كلية العلوم و التقنيات فاس
ⵜⴰⵖⵚⵓⵍⵜ ⵏ ⵜⵇⴰⵙⵙⵉⵏ ⴷ ⵜⵓⵙⵙⵙⵉⵏ
Faculté des Sciences et Techniques de Fès

جامعة سيدي محمد بن عبد الله
ⵜⴰⵙⴷⴰⵡⵜ ⵙⵙⴰⵙ ⵎⵓⵃⴰⵎⴷ ⴱⵏ ⵄⴱⴷⵍⵍⴰⵀ
Université Sidi Mohamed Ben Abdellah

Département d'Informatique

Projet de fin d'études

Master Sciences et Techniques
Systèmes Intelligents & Réseaux

Application multiplateforme pour la gestion et le tracking des tickets

Lieu du stage : Smile Maroc

Réalisé par :

- AKAABOUNE Mohamed Amine

Encadré par :

- Pr. BOUSHABA Abdelali
- Pr. BENABBOU Adil
- Mr. HAFFOU Tarik

Soutenu le 19.06.2019 devant le jury composé de :

- Pr. OUZARF Mohamed	Faculté des Sciences et Techniques de Fès	(Président)
- Pr. CHAKER Ilham	Faculté des Sciences et Techniques de Fès	(Examinateur)
- Pr. BOUSHABA Abdelali	Faculté des Sciences et Techniques de Fès	(Encadrant)
- Pr. BENABBOU Adil	Faculté des Sciences et Techniques de Fès	(Encadrant)

Année Universitaire 2018 – 2019

Dédicaces

Au nom d'Allah le tout miséricordieux, le très miséricordieux.

À mon très cher Père, À ma très chère Mère, Aucune dédicace ne saurait exprimer mon respect, mon amour et ma considération pour les sacrifices que vous avez consentis pour mon éducation et mon bienêtre. Je vous remercie pour tout le soutien et l'amour que vous me portez depuis mon enfance. Que Dieu vous accorde santé, bonheur et longue vie.

À mes très chères sœurs Siham et Ghizlane pour leurs soutiens.

À mes très chers petits Malak et Aboubaker.

À tous les membres de ma famille, pour leur soutien tout au long de mon parcours universitaire.

À mes chères amies, À tous les membres du club Espoir pour l'amitié qui nous a unies et pour tous les merveilleux moments et les souvenirs partagées.

À tous ceux que j'aime et à tous ceux qui m'aiment.

Je leur dédie ce modeste travail en leur souhaitant un immense bonheur et une joyeuse vie.

Akaaboune Mohamed Amine.

Remerciements

C’est avec un grand honneur et un grand plaisir que je dédie ces quelques lignes afin de remercier toute personne ayant contribué de près ou de loin à l’accomplissement de ce projet.

Tout d'abord, je tiens à exprimer mes sincères remerciements à mes encadrants à la Faculté des Sciences et Techniques de Fès **Pr. Abdelali BOUSHABA** et **Pr. Adil BENABBOU** pour leurs assistances et leurs conseils judicieux tout au long de PFE.

J'adresse mes remerciements à **M. Yacine BEKKAR**, Directeur de l’agence Smile Maroc, de m’avoir accueilli comme stagiaire au sein de la société et de m’accorder l’opportunité de découvrir et réaliser ce projet.

Je présente mes remerciements à **M. Tarik HAFFOU**, Responsable du pôle local, pour l’aide, le support, les précieux conseils, et de m’avoir accueilli chaleureusement.

Je souhaite remercier spécialement **M. Tarik HADDADI**, Teach Lead Front et Mobile, pour son accueil, ses conseils pertinents et son encadrement direct et amical durant ce stage.

Je saisis aussi l’occasion pour remercier tous les collaborateurs du pôle Front et Mobile pour l’aide inconditionnelle qu’ils m’ont apportés tout au long de la période de mon stage au sein de Smile Maroc.

Je remercie **Pr. OUZARF Mohamed** et **Pr. CHAKER lham** membres de jury, pour l’honneur qu’ils m’ont fait en acceptant d’examiner et juger ce travail.

J’adresse également des remerciements particuliers à l’ensemble du corps professoral et administratif de la Faculté des Sciences et Techniques de Fès, pour l’inestimable qualité de l’enseignement qu’ils assurent.

Résumé

Pour rationaliser les processus et garder les équipes au courant, l'entreprise d'aujourd'hui vise à fournir à ses collaborateurs une plateforme centralisée où ils pourront avoir une vue d'ensemble claire de toutes leurs activités et de leurs communications au sein de l'équipe.

C'est dans ce sens que s'inscrit notre projet au sein de Smile Maroc, qui consiste à mettre en place une solution de gestion des tâches et de tracking, qui est basé sur les outils manipulés quotidiennement par des équipes offshores et locales, avec l'ajout de nouvelles fonctionnalités pour répondre mieux aux besoins des équipes, et unifier leur manière de travail, on lui offrant une plateforme commune et personnalisée.

Pour ce faire, nous avons commencé par l'étude de fonctionnement des outils existants utilisés par les équipes de développement pour bien analyser les besoins fonctionnels, puis nous avons réalisé une étude comparative entre ces outils, afin de mettre en œuvre la nouvelle solution. Et pour assurer l'agilité et la réactivité, nous avons choisi Scrum comme méthodologie de gestion.

Mots clés : MEAN Stack, JavaScript, Front-End, Gestion des tickets, Suivi du temps.

Abstract

To rationalize processes and keep teams informed, today's company aims to provide employees with a centralized platform where they can have a clear overview of all their activities and communications within the team.

In this context, our project consists in setting up a personalized solution of task management and tracking, which is based on the tools used daily by offshore and local teams, by the addition of new functionalities to better meet the needs of teams, and unified their way of working, offering them a common and personalized platform.

To complete this mission, we began by studying the existing tools used by the development teams to properly analyze the functional needs, and then carried out a comparative study between these tools in order to implement the new solution. And to ensure agility and responsiveness, we chose Scrum as a management methodology.

