

Année universitaire : 2019/2020

MASTER 2 INFORMATIQUE

Rapport de stage

Développement d'un projet pour ALTEN DELIVERY CENTER MAROC concernant le système d'information pour la gestion des compétences

Étudiant	Tayeb BOUSFIHA Courriel : tayebbousfiha72@gmail.com
Entreprise d'accueil	ALTEN DELIVERY CENTER MAROC Département : Systèmes d'information et systèmes embarqués « SISE ». Adresse : Shore Harazem, Route de Fès, Gzoula Sidi Mbarek
Maître de stage	Anas EL KORCHI Courriel : anas.elkorchi@alten.com
Tuteur à l'université	Franck POIRIER Professeur des universités en informatique à l'université Bretagne-Sud « UBS ». Courriel : franck.poirier@univ-ubs.fr
Date de la soutenance	Le 4 novembre 2020

Confidentialité et respect du règlement d'ALTEN DELIVERY CENTER MAROC

Ce rapport respecte le règlement d'ALTEN DELIVERY CENTER MAROC et ne divulgue pas les noms, les fonctions et les photos des collaborateurs ou des clients sauf au niveau du remerciement où trois noms sont cités sans préciser leur fonction ainsi ce rapport n'est en aucun cas confidentiel.

Remerciements

Au terme de ce travail, nous saisissons cette occasion pour exprimer nos vifs remerciements à toute personne ayant contribué, de près ou de loin, à sa réalisation.

Nous souhaitons tout d'abord remercier Mme Bahia QOZAM, pour le vif accueil qu'elle nous a accordé au sein de la société, Mr. Anas EL KORCHI, pour son encadrement, suivi permanent et ses précieux conseils, Mr. Hicham KIKI pour ses explications et son soutien ainsi que tout le personnel de la société « ALTEN DELIVERY CENTER MAROC » pour le temps qu'ils nous ont accordé afin de nous fournir les explications nécessaires et toute l'aide dont nous avons eu besoin.

Nous tenons à adresser aussi nos remerciements les plus profonds à la directrice de notre école Rim MRANI pour son énorme soutien, le directeur de l'ingénierie Rachid NAOUAL pour le savoir partagé, le directeur du master Franck POIRIER pour son enseignement précieux et tout le personnel de l'ISGA et à l'ensemble du corps professoral pour l'effort qu'il fournit pour faire de nous des futurs ingénieurs, sans oublier de remercier nos chers parents pour tous leurs sacrifices tout au long de nos études.

Résumé (français/anglais)

Ce rapport de stage présente les étapes professionnelles d'analyse, de conception et de réalisation d'une application pour « ALTEN DELIVERY CENTER MAROC ». Il s'agit d'un projet pour la gestion des compétences, des collaborateurs et des projets. L'application doit générer les DT (dossiers techniques) détaillés automatiquement en se basant sur les compétences utilisées pour chaque projet et l'expérience de chaque collaborateur. Elle doit également permettre de prévoir les congés, connaître la disponibilité des consultants, gérer des formations et générer des rapports. Dans ce rapport n'est présentée que l'évolution de l'application depuis sa création jusqu'au moment du rendu de ce dernier.

This internship report presents the professional stages of analysis, design and implementation of an application for « ALTEN DELIVERY CENTER MAROC ». This is a project for the management of skills, collaborators and projects. The application must generate detailed technical files automatically based on the skills used for each project and the experience of each collaborator. It must also make it possible to predict holidays, know the availability of consultants, manage training and generate reports. In this report, only the evolution of the application from its creation until the time of rendering this document is presented.

Mots clés

Application web monopage, système d'information, dossiers techniques, gestion des compétences, Services web REST, OAuth, JHipster, React, Javascript, TypeScript, Spring Boot, technologies récentes et modernes, Analyse et conception, méthode agile Scrum, outil de planification Azure DevOps.

Liste des tableaux :

Tableau 1 : Entité Collaborateur.....	38
Tableau 2 : Entité Utilisateur	39
Tableau 3 : Entité Compétence	39
Tableau 4 : Entité Catégorie	39
Tableau 5 : Énumération Niveau	40
Tableau 6 : Énumération Rôle	40
Tableau 7 : Énumération Situation familiale	40
Tableau 8: Les livrables.....	41
Tableau 9: Les risques du stage	42
Tableau 10 : Acteurs.....	44

Liste des figures :

Figure 1 : Carte d'identité ALTEN.....	11
Figure 2 : Le groupe ALTEN à l'échelle internationale en 2020.....	12
Figure 3 : Organigramme d'ALTEN DELIVERY CENTER MAROC à Fès.....	16
Figure 4: Architecture REST	20
Figure 5 : Fiche projet - système d'information pour la gestion des compétences	22
Figure 6 : Project sheet - information system for skills management.....	23
Figure 7 : Issue de génération des entités avec leur CRUD	25
Figure 8 : Issue changement de la selection des relations des entités pour une meilleure visibilité	25
Figure 9 : Issue Ajout de la langue française à l'internationalisation de l'application comme langue par défaut	26
Figure 10 : Liste des US du sprint 1.....	27
Figure 11 : Backlog du sprint 2 sur Azure DevOps.....	29
Figure 12 : Règles de gestion des niveaux.....	30
Figure 13 : Règles de gestion des compétences.....	30
Figure 14 : Règles de gestion des catégories.....	31
Figure 15 : Règles de gestion des collaborateurs	31
Figure 16: Planning initial - Page1	32
Figure 17: Planning initial - Page2	33
Figure 18 : Planning initial - Page3	33
Figure 19 : Planning initial - Page4	34
Figure 20 : Planning initial - Page5	34
Figure 21 : Planning réel et écarts avec le planning initial - Page1	35
Figure 22 : Planning réel et écarts avec le planning initial - Page2	36
Figure 23 : Planning réel et écarts avec le planning initial - Page3	36
Figure 24 : Planning réel et écarts avec le planning initial - Page4	37
Figure 25 : Planning réel et écarts avec le planning initial - Page5	37
Figure 26 : Diagramme de cas d'utilisation du manager sur « Visual Paradigm »	45
Figure 27 : Diagramme de cas d'utilisation du collaborateur sur « Visual Paradigm »	46

Figure 28: Diagramme de classe du sprint 1 sur « Visual Paradigm »	47
Figure 29 : Diagramme de classe du sprint 2 sur « creately.com »	48
Figure 30: Diagramme de séquence pour un scénario d'authentification sur « Visual Paradigm »	49
Figure 31: Diagramme de séquence pour un scénario d'ajout sur « Visual Paradigm »	50
Figure 32: Diagramme de séquence pour un scénario de suppression sur « Visual Paradigm »	51
Figure 33: Diagramme de séquence pour un scénario de modification sur « Visual Paradigm »	52
Figure 34: Diagramme de séquence pour un scénario de recherche sur « Visual Paradigm »	53
Figure 35: Architecture technique du projet	54
Figure 36 : schéma explicative de la dockerisation de notre application	56
Figure 37 : liste des images docker de notre projet	57
Figure 38 : tableau d'imputation et de statistiques sur Excel du sprint 1	61
Figure 39 : Page d'accueil de l'application	70
Figure 40 : Page authentifié en tant qu'administrateur	71
Figure 41 : Gestion des utilisateurs	72
Figure 42 : Gestion du profil d'un utilisateur connecté	73
Figure 43 : Liste de l'entité dégrés	74
Figure 44 : Fiche de l'entité collaborateur	75
Figure 45 : modification d'une entité collaborateur	76
Figure 46 : Ajout d'une entité collaborateur	77
Figure 47 : Pop-up de la confirmation de suppression physique d'une entité	78
Figure 48 : Affichage de la liste des compétences du sprint 1	79
Figure 49 : filtrage avec selection multiple	81
Figure 50 : fiche détaillée d'une compétence	82
Figure 51 : modification d'une compétence	83
Figure 52 : champs requis de l'ajout d'une compétence	84
Figure 53 : Erreur lors de l'ajout d'une compétence qui existe avec le même label	85
Figure 54 : Erreur interne du serveur	85
Figure 55 : liste déroulante des compétences avec le bouton d'ajout d'une nouvelle	86
Figure 56 : affichage de la liste des tags	86
Figure 57 : Page de connexion sprint 2	87
Figure 58 : Page d'accueil Manager sprint 2	88
Figure 59 : Menu Manager sprint 2	89
Figure 60 : Page gestion des collaborateurs sprint 2	90
Figure 61 : Recherche de collaborateurs par compétences sprint 2	90
Figure 62 : Fiche collaborateur sprint 2	91
Figure 63 : gestion des compétences sprint 2	92
Figure 64 : Formulaire d'enregistrement	93
Figure 65 : notification d'activation de compte	93
Figure 66 : mail d'activation de compte	94
Figure 67 : message de confirmation d'activation de compte	94
Figure 68 : message d'avertissement pour completer la fiche du collaborateur	94
Figure 69 : Ajout d'un collaborateur sprint 2	95
Figure 70 : Fichier de configuration d'entité de JHipster	102
Figure 71 : React Hooks	103
Figure 72 : Récupération et conversion des états en propriétés	103

Figure 73 : service web de l'affichage des collaborateurs.....	104
Figure 74 : fonctions utilisées pour l'affichage des collaborateurs.....	105
Figure 75 : Envoie de requêtes HTTP grâce à Axios.....	106
Figure 76 : Interdiction d'accès aux routes et aux composants liés pour ceux qui n'ont pas les droits d'administrateur	107

Sommaire

Introduction.....	9
1. Contexte général du stage.....	10
1.1. Contexte du stage.....	11
1.1.1. Carte d'identité.....	11
1.1.2. Le groupe ALLEN.....	11
1.1.3. Domaines et secteurs d'activité	12
1.1.4. Réalisations.....	13
1.1.5. Prix et reconnaissances	14
1.1.6. Organigramme.....	15
1.2. Cadre du stage	17
2. Etude générale du projet.....	18
2.1. Cahier des charges.....	19
2.1.1. Cahier de charge de gestion	19
2.1.2. Cahier de charge technique.....	20
2.2. Projet confié	21
2.3. Fiche projet (français/anglais)	22
2.4. Méthode agile « SCRUM ».....	23
2.5. Les sprints plannings.....	24
SPRINT 0 – « Préparation de l'environnement ».....	24
Sprint goal	24
Sprint backlog	24
SPRINT 1 – « Manager : Gestions des compétences »	26
Sprint goal	26
Sprint backlog	26
SPRINT 2 – « Manager : Gestions des compétences, collaborateurs, niveaux et catégories ».....	27
Sprint goal	28

Sprint backlog	28
2.6. Plannings.....	32
2.6.1. Planning initial	32
2.6.2. Planning réel.....	35
2.7. Fiches des entités (Dictionnaire de données).....	38
2.8. Les livrables	41
2.9. Les risques	42
3. Analyse et conception	43
3.1. Identification des acteurs	44
3.1.1. Définition d'un acteur.....	44
3.1.2. Acteurs.....	44
3.2. Analyse et UML des sprints	45
3.2.1. Diagrammes de cas d'utilisation généraux.....	45
3.2.2. Diagrammes de classes des sprints	46
SPRINT 0 ET SPRINT 1	47
SPRINT 2.....	48
3.2.3. Diagrammes de séquences généraux.....	49
3.2.3.1. Diagramme de séquence pour un scénario d'authentification.....	49
3.2.3.2. Diagramme de séquence pour un scénario d'ajout	50
3.2.3.3. Diagramme de séquence pour un scénario de suppression	51
3.2.3.4. Diagramme de séquence pour un scénario de modification	52
3.2.3.5. Diagramme de séquence pour un scénario de recherche	53
3.3. Architecture technique.....	54
3.4. Conception et développement des sprints	55
SPRINT 0 – « Préparation de l'environnement »	55
SPRINT 1 – « Manager : Gestions des compétences »	58
SPRINT 2 – « Manager : Gestions des compétences, collaborateurs, niveaux et catégories »	62
3.5. Enseignements/apports du stage (connaissances - compétences).....	65
4. Réalisation	66
4.1. Outils utilisés	67
4.2. Imprimés écran des sprints	70
SPRINT 0 – « Préparation de l'environnement »	70
SPRINT 1 – « Manager : Gestions des compétences »	79

SPRINT 2 – « Manager : Gestions des compétences, collaborateurs, niveaux et catégories »....	87
Conclusion et perspectives	96
Webographie	97
Glossaire	100
Liste des abréviations	101
Annexes	102

Introduction

ALTEN, leader mondial de l'Ingénierie et du Conseil en Technologies est une entreprise qui compte plus que 37 200 effectifs au niveau international dont la majorité sont des ingénieurs. Aujourd'hui, les compétences sont très diversifiées Dans le département SISE (Système d'information et système embarqué), et donc il devient difficile à chaque fois qu'il y a le besoin, de pouvoir, de manière manuelle, identifier rapidement les profils, trouver leur DT (dossier technique), que leur DT soit à jour et qu'il soit déjà passé par un cycle de validation et par la suite validé, de créer des mini CVs ainsi de suite ...

Dans ce cadre, notre mission consiste en l'étude, l'analyse, la conception et la réalisation d'une application concernant le système d'information pour la gestion des compétences.

À travers cette application, il est dorénavant possible en interne d'exécuter toutes les tâches rattachées au domaine de la gestion des compétences d'une manière automatique, rapide et efficace.

Pour ce faire, on a utilisé comme démarche la méthode agile « SCRUM » afin d'optimiser la productivité la flexibilité et la communication de l'équipe.

Dans cet ouvrage, on expose, chronologiquement, les différentes étapes de l'implémentation de notre application. Dans un premier temps, on aborde le contexte et l'étude générale du projet, ensuite on présente le cahier des charges établi, puis on met en exergue l'analyse et la conception ainsi que la phase de réalisation de notre projet.

1. Contexte général du stage

Dans cette partie, nous présentons l'établissement qui a formulé le besoin et le cadre général du stage.

1.1. Contexte du stage

1.1.1. Carte d'identité

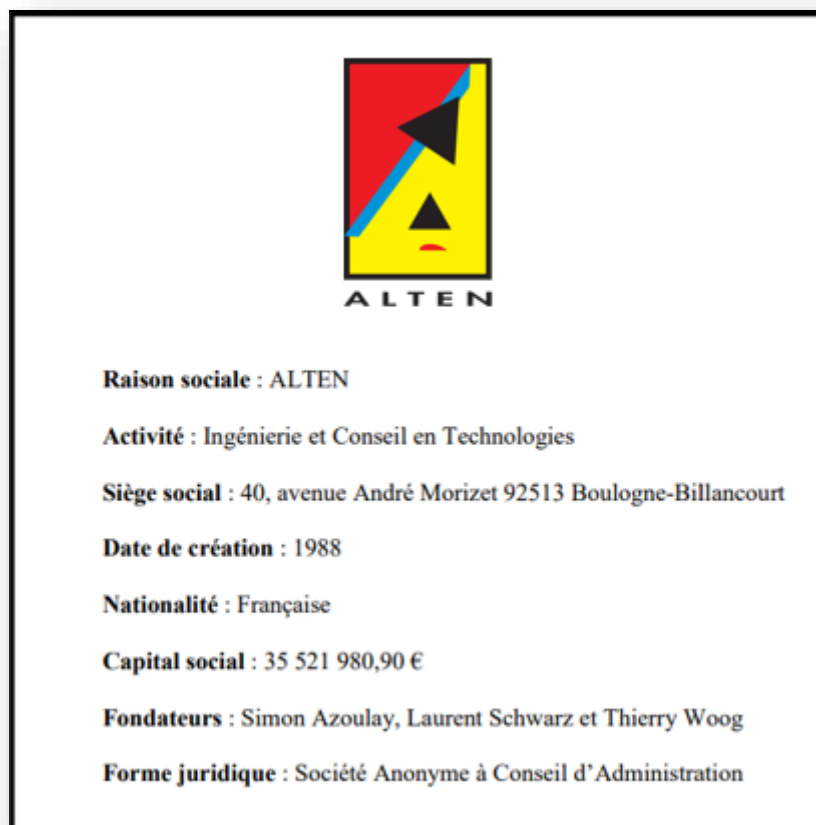


Figure 1 : Carte d'identité ALLEN

(Cette image est notre réalisation)

1.1.2. Le groupe ALLEN

Le groupe ALLEN, dont le siège social se situe à Paris en France, avec un chiffre d'Affaires de plus de 2,6 milliards d'euros en 2019 et un effectif de plus de 37200 collaborateurs aujourd'hui, est un des leaders mondiaux de l'Ingénierie et du Conseil en Technologies. Il est présent dans le monde dans plus de 25 pays et accompagne la stratégie de développement de ses clients dans les domaines de l'innovation, de la

R&D et des systèmes d'information. Il progresse, chaque année, et s'est classé au 83^{-ème} rang dans le Top 100 2019 des entreprises les plus attractives selon les étudiants des grandes écoles par « Universum » ; spécialiste dans la mesure de l'attractivité et de la marque employeur.

L'histoire du collectif ALTEN remonte en 1988 lorsque trois ingénieurs issus de grandes écoles décident de fonder l'entreprise et de l'élargir. En l'an 2000, le développement à l'international commence, ce qui a permis, en 2019, à la part de l'international d'atteindre les 56,8% du chiffre d'affaires totale.

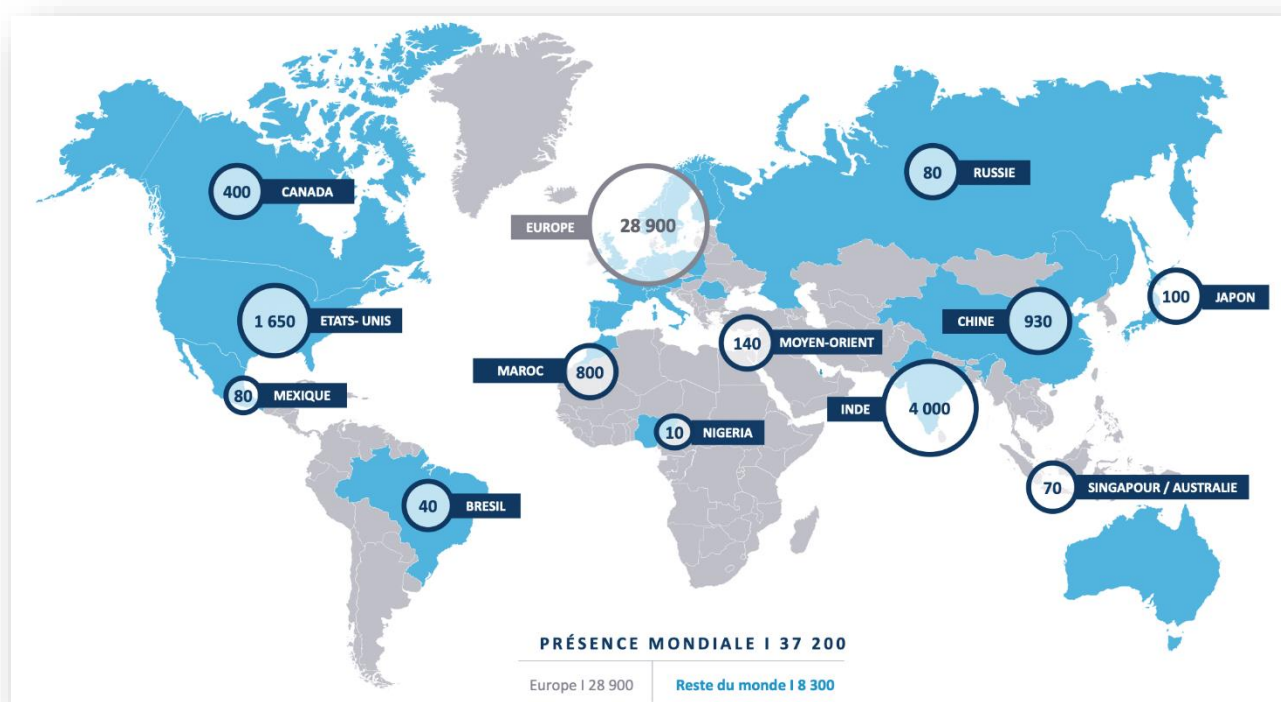


Figure 2 : Le groupe ALTEN à l'échelle internationale en 2020

(Cette image provient du site : <https://www.altenrecrute.fr/carrieres/opportunités-internationales>)

1.1.3. Domaines et secteurs d'activité

Le collectif ALTEN, dont presque 90% de son effectif sont des ingénieurs, dispose d'une équipe professionnelle avec des conseillers et des innovateurs de confiance qui

croient en leurs talents de réalisation de projets ambitieux et complexes et assurent les meilleures solutions à leurs clients. Ses missions sont multiples. Une liste non exhaustive de ses secteurs ou domaines de compétence se présente comme suit :

- Aéronautique ;
- Spatial ;
- Défense & Sécurité ;
- Naval ;
- Automobile ;
- Ferroviaire ;
- Telecoms & Medias ;
- Énergie & Environnement ;
- Sciences de la vie ;
- Banque, Finance & Assurance ;
- Retail & Services...

1.1.4. Réalisations

Le collectif ALTEN a déjà réalisé plusieurs activités à l'échelle mondiale ; notamment, des projets ambitieux et complexes pour leur clients, des missions d'expertises ainsi que plusieurs solutions. Ses réalisations sont multiples et de domaines diversifiés, avant de citer celles du domaine IT (Technologie de l'information ou aussi appelé système informatique) qui est notre domaine, voici d'abord quelques-unes des autres domaines :

- *Gestion de projet industriel pour un programme de sous-marin ;*
- *Avion de combat multi-rôle : adaptation à l'exportation ;*
- *Assemblage, Intégration et Test (AIT) des satellites ;*
- *Aménagement et conception de mini-satellites ;*
- *Conception mécanique et aérodynamique du futur lanceur européen ;*
- *Roulage Numérique : test des véhicules autonomes ;*

- *Développement de la nouvelle génération de système d'aide à la conduite (ADAS) basé sur caméra ;*
- *Modernisation d'une usine de fabrication de camions en Amérique du Nord ;*
- *Métro automatique : développement du système CBTC ;*
- *Maintenance et modernisation du parc nucléaire en exploitation...*

Réalisations dans le domaine IT :

- *Développement d'une application Streaming TV multi-devices ;*
- *Une approche Agile pour le développement d'applications e-gouvernement ;*
- *Analyse prédictive de rentabilité des campagnes publicitaires Retail ;*
- *Développement du canal e-commerce pour un constructeur automobile ;*
- *Améliorer la performance commerciale des vendeurs avec le Big Data ;*
- *Développement d'un assistant vocal intelligent multi-services ;*
- *Analyse prédictive du « Churn » avec l'approche Big Data...*

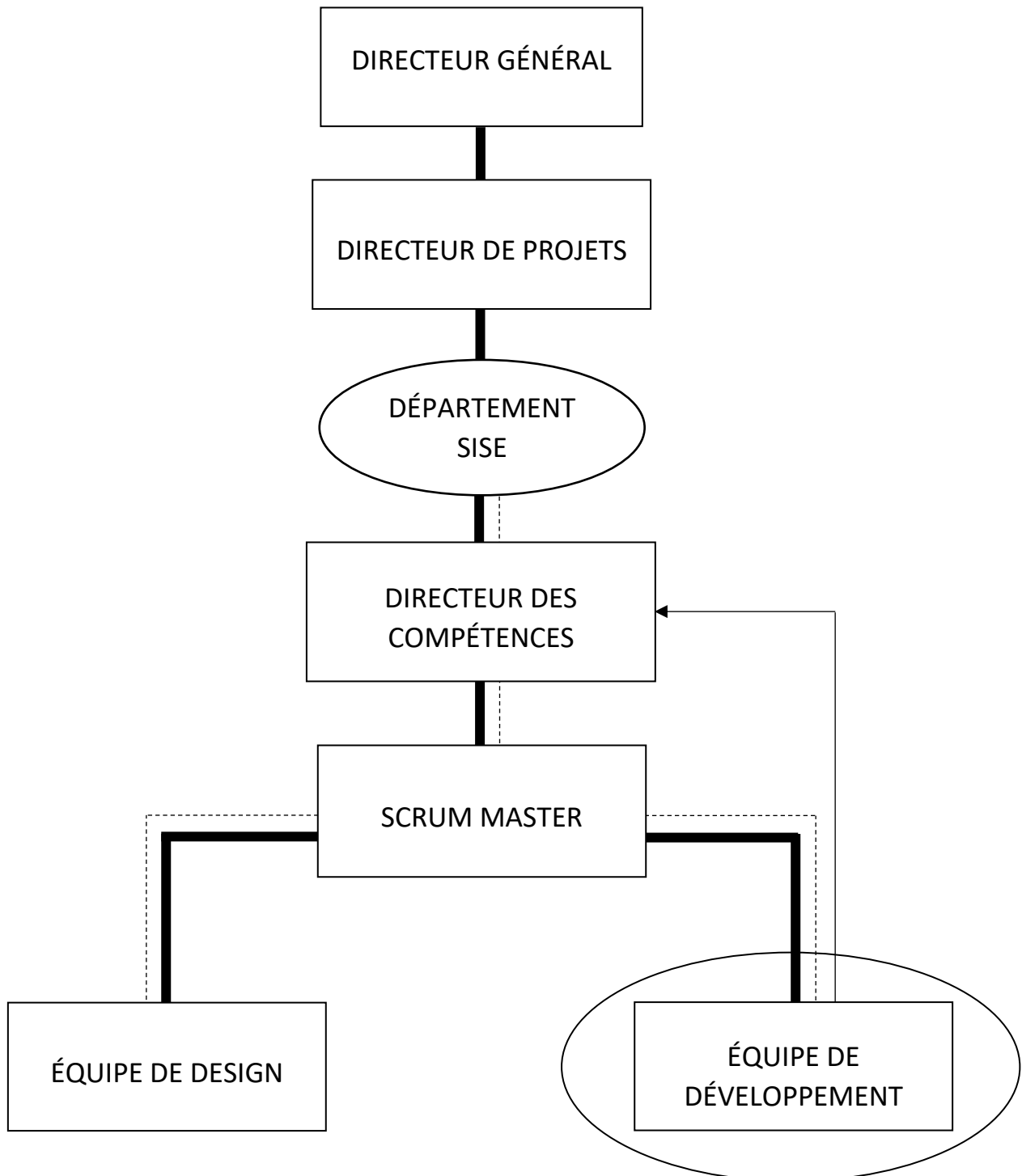
1.1.5. Prix et reconnaissances

Grâce à ces nombreuses réalisations et à sa responsabilité de l'entreprise, le groupe ALTEN a reçu plusieurs prix et reconnaissances :

- **Best Cooperation Partner Award** pour sa capacité à accompagner HUAWEI dans sa course à l'innovation et à lui apporter son expertise spécifique dans le déploiement de sa nouvelle ligne de produit d'équipement radio ;
- **HappyTrainees** en 2018 avec plus de 88 % de stagiaires satisfaits de leur expérience au sein du Groupe ;
- **CDP (Carbon Disclosure Projet)** en 2018 le Groupe obtient la note A- en reconnaissance de son leadership sur le sujet climat ;
- **Top Employer©** pour la 8ème année consécutive, ALTEN a reçu le label Top Employer© France dans 4 pays d'Europe, décerné par le Top Employers Institute ;
- **Best Place to Work 2019** ALTEN Maroc dans le TOP 5 des meilleurs employeurs au Maroc...

1.1.6. Organigramme

Voici l'organigramme de la société ALTEN DELIVERY CENTER MAROC représentant ses départements. En tant que stagiaires, on se situe dans le département SISE « systèmes d'information systèmes embarqués », et nous développons l'application pour la direction des compétences.



--- : liaison fonctionnelle.

— : liaison hiérarchique.

○ : Où on se situe.

→ : notre client.

Figure 3 : Organigramme d'ALLEN DELIVERY CENTER MAROC à Fès

(Cette image est notre réalisation)

1.2. Cadre du stage

Dans le cadre des activités d'ALTEN DELIVERY CENTER MAROC, l'identification des compétences selon un ensemble de critère, la création des DT (dossiers techniques), leur validation et mise à jour, la création de mini CVs, et connaître la disponibilité des consultants avec la prédiction de leurs congés occupe une place prépondérante. Ceci est dû essentiellement à :

- La diversification des nombreuses compétences.
- Le grand nombre des collaborateurs qui ne cesse d'augmenter chaque année.
- Le besoin de consulter les collaborateurs avec leurs formations, projets et compétences.

Dans cette perspective, une démarche qualité est amorcée ; dotant les collaborateurs et les managers de formulaires qualité ; afin de mieux cerner la collaboration aux projets, les formations effectuées et les compétences actuels.

Une application informatique se présente comme une solution opportune pour consolider les différentes informations affluentes de toute la gestion des compétences.

Il s'agit donc de développer une application interne concernant le système d'information pour la gestion des compétences.

Le but est donc de produire des informations fiables, disponibles en temps réel facilitant la prise de décisions.

2. Etude générale du projet

Dans cette partie, nous présentons le cahier des charges, la planification du stage, le projet confié, les livrables, les objectifs des sprints, les fiches des entités et les risques du stage ainsi que la démarche adoptée pour atteindre les objectifs.

2.1. Cahier des charges

2.1.1. Cahier de charge de gestion

Soit la société « ALTEN DELIVERY CENTER MAROC » souhaitant améliorer sa gestion des compétences pour la faciliter et offrir une meilleure fluidité et accessibilité. Elle veut instaurer un système capable de fournir des informations exactes, à jour et rapidement.

L'application doit proposer différentes interfaces. Une interface destinée aux managers et la seconde aux collaborateurs. L'utilisateur doit s'authentifier pour être redirigé vers l'interface qui lui correspond.

Le manager doit pouvoir identifier des compétences selon un ensemble de critère. Il doit avoir la possibilité de générer automatiquement le ou les dossier(s) technique(s) détaillé(s) du ou des collaborateur(s) choisi(s) et de le(s) valider grâce à l'intégration d'un workflow de validation. Il doit aussi être capable de générer automatiquement le ou les mini CV(s) du ou des collaborateur(s) choisi(s) à partir des dossiers techniques détaillés. Le manager doit également pouvoir gérer les collaborateurs (les visualiser, les rechercher par des critères, les ajouter ou désactiver, valider leur mise à jour et gérer leurs rôles), connaître la disponibilité des consultants ainsi que le taux d'inter contrat, générer des références projets, gérer les équipes, gérer les projets, gérer les équipes, gérer les prévisions des congés, gérer les formations, gérer les AVV (avant ventes) et générer des rapports.

Le collaborateur doit pouvoir gérer ses propres informations dont la gestion de ses compétences, de ses projets, de ses formations, de son équipe et de son profil. Il doit aussi avoir la possibilité de s'affecter (ou se désaffecter) à un projet, à une formation et à une compétence. Il doit également disposer d'une fonctionnalité lui permettant de générer son DT et son mini CV.

