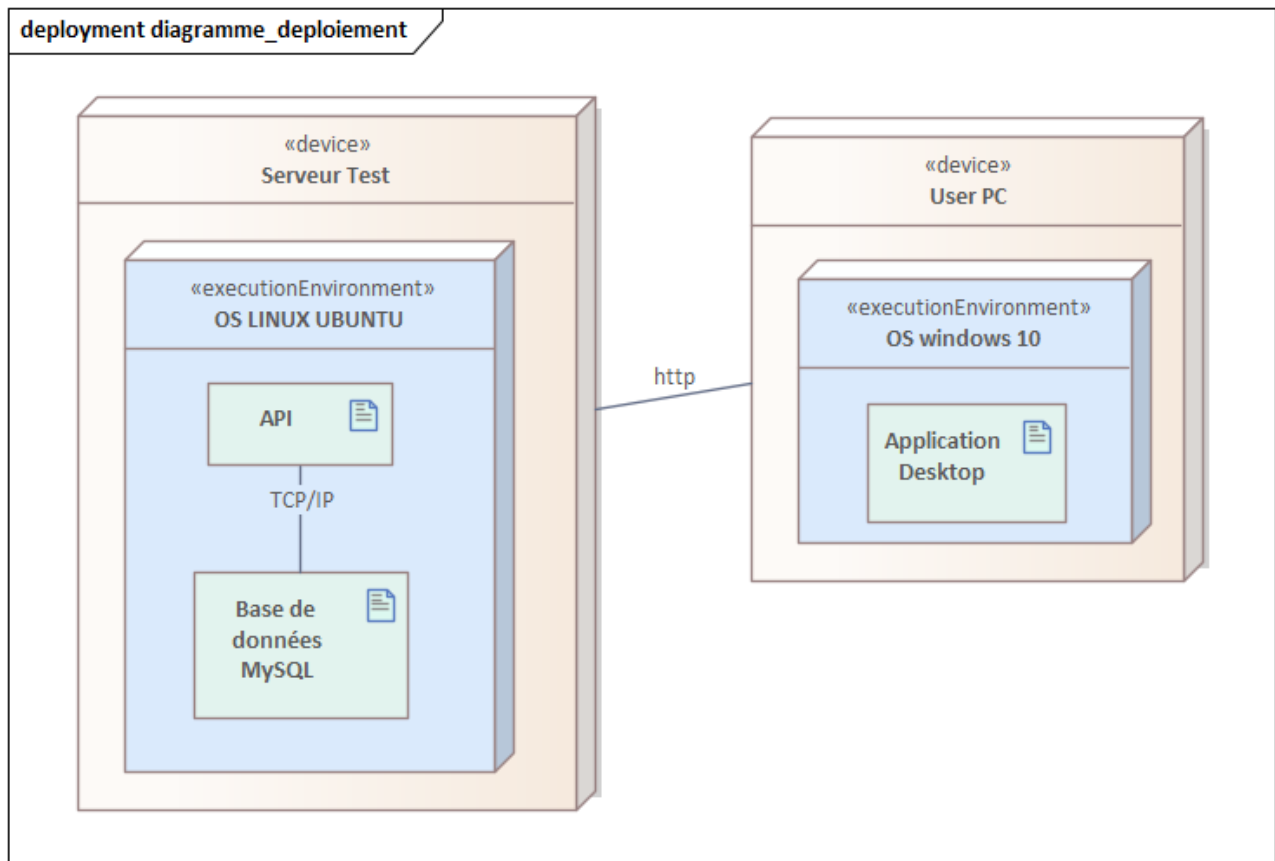


Figure 54: Diagramme de déploiement de l'application

Notre Api ainsi que notre base de données seront dans le même serveur Test de l'entreprise en premier lieu. Notre application communiquera avec l'API via le Protocole http.

## IV. Développement

Il reste toujours quelques points qui doivent être développés surtout dans la partie frontend :

- **Page traitement** : Bien que la partie backend est finie. On doit développer la page traitement qui nous permettra de faire les calculs présents sur les formules sur chaque employé
- **Page administration** : Seul l'administrateur aura le droit d'accéder à cette page. On peut y créer des utilisateurs nouveaux et leurs accroître des droits et les affectés à un rôle précis. On doit aussi pouvoir créer et modifier les rôles.

## Conclusion :

Vu que ce stage est encore en cours au moment de la rédaction de ce rapport, les fonctionnalités importantes comme le développement de la page traitement seront éventuellement terminés.



## Problèmes rencontrés

Durant toute la période de conception et développement de ce projet, plusieurs problèmes ont été rencontrés.

Le premier était de regrouper et comprendre tous les besoins de l'équipe RH et toutes les informations nécessaires au bon déroulement du stage. Des recherches ont été faites dans le domaine des ressources humaines et plus précisément la gestion de paie pour comprendre leurs jargons d'un côté et pouvoir communiquer efficacement avec eux d'un autre. Cette découverte graduelle des besoins à mener inévitablement a des changements réguliers de la structure même de la base de données au fur et à mesure que les besoins s'éclaircissent pour nous et pour le client. L'implémentation du Scrum a sans doute rendu cette tâche plus facile. Même si on a dû (Surtout au début du développement) plutôt privilégier la partie frontend car elle est moins exposée aux changements radicaux en plus d'être un moyen efficace de montrer l'avancement du projet au client durant les sprints reviews.