Keywords : MEAN Stack, JavaScript, Front-End, Tasks management, Time Tracking.

Table des matières

Liste des tableaux

Liste des figures

Liste d'Acronymes et Abréviations

Acronyme	Désignation
SPA	Single Page Application
CP	Chef de projet
DP	Directeur de projet
UI /UX	User Interface /User Experience
JSON	JavaScript Object Notation
JWT	Json Web Token
API	Application Programing Interface
REST	Representational State Transfer
ES6	ECMAScript 6
SASS	Syntactically Awesome Stylesheets
MVC	Model View Controller
DI	Dependency Injection
IT	Information Technology
ODM	Object Data Modeling
CLI	Command Line Interface
NoSQL	Not Only Structured Query Language
RDBMS	Relational Database Management System

Introduction générale

Dans le contexte d'un suivi de projet, une équipe a certes besoin d'outils qui lui permet d'avoir une gestion efficace des tâches quotidienne afin d'optimiser au plus le bon cadrage des projets, ainsi d'être irréprochable en termes de planning.

Un grand nombre d'outils de gestion de projets qui sont disponibles sur le marché mettent l'accent sur un ensemble de fonctionnalités, telles que le suivi et l'organisation du processus de gestion d'un tel projet, le suivi des tâches, les évolutions, les retours clients, ainsi que l'état d'avancement de ces projets. Malgré le grand nombre de choix, certaines sociétés ayant des exigences uniques recherchent toujours des solutions qui leur conviennent spécifiquement.

Afin de gérer leurs tâches quotidiennes, les équipes au sein de Smile utilisent trois outils qui sont Redmine, Trello et Jira, dont chacun a des limites en point de vue des membres des équipes qui l'utilisent chaque jour, à savoir la lourdeur, la complexité et le manque du suivi du temps, vu que le développement nécessite de la concentration et ce peut que le développeur rencontre des problèmes tels que des oublis à cause de la charge du travail ainsi qu'une perte de temps non négligeable.

C'est dans ce contexte s'intègre notre projet de fin d'études, qui a pour objectif la réalisation d'un système de gestion de tâches et de tracking basé sur les solutions existantes et les outils manipulés quotidiennement par des équipes offshores et locales, avec l'ajout de nouvelle fonctionnalité pour répondre mieux aux besoins des équipes et unifié leur manière de travail, on lui offrant une plateforme commune et personnalisée.

Pour mener à bien le développement de ce projet, nous avons tout d'abord effectué une phase d'étude de l'existant afin de bien définir ses limites. Après nous avons spécifié les besoins fonctionnels et non fonctionnels. Ensuite, nous avons procédé à sa conception ainsi qu'aux choix technologiques pour sa réalisation pour enfin la mettre en œuvre.

Les différentes étapes suivies pour la réalisation de ce projet sont présentées par le biais du présent rapport qui s'articule autour de quatre chapitres. D'abord, dans le premier chapitre, nous présentons le cadre général du projet ainsi que l'étude de l'existant, la méthodologie du travail adoptée et le planning provisionnel du projet. Ensuite, le deuxième chapitre sera dédié à l'analyse et la spécification des besoins dans lequel nous identifions les acteurs, les besoins fonctionnels et non fonctionnels que doit satisfaire notre projet avec une modélisation par le biais des diagrammes de cas d'utilisation, des diagrammes de séquences et de classes.

Le troisième chapitre consistera à introduire la méthodologie de développement, à savoir l'environnement logiciel du projet, les différents outils et plateformes utilisés, et l'architecture technique de la plateforme.

Ensuite, nous présenterons dans le dernier chapitre les interfaces principales qui mettent en évidence le fonctionnement de l'application développée. Enfin, nous finissons par une conclusion dans laquelle nous résumons notre solution et exposons notre futures perspectives.

Chapitre 1. Cadre général du projet

Introduction

L'objectif de ce premier chapitre est d'introduire le contexte général de notre projet de fin d'études. Nous commençons tout d'abord par la présentation de l'organisme d'accueil Smile dans lequel s'est déroulé notre stage, puis nous exposons le cadre du projet en décrivant le fonctionnement actuel dans l'entreprise. Finalement, nous présentons la solution proposée, ainsi que la démarche adoptée pour la réalisation du projet.

1. Organisme d'accueil

Dans cette première partie, nous citons quelques informations sur l'entreprise Smile ainsi que ses références, ses domaines d'activités et ses implantations, nous parlons aussi de l'agence Smile Maroc où s'est déroulé notre stage.

1.1. Groupe Smile

Créée en 1991, Smile est le leader européen du numérique ouvert, expert du digital et de l'open source, il intervient en conseil, en conception, intégration et réalisation, dans le déploiement de solutions open source, ainsi que la conception et la réalisation de plateformes appuyées sur des produits et outils open source. Son offre couvre aussi la formation, le support et l'hébergement [1].

Smile possède 16 agences réparties dans 8 pays, elle est présente en France à Paris, Lyon, Grenoble, Montpellier, Marseille, Nantes, Bordeaux, Lille, et Toulouse. Elle est présente aussi en Suisse à Lausanne, aux Pays-Bas à Utrecht, en Belgique à Bruxelles, au Luxembourg (Neopixl). La société est également présente à Casablanca au Maroc, Kiev en Ukraine, et à Abidjan en Côte d'Ivoire.

Figure. 1 : Répartition des agences Smile dans le monde

Smile eu une croissance organique et externe par acquisition très importante et s'est développé progressivement à l'international pour atteindre sa taille actuelle. Parmi les dates clés de l'histoire de Smile :

1991 : Création de Smile.

1996 : Construction d'une expertise sur l'ingénierie Internet et les grandes plateformes web.

2001 : Positionnement sur les technologies open source.

2009 : Acquisition de Cometa Technologies.

2014 : Plus de 20 % de croissance sur les cinq dernières années et près de 700 collaborateurs dans le monde.

2016 : Smile rachète Open Wide, second intégrateur français dans le logiciel libre, et passe à près de 1000 collaborateurs.