(Pour les règles de gestion détaillées à voir les user stories des sprints)

2.1.2. Cahier de charge technique

L'application web dispose d'un serveur qui stocke toutes les informations relatives à la gestion des compétences, du collaborateur, du manager, de leurs actions, ainsi que leurs comptes. Le SGBD (Système de Gestion de Base de Données) de l'application est PostgreSQL.

L'intégralité de l'application est générée par JHipster.

La gestion de la base de données s'effectue en **Java** grâce au framework backend open source **spring-boot** qui exécute le code Java côté serveur. La connexion à la base de données s'effectue dans des fichiers de configuration de l'application.

La récupération des données de la base de données se fait par l'envoi d'une requête http à l'adresse du serveur par l'application. Les données sont extraites, et converties en **JSON** grâce aux services web REST du backend. Elle traite donc les données au format **JSON** qui est un format de données facilement manipulable par le JavaScript.

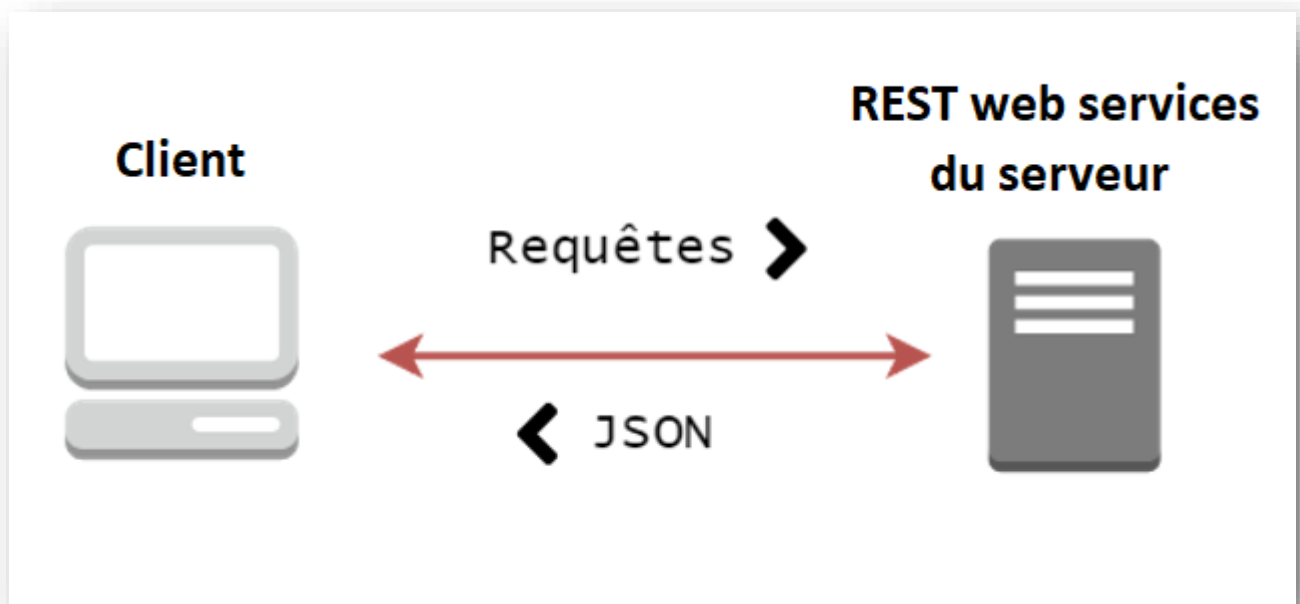


Figure 4: Architecture REST

(Cette image est notre réalisation)

La connexion au serveur est immédiatement rompue une fois que la totalité des données de la réponse associée à la requête http soit récupérée. Ce procédé est répété à plusieurs instants, lorsque l'application a besoin d'actualiser les données.

Le coté frontend de l'application est réalisé avec des technologies de développement web TypeScript, REACT (javascript), HTML et SASS. En effet, l'application repose sur la librairie javascript REACT, qui permet de concevoir des interfaces utilisateur.

Pour une meilleure mobilité, l'environnement de l'application est dockerisé et son lancement ne repose que sur 2 commandes.


L'application est monopage (SPA). Elle est accessible à travers une page web unique pour éviter le chargement de nouvelles pages.

2.2. Projet confié

Lors, de notre stage on nous a confié le projet suivant :

_ Développement d'un projet interne concernant le système d'information pour la gestion des compétences.

2.3. Fiche projet (français/anglais)

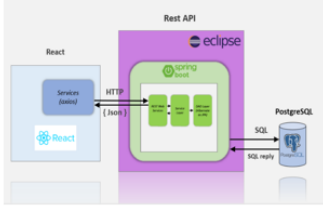


Projet Interne – système d’information pour la gestion des compétences

Département des compétences

A PROPOS

« Système d’information pour la gestion des compétences » est un outil conçu pour l’identification des compétences selon des critères bien définis. Cet outil permettra la génération automatique des dossiers techniques détaillés et des mini-CVs. Il a également pour objectif la fluidification de la gestion des prévisions des congés, des formations et des avant ventes. De plus, la direction compétence ALTEN bénéficiera de la mise à disposition des informations concernant la disponibilité des consultants et du taux d’inter-contrat.



ACTIVITES ALTEN

Analyse & Conception

- ✓ Analyse et cadrage du besoin
- ✓ Conception du projet
- ✓ Définition des fonctionnalités avec les règles de gestion
- ✓ Chiffrage des users stories

Développement

- ✓ Préparation de l’ environnement de travail
- ✓ Développement des fonctionnalités
- ✓ Modification d’une fonctionnalité existante
- ✓ Développement des interfaces responsive

test

- ✓ Ecriture des tests unitaires

Déploiement

- ✓ Préparation de déploiement
- ✓ Déploiement

Méthodologie Agile

- ✓ SCRUM

OUTILS & TECHNOLOGIES

- Azure DevOps (Outil pour planification)
- Creately (Outil pour dessiner des diagrammes)
- JHIPSTER (plate-forme de développement)
- Sprint boot (Framework java)
- React js/ PrimeReact (Framework javascript)
- PostgreSQL (Base de données SQL)
- Docker (Outil de Développement continue)
- GIT, Azure DevOps (Outils pour la versionning)

Département
Software & Outils

+ 3 Mois

2 ETP

FEZ

Figure 5 : Fiche projet - système d'information pour la gestion des compétences

(Cette image est notre réalisation)

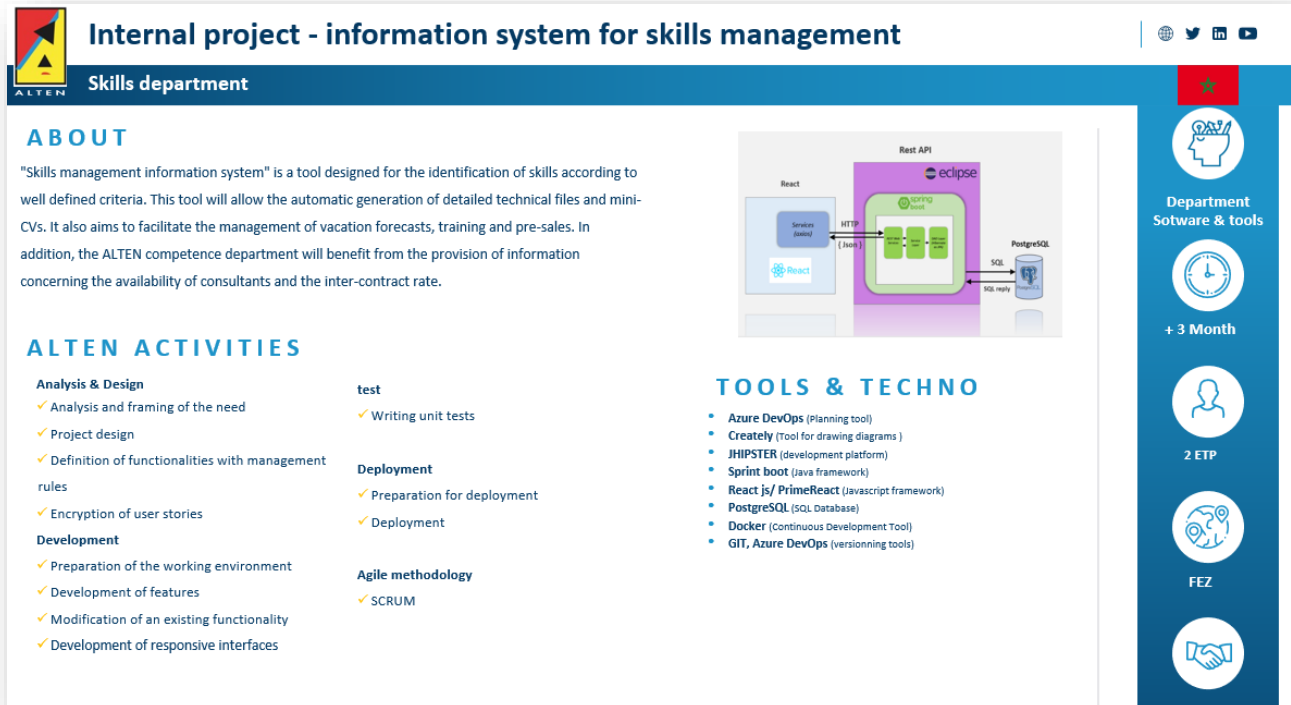


Figure 6 : Project sheet - information system for skills management

(Cette image est notre réalisation)

2.4. Méthode agile « SCRUM »

Scrum est la méthode agile qu'on utilise dans ce projet et aussi un framework qui aide les équipes à travailler ensemble. Tout comme une équipe de rugby (d'où est tiré son nom) s'entraînant pour le grand match, Scrum encourage les équipes à apprendre par des expériences, à s'auto-organiser tout en travaillant sur un problème, et à réfléchir sur leurs victoires et leurs pertes pour s'améliorer continuellement.

Bien que le Scrum dont on parle soit le plus fréquemment utilisé par les équipes de développement logiciel, ses principes et ses leçons peuvent être appliqués à toutes sortes de travail d'équipe. C'est l'une des raisons pour lesquelles Scrum est si populaire. Souvent considéré comme un cadre de gestion de projet agile, Scrum décrit un ensemble de réunions, d'outils et de rôles qui travaillent de concert pour aider les équipes à structurer et à gérer leur travail.

L'un des grands principes de scrum est la répartition d'un grand projet en sprints, ce qui permet de présenter des livrables tout au long du développement à la fin de

chaque sprint. Un sprint dur, en général, entre 2 et 4 semaines. Avant le développement et la réalisation de chaque sprint il y a le sprint planning qui est composé d'un sprint goal qui résume l'objectif du sprint et d'un sprint backlog qui contient le plan et les tâches ainsi que toute la procédure pour achever le sprint, en général il contient les user stories avec leurs règles de gestion et leur chiffrage.

2.5. Les sprints plannings

SPRINT 0 – « Préparation de l'environnement »

Le sprint 0 est fondamental et important car il représente la bonne base de tout projet IT. Ses objectifs ou missions sont :

Sprint goal

- _ Dockerisation de l'environnement (Frontend & Backend).
- _ Génération de l'application avec ses entités, classes, DAO, CRUDs, services web REST et interfaces à l'aide de JHipster.

Sprint backlog

Le backlog du sprint 0 est sous forme d'issues dans la plateforme github, où notre encadrant nous décrit la tâche qu'on doit réaliser dans une nouvelle branche et après on lui partage le code grâce à un pull request de cette dernière à la branche de développement du projet pour le vérifier et valider par la suite.

Voici les différentes issues du sprint 0 :

Modèle de données + Generation des CRUD des entités coté backend et frontend à partir du JDL #1

Closed

opened this issue on 27 Jul · 0 comments

commented on 27 Jul · edited by TayebBou

Générer le JDL et la couche DAO des modèles suivant :

- Collaborator (firstName (String), lastName (String), beginDate (Date), avatar (String base64 encoded image), department (Relation ManyToOne), skills (Relation ManyToMany), projets (Relation ManyToMany), degrees (Relation OneToMany with Degree))
- Degree (label (String), description(Text), graduationYear (Date))
- Department (label (String), description (Text), code (String), manager (OneToOne with collaborator), collaborators (OneToMany with collaborators))
- Skills (label (String), category (Relation ManyToMany with Category), code (String), description (Text))
- Category (label (String), code (String), description (Text))
- Project (label (String), description(Text), client (String), startDate (Date), endDate (Date), department (Relation ManyToOne), manager (Relation OneToOne with collaborator), projectManager (Relation OneToOne with collaborator), tl (Relation OneToOne with collaborator), skills (Relation ManyToMany))

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merged issue.

None yet

Figure 7 : Issue de génération des entités avec leur CRUD

(Cette image est notre réalisation)

Optimisation de la gestion de toutes les entités #10

Closed

TayebBou opened this issue on 3 Aug · 0 comments



TayebBou commented on 3 Aug · edited

Au niveau de chaque gestion quand on a une entité qui a une relation avec une autre, la sélection doit contenir le label ou la concaténation du firstName et lastName au lieu de l'id pour plus de visibilité.

Figure 8 : Issue changement de la selection des relations des entités pour une meilleure visibilité

(Cette image est notre réalisation)



Figure 9 : Issue Ajout de la langue française à l'internationalisation de l'application comme langue par défaut

(Cette image est notre réalisation)

SPRINT 1 – « Manager : Gestions des compétences »

Pour le premier sprint, on commence par déterminer ses objectifs et ses user stories pour en déduire les tâches qu'on priorise et estime en groupe avec le chiffrage de Fibonacci grâce à un vote collectif.

Sprint goal

L'objectif de ce sprint est la réalisation en partie de la gestion des compétences pour le manager.

Sprint backlog

Dans le sprint backlog, on détermine les user stories avec leur acteur, cas d'utilisation, étapes, règles de gestion, remarques, priorité et estimation.

Pour l'estimation on utilise le chiffrage de Fibonacci qui inclut la suite d'entiers suivante : 1 2 3 5 8 13 21 34 ...

La suite représente pour nous la complexité du travail, on suppose qu'un collaborateur peut réaliser 5 points de complexité par jour.

On organise alors un vote, chaque collaborateur propose un nombre appartenant à la suite et à la fin on calcule la moyenne pour ensuite fixer l'estimation de la tâche avec la séquence la plus proche.

On présente si dessous la liste des user stories du sprint avec les estimations des tâches :

ID	Acteur	Cas d'utilisation	Etapas	RG	remarques	Priorité	Estimation
4	Manager	Gestion des compétences	Visualisation liste complète Compétences	1-Liste contient : Label, Code, Description. 2-Prévoir la pagination de la liste. 3-Prévoir tri et filtres sur toutes les colonnes. 4-Chaque ligne de la liste est un lien vers la Fiche compétence.			13
			Visualiser Fiche (Détails) compétence	Metadata voir sheet Fiche Compétence			8
			Ajouter nouvelle compétence	1-ce lien doit être affiché dans l'ecran Comptences et à côté de toutes les listes déroulantes Compétences pour les autres pages. 2-On ne peut ajouter une compétence avec les mêmes labels (Ignorer la casse)			13

Figure 10 : Liste des US du sprint 1

(Cette image est notre réalisation)

SPRINT 2 – « Manager : Gestions des compétences, collaborateurs, niveaux et catégories »

Après la démo du sprint 1, le travail ne collait pas aux attentes de notre client, donc il a fallu refaire toute la conception et commencer depuis la branche master. Cette itération est dû à cause du client qui n'était pas disponible pour valider l'étude fonctionnelle.

Dans le deuxième sprint, on est passé à un outil professionnel pour la planification des tâches qu'on avait gratuitement grâce à nos comptes ALLEN qui est Azure DevOps. Cet outil nous permet une bonne organisation car en plus que les tâches soient affichées par sprint, estimé et assigné à un ou plusieurs collaborateurs, on peut taguer le client dans les tâches ce qui lui permet d'avoir un visuel et un suivi du travail en direct ainsi que la possibilité de commenter les tâches. Donc pour ce

sprint, on détermine ses objectifs et son backlog à savoir le plan et les user stories accompagné par leurs règles de gestion chiffrées grâce à Azure DevOps.

Sprint goal

Le sprint 2 a comme objectif de recommencer l'application depuis la branche master, de refaire la conception et de proposer une première version des gestions suivantes : la gestion des collaborateurs, des compétences, des catégories et des niveaux pour répondre aux attentes du client.

Sprint backlog

Pour ce sprint, on définit les gestions et tâches sur Azure DevOps avec leur estimation par le même procédé utilisé pour le backlog de celui d'avant et on expose dans les figures suivantes ses user stories avec les estimations de leurs règles de gestion :

Skills pro Team

Taskboard Backlog Capacity Analytics + New Work Item Column Options ...

Order	Title	State	Assigned To	Remaining Work
1	Gestion collaborateur	New		55
	CRUD collaborateur	Done	Tayeb BOUSFIHA	34
	Front End & Intégration responsive - Gestion collaborateur	In Progress		21
2	Gestion des niveaux	New		8
	CRUD Gestion des niveaux	Done	Tayeb BOUSFIHA	
	Front End & Intégration responsive - Gestion des niveaux	To Do		8
3	Gestion de compétence	New		8
	Gestion de compétence	Done	Tayeb BOUSFIHA	
	Front End & Intégration responsive - Gestion de compète...	To Do		8
4	Gestion des catégories	New		8
	CRUD Gestion des catégories	Done	Tayeb BOUSFIHA	
	Front End & Intégration responsive Gestion catégories	To Do		8

Figure 11 : Backlog du sprint 2 sur Azure DevOps

(Cette image est notre réalisation)

TASK 162

162 CRUD Gestion des niveaux

TB Tayeb BOUSFIHA 0 comments Add tag

State	● Done	Area	Skills pro
Reason	🔒 Work finished	Iteration	Skills pro\Sprint 3

Description

1. un manager peut ajouter un ou plusieurs niveaux.
2. un manager peut modifier ou supprimer un ou plusieurs niveaux.
3. Seul le manager qui peut ajouter, modifier ou supprimer un niveau.
4. un niveau est unique.

Details

Priority
2

Remaining Work
🔒

Activity
Development

Blocked
🔒

Discussion

TB Add a comment. Use # to link a work item, ! to link a pull request, or @ to mention a person.

Figure 12 : Règles de gestion des niveaux

(Cette image est notre réalisation)

TASK 159

159 Gestion de compétence

TB Tayeb BOUSFIHA 0 comments Add tag

State	● Done	Area	Skills pro
Reason	🔒 Work finished	Iteration	Skills pro\Sprint 3

Description

Les compétences sont géré par le manager

1. un manger peut ajouter un ou plusieurs compétences.
2. un manager peut visualiser les compétences existantes.
3. un manager peut modifier ou supprimer une compétence.
4. Une compétence est unique.
5. On ne peut pas supprimer une compétence en cours d'utilisation.

Details

Priority
2

Remaining Work
🔒

Activity

Blocked
🔒

Discussion

TB Add a comment. Use # to link a work item, ! to link a pull request, or @ to mention a person.

Figure 13 : Règles de gestion des compétences

(Cette image est notre réalisation)

TASK 170

170 CRUD Gestion des catégories

Tayeb BOUSFIHA 1 comment Add tag

State	Done	Area	Skills pro
Reason	Work finished	Iteration	Skills pro\Sprint 3

Description

- Le manager peut ajouter une ou plusieurs catégorie(s).
- Le manager peut modifier ou supprimer une ou plusieurs catégorie(s).
- Seul le manager qui peut consulter, ajouter, modifier ou supprimer une catégorie(s).
- une catégorie est unique.

Details

Priority
2

Remaining Work
🔒

Activity
Development

Blocked
🔒

Discussion

TB Add a comment. Use # to link a work item, ! to link a pull request, or @ to mention a person.

Figure 14 : Règles de gestion des catégories

(Cette image est notre réalisation)

TASK 156*

156 CRUD collaborateur

Tayeb BOUSFIHA 0 comments Add tag

State	Done	Area	Skills pro
Reason	Work finished	Iteration	Skills pro\Sprint 3

Description

Le manager s'occupe de valider les actions faites par les collaborateurs.

- Le collaborateur peut s'inscrire dans l'application directement sans l'aide de manager pour enrichir son profil après l'activation de son compte.
- Le manager peut activer ou désactiver un compte.
- Le manager peut visualiser les profils existants.
- Les collaborateurs sont paginés.
- Le manager peut chercher les profils par compétences.
- Le manger peut visualiser la fiche de collaborateur choisie.

Details

Priority
1

Remaining Work
🔒

Activity
Development

Blocked
🔒

Discussion

TB Add a comment. Use # to link a work item, ! to link a pull request, or @ to mention a person.

Figure 15 : Règles de gestion des collaborateurs

(Cette image est notre réalisation)

2.6. Plannings

2.6.1. Planning initial

Les figures suivantes représentent le planning prévisionnel et initial du projet dans lequel on a opté depuis le début pour la méthode agile SCRUM :

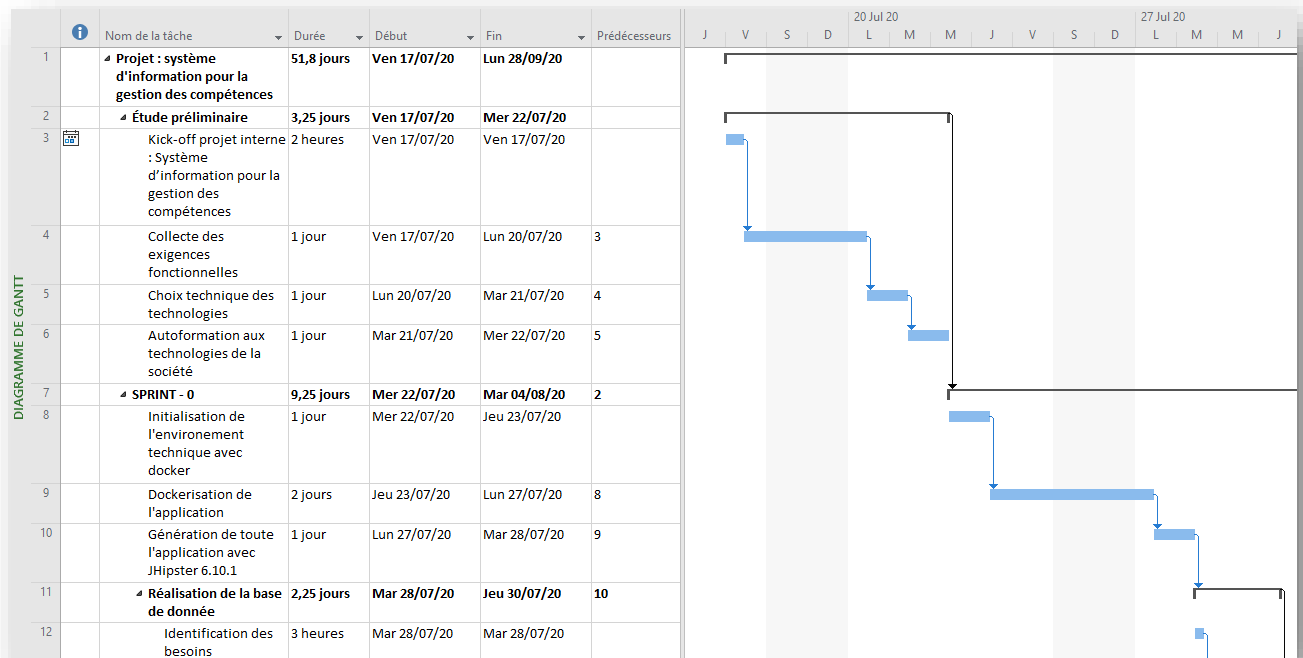


Figure 16: Planning initial - Page1

(Cette image est notre réalisation)

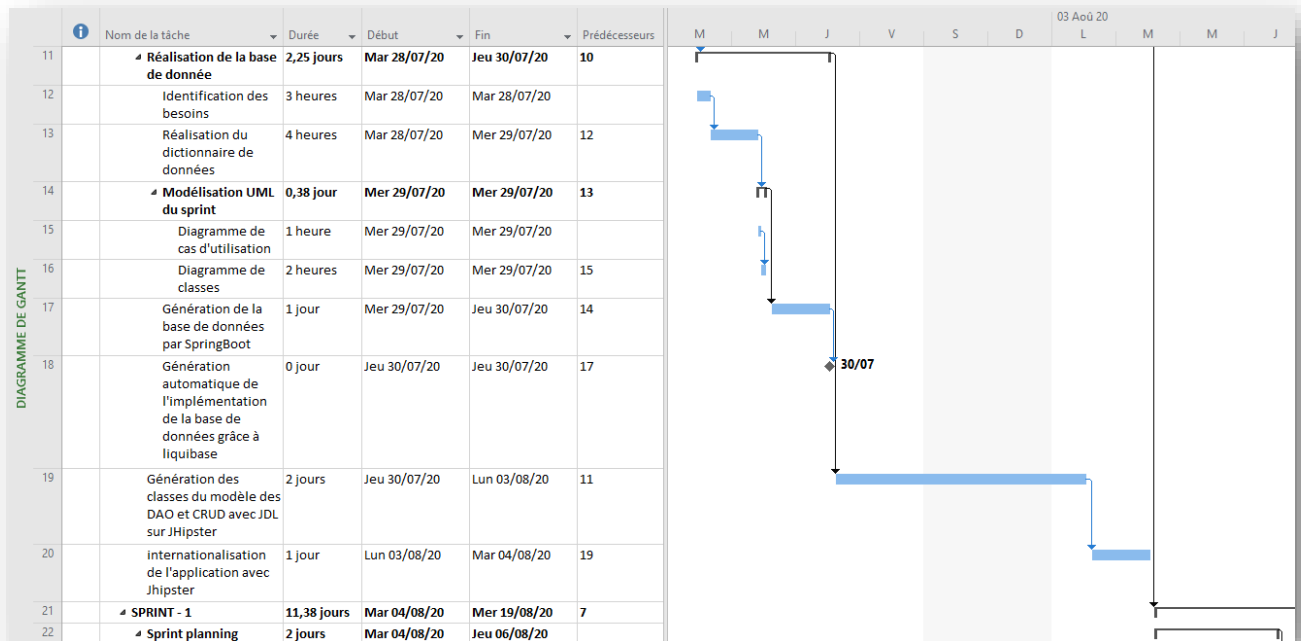


Figure 17: Planning initial - Page2

(Cette image est notre réalisation)

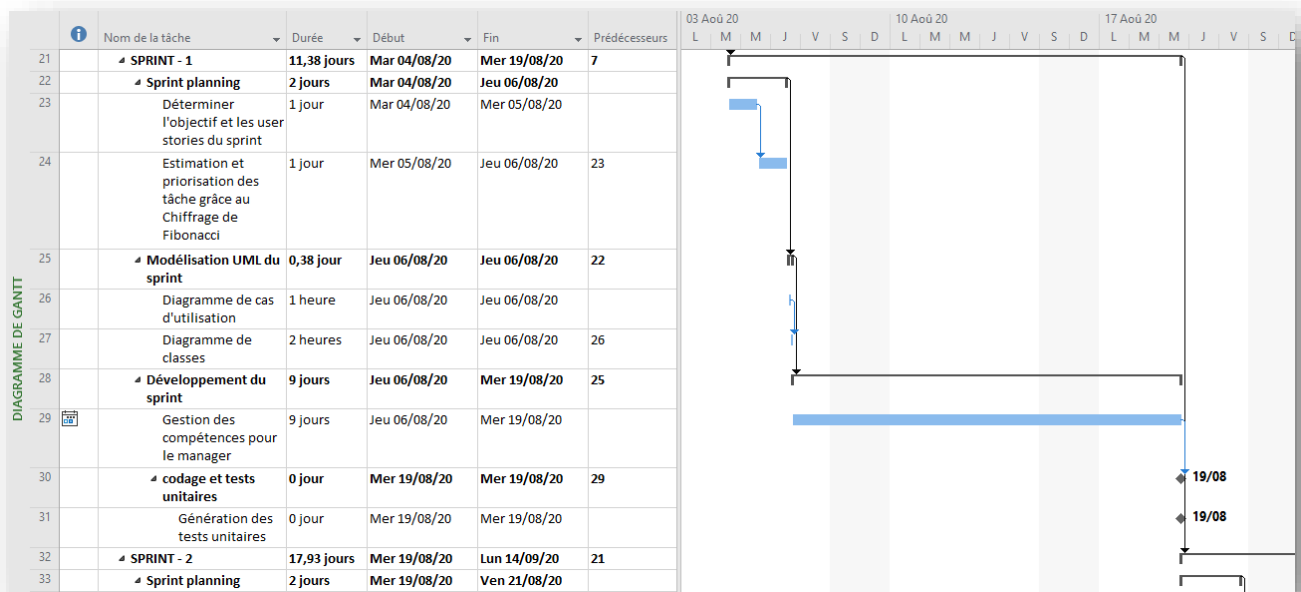


Figure 18 : Planning initial - Page3

(Cette image est notre réalisation)

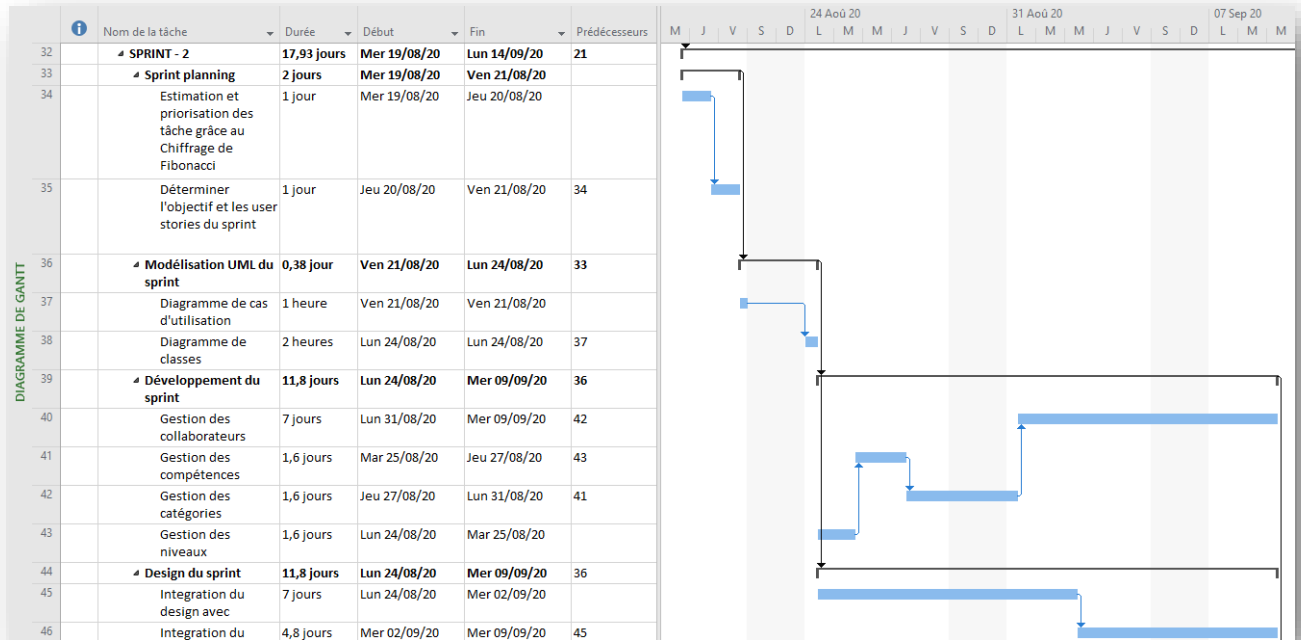


Figure 19 : Planning initial - Page4

(Cette image est notre réalisation)

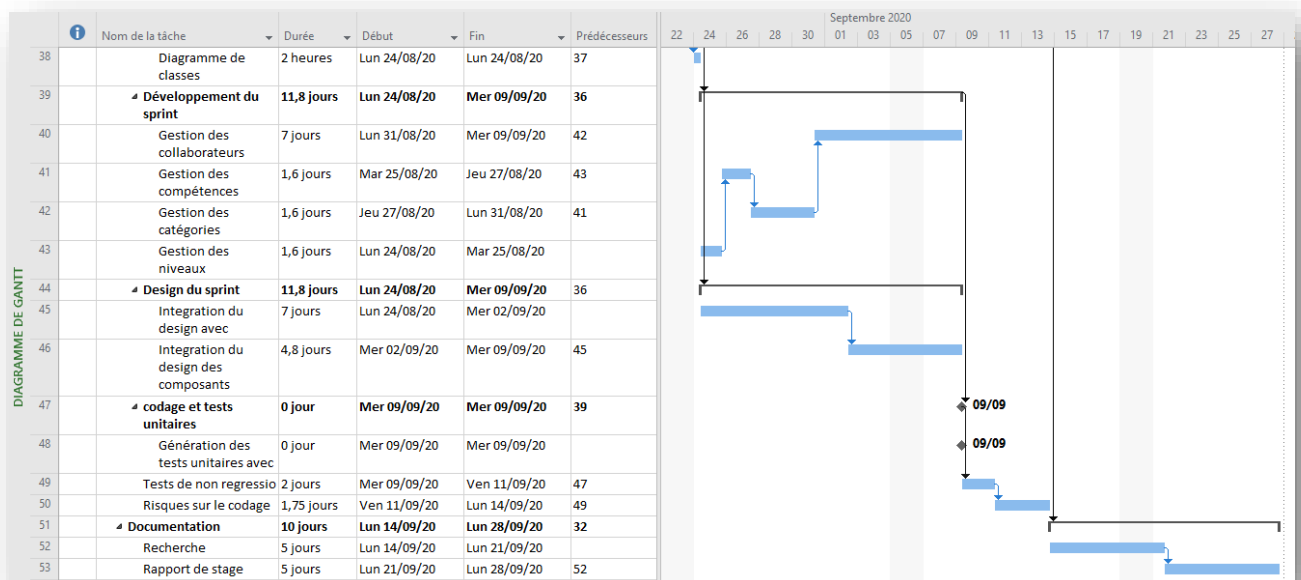


Figure 20 : Planning initial - Page5

(Cette image est notre réalisation)

2.6.2. Planning réel

En vue des itérations que notre projet a encourues et de quelques non-disponibilités du client et du SCRUM master le planning réel constate un retard par rapport au planning prévisionnel. Les figures ci-dessous illustrent l'écart entre ces plannings.