Ensuite, il est le cas de dire que nous avons cruellement sous-estimer le temps et l'effort que prendra l'apprentissage ne serait-ce que des connaissances de bases de la technologie frontend choisie Angular. Ceci sans compter le langage Typescript et Django Rest Framework.

De plus, tout au long de la période de stage, nous avons eu la chance d'apprendre plusieurs choses en parallèles et d'améliorer notre connaissance sur le métier de la logistique, Commerciale, Service client, Facturation etc. Cependant, il s'est avéré particulièrement difficile de d'équilibrer entre la conception et développement de notre projet et l'apprentissage et l'exécutions des taches suivantes :

- Découvertes du système d'information Odoo version 13
- Traitements de demandes internes ou externes en relation en relation avec Odoo
- Gérer avec l'équipe SI différents projets clients. (Puisque l'entreprise possède un prestataire qui s'occupe de la partie code, il s'agissait plus de définir et comprendre les besoins des clients de l'entreprise. Puis de les gérer et faire leurs suivis. Et ensuite de les communiquer clairement et fidèlement au prestataire)

- développer plusieurs code python qu'on appelle « moulinettes » dont le but est de prendre des fichiers Excels exportés de notre système d'information et de les traiter pour qu'ils deviennent compatible avec le SI du client ou ils seront ensuite importés. (Développé en utilisant pandas pour le traitement, tkinter pour la partie UI et openpyxl pour créer un exécutable).
- Développement de moulinette qui fait la planification des colis (spécifie dans quelle agence et quels date les colis vont être ramassés en se basant sur plusieurs paramètres)
- En cours de développement d'une moulinette pour la gestion de stock qui jusque-là se faisait manuellement.
- Découverte de l'API XML-RPC de Odoo de l'entreprise

Ce travail est d'une grande importance car ces moulinettes sont utilisées actuellement en production et accélère le travail des employés au jour le jour. Ceci dit, dans le contexte de ce projet, on peut les considérés comme des problèmes qui ralentissent l'avancement du projet.

## Conclusion générale et perspectives

Finalement, Ceci conclut mon stage de fin d'étude de 6 mois pour l'obtention de diplôme d'ingénieur d'état dans la spécialité IA&GI au sein du siège de l'entreprise Sapress à Casablanca.

Je peux désormais fièrement dire que ce stage a été une expérience satisfaisante et riches d'enseignements. Professionnellement, ce projet m'a permis avant tout de me confronter au domaine du développement d'application Desktop. Ce qui m'a permis de concrétiser mes connaissances théoriques et de découvrir de nouvelles technologie (Angular, Django, Electron ...). Ce stage a été pour moi un portail vers l'entreprise et tous ses métiers (RH, CS, facturation, caisse, logistique ...) ainsi qu'une mine au niveau relationnel. Ceci dit, il s'est avéré difficile d'équilibrer entre mes missions quotidiennes dans l'entreprise (En relation avec le SI Odoo) et le projet de ce rapport. De plus, Je trouve qu'une expérience future de développement dans le cadre d'une équipe me serait bénéfique.

Le résultat de ce stage est une application Desktop utilisée par l'équipe Rh qui vise à faciliter la gestion des employés et à faire des traitements et calculs sur ces données.

J'ai réussi en premier lieu à identifier les différentes exigences de l'équipe RH. Puis de préparé un planning conforme à la démarche SCRUM respecté durant toute la conception et le développement.

Les employés ont été stocké dans une base de données MySQL et l'utilisateur arrive à les gérer efficacement. Notre solution leurs permet aussi d'ajouter des formules qui seront utilisé durant le traitement. Le tout accessible dans une interface graphique conviviale développé avec Angular qui communique avec une API Rest développé avec Django.

A pars l'interface frontend de traitement, On peut dire qu'on a répondu aux besoins de notre client.

Sachant que mon stage n'est pas fini à la date du dépôt du rapport, de nouvelles pistes été améliorations peuvent être exploré une fois ce projet terminé. Je site par exemple la possibilité d'intégrer cette application avec le système d'information déjà existant Odoo. Il se peut aussi

qu'elle englobe plus de fonctionnalité de gestion de paie. Et sachant qu'il est développé avec des technologie web, cette application pourrait très bien être présenté comme application SaaS déployée dans le cloud.

## Bibliographie

- <https://angular.io/>
- <https://material.angular.io/>
- <https://github.com/>
- <https://ngrx.io/>
- <https://docs.djangoproject.com/en/4.0/topics/db/queries/>
- <https://changethework.com/ressources-humaines-definition/>
- CMSC 430, Introduction to Compilers, Lexing and Parsing, 2017, p79
- <https://www.youtube.com/> Mohamed Youssefi chaine
- Martin Fowler, Refactoring improving the Design of Existing Code, Second edition, 2018,p 454