2018 : Smile fait les acquisitions de Virtua, une agence digitale indépendante suisse, puis de Adyax spécialiste du CMS Drupal.

2019 : Smile devient actionnaire majoritaire de SensioLabs, société à l'origine du Framework Symfony.

1.2. Références Smile

Toujours en quête de nouveaux challenges, Smile accompagne au quotidien de nombreuses entreprises dans leurs projets de transformation digitale. Que ce soit pour la refonte d'un site, d'un intranet ou la mise en place d'une plateforme de gestion documentaire ou d'outils connectés. Elle s'adresse à tous les secteurs allant au secteur public en passant par celui de l'industrie, des médias et bien d'autres.

Figure. 2 : Quelques références du groupe Smile

1.3. Livres blancs

Smile mène une action active de veille technologique qui lui permet de découvrir les produits les plus prometteurs de l'open source, de les qualifier et de les évaluer, de manière à proposer à ses clients les produits les plus aboutis, les plus robustes et les plus pérennes.

Cette démarche a donné lieu à toute une gamme de livres blancs couvrant différents domaines d'application. De nos jours, le catalogue de Smile dispose de plus de 40 livres blancs, téléchargeables gratuitement, qui présentent les concepts fondamentaux, les bonnes pratiques et les meilleures solutions open source du marché, sur les domaines d'expertise de Smile.

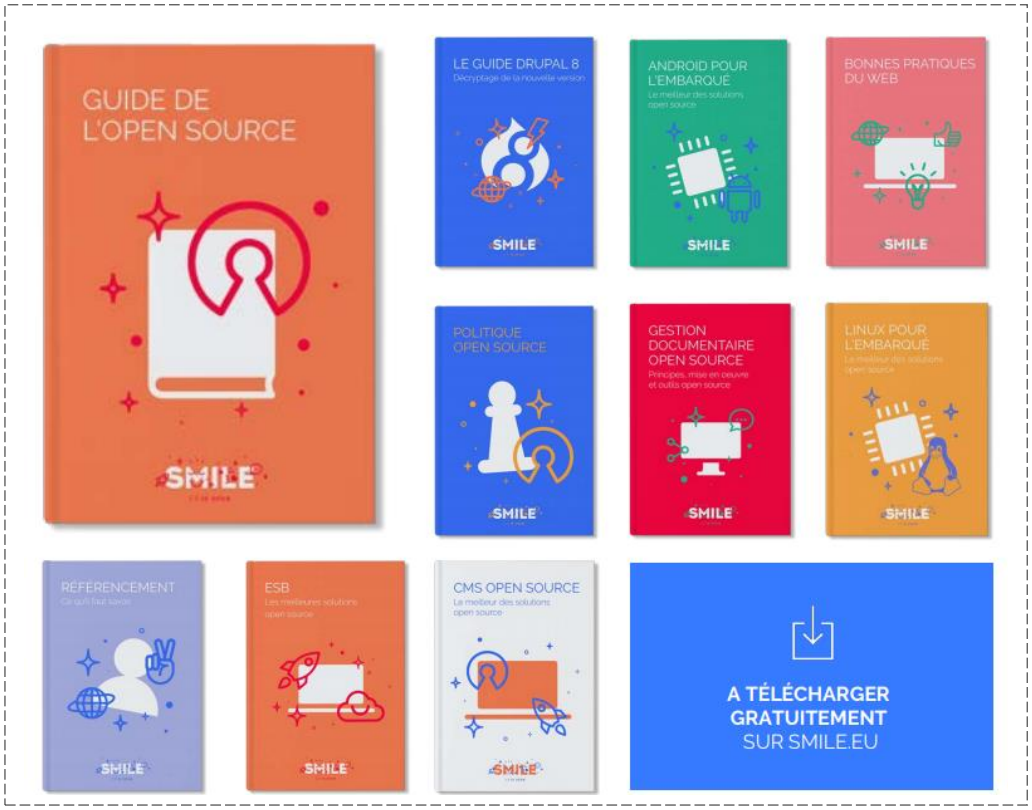


Figure. 3 : Livres blancs du groupe Smile

1.4. L'offre digitale

De nombreux clients, de tous secteurs et de toutes tailles, ont fait de Smile leur partenaire IT (Information Technology) de préférence sur le long terme. Cette reconnaissance est due à l'expertise technique ainsi que la compréhension des enjeux business.

Avec plus de 600 experts des projets digitaux et e-commerce, Smile accompagne leurs clients dans leur transformation numérique en proposant une gamme de solutions digitales de haut niveau pour garder sa place sur le podium des acteurs majeurs du digital [2].

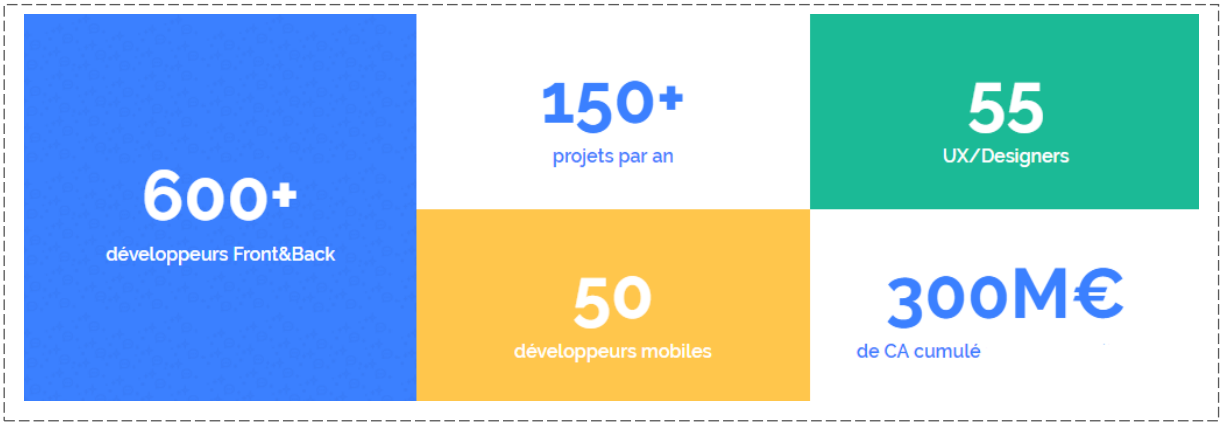


Figure. 4 : Chiffres clés de l'offre digitale de Smile

1.5. Organisation Smile Maroc

Présente depuis 2004 à Casablanca, Smile Maroc est la première agence internationale du groupe, l'agence bénéficie d'une expérience de longue date dans la réalisation de projets digitaux d'envergure.