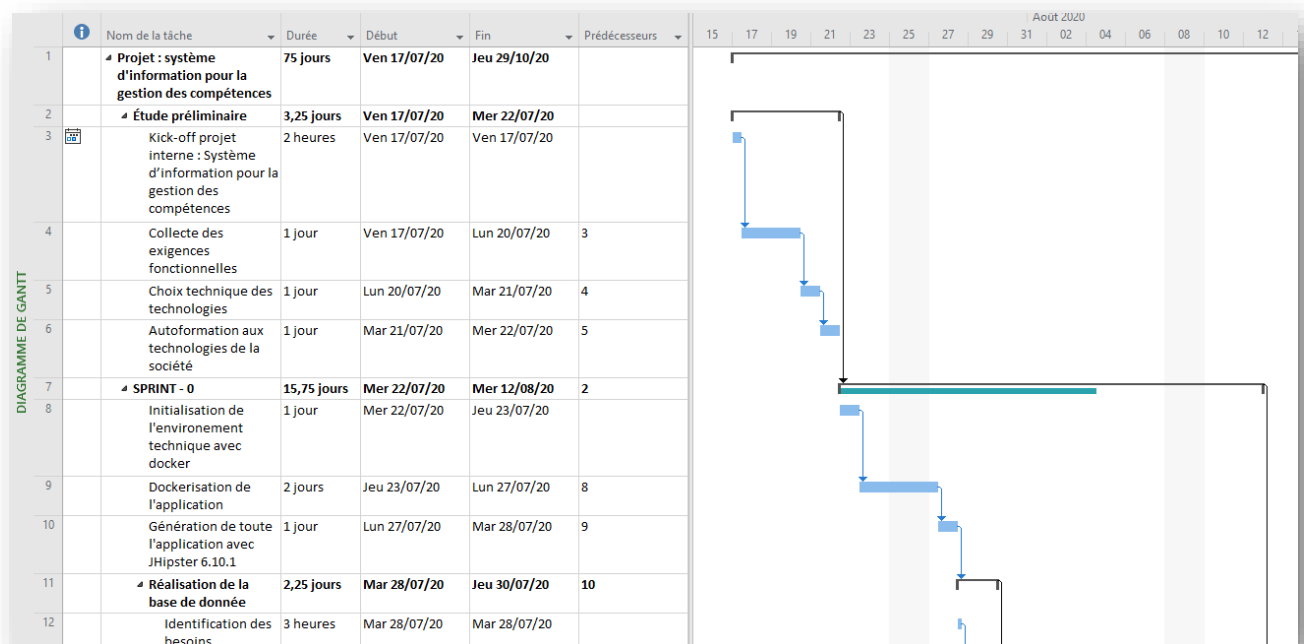


Figure 21 : Planning réel et écarts avec le planning initial - Page1

(Cette image est notre réalisation)

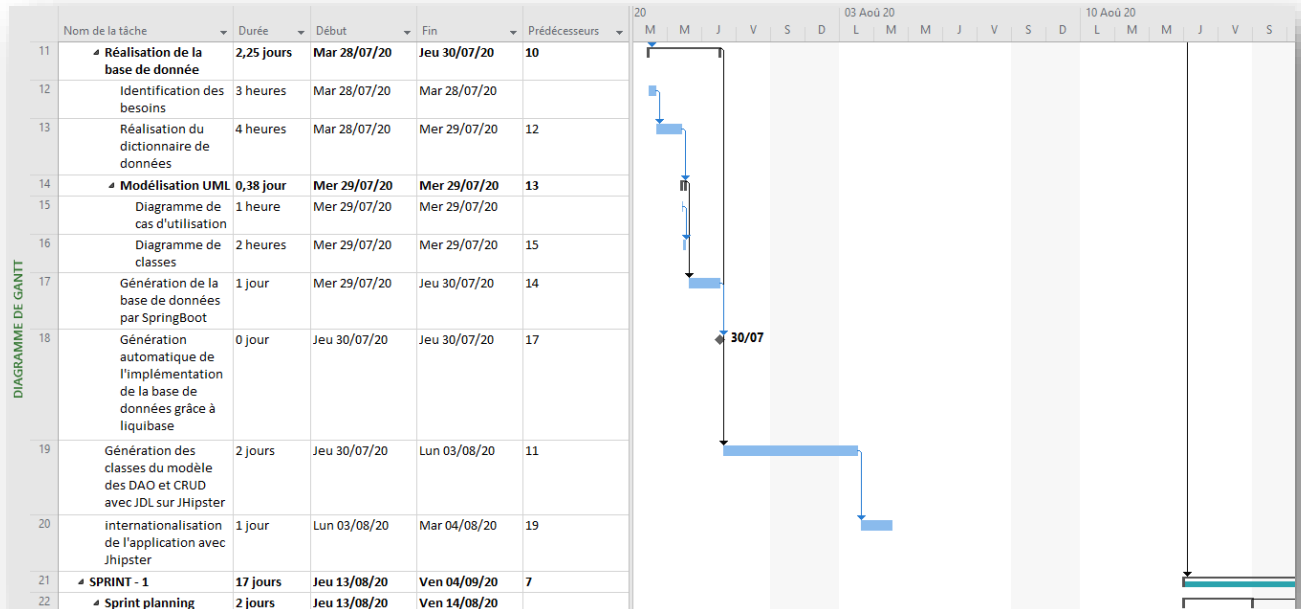


Figure 22 : Planning réel et écarts avec le planning initial - Page2

(Cette image est notre réalisation)

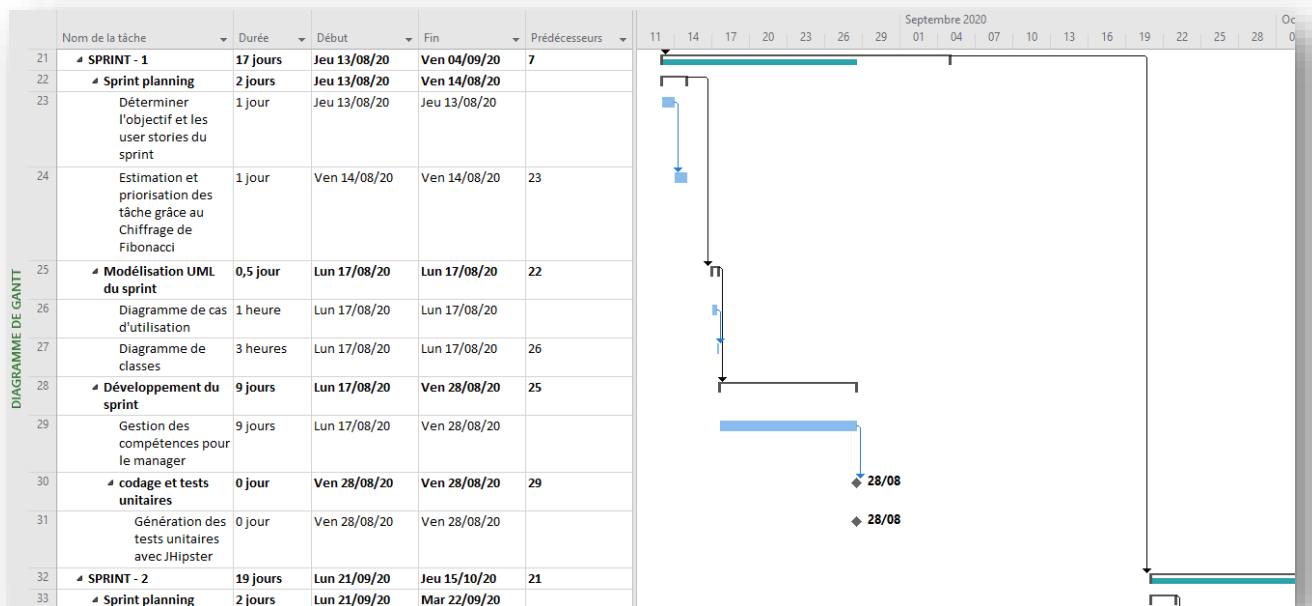


Figure 23 : Planning réel et écarts avec le planning initial - Page3

(Cette image est notre réalisation)

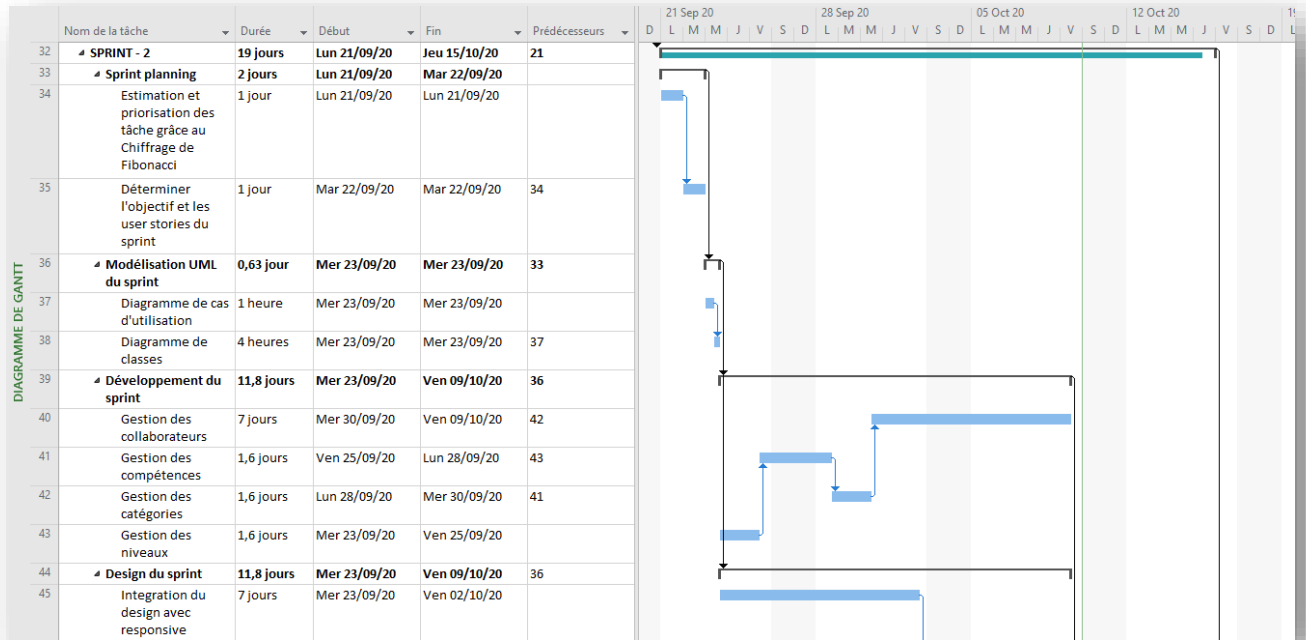


Figure 24 : Planning réel et écarts avec le planning initial - Page4

(Cette image est notre réalisation)

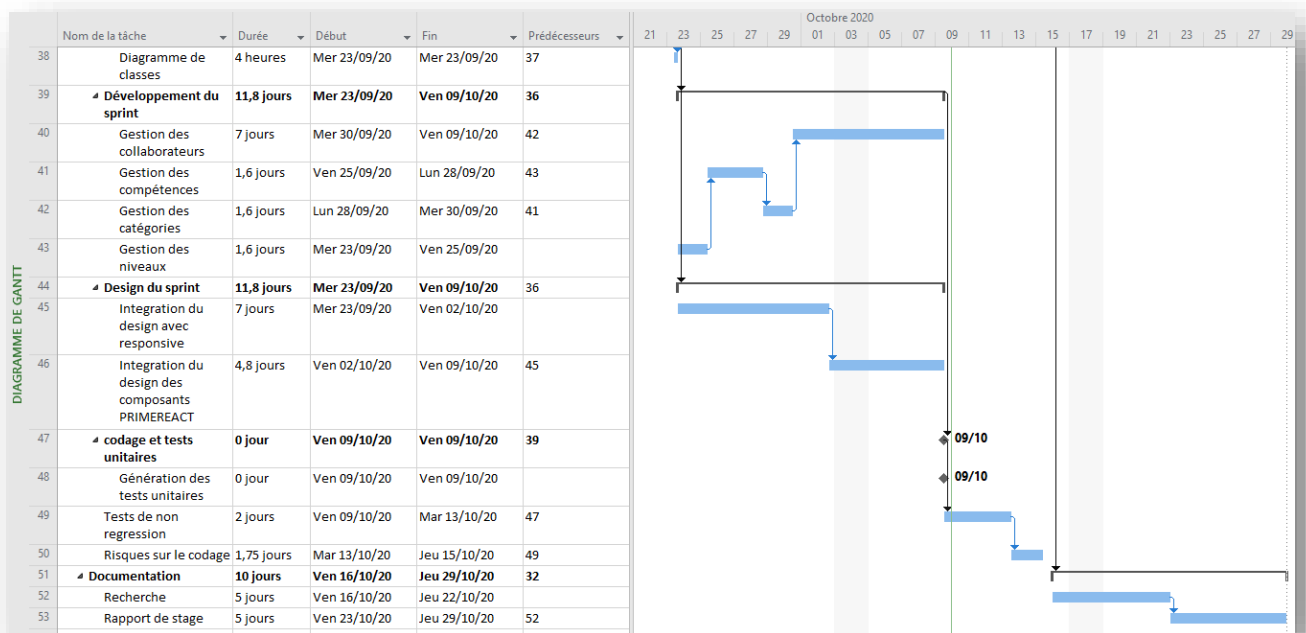


Figure 25 : Planning réel et écarts avec le planning initial - Page5

(Cette image est notre réalisation)

Dans le premier point, on a été retardé à cause de la non disponibilité du client. En effet, l'itération nous a retardé faisant un effet boule de neige sur les tâches suivantes. Dans le second point, on constate que le risque d'itération prévu a été amorcé grâce à Azure DevOps et à plusieurs réunions organisées pour permettre une bonne communication qui assure la bonne compréhension et atteinte des objectifs du client.

2.7. Fiches des entités (Dictionnaire de données)

Entité : Collaborator (Collaborateur)		
Attribut		
Nom	Description	Type
ID	ID du collaborateur	Long
RegisterNumber	Matricule du collaborateur	String
FamilySituation	Situation familiale du collaborateur	Enumeration
Phone	Numéro de téléphone du collaborateur	String
DateOfBirth	Date de naissance du collaborateur	Date
StartDate	Date de début où le collaborateur a rejoint la société	Date
Avatar	Photo du collaborateur	Base64
LineManager	Responsable hiérarchique du collaborateur	String
AvailabilityDate	Date de disponibilité du collaborateur	Date (Optionnel)
IsValidated	Si la fiche de collaborateur est valide ou non	Boolean
Status	Statut du collaborateur, s'il est disponible affecté ou inter-contrat.	Calculable

Tableau 1 : Entité Collaborateur

Entité : User (Utilisateur)		
Attribut		
Nom	Description	Type
ID	ID de l'utilisateur	Long
FirstName	Prénom de l'utilisateur	String
LastName	Nom de l'utilisateur	String
Login	L'identifiant de l'utilisateur	String
Role	Droits de l'utilisateur	Enumeration
Email	Le mail de l'utilisateur	String
Password	Le mot de passe de l'utilisateur	String
IsActivated	Si le compte de l'utilisateur est activé ou non	Boolean

Tableau 2 : Entité Utilisateur

Entité : Skill (Compétence)		
Attribut		
Nom	Description	Type
ID	ID de la compétence	Long
Label	Label de la compétence	String
Code	Code de la compétence	String

Tableau 3 : Entité Compétence

Entité : Category (Catégorie)		
Attribut		
Nom	Description	Type
ID	ID de la catégorie	String
Label	Label de la catégorie	String
Code	Code de la catégorie	String
Description	Description de la catégorie	String (Optionnel)

Tableau 4 : Entité Catégorie

Énumération : Level (Niveau)			
Attribut			
Nom	Description	Type	Occurrences
ID	ID du niveau	Long	1,2,3,4,5...
Label	Label du niveau	String	Débutant, Junior, Confirmé, Sénior, Expert...
Code	Code du niveau	String	1,2,3,4,5...
Description	Description du niveau	String (Optionnel)	A reçu une introduction à la compétence. / Peut le faire avec de l'aide. / Peut le faire sans aucune aide. / A une grande expérience sur la compétence et travaille à l'aise avec. / Est un expert de la compétence et peut former les autres.

Tableau 5 : Énumération Niveau

Énumération : Role (Rôle)			
Attribut			
Nom	Description	Type	Occurrences
ID	ID du rôle	Long	1,2...
Name	Nom du rôle	String	ROLE_COLLABORATOR, ROLE_Manager...

Tableau 6 : Énumération Rôle

Énumération : FamilySituation (Situation familiale)			
Attribut			
Nom	Description	Type	Occurrences
ID	ID de la situation familiale	Long	1,2,3,4...
Name	Nom de la situation familiale	String	MARRIED, CELIBATE, WINDOWER, DIVORCED...

Tableau 7 : Énumération Situation familiale

2.8. Les livrables

Le tableau ci-dessous représente les livrables demandés, les dates de livraison prévisionnelle et réelle ainsi que le responsable de leur validation.

Phase	Livrable	Responsable de validation	Livraison prévue	Livraison réelle	Validation prévue
Etude Préalable sprint 0	Identification des besoins, choix technique et modélisation UML	SCRUM MASTER	29/07/2020	29/07/2020	30/07/2020
Conception sprint 0	Interfaces et base de données	SCRUM MASTER	04/08/2020	04/08/2020	04/08/2020
Etude Préalable sprint 1	Planning avec user stories et leurs règles de gestion	SCRUM MASTER	06/08/2020	17/08/2020	17/08/2020
Conception sprint 1	Interfaces et base de données	PRODUCT OWNER	19/08/2020	28/08/2020	15/09/2020
Etude Préalable sprint 2	Planning avec user stories et leurs règles de gestion	PRODUCT OWNER	18/09/2020	18/09/2020	18/09/2020
Conception sprint 2	Interfaces et base de données	PRODUCT OWNER	05/10/2020	07/10/2020	07/10/2020
Documentation	Rapport (bilan du projet)	RESPONSABLE DES COMPÉTENCES	28/09/2020	14/10/2020	16/10/2020

Tableau 8: Les livrables

2.9. Les risques

Le tableau suivant présente les risques potentiels encourus durant le stage.



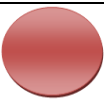

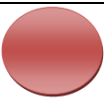
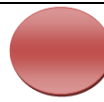
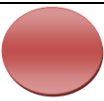
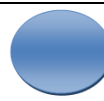


Risques	Type	Impact	Probabilité	Poids	Actions correctives	Actions Préventives
Technologies non maîtrisées	Organisationnel			4	Se documenter.	Se former aux outils.
Non-respect des délais	Organisationnel			8	Augmenter les heures de travail (heures supplémentaires).	Prévoir un éventuel retard en considération lors de la planification du projet.
Perte de données	Crash			16	Utiliser la dernière sauvegarde opérationnelle.	Utilisation d'outils d'intégration et de serveur de secours. Dupliquer les sauvegardes.
Panne Matérielle	Crash			4	Utiliser le matériel de secours	Prévoir un matériel de secours
Sujet incompris	Projet			4	Faire valider les livrables par l'encadrant et/ou le client.	Livrer des documents et des interfaces pour validation à chaque phase du projet.

Tableau 9: Les risques du stage

Légende :



: 1 Bas



: 2 Moyen



: 3 Elevé



: 4 Grave

_ L'impact et la probabilité vont de 1 à 4 tel que 4 est le niveau le plus dangereux.

_ Poids = Impact x Probabilité.

L'identification, la planification et la démarche ainsi réalisées ; on peut établir l'analyse et la conception.

3. Analyse et conception

Dans cette partie, on définit les acteurs de l'application, on présente l'analyse et la conception de chaque sprint ainsi que l'architecture technique utilisée et pour finir les compétences acquises lors de ce stage.

3.1. Identification des acteurs

3.1.1. Définition d'un acteur

Un acteur est un utilisateur qui communique avec le système (opérateur, autre système...) et en réponse le système lui fournit le service qui correspond à son besoin.

3.1.2. Acteurs

Les acteurs qui interagissent avec l'application sont :

Manager	<p>Le manager joue un rôle primordial et fondamental, c'est la seule personne qui dispose du droit d'exécuter toutes les différentes tâches offertes par notre application. En effet, il peut saisir des informations en cas de nécessité et il peut aussi soit les mettre à jour soit les rechercher ou les supprimer soit générer des documents.</p> <p>C'est une personne qui se connecte à notre application pour gérer et utiliser les services suivants : gestion des compétences, gestion des collaborateurs, la recherche avancée, gestion des formations, gestion des projets, gestion des équipes, génération des DT et mini CVs.</p>
Collaborateur	<p>Acteur principal, le collaborateur n'a que la possibilité de consulter et gérer ses propres informations relatives à sa vie professionnelle et de générer ses documents.</p> <p>C'est une personne qui se connecte à l'application pour gérer et appliquer les gestions et fonctionnalités suivantes : gestion de ses compétences, gestion de ses projets, gestion de ses formations, gestion de son profil, génération de son DT et mini CV.</p>

Tableau 10 : Acteurs

3.2. Analyse et UML des sprints

3.2.1. Diagrammes de cas d'utilisation généraux

Le diagramme des cas d'utilisation représente les relations entre les acteurs et les fonctionnalités du système. Le diagramme des cas d'utilisation montre l'ensemble des processus du domaine d'étude. Chaque processus, ou plus précisément, chaque variante de processus, sera modélisé au moyen d'un diagramme de séquence et/ou d'un diagramme d'états-transitions et/ou d'un diagramme d'activités. Ces diagrammes sont généraux et donc valables pour tous les sprints du projet.

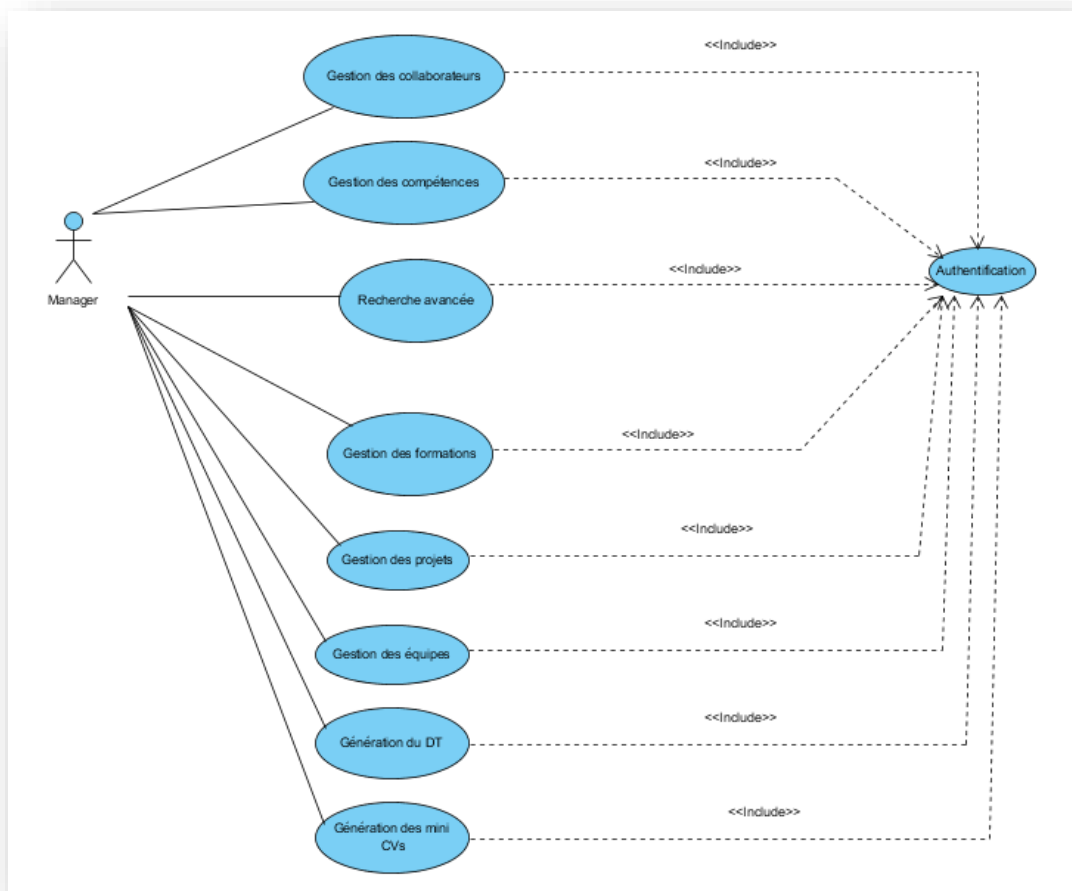


Figure 26 : Diagramme de cas d'utilisation du manager sur « Visual Paradigm »

(Cette image est notre réalisation)

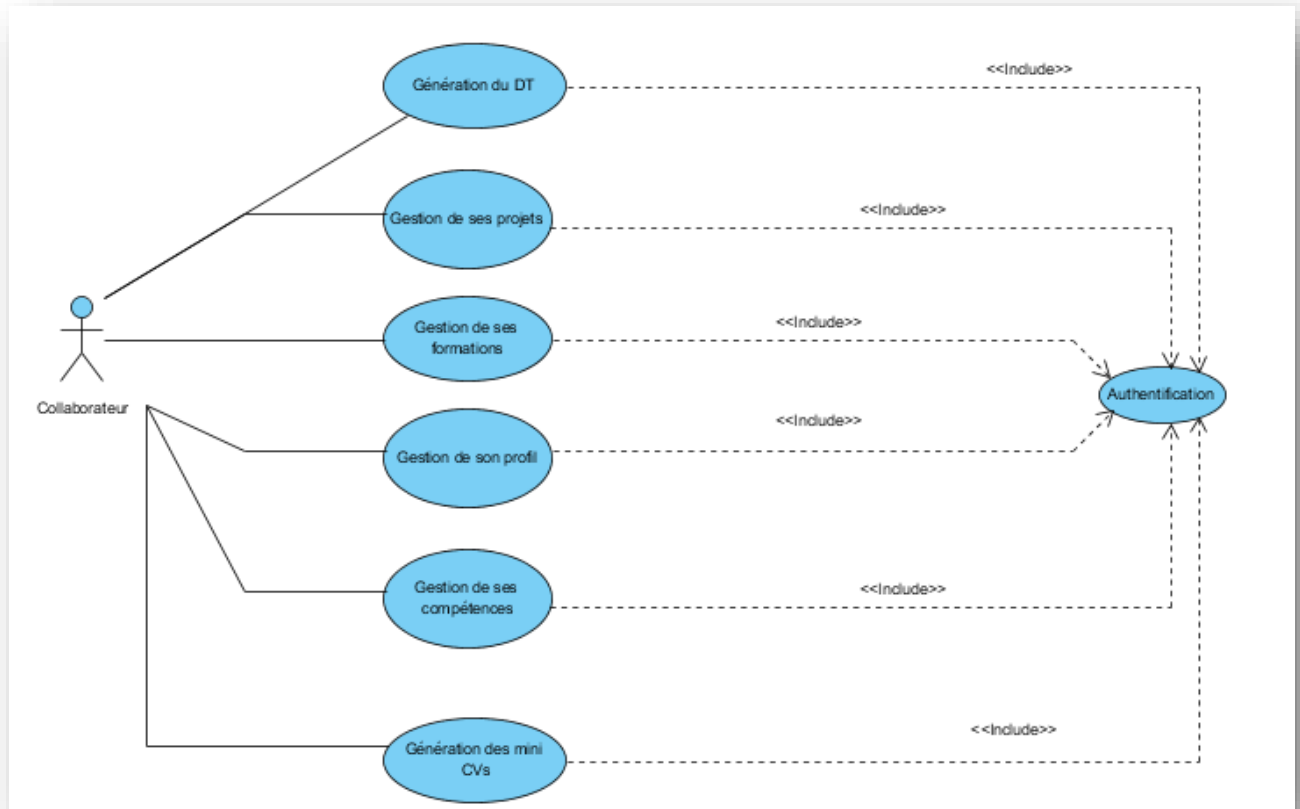


Figure 27 : Diagramme de cas d'utilisation du collaborateur sur « Visual Paradigm »

(Cette image est notre réalisation)

3.2.2. Diagrammes de classes des sprints

Il représente la structure statique d'un système. Il contient principalement les classes ainsi que leurs associations mais on peut aussi y trouver des objets. En pratique, l'intérêt majeur du diagramme de classes est de modéliser les entités du système d'information.