Smile Maroc contribue activement à la réalisation des projets digitaux, à la fois sur les marchés français et marocain. Et pour répondre au mieux aux besoins de ses clients, l'agence s'organise autour de deux pôles complémentaires :

- **Le pôle Local** : qui accompagne les grands comptes marocains (Orange, OCP, Bank Al-Maghrib, ONCF, Ministères, Al Omran etc.) en apportant des solutions open source pérennes, matures et à forte valeur ajoutée.

- **Le pôle Skill Center Groupe** : qui représente plus de 70% du chiffre d'affaires de l'agence grâce à plusieurs CDS (Centres De Service) pour plusieurs grands comptes.

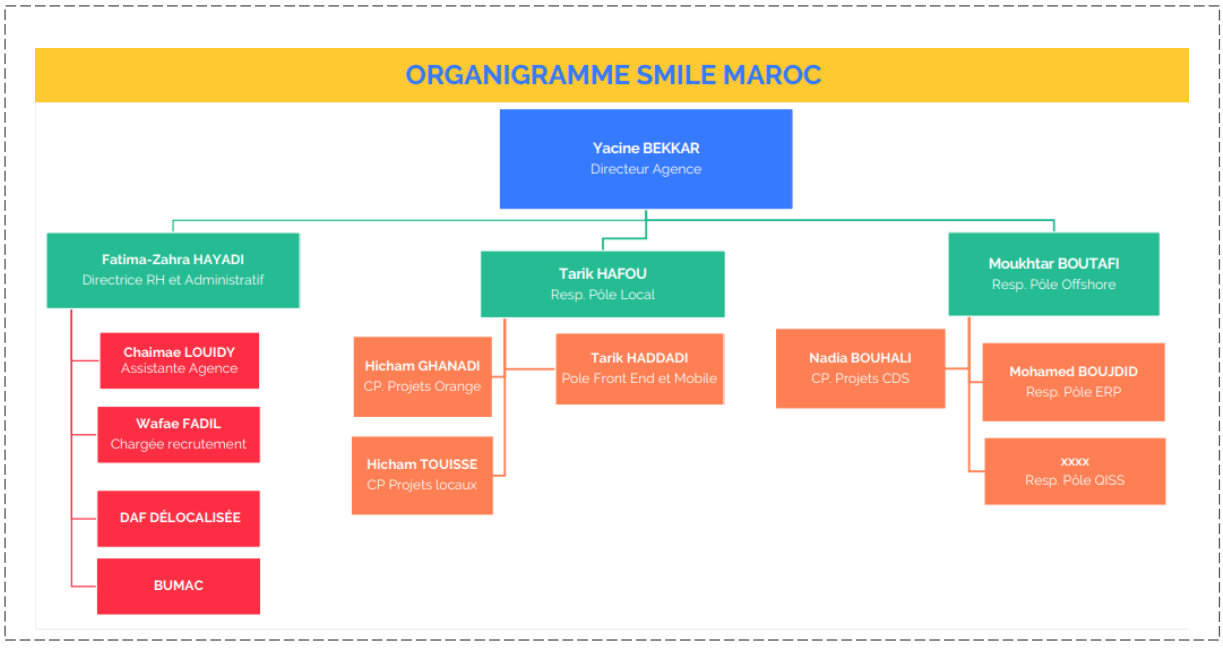


Figure. 5 : Organigramme Smile Maroc

2. Étude de l'existant

Dans cette partie, nous présentons les outils de gestions de projets utilisés à l'entreprise Smile, ainsi que nos observations sur le fonctionnement actuel afin de proposer une solution qui répond mieux aux besoins des équipes.

2.1. Fonctionnement actuel

Afin de gérer leurs tâches quotidiennes, les équipes au sein de Smile utilisent trois outils, qui sont **Redmine**, **Trello** et **Jira**.

2.1.1. Redmine

Redmine est un logiciel de gestion de projet Open Source et gratuit basé sur le web, qui est sorti pour la première fois en 2006 sous la licence GPLv2. Il est écrit en langage Ruby et se base sur Ruby on Rails. Pour qu'on puisse utiliser ce programme de management de projets, nous avons besoin d'un environnement d'hébergement approprié et d'une base de données (MySQL, MariaDB, PostgreSQL) [3].

Il reste la solution la plus utilisée par les équipes de développement à Smile pour la gestion et le suivi des tâches à cause de sa simplicité. La navigation se fait via un moteur de recherche qui permet de trouver des demandes (tickets), des documents de suivi de projet, des wikis et tout autre contenu, ainsi que des onglets :

« **Overview** » qui offre une vue générale du projet [cf. Figure 6] (membres, bugs rencontrés, etc.)

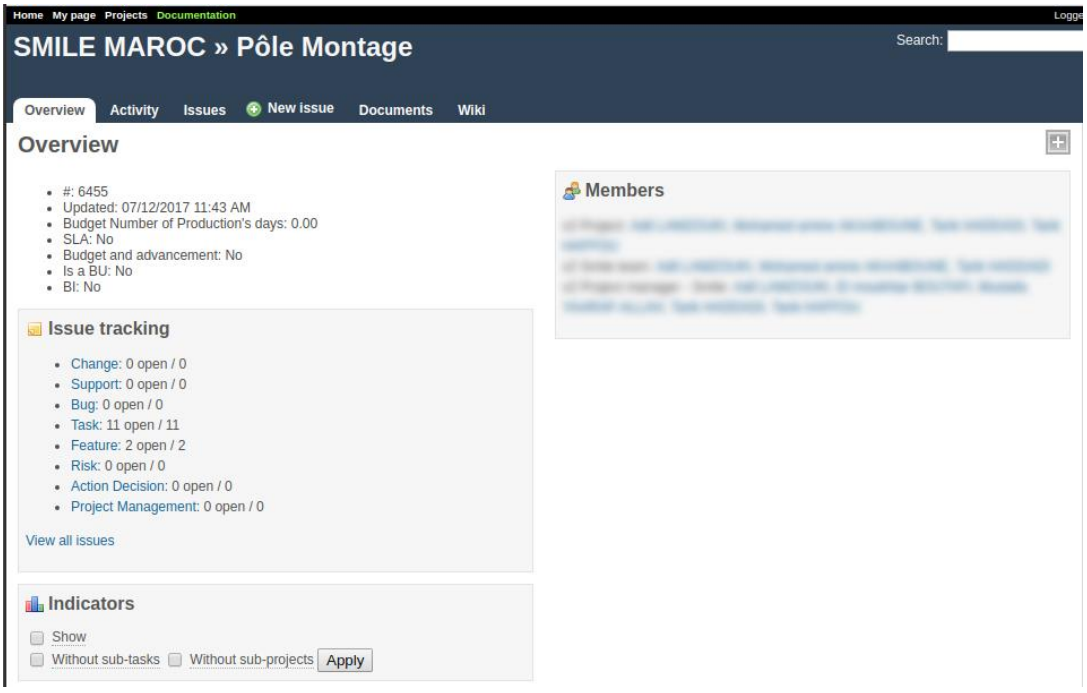
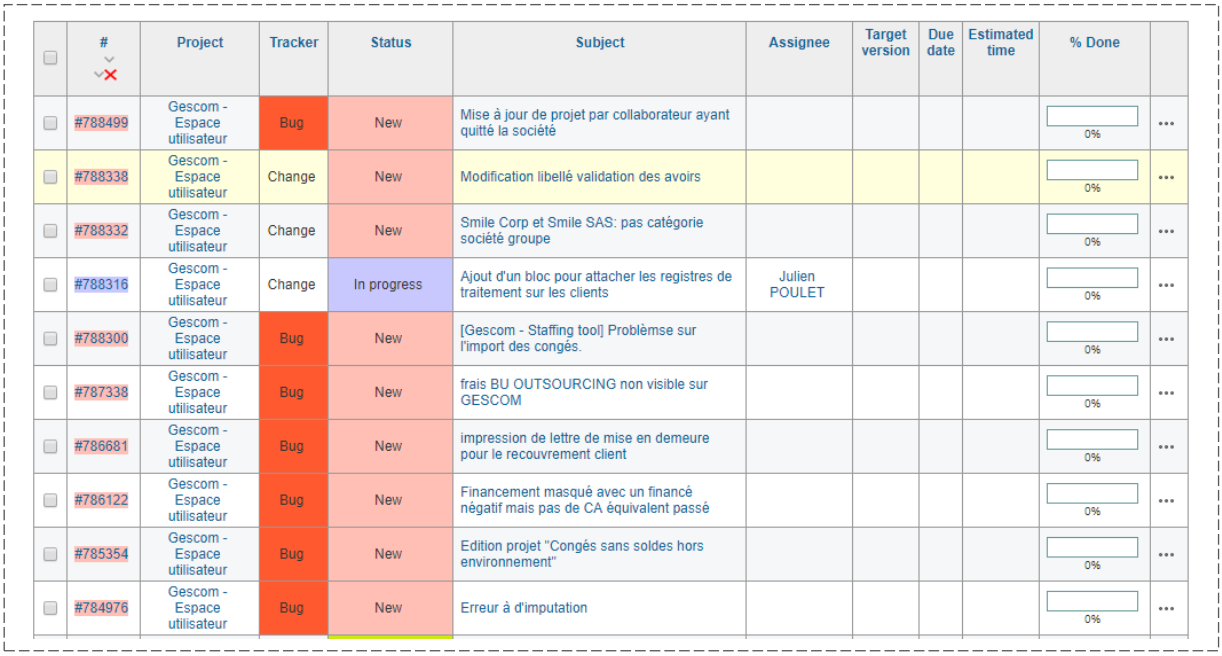


Figure. 6 : Onglet Overview de Redmine

« **Activity** » est un fil d'actualités qui permet de suivre l'évolution des tâches en cours.
« **Issues** » est une liste qui affiche les tâches en cours dans un projet [cf. Figure 7]. Il est possible de créer autant de listes qu'on le souhaite et d'en partager certaines avec des collaborateurs.



#	Project	Tracker	Status	Subject	Assignee	Target version	Due date	Estimated time	% Done
#788499	Gescom - Espace utilisateur	Bug	New	Mise à jour de projet par collaborateur ayant quitté la société					0%
#788338	Gescom - Espace utilisateur	Change	New	Modification libellé validation des avoirs					0%
#788332	Gescom - Espace utilisateur	Change	New	Smile Corp et Smile SAS: pas catégorie société groupe					0%
#788316	Gescom - Espace utilisateur	Change	In progress	Ajout d'un bloc pour attacher les registres de traitement sur les clients	Julien POULET				0%
#788300	Gescom - Espace utilisateur	Bug	New	[Gescom - Staffing tool] Problèmse sur l'import des congés.					0%
#787338	Gescom - Espace utilisateur	Bug	New	frais BU OUTSOURCING non visible sur GESCOM					0%
#786681	Gescom - Espace utilisateur	Bug	New	impression de lettre de mise en demeure pour le recouvrement client					0%
#786122	Gescom - Espace utilisateur	Bug	New	Financement masqué avec un financé négatif mais pas de CA équivalent passé					0%
#785354	Gescom - Espace utilisateur	Bug	New	Edition projet "Congés sans soldes hors environnement"					0%
#784976	Gescom - Espace utilisateur	Bug	New	Erreur à d'imputation					0%

Figure. 7 : Listes des demandes dans Redmine

« **New issue** » permet de créer un nouveau ticket dans le projet en cours [cf. Figure 8], avec les champs classiques (nom, description, pièces jointes, catégorie, tickets parents et tickets liés). Cela rend Redmine un outil de ticketing gratuit.