Nous décrivons dans cette partie la structure de notre base de données à travers les diagrammes de classes suivants.

SPRINT 0 ET SPRINT 1

Durant le sprint 0, on commence la modélisation de notre base de données et on conçoit le diagramme de classe suivant qui reste le même pour le sprint 1 à la seule différence qui est l'ajout de l'entité « Tags » en relation « ManyToMany » avec les compétences.

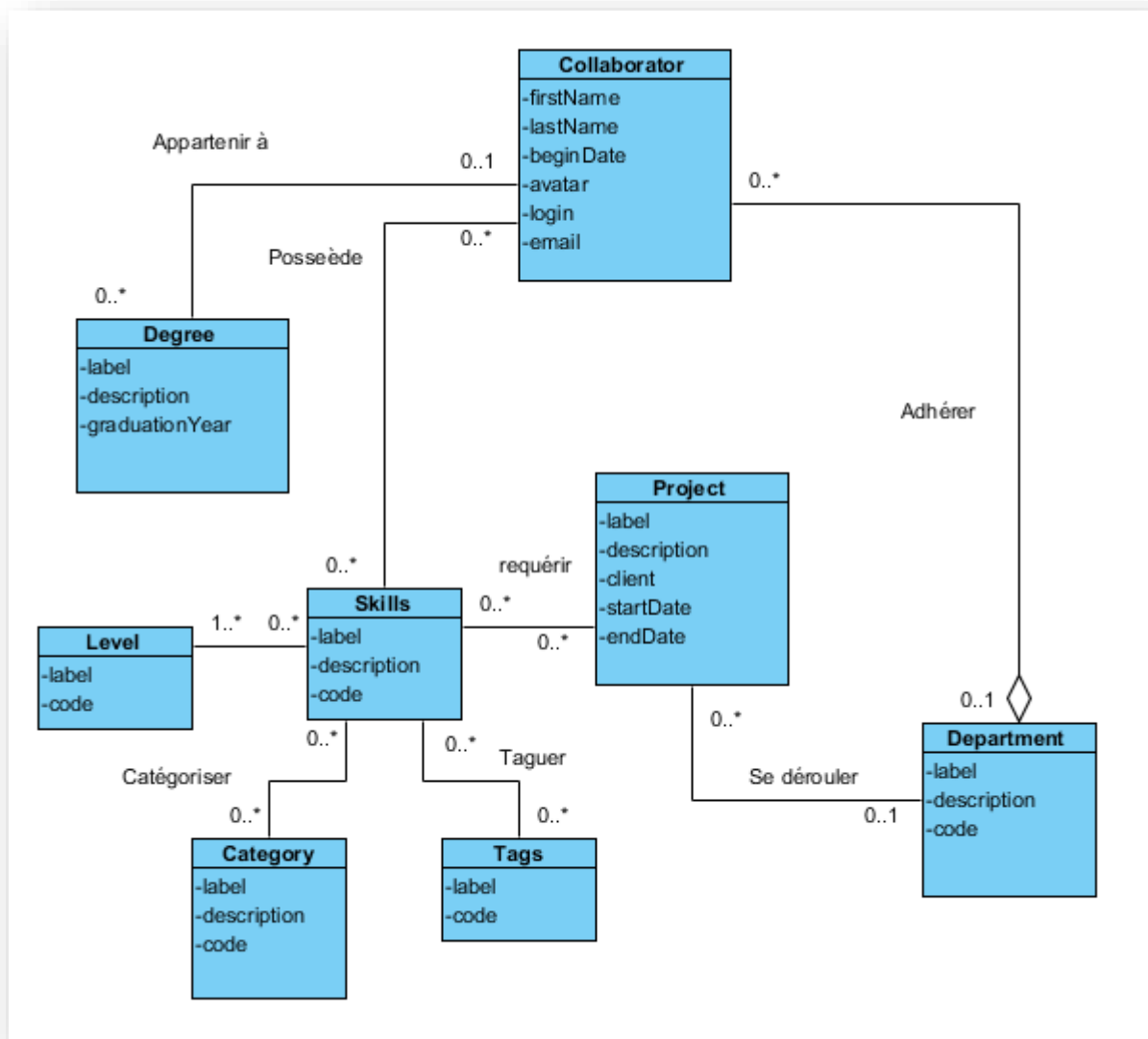


Figure 28: Diagramme de classe du sprint 1 sur « Visual Paradigm »

(Cette image est notre réalisation)

SPRINT 2

Arrivé au sprint 2, il a fallu refaire toute l'analyse et la conception depuis le début pour répondre au besoin du client et donc on change le diagramme de classes des sprints précédents pour obtenir le résultat suivant :

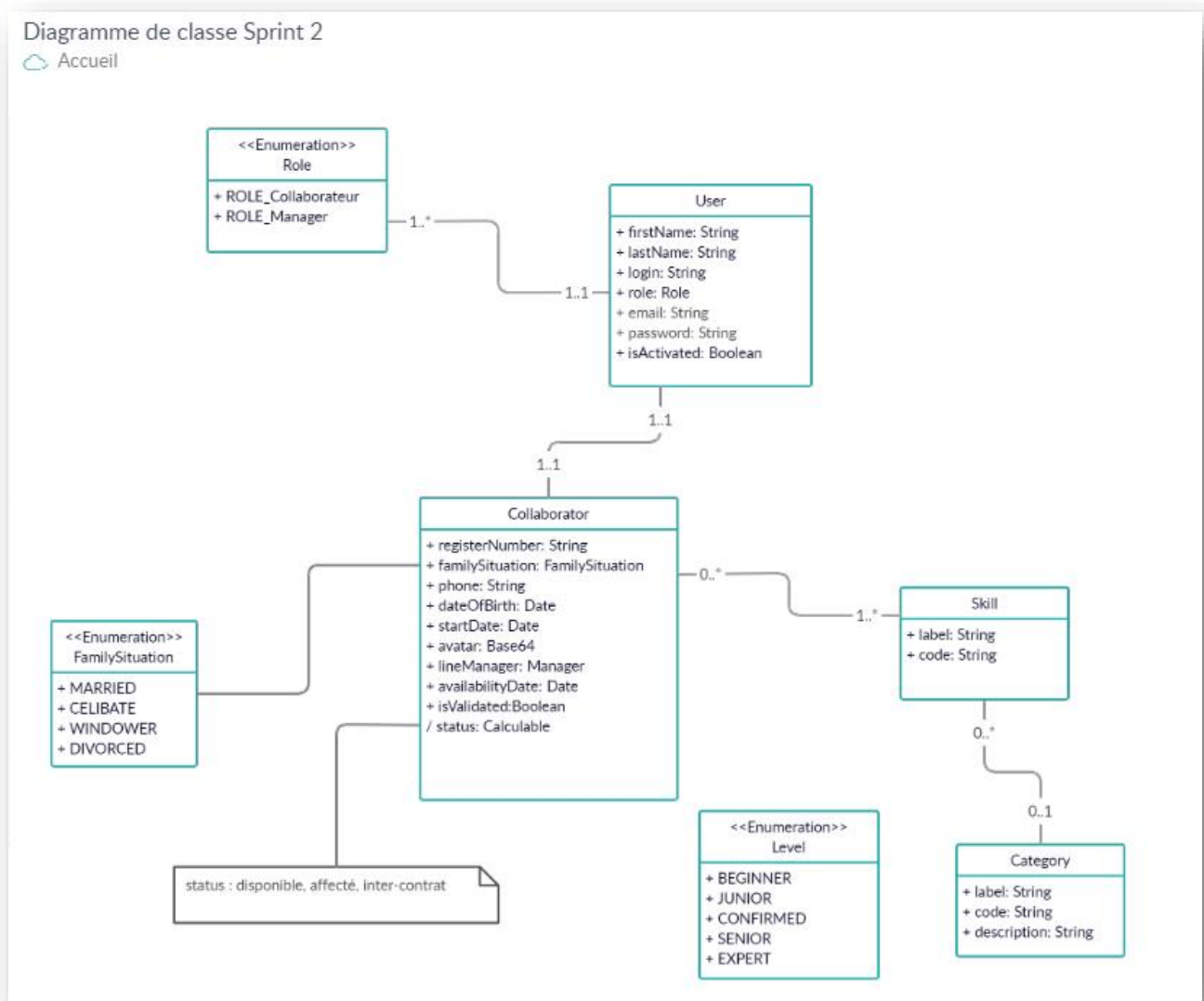


Figure 29 : Diagramme de classe du sprint 2 sur « creately.com »

(Cette image est notre réalisation)

3.2.3. Diagrammes de séquences généraux

C'est une variante du diagramme de collaboration. Il permet de mieux visualiser la séquence des messages en mettant l'accent sur les aspects temporels. Le diagramme de séquence représente la succession chronologique des opérations effectuées par un acteur pour la réalisation d'un cas d'utilisation, il indique les objets que l'acteur va manipuler et les opérations qui font passer d'un objet à l'autre. Il est à invoquer que toutes les opérations mentionnées dans les différents diagrammes de cas d'utilisation se ressemblent, dans ce sens, on présentera les diagrammes de séquences de chaque opération pour une meilleure compréhension. Ces diagrammes sont généraux et donc valables pour tous les sprints du projet.

3.2.3.1. Diagramme de séquence pour un scénario d'authentification

Le diagramme, exposé dans la figure ci-dessous, décrit les scénarios possibles lors d'une opération d'authentification.

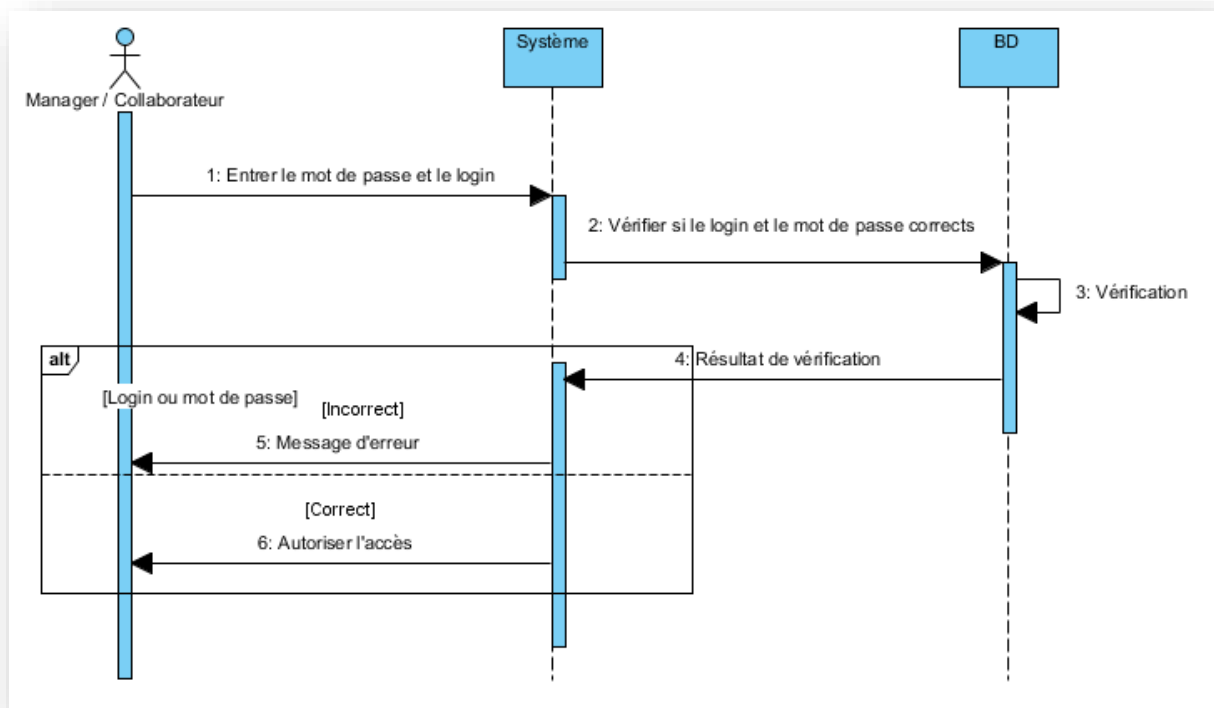


Figure 30: Diagramme de séquence pour un scénario d'authentification sur « Visual Paradigm »

(Cette image est notre réalisation)

3.2.3.2. Diagramme de séquence pour un scénario d'ajout

Le diagramme, exposé dans la figure ci-dessous, décrit les scénarios possibles lors d'une opération d'ajout.

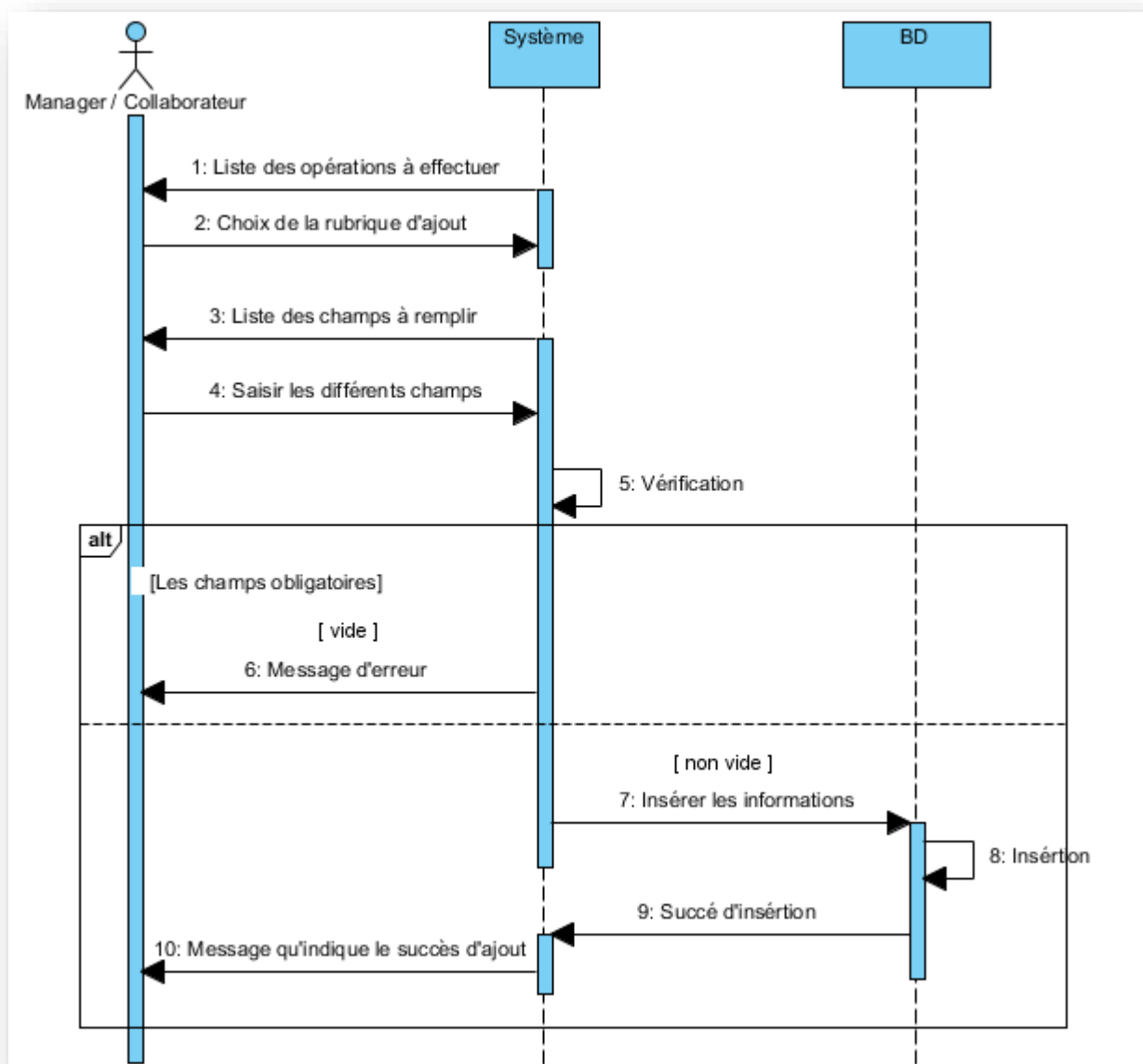


Figure 31: Diagramme de séquence pour un scénario d'ajout sur « Visual Paradigm »

(Cette image est notre réalisation)

3.2.3.3. Diagramme de séquence pour un scénario de suppression

Le diagramme, exposé dans la figure ci-dessous, décrit les scénarios possibles lors d'une opération de suppression.

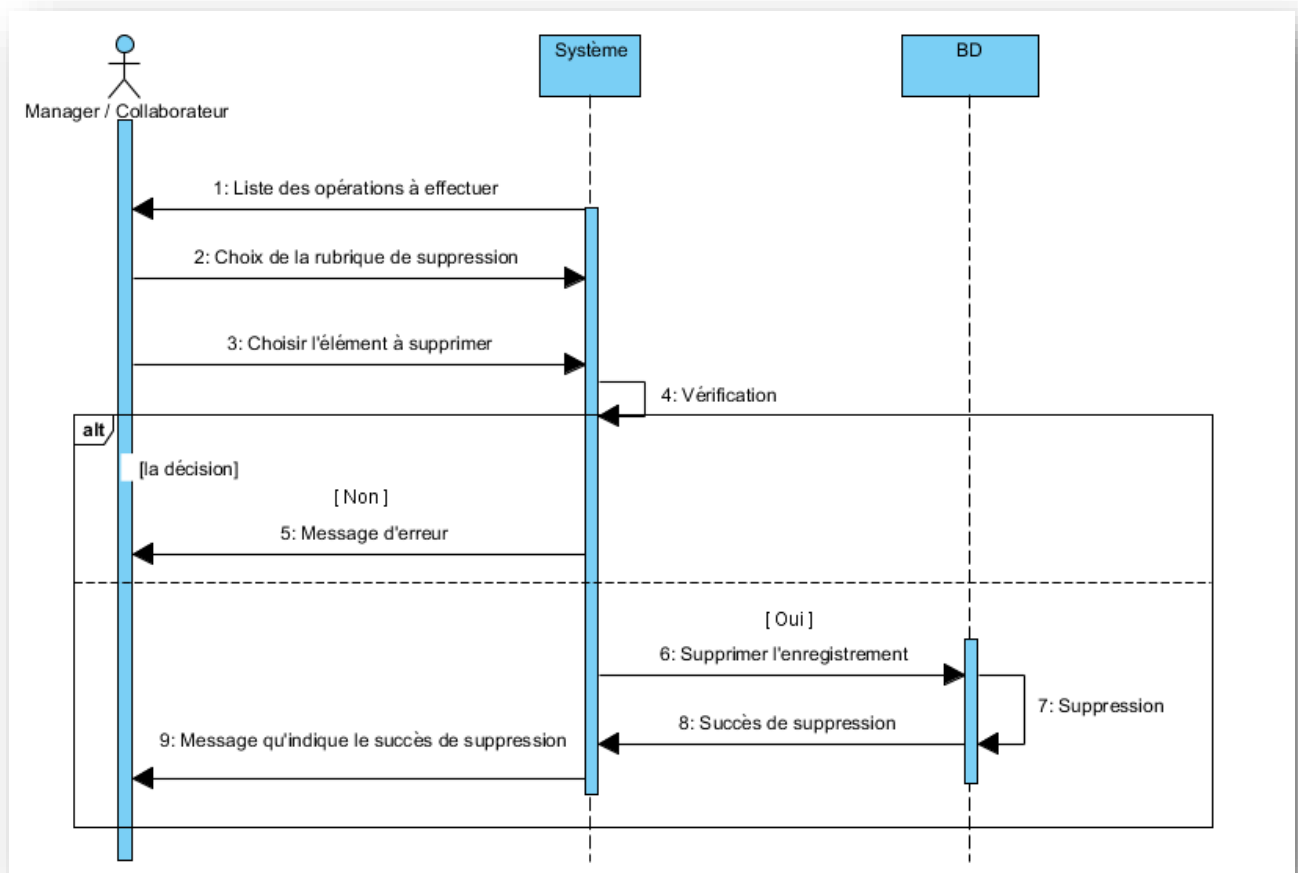


Figure 32: Diagramme de séquence pour un scénario de suppression sur « Visual Paradigm »

(Cette image est notre réalisation)

3.2.3.4. Diagramme de séquence pour un scénario de modification

Le diagramme, exposé dans la figure ci-dessous, décrit les scénarios possibles lors d'une opération de modification.

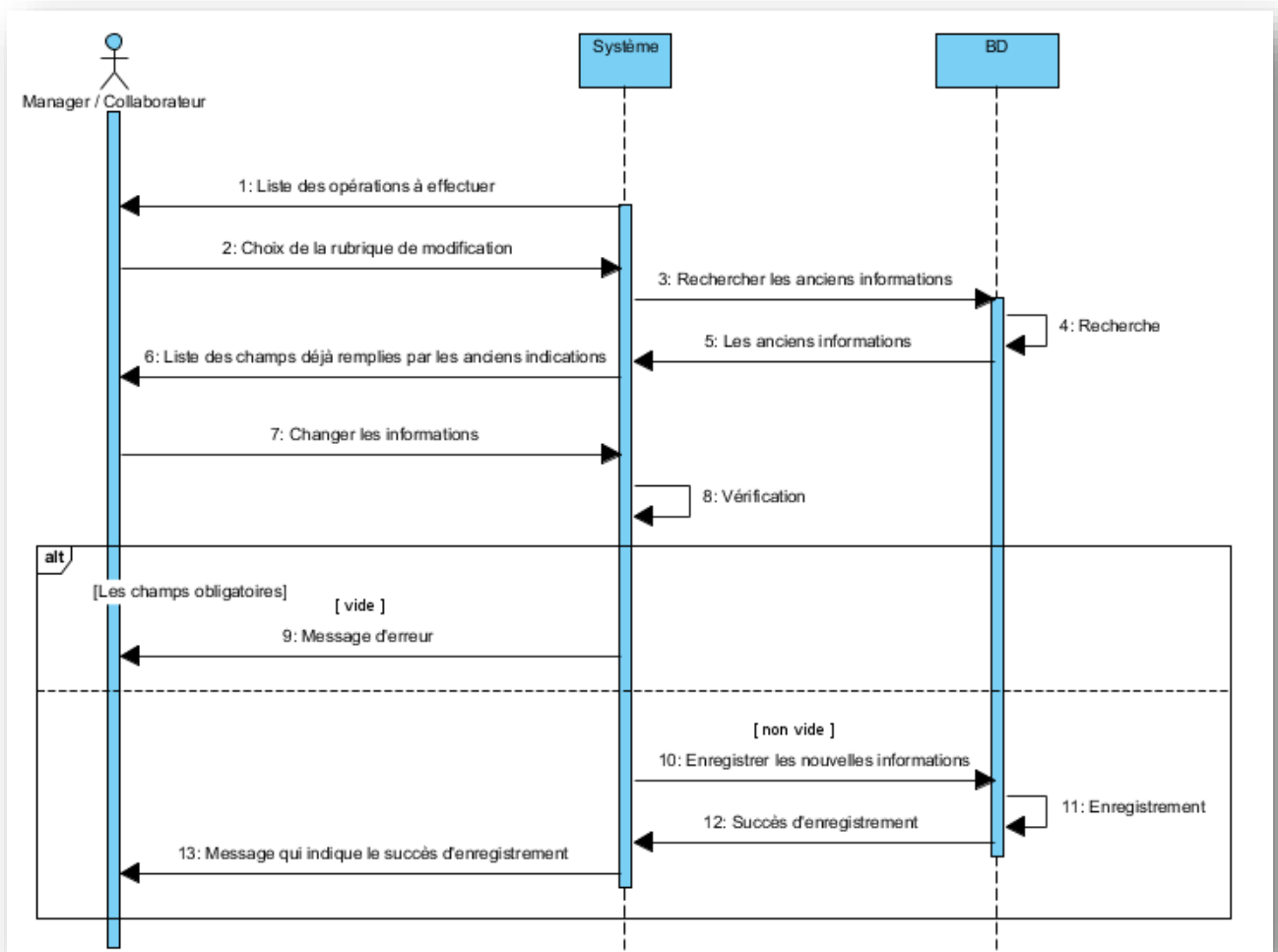


Figure 33: Diagramme de séquence pour un scénario de modification sur « Visual Paradigm »

(Cette image est notre réalisation)

3.2.3.5. Diagramme de séquence pour un scénario de recherche

Le diagramme, exposé dans la figure ci-dessous, décrit les scénarios possibles lors d'une opération de recherche.

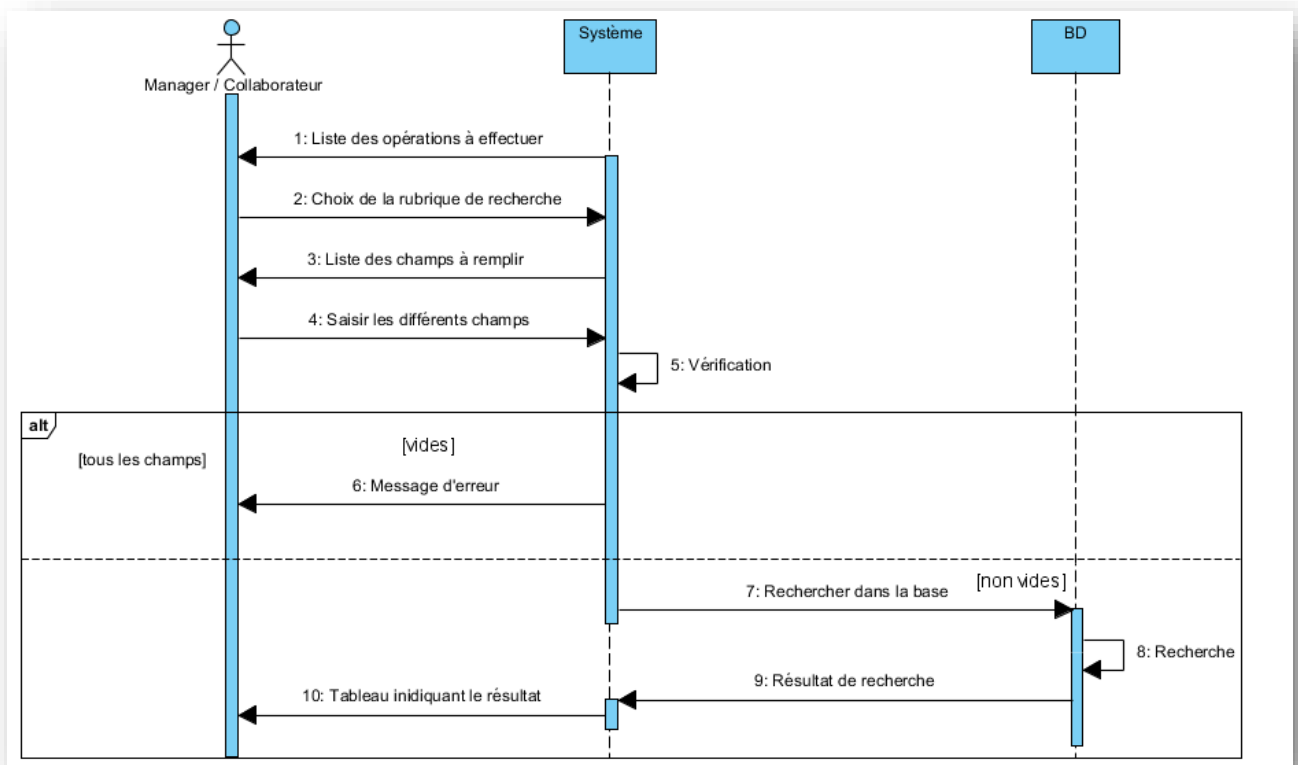


Figure 34: Diagramme de séquence pour un scénario de recherche sur « Visual Paradigm »

(Cette image est notre réalisation)

3.3. Architecture technique

La figure suivante présente les différents composants de notre architecture technique.