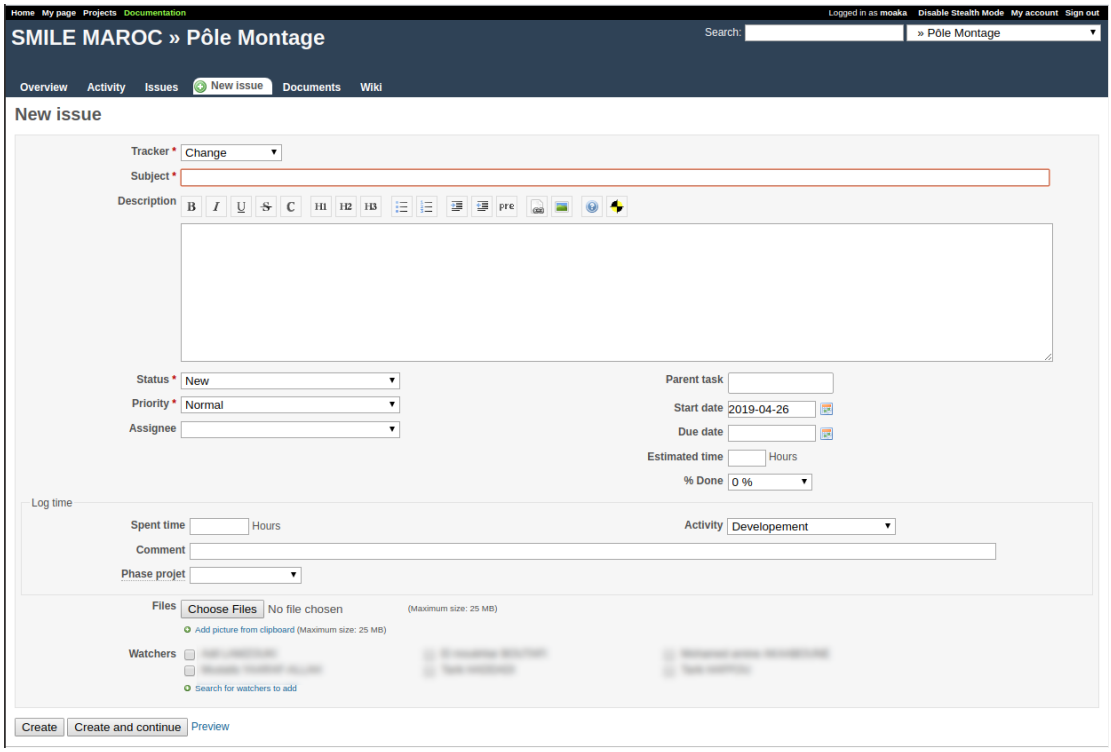


Figure. 8 : L'ajout d'une demande dans Redmine

« **Documents** » est un espace de gestion documentaire. Il est possible d'y ajouter des documents stockés sur PC.

« **wiki** » est un espace de gestion du savoir indexer dans la recherche de Redmine. Cette fonction est particulièrement utile pour rédiger de la documentation technique ou de bonnes pratiques à partager en interne.

Redmine offre aussi les fonctionnalités suivantes :

Multiprojets : il est possible de créer plusieurs projets en parallèle et de les gérer de manière indépendante.

Contrôle des accès des utilisateurs : chaque compte utilisateur dispose d'un ou plusieurs rôle(s) (administrateur, accès restreint, utilisateur). L'administrateur attribue des droits spécifiques à chacun d'eux pour contrôler les accès de chacun.