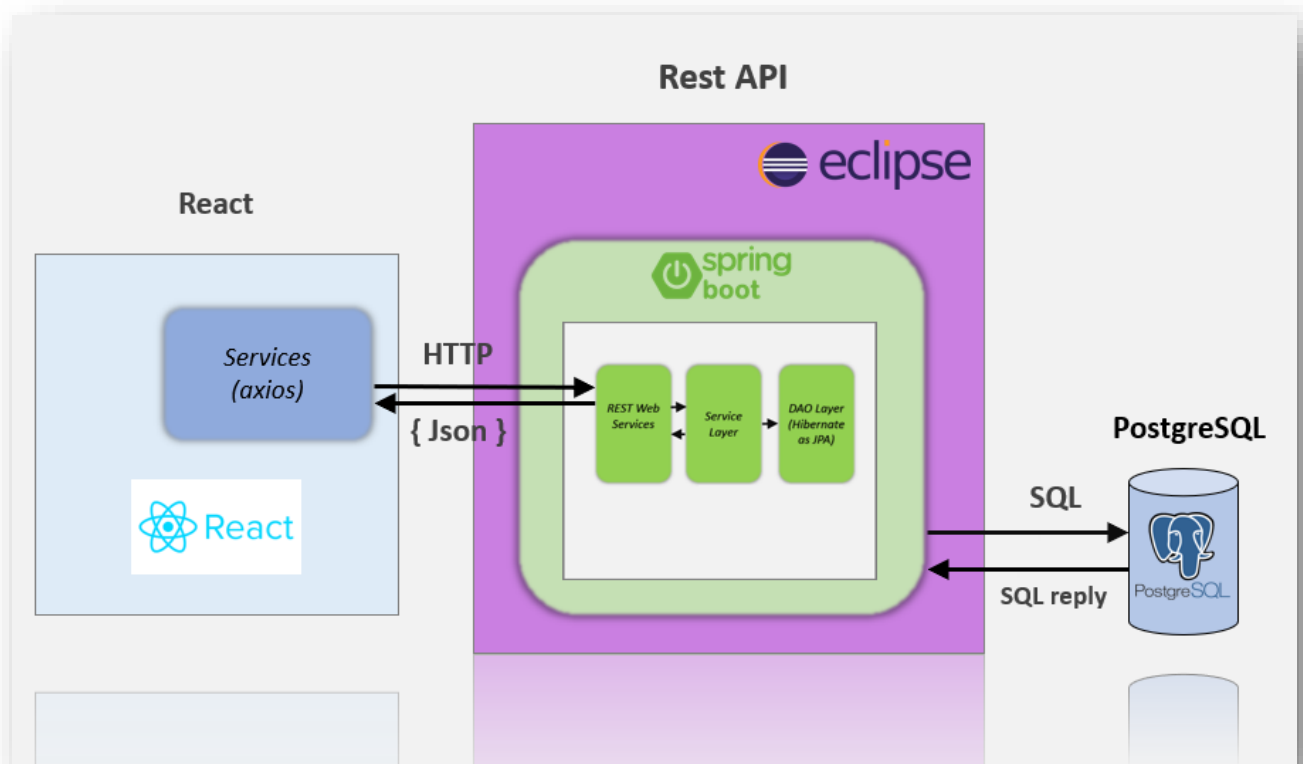


Figure 35: Architecture technique du projet

(Cette image est notre réalisation)

Notre application web dispose d'une interface de programmation applicative « API » qui est la partie du programme qu'on expose officiellement au monde extérieur pour manipuler celui-ci. Elle permet d'entrer des données et de les récupérer à la sortie d'un traitement.

Les API sont un ensemble de fonctions et de procédures qui permettent la création d'applications qui accèdent aux données et aux fonctionnalités d'autres applications, services ou systèmes d'exploitation.

On a choisi pour notre API le type d'architecture Transfert d'état représentationnel « REST » qui donne la possibilité de mettre en place une méthode vraiment simple afin d'accéder à des services web. L'un de ses plus grands avantages est le fait que les

services web peuvent être utilisés par n'importe quel genre d'application front (web, mobile, bureau ...).

On utilise comme frontend la librairie javascript react. Lorsque l'utilisateur interagit avec notre application, il consomme en parallèle des services web auxquels on fait appel depuis react grâce à la librairie javascript axios qui facilite l'envoi de requête HTTP. Cette dernière est interceptée côté backend par les contrôleurs web REST implémentés avec la couche service qui utilise les méthodes de la couche DAO basées sur des requêtes HQL « Hibernate Query Language » converties par l'ORM Hibernate en SQL afin que le SGBD puisse les reconnaître. Ce framework permet de persister les objets java dans la base de données d'une manière configurable sous forme de tables en implémentant la spécification JPA.

Après le SGBD envoie la réponse à travers le protocole SQL au backend pour ensuite retourner au frontend le résultat de la requête HTTP en JSON.

3.4. Conception et développement des sprints

SPRINT 0 – « Préparation de l'environnement »

La première étape du sprint 0 est la Dockerisation de l'environnement (Frontend & Backend). Elle a pour but d'automatiser l'installation de l'environnement du projet en préparant toutes ses dépendances sous forme d'un environnement docker pour éviter la perte de temps.

Pour cela, on commence par créer un fichier texte avec des instructions pour construire notre image docker, ce dernier s'appelle un dockerFile.

Ensuite, on crée un fichier de configuration « docker-compose.yml » qui va utiliser notre dockerFile qui se trouve dans le même répertoire mais qui va aussi nous permettre d'ajouter une base de données à notre environnement dans un conteneur consacré à cette dernière.

Un conteneur docker est un package de logiciels imitant une machine virtuelle mais sans le système d'exploitation client pour une meilleure portabilité et qui comprend tout le nécessaire pour exécuter une application.

Puis, on installe Docker qui va nous permettre d'exécuter la commande « docker-compose -f docker-compose.yml build » pour créer nos images docker à partir de notre fichier de configuration docker « docker-compose.yml » et de notre dockerFile et la commande « docker-compose -f docker-compose.yml up » pour déployer l'application et instancier les images docker dans leur conteneur docker.

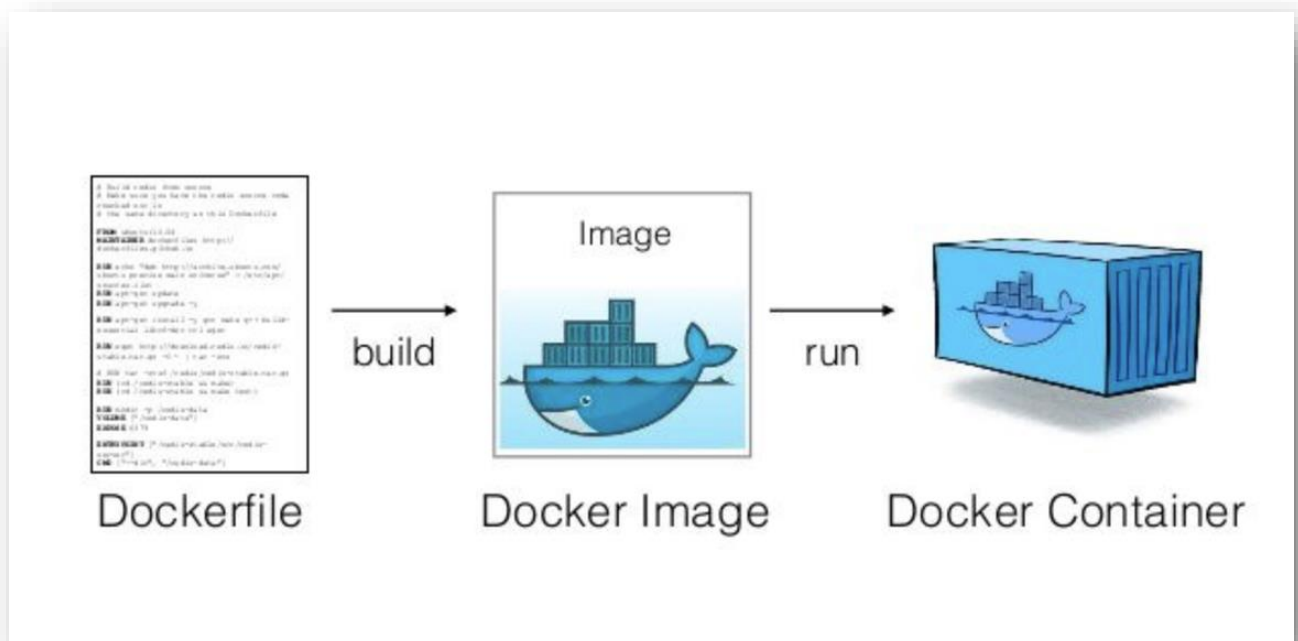


Figure 36 : schéma explicative de la dockerisation de notre application

(Cette image provient du site : <https://medium.com/fiolabs-datascience/how-to-make-dockerfile-build-docker-image-and-run-docker-container-part-2-of-2-82bfb30a6859>)

On utilise 3 images dans notre environnement, la première c'est le système d'exploitation hôte ubuntu, la deuxième c'est l'image qui contient toutes les dépendances du projet et la troisième c'est celle de postgresSQL qui contient notre base de données.

Les deux premières s'exécutent dans le conteneur « crm-skills » et la troisième et dernière image dans le conteneur « crm-postgres ».


```
C:\Users\Mr-Ta>docker image ls
REPOSITORY          TAG                IMAGE ID           CREATED           SIZE
altencrm_crm-skills latest            47069045a375      3 weeks ago      1.78GB
ubuntu              bionic            2eb2d388e1a2      4 weeks ago      64.2MB
postgres            10                95a44b3fd42b      4 weeks ago      200MB
```

Figure 37 : liste des images docker de notre projet

(Cette image est notre réalisation)

Après, on passe à l'étape suivante qui est la génération de l'application avec ses entités, classes, DAO, CRUDs, services web REST et interfaces à l'aide de JHipster.

Tout d'abord, on installe JHipster et on génère la totalité de l'application avec la commande « jhipster » et on choisit parmi les options ce qui nous convient.

Avant de passer à la génération des entités, on commence par ajouter la langue française car la langue par défaut qu'on a choisi est l'anglais. On utilise alors la commande « jhipster languages » et on choisit le français, après, on ajuste la traduction qui se trouve dans le fichier JSON générés dans le dossier i18n dans la partie front de notre application ce qui nous permet d'avoir une application qui dispose d'une internationalisation avec deux langues le français et l'anglais.

On change alors la langue par défaut en français grâce à un fichier de configuration déjà mis en place dans le projet. L'internationalisation et la langue par défaut sont des options qu'on peut choisir lors de la génération de l'application.

Puis, on détermine les entités avec leurs attributs et relations depuis la modélisation et la conception UML. Ensuite, on crée un fichier JDL « JHipster Domain Language » dans la racine de notre projet, il s'agit d'un fichier simple où on introduit nos entités avec leurs attributs et relations à l'aide de la documentation que nous offre JHipster (voir webographie) afin de pouvoir générer les entités, classes, DAO, CRUDs, services web REST et interfaces de notre application avec une seule commande « jhipster import-jdl your-jdl-file.jh ». Toutes les entités de l'application sont configurables dans leur fichier de configuration généré par JHipster.

La base de données est initialement implémentée avec Liquibase qui utilise Faker pour générer de fausses données pour une meilleure visualisation.

Pour affiner le rendu coté client, au niveau de chaque gestion quand une entité a une relation avec une autre, la sélection doit contenir le label ou la concaténation du nom et prénom au lieu de l'id pour plus de visibilité.

SPRINT 1 – « Manager : Gestions des compétences »

On commence par créer la branche `Sprint1_Manager_Gestion_Compétences`, à partir de la branche `dev`, où seront apportées toutes les modifications du sprint.

Ensuite on se documente bien sur JHipster pour trouver une manière de générer une entité au lieu de changer à chaque fois le JDL qui est la commande « `jhipster entity <entity-name>` » ce qui nous a permis de créer une nouvelle entité `Tags` avec ses attributs `label` et `code`, définir ses validations (`required` et `unique`), ajouter de nouvelles propriétés à l'entité `compétences` et joindre les deux entités `compétences` et `Tags` avec une relation `ManyToMany` des deux coté.

Après chaque ajout de propriété ou d'entité on supprime la base de données pour qu'elle se génère avec les nouvelles colonnes et tables grâce à quelques commandes de PostgreSQL.

Puis, on génère la pagination avec un groupement `SORT` de chaque attribut d'une entité sauf pour les attributs ayant une liste grâce à JHipster en changeant la propriété pagination de « `no` » à « `pagination` » dans le fichier de configuration de l'entité `compétences` et par la suite la régénérer pour prendre les modifications en compte.

JHipster n'a commencé de générer des applications avec React que récemment donc il ne procure pas tout correctement pour le moment du moins dans sa version 6.10.1 qu'on utilise dans notre projet et donc il y a quelques bugs ou mal fonctionnements qu'on corrige tout au long du sprint.

Ensuite, on corrige quelques problèmes avec la pagination qui se produisent lorsque l'utilisateur ajoute, modifie ou supprime une entité qui dispose d'un attribut avec relation en modifiant les redirections vers les pages par les méthodes suivantes : « `history.goBack()` » et « `history.push(path, [state])` » car les mauvaises redirections créent des problèmes dans les paramètres de la pagination (à voir le lien de documentation sur l'objet `history` de react dans la webographie).

On corrige aussi par la suite l'erreur dans le cas où, une entité a dans ses attributs une relation et qu'on ajoute pour la deuxième fois l'entité sans actualiser la page elle retourne « undefined » dans les attributs qui sont en relation sous forme d'un select multiple, cette correction est réalisée grâce à l'opérateur logique OR de javascript « || » qui nous permet d'attribuer aux propriétés avec relation soit leur valeur, soit un tableau vide afin d'éviter la valeur « undefined ».

On organise alors la liste des compétences en affichant que l'id, label, code et description et pour plus de détails on affiche tous les attributs de l'entité dans sa fiche dans l'ordre voulu et avec des relations sous forme de liens vers les fiches des entités liées ainsi que la traduction de chaque attribut, même la valeur de l'attribut statut fiche qui est une énumération est traduite.

Pour le filtrage, on a d'abord ajouté une barre de recherche sur le côté front qui nous a permis de comprendre qu'il faut absolument que la barre de recherche soit réalisé dans le coté backend avec des requêtes car si on la réalise sur le frontend elle ne pourras filtrer que les données qui se trouve dans le front, c'est alors qu'on crée une barre de recherche automatique qui permet de chercher une compétence avec son label, code et technologies, on peut switcher entre les trois avec les boutons label, code et technologies, cette barre de recherche fait appel à une requête http du backend soit si elle est vide pour afficher toutes les compétences sans filtre, soit à chaque caractère ajouté depuis le troisième et donne comme résultat les compétences contenant celui-ci. Les requêtes de filtrage d'une entité dans la partie backend sont générées par JHipster en modifiant la propriété « service » de « no » à « serviceClass » ou « serviceImpl » et la propriété « jpaMetamodelFiltering » de « false » à « true » dans le fichier de configuration de l'entité qui se trouve dans le dossier « .jhipster ».

On corrige après, lors de la génération des filtres des entités, les tests qui peuvent parfois considérer une relation many to many comme étant un attribut simple d'une entité et donc lui attribuer le mauvais test, c'est alors qu'on attribue le bon test à notre relation catégories en copiant la structure d'un test many to many qui a bien été généré.

Après, on change le type des catégories en les rendant requises à la création d'une compétence grâce au fichier de configuration et à la propriété « "relationshipValidateRules": "required" » de l'entité puis JHipster s'occupe d'ajouter l'annotation @NotNull pour éviter les valeurs nulles et @Valid pour la

prendre en compte au backend. Pour notre cas, on remplace `@NotNull` par `@NotEmpty` qui évite non seulement les valeurs nulles mais même les valeurs vides. On ajoute aussi la validation au niveau du frontend en rendant les champs requis et en affichant une erreur comme quoi le champ est obligatoire qui est traduite.

Par la suite, on applique une requête JPQL dans le repository de l'entité compétence pour vérifier lors de l'ajout d'une compétence qu'elle n'existe pas déjà, ensuite on place la condition dans le service web REST qui renvoie une erreur, comme quoi l'entité existe déjà si c'est bien sûr le cas, traduite dans les différentes langues de l'application à savoir le français et l'anglais. De plus, on convertit aussi le label entré par l'utilisateur en minuscule côté front pour ignorer la casse (c'est-à-dire qu'on ignore les majuscule).

Pour éviter que l'utilisateur ne se perde et profite d'une navigation rapide et efficace on remplace les boutons retour et cancel des fiches et entités par un « `history.goBack()` » de React qui permet de le rediriger vers la page d'avant.

Entre autres, on place le lien de l'ajout d'une nouvelle compétence à côté de toutes les listes déroulantes de celle-ci, soit dans la page ajout et modification des entités Projets et Collaborateurs.

Puis, on enlève les attributs statut fiche et date de validation de l'ajout d'une nouvelle compétence et les laisse dans la modification.

En dernier lieu, On ajoute finalement les filtres qui restaient, à savoir les catégories et tags à notre entité compétence sous forme d'un select multiple qui est affiché à la place de la barre de recherche des autres filtres si l'utilisateur clique sur l'un des deux boutons des filtres qui sont en relation ManyToMany soit « catégories » et « tags ». Cette permutation se fait avec le rafraîchissement du DOM et donc quand le select multiple est affiché la barre de recherche n'est pas caché mais n'existe pas et vice versa. La première option du select est vide dans le cas où l'utilisateur souhaite réafficher les compétences sans filtre, donc l'utilisateur peut filtrer les compétences avec les catégories ou tags en les sélectionnant, il peut choisir plusieurs s'il reste appuyé sur la touche CTRL. Le résultat est l'ensemble des compétences qui ont toutes les options sélectionnées. Les deux requêtes qui permettent ce résultat ont été réalisé manuellement avec JPQL car JHipster ne prend pas en compte le filtre qui affiche que les entités ayant toutes les sélections mais seulement ayant une sélection parmi plusieurs.

Le code qu'on utilise est un code professionnel soutenu avec ESLint dans le coté front javascript on emploie des opérateurs logiques, ternaires et de décomposition ainsi que les hooks de React qui sont une nouveauté qui vient d'apparaître depuis sa version 16.8 alors que maintenant lors de l'écriture de ce rapport on est à la version 16.13.1 et dans le coté back end on utilise le JPQL pour le requêtage, les annotations, le minimum de code possible et bien sûr la totalité du code backend et frontend est commentée pour que chacun puisse comprendre le code et se repérer facilement. Les React hooks permettent de créer des états et de les changer dans une seule ligne de code sans même avoir besoin de créer une nouvelle classe (à voir dans l'annexe).

Lors du sprint 1, à chaque fin de journée on doit imputer combien de points de complexité on a réalisé dans le tableau de statistiques de sprint suivant dans Excel :

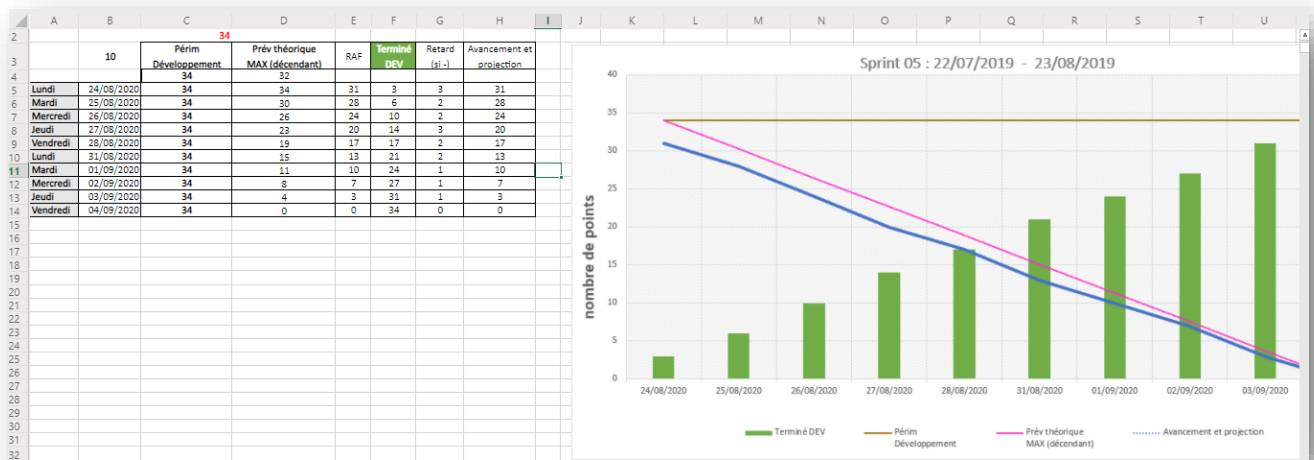


Figure 38 : tableau d'imputation et de statistiques sur Excel du sprint 1

(Cette image est notre réalisation)

SPRINT 2 – « Manager : Gestions des compétences, collaborateurs, niveaux et catégories »

Après avoir remodelisé notre base de données pour répondre aux besoins de notre client, On a commencé par créer notre branche « Feature_T01 ». On recommande fortement ce workflow (façon de travailler) que nous avons suivi, le nom d'une branche doit contenir « Feature_ » qui veut dire fonctionnalité en français + des initiaux du nom de l'auteur + le numéro de la fonctionnalité, tandis que les commentaires doivent être dans le commit si on veut préciser quelque chose. Après avoir terminé le travail sur la branche le collaborateur doit rebaser sa branche sur la branche générale de développement pour notre cas « dev » pour gérer les conflits avec le travail des autres et doit absolument les contacter pour gérer le conflit ensemble.

Durant le sprint 2, on développe 4 gestions avec leur design basé sur prime React qui est un design de composants React.

Voici une liste des gestions avec l'ordre de leur réalisation : gestion des niveaux, gestion des compétences, gestion des catégories et gestion des collaborateurs.

D'abord, on écrit dans le fichier JDL nos entités, leurs relations, leurs contraintes de validation on peut même ajouter la pagination mais on utilise le fichier JDL ici pour générer une première version des entités et si on veut changer plus tard on a qu'à utiliser la commande suivante : « `jhipster entity <entity-name>` » pour modifier notre entité comme on le souhaite.

Donc on complète la première gestion qui concerne les niveaux, en générant son CRUD, en ajoutant la validation unique au label et code du niveau et en privant le collaborateur de l'ajout, modification et la suppression ne lui laissant que le droit de consulter les niveaux. Pour cette dernière fonctionnalité, on change le composant de la route de modification, ajout et suppression de niveau par « PrivateRoute » importé de « `app/shared/auth/private-route` » et on lui ajoute la propriété « `hasAnyAuthorities=[AUTHORITIES.ADMIN]` » comme quoi seul les utilisateur avec le rôle admin on le droit d'accéder à cette route. C'est une propriété qui renvoie vers une fonction qui à son tour vérifie si le ou les rôle(s) de l'utilisateur connecté correspondent à ceux de la route, si ce n'est pas le cas l'utilisateur reçoit un message lui indiquant qu'il n'a pas les droits nécessaires pour accéder à la route.

Puis, on passe à la gestion des compétences pour lui générer son CRUD, lui ajouter sa validation unique à son label et son code et en lui ajoutant une contrainte lors de la suppression, comme quoi de ne pas supprimer une compétence en cours d'utilisation qui est déjà attribuée à un collaborateur. Pour ce faire, nous avons ajouté une requête dans la partie de suppression au niveau back end en JPQL qui permet de vérifier si la compétence existe dans au moins un collaborateur pour envoyer à l'utilisateur un message d'erreur traduit dans les langues de l'application qui le prévient que la compétence est en cours d'utilisation sinon la suppression se passe normalement.

Par la suite, on continue avec la gestion des catégories cette fois, on lui génère elle aussi son CRUD et sa validation unique pour son label et code puis on interdit l'accès à toutes les fonctionnalités de cette gestion au collaborateur, pour cela on va suivre la même méthode précédente en utilisant le composant « PrivateRoute » et sa propriété « `hasAnyAuthorities=[AUTHORITIES.ADMIN]` » avec le rôle admin comme obligation mais cette fois ce n'est pas la peine de l'attribuer à chaque route de chaque fonctionnalité de l'entité catégorie, puisqu'on va l'appliquer à toutes les fonctionnalités de l'entité autant lui interdire l'accès au composant parent de toute l'entité catégorie, et donc on aura qu'à l'appliquer sur une seule route qui est générale avec le composant parent de toute l'entité catégorie.

Après, on arrive à la dernière gestion du sprint, la gestion des collaborateurs où on commence par rallier l'entité collaborateur avec l'entité User grâce à une relation « OneToOne » pour considérer le collaborateur comme un profil qui peut se connecter à notre application et cela soit par JDL ou avec la commande « `jhipster entity <entity-name>` », ensuite on génère le CRUD du collaborateur et configure le serveur SMTP avec la configuration de Gmail pour que l'envoi des mails pour activation de compte fonctionne, l'envoi de mail est déjà généré par JHipster ainsi que l'enregistrement et l'activation mais aussi la désactivation de compte autant qu'administrateur qui est le Manager dans notre cas. Lorsque le collaborateur se connecte dans l'application il trouve un nouveau lien dans le menu qu'on ajoute à savoir « Saisir ma fiche collaborateur » avec le composant « navLink » de « react-router-dom » qui le redirige vers l'ajout d'un nouveau collaborateur mais sans l'attribut compte qui est automatique dans le formulaire. Cet attribut reçoit automatiquement l'id de l'utilisateur connecté et l'insère dans le formulaire collaborateur comme ça l'utilisateur n'aura qu'à entrer ces données directement et la liaison est faite automatiquement. Pour récupérer l'id ou autre information de l'utilisateur connecté il suffit d'aller dans la fonction « `mapStateToProps` » et de

recupérer l'état « `storeState.authentication.account` » et de l'ajouter aux propriétés existantes puis lui soutirer l'id ou les informations voulu comme JHipster on fait dans la page d'accueil où ils soutirent le prénom et nom de l'utilisateur connecté.

Puis, on ajoute la pagination à l'entité collaborateur ce qui provoque un problème dans l'affichage de ses compétences, les objets collaborateurs reçu de l'API contiennent la propriété compétence « null » et cela à cause de JHipster qui ne génère pas encore un code permettant d'afficher les relations avec la pagination dans la version que nous utilisons 6.10.1, donc pour y remédier on utilise la requête « `findAllWithEagerRelationships` » avec aucun paramètre qui est générée par JHipster pour obtenir la liste des collaborateurs qu'on met comme paramètre dans la fonction qu'on développe qui permet de convertir une liste de collaborateur en Page de collaborateur qu'on affiche par la suite. « Page » est un type de données du Framework Spring. Cette fonction est créée dans une implémentation d'une interface qui doit porter le même début de nom que le repository du collaborateur avec Custom à la fin « `CollaboratorRepositoryCustom` » et elle prend en paramètre la liste des collaborateurs et l'objet Pageable pour paginer la nouvelle Page qu'on crée grâce à la classe « `PageImpl` » et la retourne ensuite. À voir dans l'annexe la méthode pour paginer la Page créée.

Au final, on ajoute la recherche de collaborateur par compétence comme pour les filtres tags et catégories dans le sprint 1, cette fois dans le front on envoie une requête HTTP avec la pagination et les compétences sélectionnées, qui est reçu par l'API qui transforme, grâce à une fonction que nous avons réalisé disponible en annexe, les ids reçus en une collection Long pour ensuite les utiliser dans une requête JPQL qui retourne la liste des collaborateurs qui ont toutes les compétences sélectionnées et ensuite on utilise la même fonction de tout à l'heure pour convertir la liste en page et on la renvoie pour l'affichage.

On a aussi réalisé quelques modifications dans le composant update du collaborateur avec l'état « `isNew` » déjà généré par JHipster qui nous permet de faire des conditions, pour mettre en place des changements entre la page d'ajout et de modification dans la partie client puisqu'au niveau du code on a un seul composant pour les deux. Il permet de différencier entre une page d'ajout et de modification par le paramètre du lien. Si « `isNew` » existe alors il s'agit de l'ajout et vice versa.

3.5. Enseignements/apports du stage (connaissances - compétences)

Ce stage nous a apporté plusieurs compétences :

- _ Maîtrise avancée de React avec redux et les hooks.
- _ Maîtrise avancée en javascript, ECMAScript, TypeScript.
- _ Maîtrise avancée des bonnes manipulations et techniques de codage basées sur le code de JHipster qui est généré par des professionnels et aussi grâce à ESLint.
- _ Dockeriser et déployer une application avec docker.
- _ Génération d'une application avec JHipster.
- _ Génération des éléments essentiels d'un système d'information avec JHipster (modèles, relations, services web, base de données, CRUD, interfaces, paginations...).
- _ Réalisation d'une API REST FULL.
- _ Analyse et débogage du code avancée coté front et back.
- _ Maîtrise du requêtage sécurisé en JPQL.
- _ Maîtrise de java, JPA et Spring boot.
- _ Maîtrise de PostgreSQL.
- _ La sécurité avec JWT et OAuth.
- _ Fixer des erreurs de codage avancées et complexes.
- _ L'autoformation avancée.
- _ La recherche et la documentation poussée.
- _ La persévérance avancée.
- _ S'adapter aux besoins d'un client.
- _ S'organiser et travailler en équipe en agile avec des outils tels qu'Azure DevOps ou Jira.
- _ Création d'interfaces utilisateur en suivant les principes de l'IHM.
- _ Modélisation UML.

4. Réalisation

Dans ce chapitre, nous présentons les outils utilisés pour l'application ainsi que quelques interfaces réalisées pour chaque sprint.

4.1. Outils utilisés

_ **JHipster** : « Java Hipster », est un générateur d'applications pratique qui crée des applications Spring Boot et AngularJS / React.

_ **PostgreSQL** : est un système de gestion de base de données objet et relationnelle. Il prend en charge SQL pour les requêtes relationnelles et JSON pour les requêtes non relationnelles.

_ **Liquibase** : est l'un des outils les plus polyvalents pour la migration de bases de données. Il fonctionne sur presque toutes les plates-formes de base de données SQL.

_ **Faker** : est une librairie javascript qui permet de générer des fausses données et qui nous a permis d'implémenter notre base de données pour une meilleure visualisation.

_ **Spring Boot** : est un projet qui repose sur le framework Spring. Il offre un moyen plus simple et plus rapide d'installer, de configurer et d'exécuter des applications simples et basées sur le Web.

_ **React** : est une librairie javascript qui sert à créer des interfaces utilisateur.

_ **Git** : Git est un outil DevOps utilisé pour la gestion du code source. Il s'agit d'un système de contrôle de version gratuit et open source utilisé pour gérer efficacement les petits et très grands projets.

_ **GitHub** : est un service d'hébergement de projet en ligne.

_ **Azure DevOps** : anciennement connu sous le nom de Visual Studio Team Services, est un service hébergé fournissant un outil de développement et de collaboration.

_ **Creately** : est un outil de création de diagrammes basé sur le Web qui peut être utilisé pour dessiner des organigrammes, des cadres filaires, des maquettes, des diagrammes UML, des cartes mentales, des organigrammes, des infographies, des diagrammes de réseau et bien d'autres types de diagrammes.

_ **ESLint** : est un outil qui permet d'analyser et de détecter les erreurs et problèmes du code javascript pour assurer un code propre.

_ **Axios** : est une librairie javascript qui joue le rôle d'un client HTTP et qui permet d'envoyer des requêtes de ce même protocole.

_ **Visual Studio Code** : est un éditeur de code source léger mais puissant qui s'exécute sur le bureau et est disponible pour Linux, macOS et Windows.

_ **Eclipse IDE for Enterprise Java Developers** : est un logiciel qui fournit un environnement de développement et des outils pour créer une application Jakarta EE ou une application Web.

_ **Docker** : est une plateforme open source pour la création, le déploiement et la gestion d'applications conteneurisées. Il permet une meilleure portabilité et résout les problèmes de compatibilité avec l'environnement qui se lance dans des conteneurs avec toutes les dépendances de l'application ou d'un ensemble de services. Les conteneurs imitent le fonctionnement d'une VM avec une consommation très réduite d'énergie ce qui représente l'un des plus grands avantages de docker et qui a permis de le rendre populaire.

_ **JavaScript** : JavaScript est un langage de script qui vous permet d'interagir avec divers éléments d'une page Web en temps réel. Il a la particularité de s'exécuter directement sur le navigateur web et d'être un langage interprété.

_ **TypeScript** : est un langage de développement Javascript. Il est compilé statiquement pour écrire du code Javascript clair, simple et très bien structuré pour plus de sécurité.

_ **ECMAScript** : est un ensemble de normes qui sont largement utilisées sur le Web pour les langages script comme JavaScript.

_ **JavaScript Object Notation** : est un format standardisé couramment utilisé pour transférer des données sous forme de texte pouvant être envoyé sur un réseau. Il est utilisé par de nombreuses API et bases de données et il est facile à lire autant pour les humains que pour les machines.

_ **L'HyperText Markup Language** : c'est un langage pour décrire des pages Web en utilisant un simple texte avec des balises.

_ **Les feuilles de style en cascade** : CSS en anglais pour Cascading Style Sheets, est un langage informatique pour définir l'apparence et le style d'un site Web.

_ **Sass** : est un langage de script pré-processeur qui peut être interprété ou compilé en CSS. C'est une version plus stable et plus puissante de CSS qui décrit le style de tout document d'une manière structurée.

_ **Java** : est un langage de POO (Programmation orientée objet) d'Oracle.

_ **JEE** : est un ensemble de spécifications offertes par Eclipse Foundation pour les fonctionnalités d'entreprise.

_ **Hibernate** : est un framework ORM open source qui permet de persister les objets java dans la base de données relationnelle.

_ **JPA** : est une des spécifications de Jakarta EE qu'implémente hibernate et d'autres framework ORM, elle leur définit un ensemble d'interfaces ou de règles pour qu'ensuite ils puissent bien gérer la gestion de correspondance entre les objets classes Java et les tables d'une base de données.

_ **Bootstrap** : est un framework qui aide à concevoir des sites Web plus rapidement et plus facilement. Il comprend des modèles de style basés sur HTML et CSS pour la typographie, les formulaires, les boutons, les tableaux, la navigation, etc. Il permet également de prendre en charge les plugins JavaScript.

_ **Prime React** : est une bibliothèque de composants React gratuite. Il comprend plus de 60 composants et est publié sous la licence MIT permissive.

_ **Swagger** : est une collection de ressources HTML, Javascript et CSS qui génèrent dynamiquement une belle documentation à partir d'une API.

_ **Postman, Inc.** : est une plateforme pour les tests des requêtes http et le bon développement d'API.

_ **JSON Web Token (JWT)** : est un jeton (token) composé de trois parties, la première qui contient l'encodage, la deuxième qui contient les données et la troisième qui concerne la clé de sécurité. Ce jeton permet les échanges entre client et serveur ou entre serveurs de manière sécurisée. La vérification de la clé secrète s'effectue par l'algorithme HMAC ou RSA.

_ **REST API** : est une API qui utilise le modèle de conception architecturale REST. Ce dernier se base sur une communication client serveur à l'aide de requêtes http et de réponses en JSON ou parfois CSV ou même RSS.

4.2. Imprimés écran des sprints

SPRINT 0 – « Préparation de l'environnement »

Les interfaces du sprint 0 sont toutes générées par JHipster.

D'abord, la première page à laquelle on accède est la page d'accueil de notre application comme le montre la capture suivante :

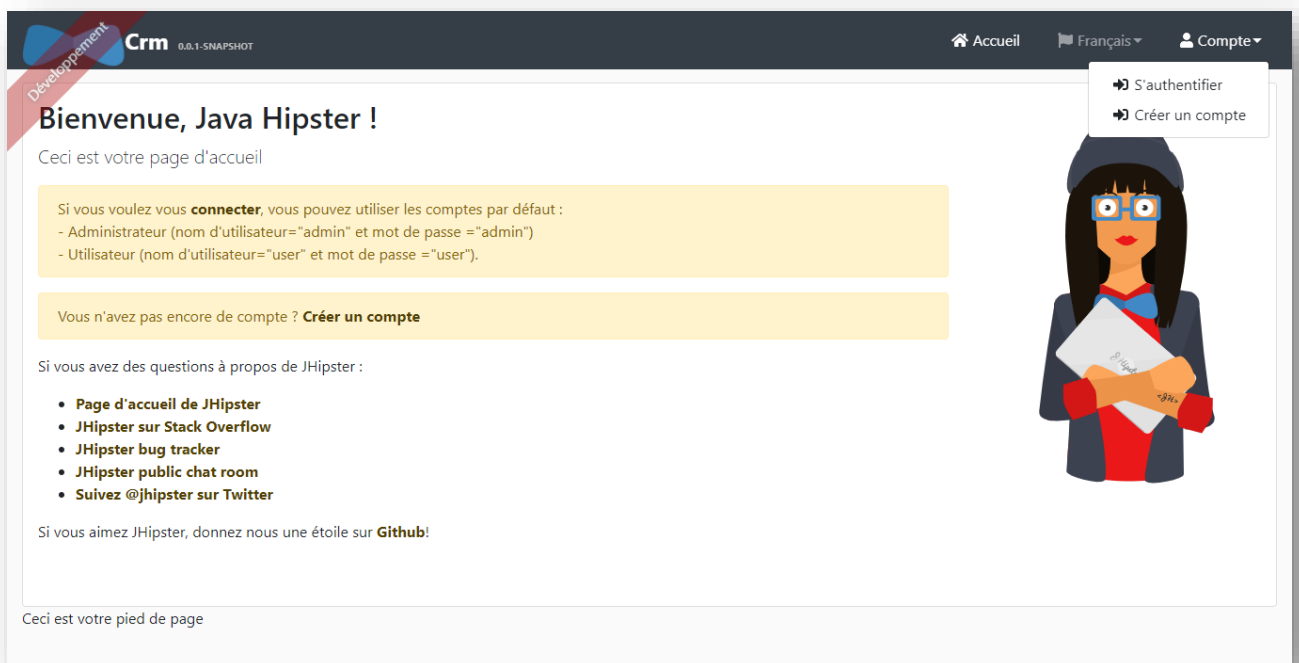


Figure 39 : Page d'accueil de l'application

(Cette image est notre réalisation)

Ici l'utilisateur dispose d'une barre de navigation qui représente le menu pour cette interface et toutes les autres. Il peut changer la langue grâce à l'internationalisation offerte par notre application et bien sûr se connecter à son compte.

Deux types d'acteurs peuvent se connecter à notre application, l'utilisateur simple qui est le collaborateur et le manager qui est l'administrateur. L'authentification si correcte renvoie vers l'interface de l'admin si l'utilisateur en est un :

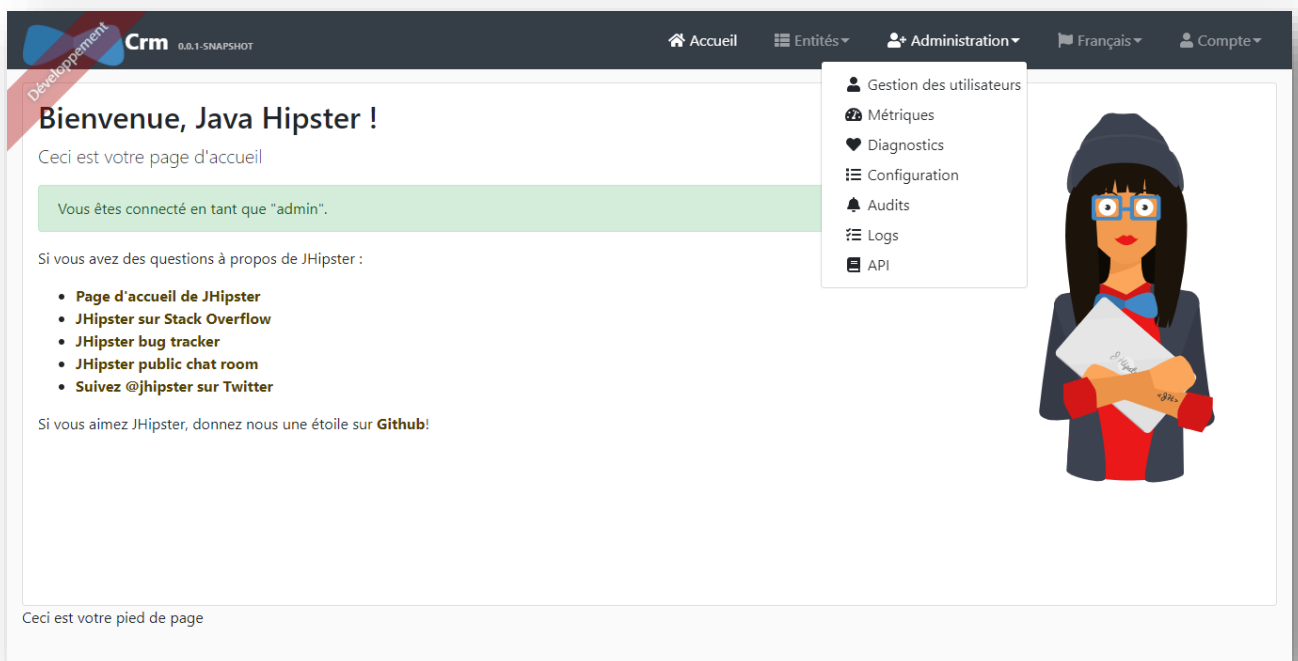


Figure 40 : Page authentifié en tant qu'administrateur

(Cette image est notre réalisation)

Et donc, après connexion l'administrateur peut accéder aux deux nouveaux menus « Entités » et « Administration » et la fonctionnalité de gérer son profil dans le menu compte. Le premier est celui des entités où on peut les visualiser en détail, les ajouter, modifier et supprimer. Tandis que le deuxième permet l'administration de l'application qui représente la seule différence entre un utilisateur simple et un administrateur. Ce dernier peut alors gérer les utilisateurs simples en les activant ou désactivant, les visualiser en détail, les ajouter, modifier et les supprimer physiquement comme figuré en bas :

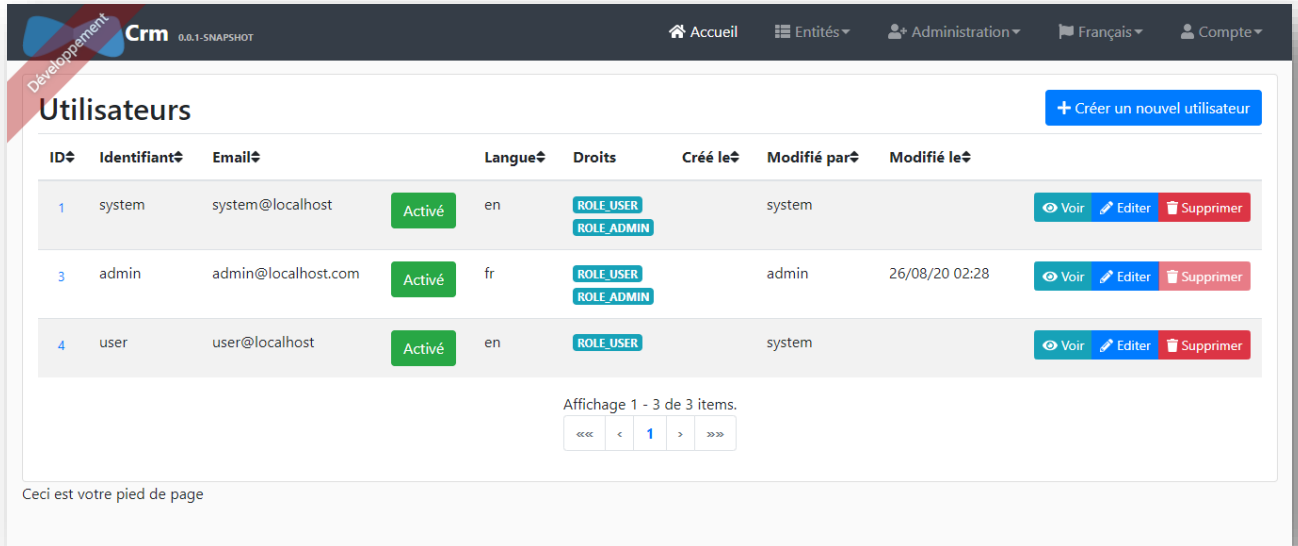


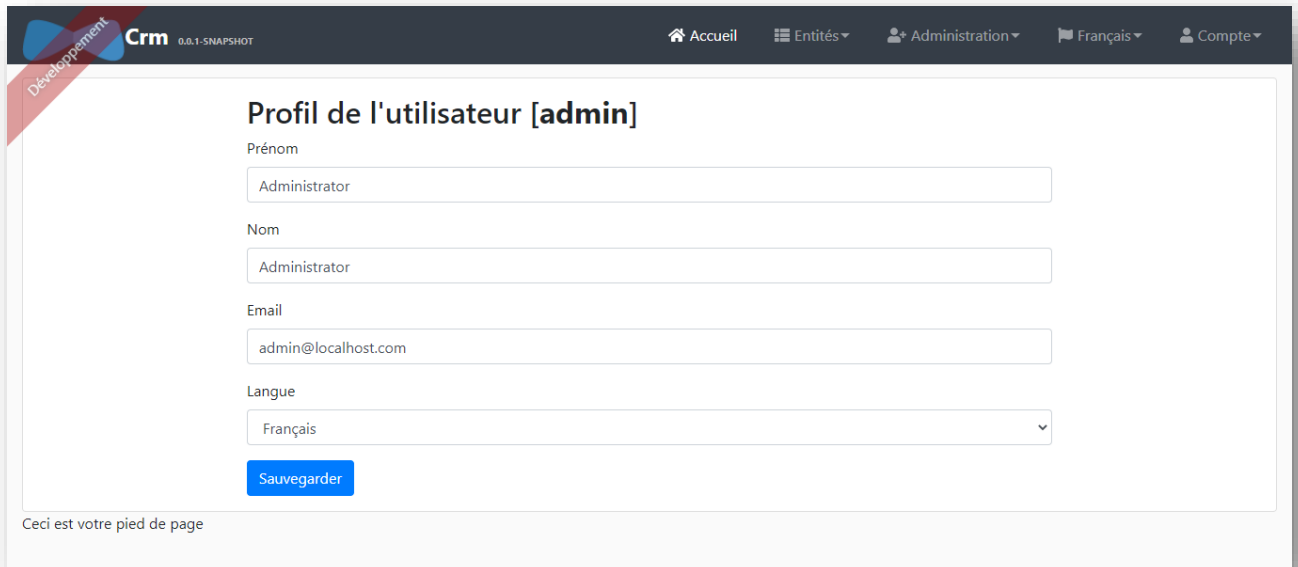
Figure 41 : Gestion des utilisateurs

(Cette image est notre réalisation)

Il peut aussi visualiser les métriques de l'application soit tout ce qui est statistiques, temps de réponses, mémoire etc...

Le manager peut même diagnostiquer la santé de l'application au niveau de la base de données de l'espace et du ping, accéder rapidement aux configurations de l'application, surveiller les audits de l'application en termes de connexions entrantes, vérifier les journaux (loggers) concernant les fichiers ainsi que la partie technique de l'application, et se documenter sur le requêtage de l'API grâce à Swagger.

Chaque utilisateur connecté peut modifier les options de son profil dans le menu compte :



The screenshot shows a web application interface for user profile management. The header includes a 'Crm' logo, version '0.0.1-SNAPSHOT', and navigation links: 'Accueil', 'Entités', 'Administration', 'Français', and 'Compte'. The main content area is titled 'Profil de l'utilisateur [admin]'. It contains four input fields: 'Prénom' (Administrator), 'Nom' (Administrator), 'Email' (admin@localhost.com), and 'Langue' (Français). A blue 'Sauvegarder' button is located below the 'Langue' field. A red banner in the top left corner reads 'Développement'. The footer text says 'Ceci est votre pied de page'.

Figure 42 : Gestion du profil d'un utilisateur connecté

(Cette image est notre réalisation)

Ces options (prénom, nom, email, langue) restent alors enregistrées dans la base de données pour sa prochaine connexion après les avoir sauvegardées grâce au bouton du formulaire. L'utilisateur doit entrer un email avec un format valide sinon il recevra une erreur et ces informations ne pourront pas être sauvegardées.

Toutes les entités sont gérées de la même manière et donc ont les mêmes interfaces. Lorsque l'utilisateur choisi une parmi le menu des entités elle est alors listée comme suit :

Degrés

ID	Label	Description	Année de graduation	Collaborateur	
1	Alabama Rest	JHipster is a development platform to generate, develop and d Angular / React / Vue Web applications and Spring microservices.	06/08/2020		Voir Editer Supprimer
2	Cotton Producer	JHipster is a development platform to generate, develop and deploy Spring Boot + Angular / React / Vue Web applications and Spring microservices.	07/08/2020		Voir Editer Supprimer
3	payment	JHipster is a development platform to generate, develop and deploy Spring Boot + Angular / React / Vue Web applications and Spring microservices.	07/08/2020		Voir Editer Supprimer
4	Innovative Avon Rubber	JHipster is a development platform to generate, develop and deploy Spring Boot + Angular / React / Vue Web applications and Spring microservices.	07/08/2020		Voir Editer Supprimer
5	Toys auxiliary Ohio	JHipster is a development platform to generate, develop and deploy Spring Boot + Angular / React / Vue Web applications and Spring microservices.	07/08/2020		Voir Editer Supprimer
6	parsing Rubber Money Market Account	JHipster is a development platform to generate, develop and deploy Spring Boot + Angular / React / Vue Web applications and Spring microservices.	07/08/2020		Voir Editer Supprimer
7	Fantastic Plastic Chips non-volatile	JHipster is a development platform to generate, develop and deploy Spring Boot + Angular / React / Vue Web applications and Spring microservices.	07/08/2020		Voir Editer Supprimer
8	transmitting	JHipster is a development platform to generate, develop and deploy Spring Boot + Angular / React / Vue Web applications and Spring microservices.	07/08/2020		Voir Editer Supprimer
9	AGP Gorgeous	JHipster is a development platform to generate, develop and deploy Spring Boot + Angular / React / Vue Web applications and Spring microservices.	07/08/2020		Voir Editer Supprimer
10	Home Loan Account	JHipster is a development platform to generate, develop and deploy Spring Boot + Angular / React / Vue Web applications and Spring microservices.	07/08/2020		Voir Editer Supprimer

Ceci est votre pied de page

Figure 43 : Liste de l'entité degrés

(Cette image est notre réalisation)

Chaque entité listée dispose de 3 liens.

_ Le premier permet d'afficher la fiche de l'entité avec ses détails :

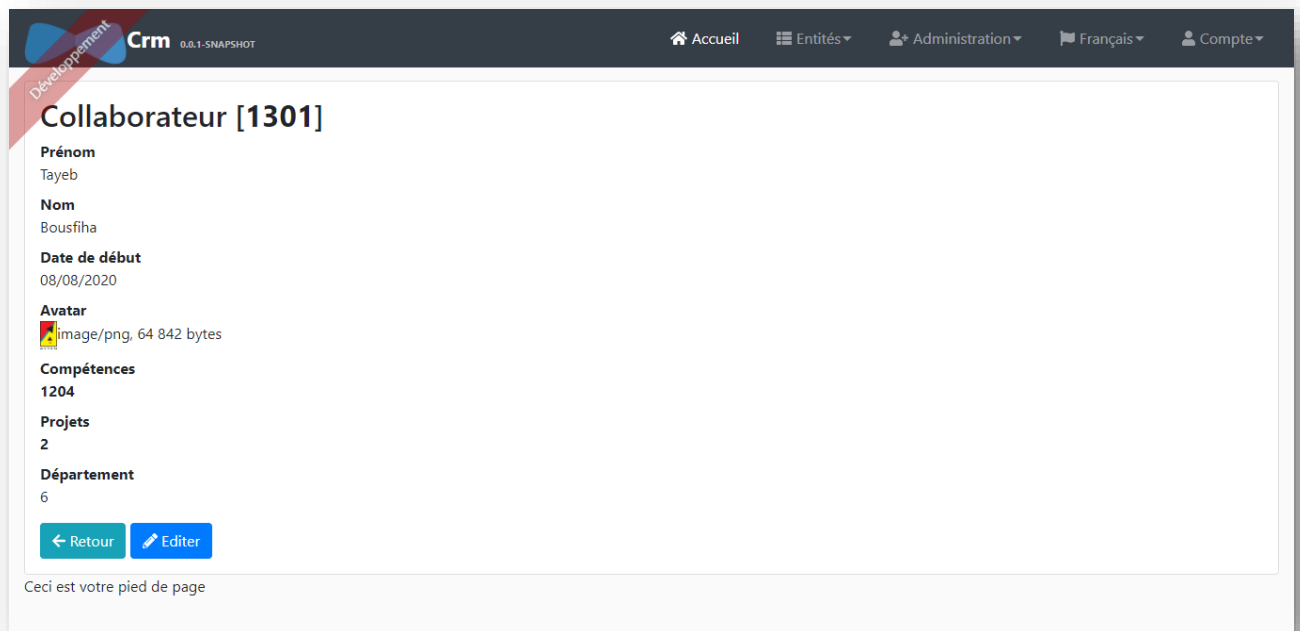
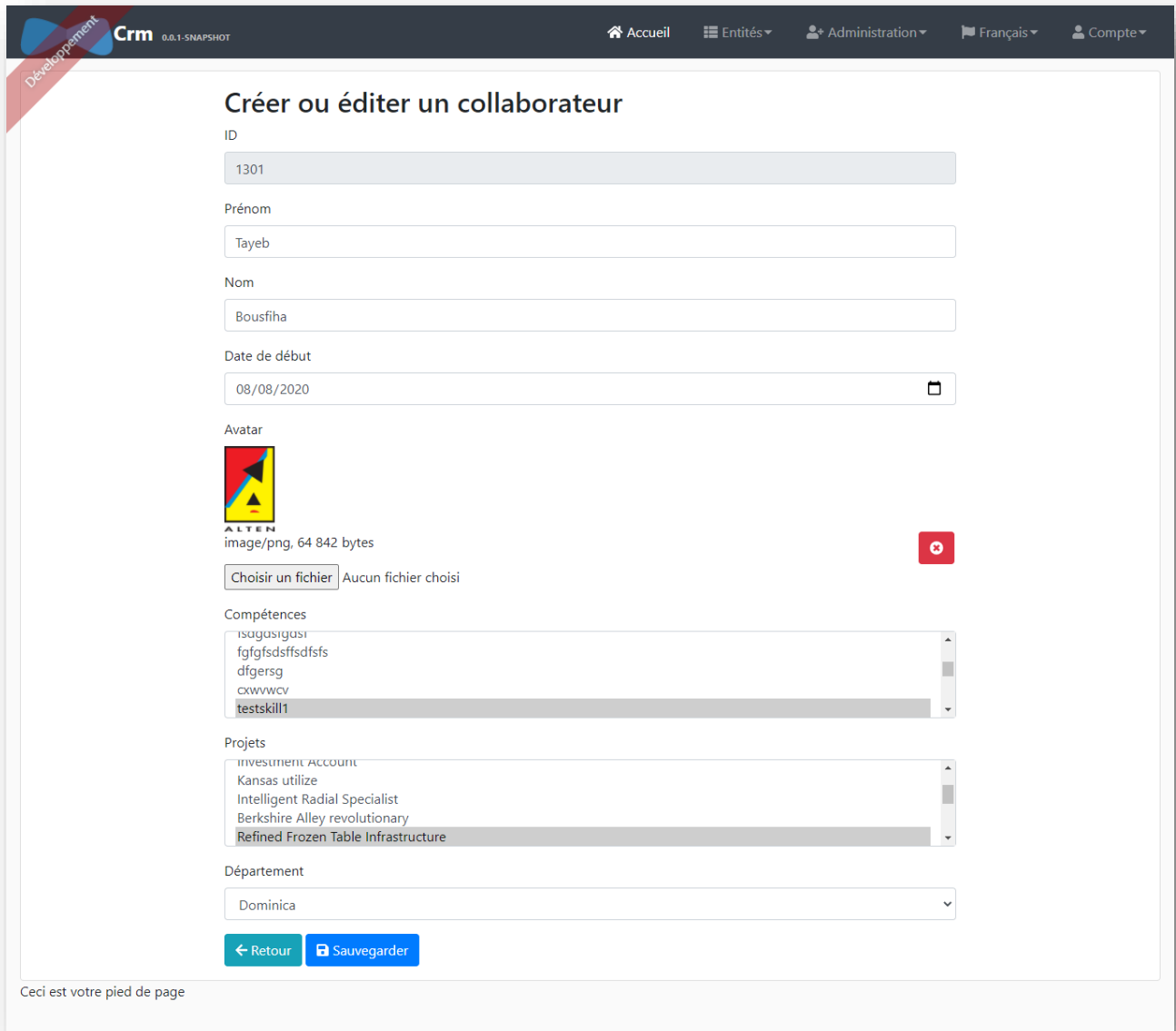


Figure 44 : Fiche de l'entité collaborateur

(Cette image est notre réalisation)

Voici la fiche de l'entité collaborateur avec ses détails. Dans l'interface de la fiche d'une entité, l'utilisateur peut la vérifier, la modifier ou retourner à la page précédente.

_ Le deuxième lien permet de modifier une entité :




Créer ou éditer un collaborateur

ID
1301

Prénom
Tayeb

Nom
Bousfiha

Date de début
08/08/2020

Avatar

image/png, 64 842 bytes
Choisir un fichier Aucun fichier choisi

Compétences
isugusrgusr
fgfgdsffdsfs
dfgersg
cxwwvcv
testskill1

Projets
Investment Account
Kansas utilize
Intelligent Radial Specialist
Berkshire Alley revolutionary
Refined Frozen Table Infrastructure

Département
Dominica

[← Retour](#) [Sauvegarder](#)

Ceci est votre pied de page

Figure 45 : modification d'une entité collaborateur

(Cette image est notre réalisation)

Cette page récupère d'abord les informations de l'entité sélectionnée qui sont ensuite attribuées à leurs champs adéquats. On peut alors modifier tous types d'informations (texte, image, date...), choisir une valeur ou des valeurs pour un

attribut comme pour Projets et Département dans la figure 33 et aussi saisir la date grâce à un calendrier. Les modifications ainsi faites l'utilisateur n'a qu'à les sauvegarder en cliquant sur le bouton adéquat ou s'il change d'avis et veut annuler ses changements de cliquer sur retour pour revenir à la page précédente. L'interface modification et ajout d'une entité sont identiques sauf que les champs sont vides pour celle de l'ajout :

Créer ou éditer un collaborateur

Prénom

Nom

Date de début

Avatar
 Aucun fichier choisi

Compétences

wfgfdgdf

fgrsgtdgdf

dgdsgsdft

Projets

contextually-based Home Loan Account Books

Investment Account

Kansas utilize

Département

Ceci est votre pied de page

Figure 46 : Ajout d'une entité collaborateur

(Cette image est notre réalisation)

Puisqu'il s'agit de la même interface elle dispose alors du même fonctionnement. On peut y accéder à partir du bouton ajouter présent dans chaque liste d'entités.

_ Le troisième lien d'une entité listée est la suppression physique de la base de données :

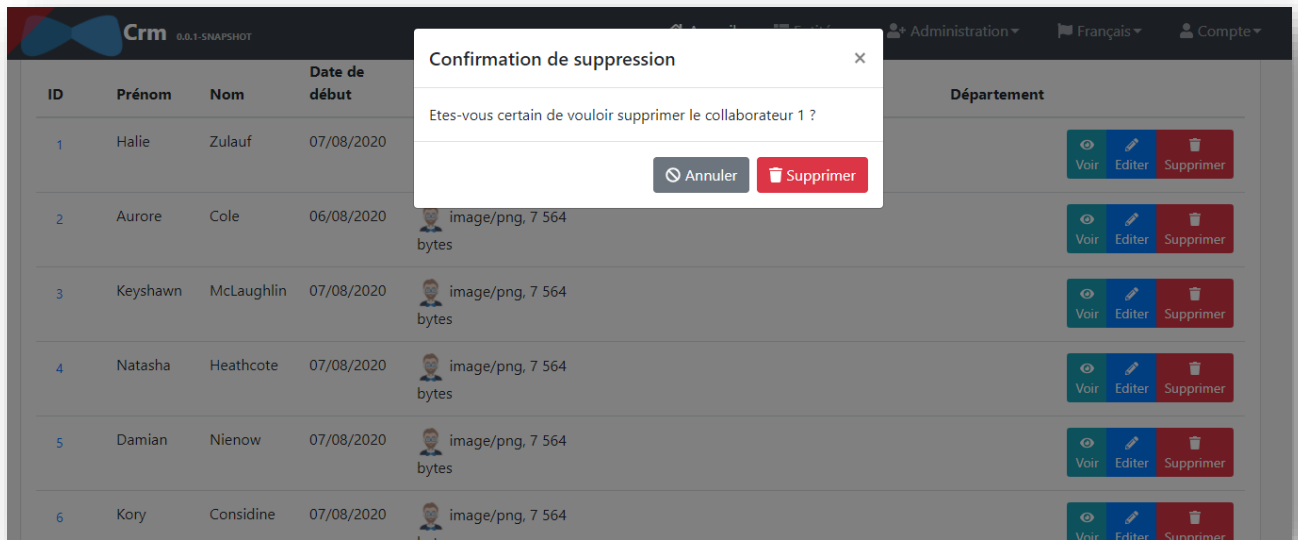


Figure 47 : Pop-up de la confirmation de suppression physique d'une entité

(Cette image est notre réalisation)

Pour sécuriser et éviter les fautes d'inattention, un pop-up de confirmation de suppression physique d'une entité a été mis en place. Il apparaît juste après avoir cliqué sur le bouton supprimer et propose deux nouvelles options, la suppression ou l'annulation.

SPRINT 1 – « Manager : Gestions des compétences »

Dans cette partie, on présente les changements réalisés et les fonctionnalités ajoutées du sprint 1 dans les interfaces destinées aux utilisateurs.