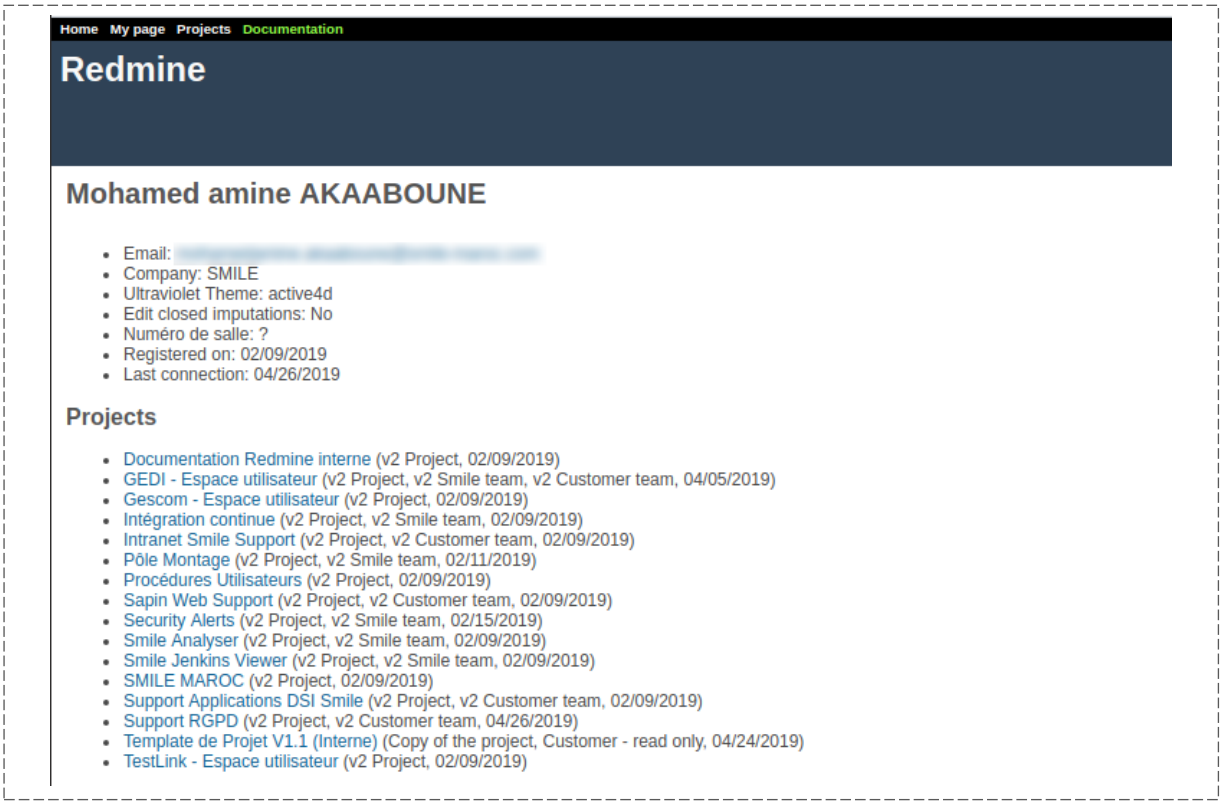


Figure. 9 : Consultation du profil dans Redmine

2.1.2. Trello

C'est une application de gestion de projet gratuite, conçue pour aider les utilisateurs individuels et les entreprises à mieux collaborer et à organiser leurs projets. Même s'il lui manque encore des fonctionnalités indispensables, Trello demeure un gestionnaire de projet intéressant pour tous ceux qui travaillent et gèrent des missions à distance ou équipe [4].

L'outil collaboratif Trello repose sur trois principes de bases, qui composeront l'ensemble des projets à gérer :

- **Les tableaux**
- **Les listes**
- **Les cartes**

On peut avoir plusieurs tableaux de bord « **Boards** » pour gérer différents projets. Chaque tableau de bord peut contenir autant de listes que l'on souhaite, et chaque liste peut contenir un nombre de cartes « **Card** ».

Chaque « **Card** » [cf. Figure 10] peut avoir une description avec une mise en forme (titres, sous titres, liens, etc.), plusieurs listes de tâches, des labels colorisés, des images et fichiers attachés et des commentaires. Chaque carte peut être associée à des utilisateurs distincts s'ils sont invités sur le projet ou à des informations utiles comme l'ajout d'un code couleur. La carte peut avoir aussi une date limite.

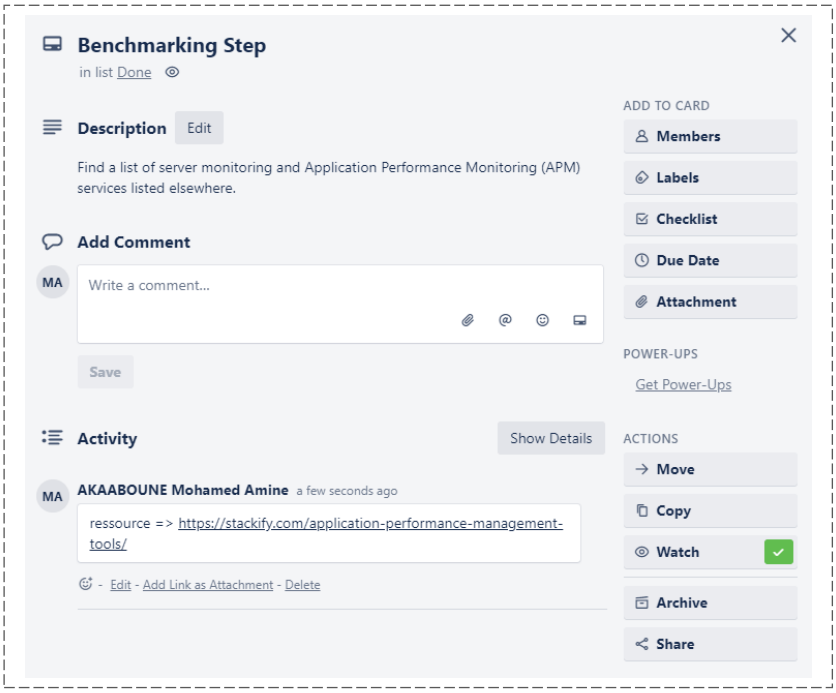


Figure. 10 : Détail d'une carte sous Trello

Les listes et les cartes se manipulent par un principe de « **Drag and Drop** » en glissant et déposant les éléments qui composent les tableaux, pour remettre de l'ordre ou définir des priorités dans les tâches assignées aux membres du projet.

Par défaut, un tableau de bord est composé de trois listes : « **To Do** », « **Doing** », « **Done** », et utilise donc la méthode « **Kanban** » pour la gestion de projet. Cependant il est possible d'utiliser Trello différemment.

Le kanban est connu comme étant un tableau signalétique qui permet de classer des informations par colonnes (listes) au sein desquelles on insère des informations pour gérer notamment les plannings. Aujourd'hui on l'utilise beaucoup avec les équipes qui travaillent sur des projets fonctionnant sur la méthode Agile [5].