On commence par l’affichage de la liste des compétences :

Compétences + Créer une nouvelle compétence

Label Code Technologie Catégorie Tags

ID	Label	Code	Description	
1001	javascript	JS	JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une partie essentielle des applications web.	Voir Editer Supprimer
1002	postgresql	PSQL	PostgreSQL est un système de gestion de base de données relationnelle et objet.	Voir Editer Supprimer
1003	hypertext preprocessor	PHP	PHP: Hypertext Preprocessor, plus connu sous son sigle PHP, est un "langage de programmation" libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet.	Voir Editer Supprimer
1004	symfony	SF	Symfony est un ensemble de composants PHP ainsi qu'un framework MVC libre écrit en PHP.	Voir Editer Supprimer
1005	laravel	LRVL	Laravel est un framework web open-source écrit en PHP respectant le principe modèle-vue-contrôleur et entièrement développé en programmation orientée objet.	Voir Editer Supprimer
1006	react	RT	React est une librairie javascript qui sert à créer des interfaces utilisateur.	Voir Editer Supprimer
1007	java	JV	Java est une technique informatique.	Voir Editer Supprimer
1008	nodejs	NJ	Node.js est une plateforme logicielle libre en JavaScript orientée vers les applications réseau événementielles hautement concurrentes qui doivent pouvoir monter en charge.	Voir Editer Supprimer
1009	angular	NG	Angular est un cadriciel côté client, open source, basé sur TypeScript, et co-dirigé par l'équipe du projet « Angular » à Google et par une communauté de particuliers et de sociétés.	Voir Editer Supprimer
1010	ionic	ION	Ionic est un framework open-source créé en 2013 par Max Lynch, Ben Sperry, et Adam Bradley.	Voir Editer Supprimer

Affichage 11 - 20 de 22 items.

Ceci est votre pied de page

Figure 48 : Affichage de la liste des compétences du sprint 1

(Cette image est notre réalisation)

Dans cet affichage on remarque dans un premier temps qu'il dispose d'un système de pagination limité à 10 éléments par page ce qui permet à l'application de ne charger que les données d'une seule page à la fois et à l'utilisateur une expérience plus claire, rapide et présentable. L'utilisateur peut bien sûr naviguer dans la page souhaitée si elle est proche de celle actuelle ou bien accéder directement à la première ou dernière via les boutons et chevrons disponibles dans la barre de pagination. Ensuite il y a le tri de l'entité selon ses propriétés, si une parmi ces dernières contient du texte et que l'utilisateur la trie en cliquant sur son nom avec les chevrons haut et bas qui signifient l'arrangement, elle sera alors en ordre alphabétique, si c'est elle est numérique elle sera alors en ordre croissant ou décroissant. Le sens du groupement change si l'utilisateur clique encore une fois. Pour le cas d'une propriété qui contient une liste, JHipster ne le prend pas en compte et doit être fait manuellement.

De plus, il y a une barre de recherche en haut qui permet de filtrer les éléments d'une entité avec ses attributs. Il suffit à l'utilisateur de taper 3 caractères pour que la page n'affiche que les résultats qui contiennent la valeur tapée dans la barre de recherche. Après le filtrage si l'utilisateur souhaite annuler la recherche par filtre il n'a qu'à effacer le contenu de la barre.

Par défaut, la recherche se fait par label mais si l'utilisateur veut changer le filtre il n'a qu'à cliquer sur le bouton du filtre souhaité. Si jamais l'utilisateur veut filtrer par un attribut avec relation ManyToMany comme pour les filtres « Catégories » et « Tags » dans notre cas alors la barre de recherche est remplacée par un select multiple où il pourra sélectionner une ou plusieurs valeurs et ainsi filtrer l'entité présente avec les éléments de l'autre :

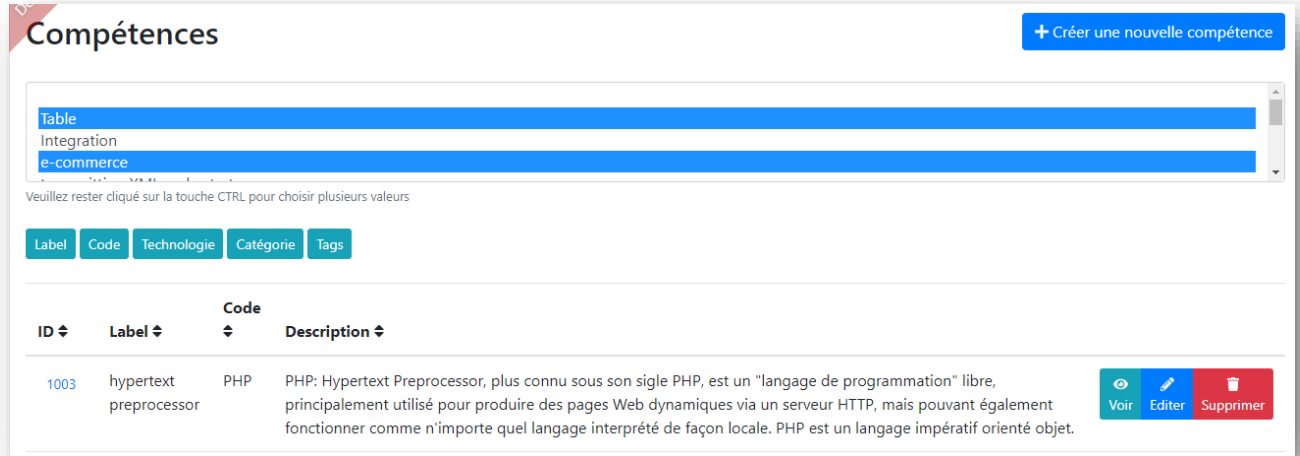


Figure 49 : filtrage avec selection multiple

(Cette image est notre réalisation)

Si l'utilisateur souhaite encore une fois revenir à l'affichage sans filtre, cette fois il n'a qu'à choisir seulement la première option vide.

Puis, dans la liste, seulement 4 attributs de l'entité « compétences » sont affichés qui sont l'ID, le label, le code et la description. Pour visualiser la fiche détaillée de chaque compétence il suffit de cliquer sur le bouton « voir » de celle-ci.

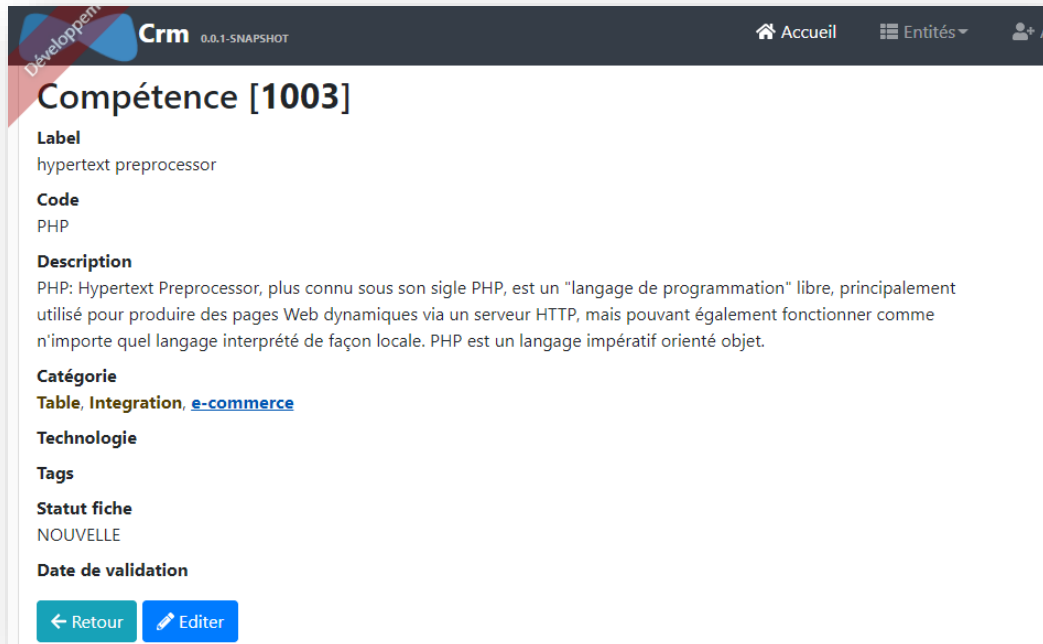
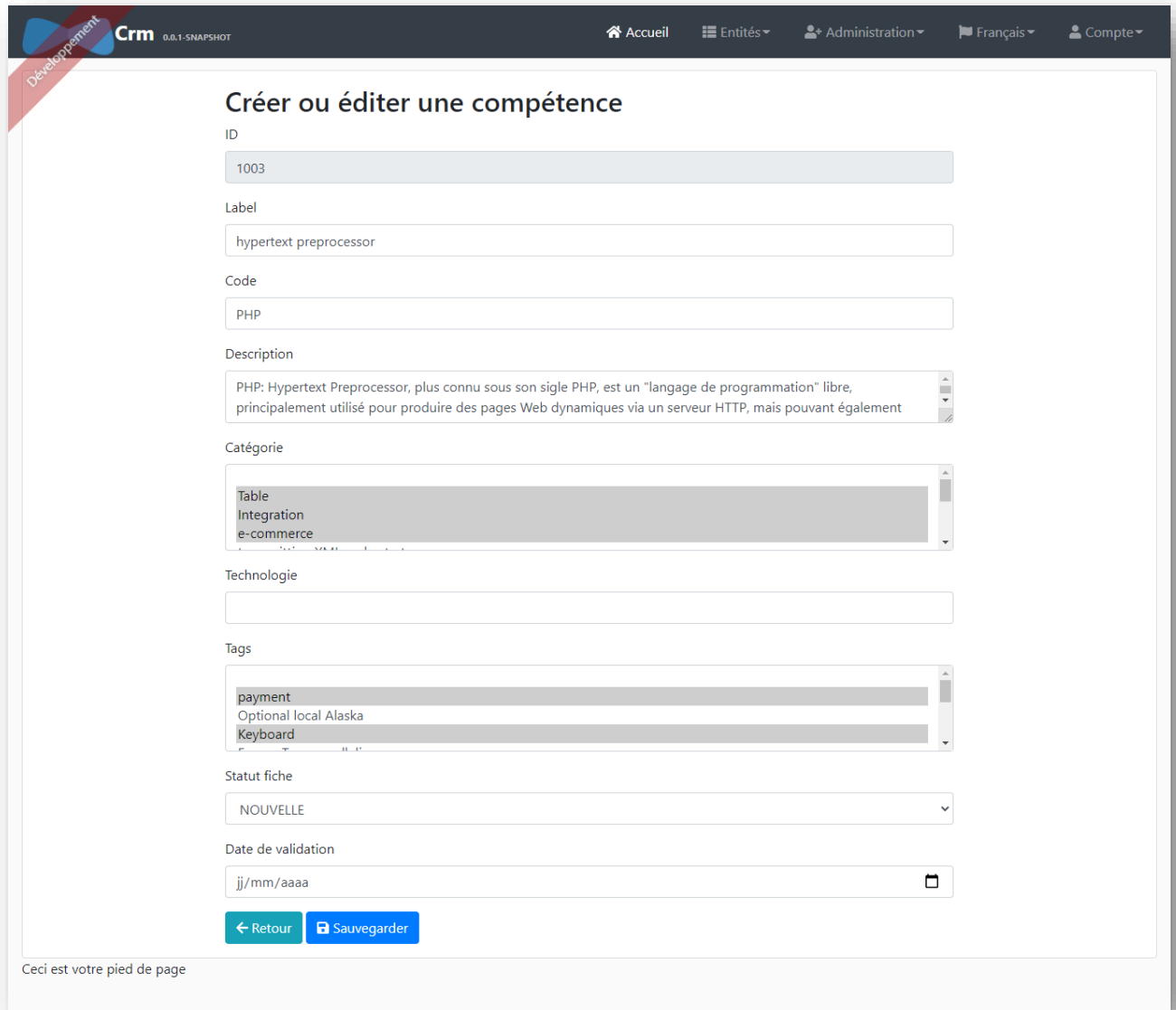


Figure 50 : fiche détaillée d'une compétence

(Cette image est notre réalisation)

La fiche d'une compétence contient toutes ses propriétés avec leurs valeurs, chaque nom d'attribut est traduit même pour la valeur de l'énumération « Statut fiche ». Les propriétés avec relation, sous forme de liste, contiennent des liens qui mènent vers chaque fiche de chaque entité liée. Le bouton « Retour » permet de retourner dans la page précédente de l'historique de React et le bouton « Éditer » pour modifier la fiche comme le montre la figure suivante :



Créer ou éditer une compétence

ID
1003

Label
hypertext preprocessor

Code
PHP

Description
PHP: Hypertext Preprocessor, plus connu sous son sigle PHP, est un "langage de programmation" libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également

Catégorie
Table
Integration
e-commerce

Technologie

Tags
payment
Optional local Alaska
Keyboard

Statut fiche
NOUVELLE

Date de validation
jj/mm/aaaa

[← Retour](#) [Sauvegarder](#)

Ceci est votre pied de page

Figure 51 : modification d'une compétence

(Cette image est notre réalisation)

L'utilisateur peut modifier chaque champ même le statut de la fiche et la date de validation qui ne sont pas disponible lors de l'ajout de l'entité.

Créer ou éditer une compétence

Label

Ce champ est obligatoire.

Code

Ce champ est obligatoire.

Description

Catégorie

Table
Integration
e-commerce

Ce champ est obligatoire.

Technologie

Tags

payment
Optional local Alaska
Keyboard

← Retour Sauvegarder

Ceci est votre pied de page

Figure 52 : champs requis de l'ajout d'une compétence

(Cette image est notre réalisation)

Lors de l'ajout, si l'utilisateur ne remplit pas les champs obligatoires il ne pourra pas sauvegarder sa compétence. On contrôle aussi le cas où l'utilisateur ajoute une compétence qui existe déjà avec le même label en ignorant la casse (c'est-à-dire qu'on ignore les majuscules), il recevra alors une erreur lui indiquant le problème qui est traduite dans les différents langages de notre internationalisation qui sont le français et l'anglais.

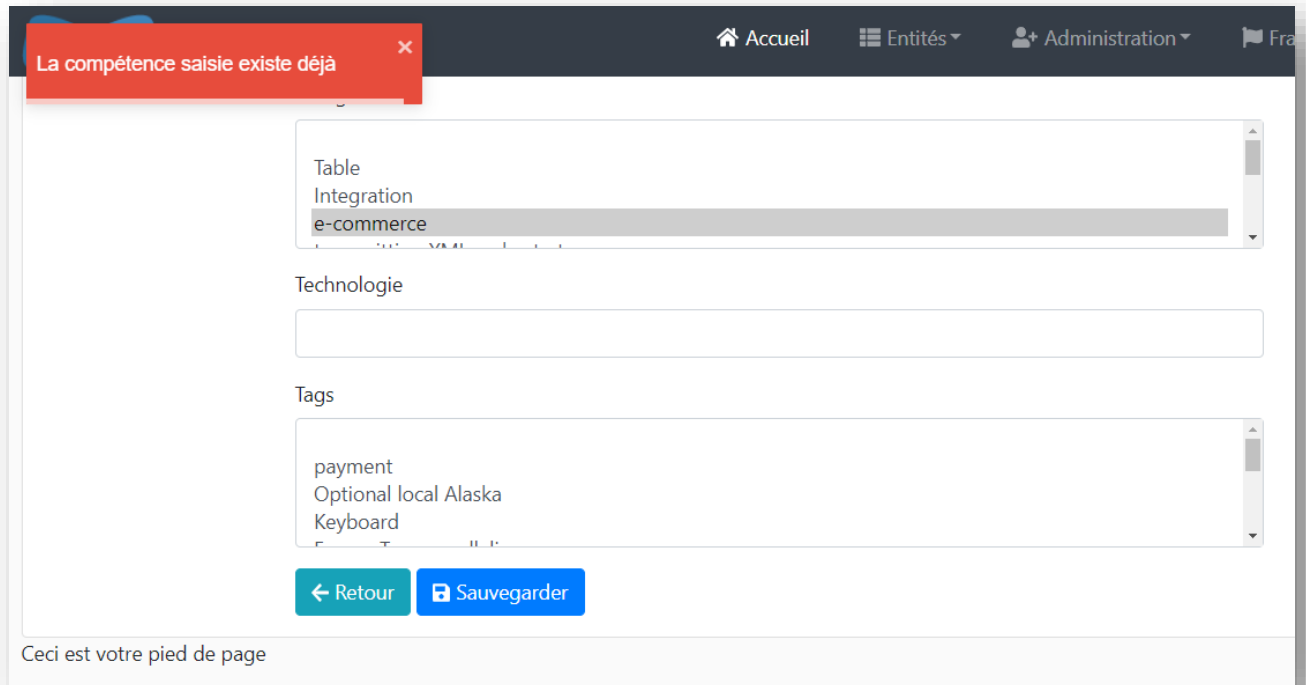


Figure 53 : Erreur lors de l'ajout d'une compétence qui existe avec le même label

(Cette image est notre réalisation)

Parfois, quand un utilisateur ne respecte pas les relations ou la validation d'une propriété et cause une erreur dans le serveur qui n'est pas encore traitée il reçoit l'erreur suivante :

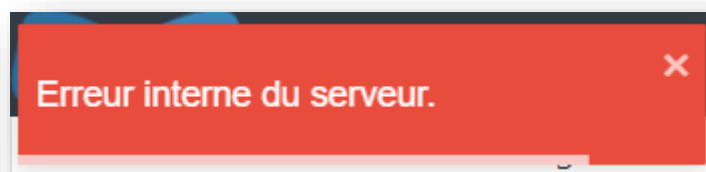


Figure 54 : Erreur interne du serveur

(Cette image est notre réalisation)

Le bouton « Créer une nouvelle compétence » se trouve dorénavant devant chaque liste déroulante des compétences pour permettre à l'utilisateur d'ajouter une compétence qui n'existe pas encore dans le système.

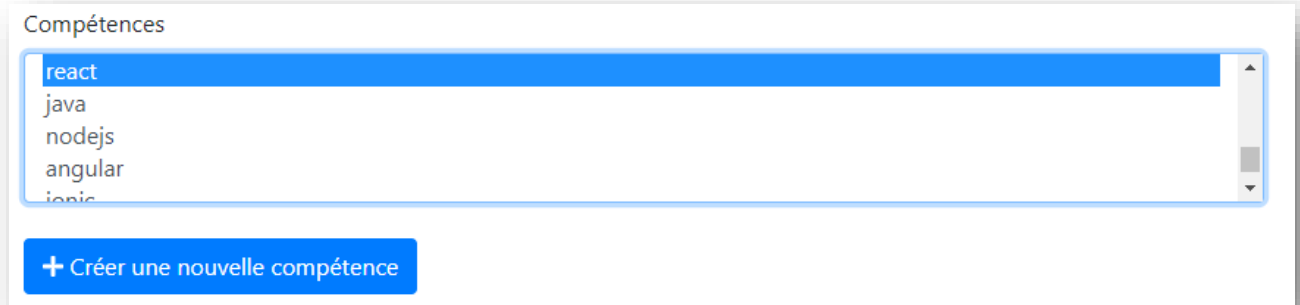


Figure 55 : liste déroulante des compétences avec le bouton d'ajout d'une nouvelle

(Cette image est notre réalisation)

Dans ce sprint, on ajoute une nouvelle entité « Tags » en relation ManyToMany avec les compétences. Elle dispose de seulement deux attributs autres que son ID, à savoir le label et le code.

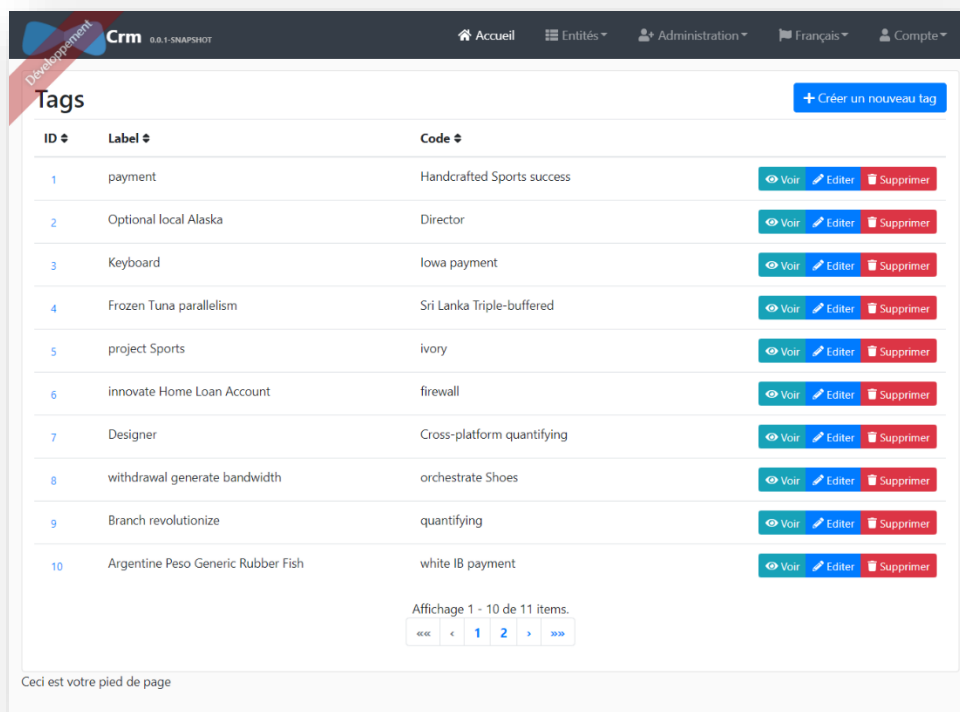


Figure 56 : affichage de la liste des tags

(Cette image est notre réalisation)

L'entité est générée avec la pagination, le tri ainsi que tout le CRUD avec les mêmes fonctionnalités que celui des compétences et ne dispose pas de filtres.

SPRINT 2 – « Manager : Gestions des compétences, collaborateurs, niveaux et catégories »

Durant le sprint 2, on refait tout le design en utilisant un nouveau qui est à la fois moderne, simple, léger et appréciable. On choisit comme design de composant « PrimeReact » et en ajoutant nos propres modifications inspirées d'autres interfaces ainsi que plusieurs fonctionnalités qu'on présente.

Voici à quoi ressemble maintenant la page de connexion :

Figure 57 : Page de connexion sprint 2

(Cette image est notre réalisation)

On peut imaginer plus tard une photo ou une couleur de fond dans l'espace à gauche pour une meilleure présentation.

Après la connexion autant que Manager on arrive sur la page d'accueil suivante avec les graphes et les statistiques de l'application comme le nombre de collaborateurs en fonction des compétences ou l'affichage des personnes disponible ou en mission par mois.

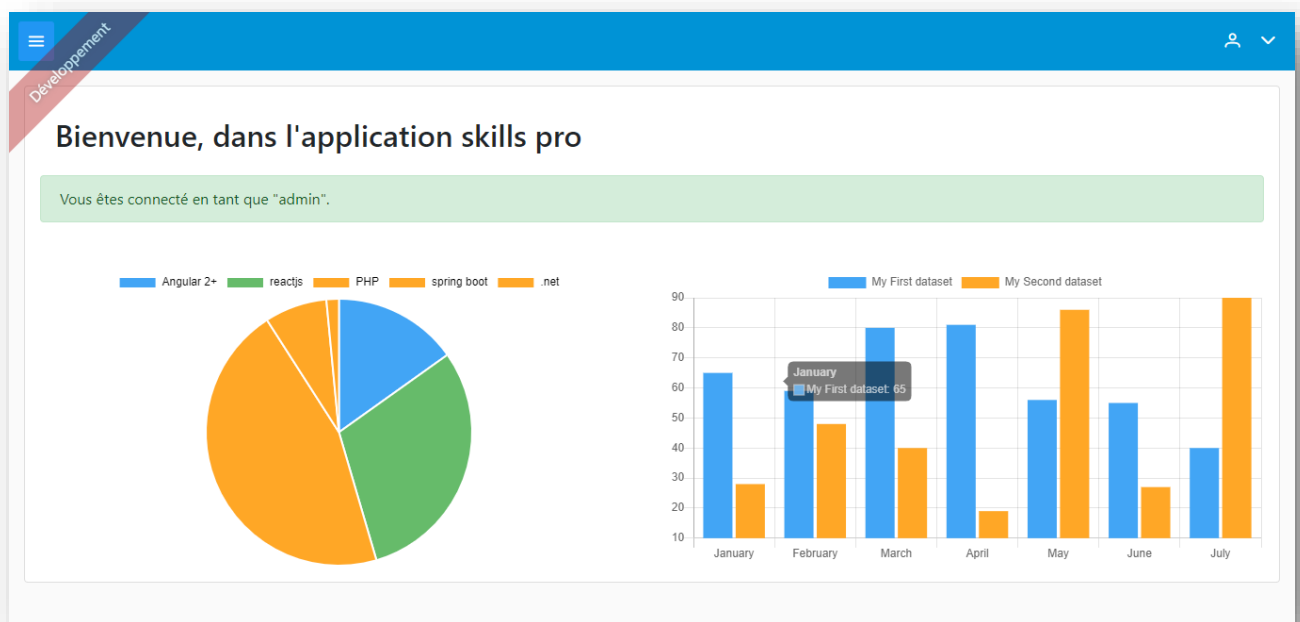


Figure 58 : Page d'accueil Manager sprint 2

(Cette image est notre réalisation)

Pour naviguer le Manager doit cliquer sur le « Burger » menu en haut à gauche.

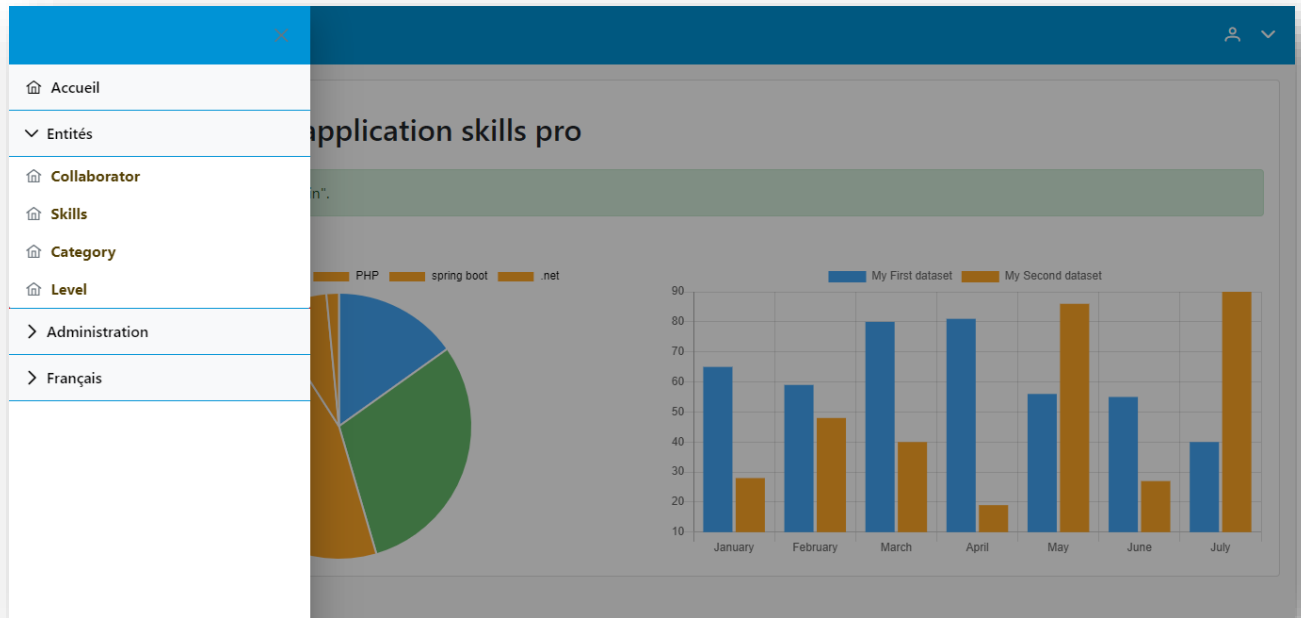


Figure 59 : Menu Manager sprint 2

(Cette image est notre réalisation)

On organise la gestion des collaborateurs avec un design léger et moderne à la fois comme le montre la figure suivante :

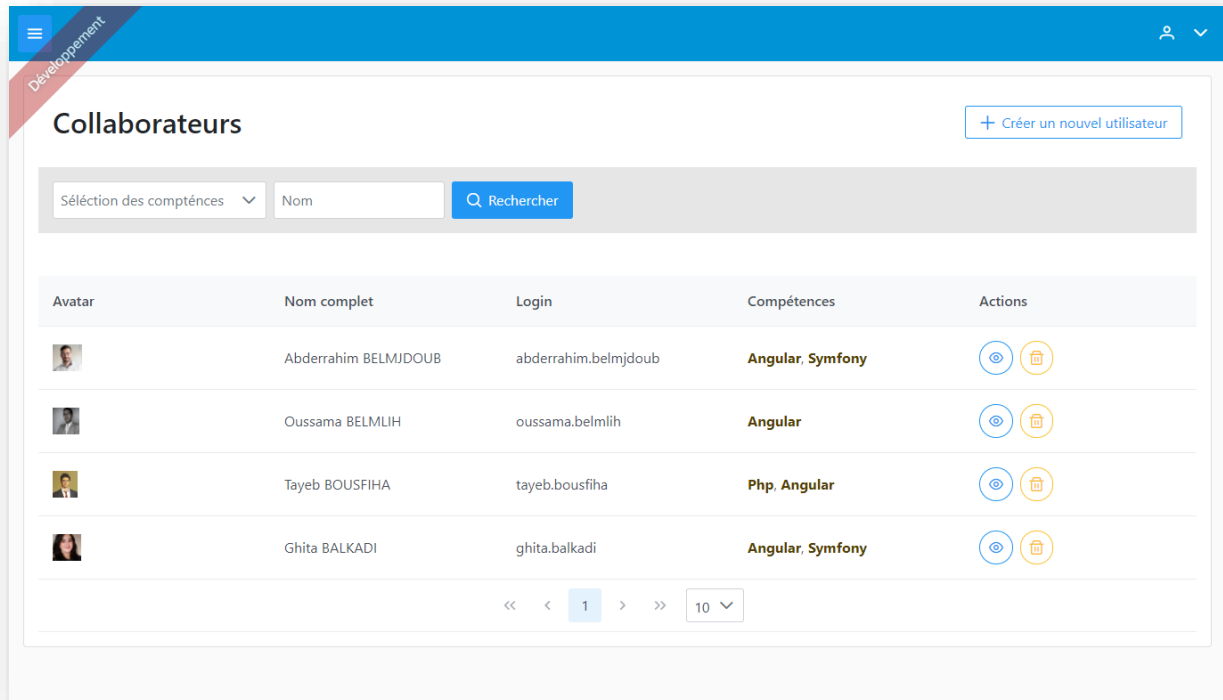


Figure 60 : Page gestion des collaborateurs sprint 2

(Cette image est notre réalisation)

Ici le manager peut chercher les profils collaborateurs par compétences :

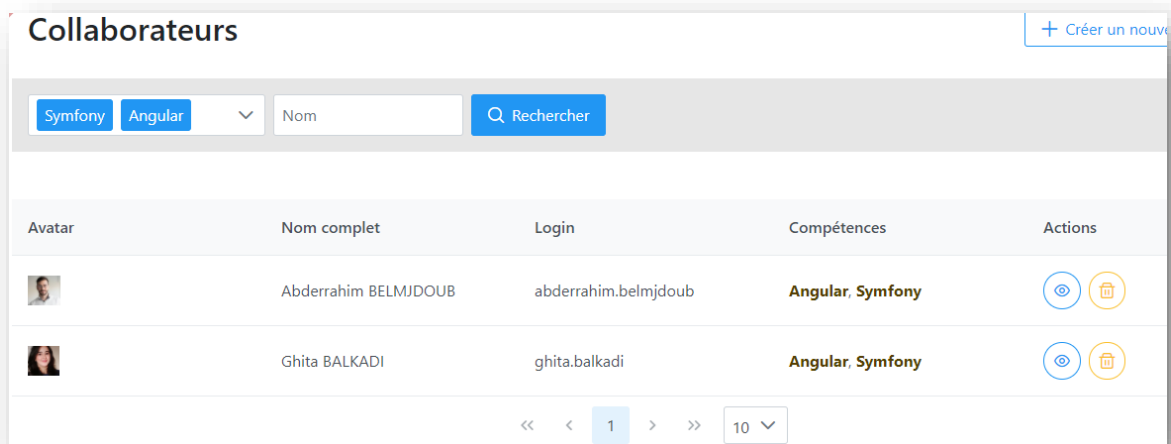


Figure 61 : Recherche de collaborateurs par compétences sprint 2































(Cette image est notre réalisation)

Il peut alors supprimer le collaborateur choisi ou visualiser sa fiche :

Figure 62 : Fiche collaborateur sprint 2

91

Le design de cette partie est inspiré du géant LinkedIn. Ensuite pour les différentes gestions de l'application on reste sur le même design :

Label	Code	Catégorie	Actions
Php	PHP	Back end	  
Angular	NG	Front end	  
Deltagen	DG	Imagerie	  
Spring	SP	Back end	  
Laravel	LRVL	Full stack	  
Java	JV	Back end	  
Symfony	SF	Full stack	  
JavaScript	JS	Full stack	  
React	REACT	Front end	  
Jakarta Enterprise Edition	JEE	Back end	  

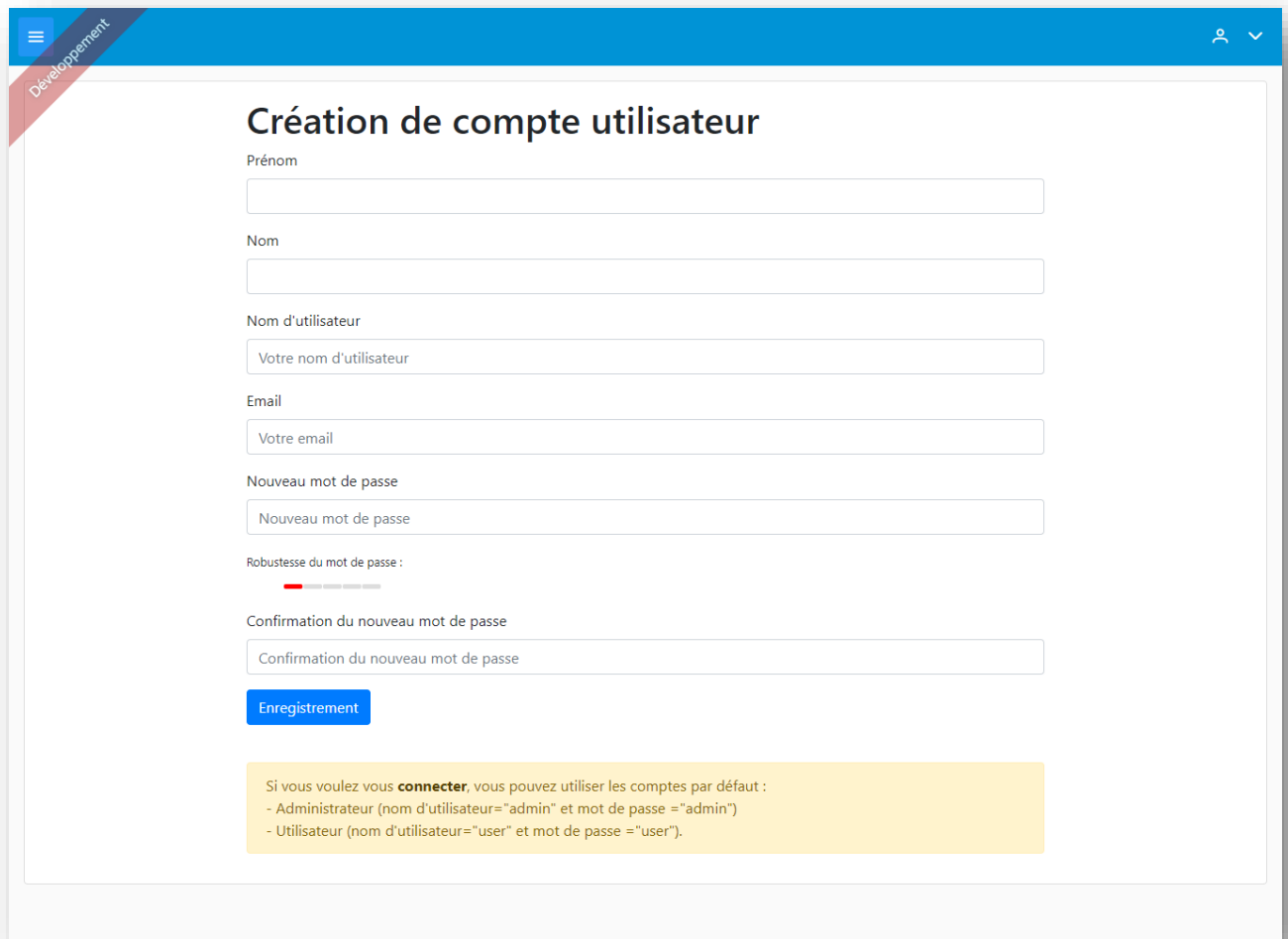
<< < 1 > >>

Figure 63 : gestion des compétences sprint 2

(Cette image est notre réalisation)

Dans la page de connexion présenté plus tôt, l'utilisateur peut soit se connecter ou s'enregistrer en passant par le workflow suivant :

D'abord, il insère ses informations dans le formulaire suivant :



The screenshot shows a web application interface for user registration. At the top left, there is a blue header bar with a hamburger menu icon and a red diagonal banner that says 'Développement'. The main content area is white and titled 'Création de compte utilisateur'. It contains several input fields: 'Prénom', 'Nom', 'Nom d'utilisateur' (with a placeholder 'Votre nom d'utilisateur'), 'Email' (with a placeholder 'Votre email'), 'Nouveau mot de passe', and 'Confirmation du nouveau mot de passe'. Below the password fields is a 'Robustesse du mot de passe' indicator showing a red bar. A blue 'Enregistrement' button is positioned below the confirmation field. At the bottom, a yellow box contains instructions for default login credentials: 'Administrateur (nom d'utilisateur="admin" et mot de passe ="admin")' and 'Utilisateur (nom d'utilisateur="user" et mot de passe ="user")'.

Figure 64 : Formulaire d'enregistrement

(Cette image est notre réalisation)

Puis, une notification le prévient qu'il a reçu un mail pour activer son compte :

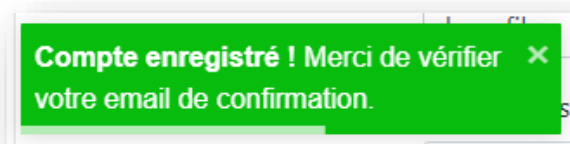


Figure 65 : notification d'activation de compte

(Cette image est notre réalisation)

Par la suite il se dirige vers sa boîte mail pour trouver le mail suivant :



Figure 66 : mail d'activation de compte

(Cette image est notre réalisation)

Pour le moment le mail ne s'affiche que dans les spams. Ensuite, l'utilisateur active son compte en cliquant sur le lien.

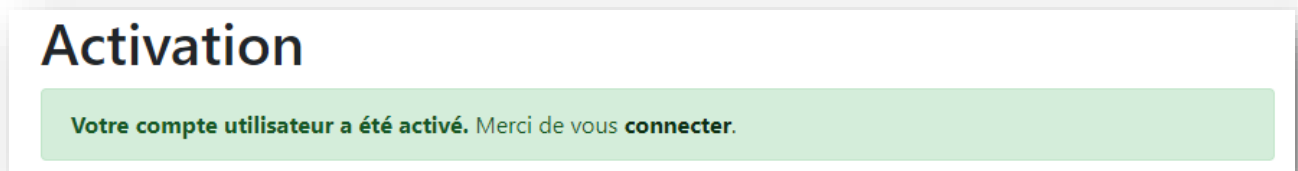


Figure 67 : message de confirmation d'activation de compte

(Cette image est notre réalisation)

Il reçoit un message de confirmation comme quoi son compte a bien été activé et se connecte après pour recevoir le message suivant :

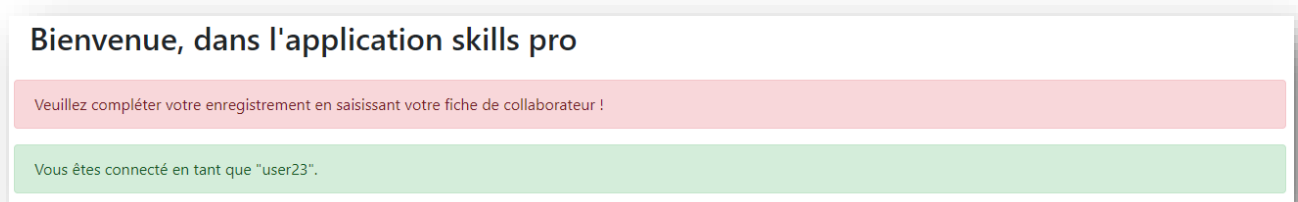


Figure 68 : message d'avertissement pour compléter la fiche du collaborateur

(Cette image est notre réalisation)

Ce message avertit le collaborateur que les informations de sa fiche ne sont pas encore complétées, il est alors amené à cliquer sur le lien « Saisir ma fiche collaborateur » disponible dans le menu, pour être redirigé vers la page ajout collaborateur, l'association entre le collaborateur et son compte se fait automatiquement au niveau front end, le formulaire est envoyé avec l'id du compte et il ne reste à l'utilisateur qu'à saisir les informations suivantes :

Créer ou éditer un collaborateur

Matricule

Situation familiale
 Sélectionnez votre situation familiale ▼

Téléphone

Date de naissance

Date de début

Avatar
 Aucun fichier choisi

Responsable hiérarchique

Date de disponibilité

Compétences

[← Retour](#) [Sauvegarder](#)

Figure 69 : Ajout d'un collaborateur sprint 2

(Cette image est notre réalisation)

Finalement, le collaborateur est ajouté dans la gestion des collaborateurs du manager et son compte à la gestion correspondante aussi.

Conclusion et perspectives

A l'issue de ce stage, on est arrivé à obtenir une vue générale sur les meilleures techniques, manipulations et procédures pratiques utilisées par des professionnels qui ont conçu JHipster pour la création dans le côté frontend des interfaces utilisateur en TypeScript avec React et dans le côté backend des services web, requête sécurisée en JPQL et API RESTFULL en JEE sur SpringBoot, commençant par le design, jusqu'au traitement java.

On a aussi appris comment répondre aux besoins d'un client en agile et à gérer les itérations tout en procurant un code propre facile à comprendre qui suit les standards pour faciliter la tâche aux futurs développeurs de l'application.

Lors de notre stage, on a surmonté toutes les difficultés grâce à de nombreuses recherches, autoformations et documentations sur le web car un bon développeur c'est une personne qui ne cesse d'apprendre chaque jour.

Plus que 80% du code est généré par JHipster, peut être qu'un jour on aura des Framework qui généreront la totalité du code ? ou bien même peut être que ça existe déjà mais payant ? il se peut que dans le futur il y aura moins de poste de développements ou bien totalement l'inverse car le monde a besoin d'innovations.

Pour ce qui est des perspectives, nous proposons les améliorations suivantes :

- _ Procurer pour le Manager un meilleur affichage des collaborateurs.
- _ Afficher les compétences du collaborateur grâce à un graphe radar.
- _ Ajouter dans les critères de recherche la disponibilité des collaborateurs et afficher ceux disponible et en intercontrat en premier.
- _ Ajouter la gestion des formations et des expériences.

Webographie

_ <https://www.udemy.com/course/react-the-complete-guide-incl-redux/>

[Auto formation React]

Date de consultation : tout au long du stage 08/07/2020 – fin décembre 2020

_ <https://openclassrooms.com/fr/courses/2035766-optimisez-votre-deploiement-en-creant-des-conteneurs-avec-docker/6211677-creez-un-fichier-docker-compose-pour-orchestrer-vos-conteneurs>

[Informations concernant docker]

Date de consultation : 23/07/2020

_ <https://www.jhipster.tech/>

[Documentation JHipster]

Date de consultation : 27/07/2020

_ <https://www.alten.fr/>

[Informations concernant « ALTEN »]

Date de consultation : 10/08/2020

_ <https://www.alten.com/fr/secteur/>

[Informations concernant « ALTEN »]

Date de consultation : 11/08/2020

_ <https://www.altenrecrute.fr/blog-alten/on-en-parle/alten-dans-le-top-100-des-entreprises-les-plus-attractives-en-france#:~:text=ALTEN%20se%20place%20%C3%A0%20la,et%20des%20%C3%A9coles%20d'ing%C3%A9nieurs.>

[Informations concernant « ALTEN »]

Date de consultation : 11/08/2020

_ https://www.alten.com/wp-content/uploads/2019/06/ALTEN_DDR_2018.pdf

[Informations concernant « ALTEN »]

Date de consultation : 12/08/2020

_ <https://stackoverflow.com/>

[Forum mondial qui permet la documentation sur les erreurs rencontrées lors du développement]

Date de consultation : tout au long du stage 08/07/2020 – fin décembre 2020

_ <https://www.supinfo.com/articles/single/104-qu-est-que-docker-pourquoi-est-il-different-machines-virtuelles>

[Informations concernant docker]

Date de consultation : 23/08/2020

_ <https://www.docker.com/resources/what-container>

[Informations concernant docker]

Date de consultation : 23/08/2020

_ <https://blog.thiga.co/glossaire/definition-fibonacci/>

[Informations concernant la suite de Fibonacci]

Date de consultation : 23/08/2020

<https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Op%C3%A9rateurs>

[Informations concernant les opérateurs JavaScript]

Date de consultation : 25/08/2020

<https://reactrouter.com/web/api/history>

[Informations concernant l'objet history de React]

Date de consultation : 26/08/2020

<https://fr.reactjs.org/docs/hooks-intro.html>

[Informations concernant React hooks]

Date de consultation : 26/08/2020

<https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Op%C3%A9rateurs>

[Informations concernant les opérateurs de javascript]

Date de consultation : 31/08/2020

<https://www.objectdb.com/java/jpa/query/jpql/collection>

[Informations concernant JPQL]

Date de consultation : 05/09/2020

<https://www.geeksforgeeks.org/split-string-java-examples/>

[Informations concernant JAVA]

Date de consultation : 06/09/2020

<https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/domain/>

[Documentation concernant SPRING]

Date de consultation : 01/10/2020

Glossaire

Mots	Signification
Workflow	Il s'agit d'un suivi d'étapes (tâches, événements, interactions) qui composent un processus de travail.
Dockerisation	C'est le fait de conteneuriser une application ou plusieurs avec leurs dépendances nécessaires.
IHM	L'étude de l'interaction entre l'humain et la machine en tant que phénomène sociotechnique et co-adaptatif.
Les opérateurs (dans le domaine de la programmation informatique)	Ce sont des fonctions représentées par des symboles qui permettent de manipuler des variables ou évaluer des constantes.
Énumération	Elle consiste à citer un nombre de constantes une par une, c'est souvent un type de classe dans plusieurs langages de programmation et dans la base de données il s'agit d'une table avec souvent un petit nombre d'éléments définis.
Annotation	Une annotation est une sorte de métadonnées dans plusieurs langages de programmation qui peut être appliquée à divers éléments du code source afin que plus tard, un outil, un débogueur ou un programme d'application puisse tirer parti de ces annotations ; et aider à analyser le programme de manière positive et constructive.
Opérateur ternaire	Ils se trouvent dans plusieurs langages informatiques et sont composés de trois parties, d'où vient le mot « ternaire ». Ces parties comprennent une déclaration conditionnelle et deux résultats possibles comme pour le fameux « if - else » de la programmation.
La décomposition	Ça fait référence aux opérateurs qui permettent de décomposer un ou tous les éléments d'un objet, en anglais ils sont nommés « spread operators ».
JPQL	C'est un langage de requête de Jakarta persistance qui est venu remplacer celui de Hibernate « HQL », lorsqu'il est compilé il est traduit d'abord en arbre AST qui est un modèle Orienté objet avant d'être traduit en SQL ce qui lui procure une sécurité de plus que le HQL le rendant invulnérable aux injections SQL.

Validation	La validation ici renvoie vers les contraintes et les vérifications des données avant de les stocker dans leur base, elle se trouve normalement dans le frontend et le backend pour assurer la fiabilité des données.
------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Liste des abréviations

Abréviation	Signification
SISE	Système d'Information et Système Embarqué.
DT	Dossier Technique.
CV	Curriculum Vitæ.
IT	Technologie de l'Information ou aussi appelé système informatique.
R&D	Recherche et Développement.
AVV	AVant Ventes.
SGBD	Système de Gestion de Base de Données.
JSON	JavaScript Object Notation.
REST	REpresentational State Transfer.
HTTP	L'HyperText Transfer Protocol.
HTML	Le HyperText Markup Language.
CSS	Cascading Style Sheets.
SPA	Single-Page Application.
CRUD	Create, Read, Update, Delete : Ajout, lecture, édition et suppression.
SQL	Structured Query Language.
IDE	Integrated Development Environment.
Jakarta EE ou JEE	Jakarta Enterprise Edition.
ECMAScript	European Computer Manufacturers Association Script.
API	Application Programming Interface.
SASS	Syntactically Awesome Style Sheets.
JWT	JSON Web Token.
POO	Programmation Orientée Objet.
ORM	Object-Relational Mapping.
CSV	Comma-Separated Values.
RSS	Really Simple Syndication.
VM	Virtual Machine.
JDL	JHipster Domain Language.
HQL	Hibernate Query Language.
DOM	Document Object Model
IHM	Interaction Homme-Machine
JPQL	Jakarta Persistence Query Language

Annexes

Voici en annexes les fichiers de configuration de JHipster des entités, les pages d'entités du front en tsx (TypeScriptXML) avec react et les services web en JEE dans le back end.

```

jhipster > {} Skills.json > [ ] relationships > {} 0 > ownerSide
1  {
2    "name": "Skills",
3    "fields": [
4      {
5        "fieldName": "label",
6        "fieldType": "String",
7        "fieldValidateRules": ["required", "unique"]
8      },
9      {
10       "fieldName": "code",
11       "fieldType": "String",
12       "fieldValidateRules": ["required", "unique"]
13     }
14   ],
15   "relationships": [
16     {
17       "relationshipType": "many-to-many",
18       "otherEntityName": "collaborator",
19       "otherEntityRelationshipName": "skills",
20       "relationshipName": "collaborators",
21       "otherEntityField": "id",
22       "ownerSide": false
23     },
24     {
25       "relationshipName": "category",
26       "otherEntityName": "category",
27       "relationshipType": "many-to-one",
28       "otherEntityField": "label",
29       "otherEntityRelationshipName": "skills"
30     }
31   ],
32   "changelogDate": "20200921140045",
33   "entityTableName": "skills",
34   "dto": "no",
35   "pagination": "no",
36   "service": "no",

```

Figure 70 : Fichier de configuration d'entité de JHipster

(Cette image est notre réalisation)

Ici on donne un exemple des fichiers de configuration de JHipster des entités.

```

33
34 export const Collaborator = (props: ICollaboratorProps) => {
35   const [paginationState, setPaginationState] = useState(
36     overridePaginationStateWithQueryParams(getSortState(props.location, ITEMS_PER_PAGE), props.location.sea
37   );
38
39   const [searchState, setSearchState] = useState({ search: '', values: [] });
40
41   const [selectedSkillsState, setSelectedSkillsState] = useState({ selectedSkills: null });
42
43   const idsSelected = [];
44
45   const getAllEntities = () => {
46     searchState.search === '' ?
47     props.getCollaborators(paginationState.activePage - 1, paginationState.itemsPerPage, `${paginationSta
48     : props.getEntitiesBySkills(paginationState.activePage - 1, paginationState.itemsPerPage, `${paginat
49   };
50
51   useEffect(() => {
52     props.getSkills();
53   }, []);
54
55   const sortEntities = () => {
56     getAllEntities();
57     const endURL = `?page=${paginationState.activePage}&sort=${paginationState.sort},${paginationState.orde
58     if (props.location.search !== endURL) {
59       props.history.push(`${props.location.pathname}${endURL}`);
60     }
61   };
62
63   useEffect(() => {
64     sortEntities();
65   }, [paginationState.activePage, paginationState.order, paginationState.sort]);
66
67

```

Figure 71 : React Hooks

(Cette image est notre réalisation)

Dans cette figure au-dessus, on démontre l'utilisation des React Hooks, il s'agit des fonctions qui commencent avec le terme « Use ».

```

const mapStateToProps = (storeState: IRootState) => ({
  collaboratorList: storeState.collaborator.entities,
  loading: storeState.collaborator.loading,
  totalItems: storeState.collaborator.totalItems,
  skills: storeState.skills.entities,
});

const mapDispatchToProps = {
  getCollaborators,
  getSkills,
  getEntitiesBySkills
};

type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof mapDispatchToProps;

export default connect(mapStateToProps, mapDispatchToProps)(Collaborator);

```

Figure 72 : Récupération et conversion des états en propriétés

(Cette image est notre réalisation)

Dans la figure 72, on vous fait part de la manière de récupérer les états du « StoreState » de redux et leur conversion en propriétés pour les utiliser dans le code.

```
/**
 * {@code GET /collaborators} : get all the collaborators.
 *
 * @param pageable the pagination information.
 * @param eagerload flag to eager load entities from relationships (This is applicable for many-to-many).
 * @return the {@link ResponseEntity} with status {@code 200 (OK)} and the list of collaborators in body.
 */
@GetMapping("/collaborators")
public ResponseEntity<List<Collaborator>> getAllCollaborators(Pageable pageable) {
    log.debug("REST request to get a page of Collaborators");
    List<Collaborator> collaborators;
    collaborators = collaboratorRepository.findAllWithEagerRelationships();
    Page<Collaborator> page = collaboratorRepositoryCustom.collaboratorsListToPage(collaborators, pageable);
    HttpHeaders headers = PaginationUtil.generatePaginationHttpHeaders(ServletUriComponentsBuilder.fromCurrentRequest(), page);
    return ResponseEntity.ok().headers(headers).body(page.getContent());
}

/**
 * {@code GET /collaborators/skills} : get all the collaborators by skills.
 *
 * @return the {@link ResponseEntity} with status {@code 200 (OK)} and the list of skills in body.
 */
@GetMapping("/collaborators/skills")
public ResponseEntity<List<Collaborator>> getCollaboratorsBySkills(@RequestParam String skillId, Pageable pageable) {
    log.debug("requestParam {}", skillId);
    Long lengthSkills = Long.valueOf(StringUtils.countMatches(skillId, ",")+1);
    Collection<Long> skillsId = collaboratorRepositoryCustom.stringToCollectionLong(skillId, lengthSkills);
    List<Collaborator> collaborators = collaboratorRepository.findBySkills(skillsId, lengthSkills);
    Page<Collaborator> page = collaboratorRepositoryCustom.collaboratorsListToPage(collaborators, pageable);
    HttpHeaders headers = PaginationUtil.generatePaginationHttpHeaders(ServletUriComponentsBuilder.fromCurrentRequest(), page);
    return ResponseEntity.ok().headers(headers).body(page.getContent());
}
```

Figure 73 : service web de l'affichage des collaborateurs

(Cette image est notre réalisation)

Ici, on présente les services web chargés de l'affichage des collaborateurs et voici les fonctions et requêtes utilisés :

```
@Repository
public interface CollaboratorRepository extends JpaRepository<Collaborator, Long> {

    @Query(value = "select distinct collaborator from Collaborator collaborator left join fetch collaborator.skills",
        countQuery = "select count(distinct collaborator) from Collaborator collaborator")
    Page<Collaborator> findAllWithEagerRelationships(Pageable pageable);
```



```
@Query("select distinct collaborator from Collaborator collaborator left join f
etch collaborator.skills")
List<Collaborator> findAllWithEagerRelationships();

@Query("select collaborator from Collaborator collaborator left join fetch coll
aborator.skills where collaborator.id =:id")
Optional<Collaborator> findOneWithEagerRelationships(@Param("id") Long id);

@Query("select distinct collaborator from Collaborator collaborator left join f
etch collaborator.skills where collaborator IN (select collaborator from Collaborat
or collaborator join collaborator.skills skill where skill.id IN :skillsId GROUP BY
collaborator having count (distinct skill.id)=:lengthSkills)")
List<Collaborator> findBySkills(@Param("skillsId") Collection<Long> skillsId, @
Param("lengthSkills") Long lengthSkills);
}
```

```
17 public class CollaboratorRepositoryCustomImpl implements CollaboratorRepositoryCustom
18 {
19     private final CollaboratorRepository collaboratorRepository;
20
21     // constructor-based injection
22     public CollaboratorRepositoryCustomImpl(CollaboratorRepository collaboratorRepository){
23
24         this.collaboratorRepository = collaboratorRepository;
25     }
26
27     // Convert String to Collection<Long>
28     public Collection<Long> stringToCollectionLong(String string, Long lengthString){
29         Collection<Long> result = new ArrayList<Long>();
30         String[] arrOfStr = string.split(",", lengthString.intValue());
31         for (String a : arrOfStr){
32             result.add(Long.parseLong(a));
33         }
34         return result;
35     }
36
37     // Convert a List of collaborators to a page of collaborators
38     public Page<Collaborator> collaboratorsListToPage(List<Collaborator> collaborators, Pageable pageable){
39         int start = (int) pageable.getOffset();
40         int end;
41         if((start + pageable.getPageSize()) > collaborators.size()){
42             end = collaborators.size();
43         } else {
44             end = (start + pageable.getPageSize());
45         }
46         Page<Collaborator> page = new PageImpl<Collaborator>(collaborators.subList(start, end), pageable, collaborators.size());
47         return page;
48     }
49 }
```

Figure 74 : fonctions utilisées pour l'affichage des collaborateurs

(Cette image est notre réalisation)

Et pour finir, on vous montre comme bonus comment on envoie des requêtes HTTP depuis le frontend grâce à Axios et comment on interdit l'accès aux routes ou composants pour ceux qui n'ont pas les droits nécessaires :

```
// Actions

export const getEntities: ICrudGetAllAction<ICollaborator> = (page, size, sort) => {
  const requestUrl = sort ?
    sort.split(",")[0] === "login" || sort.split(",")[0] === "firstName" ?
    `${apiUrl}${'?page=${page}&size=${size}&sort=account.${sort}'}`
    : `${apiUrl}${'?page=${page}&size=${size}&sort=${sort}'}`
    : `${apiUrl}${sort ? '?page=${page}&size=${size}&sort=${sort}' : ''}`;
  return {
    type: ACTION_TYPES.FETCH_COLLABORATOR_LIST,
    payload: axios.get<ICollaborator>(`${requestUrl}${sort ? '&' : ''}cacheBuster=${new Date().getTime()}`),
  };
};

export const getEntitiesBySkills = (page, size, sort, ArrayOfSkills) => {
  let flag = false;
  let requestUrl = `${apiUrl}/skills?`;
  for (let i = 0; i < ArrayOfSkills.length; i++){
    if(ArrayOfSkills[i] !== "" && !flag) {
      flag = true;
      requestUrl = requestUrl.concat(`skillId=${ArrayOfSkills[i]}`)
    } else if (ArrayOfSkills[i] !== "" && flag) {
      requestUrl = requestUrl.concat(`&skillId=${ArrayOfSkills[i]}`)
    }
  }
  return {
    type: ACTION_TYPES.FETCH_COLLABORATOR_LIST,
    payload: axios.get<ICollaborator>(`${requestUrl}${sort ? '&' : ''}&page=${page}&size=${size}&sort=${sort}' : ''}&cacheBuster=${new Date().getTime()}`),
  };
};
```

Figure 75 : Envoie de requêtes HTTP grâce à Axios

(Cette image est notre réalisation)

```

1  import React from 'react';
2  import { Switch } from 'react-router-dom';
3
4  // eslint-disable-next-line @typescript-eslint/no-unused-vars
5  import ErrorBoundaryRoute from 'app/shared/error/error-boundary-route';
6  import PrivateRoute from 'app/shared/auth/private-route';
7  import { AUTHORITIES } from 'app/config/constants';
8
9  import Collaborator from './collaborator';
10 import Skills from './skills';
11 import Category from './category';
12 import Level from './level';
13 /* jhipster-needle-add-route-import - JHipster will add routes here */
14
15 const Routes = ({ match }) => (
16   <div>
17     <Switch>
18       { /* prettier-ignore */ }
19       <ErrorBoundaryRoute path={`/${match.url}collaborator`} component={Collaborator} />
20       <ErrorBoundaryRoute path={`/${match.url}skills`} component={Skills} />
21       <PrivateRoute path={`/${match.url}category`} hasAnyAuthorities={['AUTHORITIES.ADMIN']} component={Category} />
22       <PrivateRoute path={`/${match.url}level`} hasAnyAuthorities={['AUTHORITIES.ADMIN']} component={Level} />
23       { /* jhipster-needle-add-route-path - JHipster will add routes here */ }
24     </Switch>
25   </div>
26 );
27
28 export default Routes;
29

```

Figure 76 : Interdiction d'accès aux routes et aux composants liés pour ceux qui n'ont pas les droits d'administrateur

(Cette image est notre réalisation)