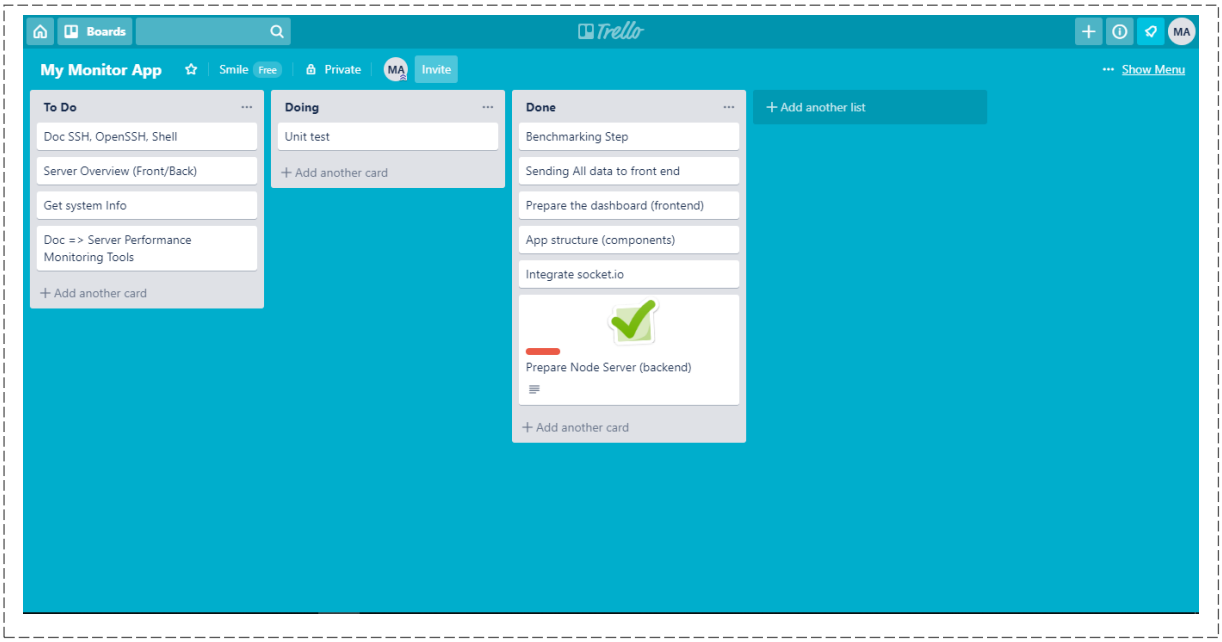


Figure. 11 : Tableaux de bord de Trello

2.1.3. Jira

Jira est un outil de gestion de projet agile développé par la société Australienne Atlassian et destiné aux équipes de développement. Il prend en charge toute méthodologie Agile, qu'il s'agisse d'un Scrum ou bien Kanban, qui aident à planifier, suivre et gérer les projets de développement logiciel Agile avec des tableaux aux rapports [6]. Jira peut paraitre un peu intimidant au premier abord, par rapport à des produits comme **Trello** et **Redmine**, car il est fourni avec beaucoup plus de fonctionnalités. Il repose sur trois concepts dans son intégralité :

- **Les projets :**

Jira propose deux types de projets : « **Software** » pour les projets de développement logiciel et « **Business** » pour les autres.

Pour chaque type de projet, trois options s'offrent aux équipes selon la méthodologie qu'ils souhaitent employer. Les fonctionnalités diffèrent d'une méthode à l'autre.

Si le projet est de nature Software, nous avons le choix entre :

- Le développement logiciel Scrum avec un « **Backlog** », la création de sprints et de rapports d'avancement, un flux de travaux (Workflow) Agile et l'édition de demandes.
- Le développement logiciel basique avec un flux de travaux classique et la création de demandes.
- Le développement logiciel Kanban avec un tableau Kanban, un flux de travaux basique et la création de demandes.

Si le projet est de nature Business, nous avons le choix entre :

- La gestion de projet avec la possibilité de créer des tâches, de les planifier et de suivre leur avancement.
- La gestion de processus pour créer des tâches et suivre leur progression.
- La gestion de tâche pour créer des tâches simples, les organiser et les attribuer.

- **Les demandes :**

Une fois notre projet est créé, nous accédons à son « **Backlog** ». Ce terme désigne la liste priorisée des fonctionnalités nécessaires au développement du produit final. Jira les appelle les demandes (issues). Après avoir invité les membres de l'équipe à rejoindre le projet, il est temps de compléter le « **Backlog** » avec nos demandes.

Une demande peut prendre trois formes :

- User Story : il s'agit de la description, en une à deux phrases et dans le langage de l'utilisateur, d'une fonctionnalité à développer.
- Une tâche.
- Un bug.

On peut définir un niveau de priorité pour chaque demande (Lowest, Low, Medium, High, Highest), estimer sa durée, y ajouter des fichiers, etc.

- **Le Workflow :**

Il s'agit d'une suite d'états, liés entre eux par des transitions. Un workflow est associé à un type de demande, au sein d'un système de workflow.

Jira a ses propres workflows intégrés. Ceux-ci incluent :

- La gestion des tâches : Le workflow le plus simple possible pour effectuer les tâches le plus rapidement possible.
- La gestion de projet : Un workflow plus complexe, qui inclut un statut « En cours » pour mieux marquer le travail effectué sur la tâche.
- La gestion de processus : Une structure dotée de plusieurs statuts et résolutions, qui commence à refléter la complexité des processus métier et de développement.

Cependant, il pourrait être préférable de créer notre propre workflow personnalisé. Le logiciel Jira offre 2 méthodes pour ça :

- Une interface textuelle.
- Une interface visuelle sous forme de diagramme (disponible à partir de la version 6.1 de Jira), aussi appelée workflow designer.

2.2. Étude comparative

Dans cette partie, nous allons comparer **Redmine**, **Trello** et **Jira** pour mettre en lumière leurs différences fondamentales en nous basant sur des critères bien précis, ainsi que l'observation des atouts de chacun afin de commencer la réalisation d'une solution personnalisée qui sera plus adaptée au besoin de l'entreprise.

Le tableau ci-dessous synthétise la comparaison de ces trois outils présentés préalablement, selon les critères suivants [cf. Table 1] :

Outil / Critère	REDMINE	Trello	Jira
Open source	Oui	Non	Non
Licence	Gratuit	Gratuit/payant	Payant
Ergonomie	Mauvaise	Bonne	Bonne
Suivi du temps	Non	Non	Non
Progression et suivi de projets	Oui	Non	Oui
Glisser-Déposer	Non	Oui	Oui
Reporting (Tableau de bord)	Non	Non	Oui
Kanban (tableau)	Non	Oui	Oui
Mis en place	Complexe	Simple	Moyenne
Multilingue	Oui	Oui	Oui
Technologie	Ruby on Rails	CoffeeScript	Java

Application web	Oui	Oui	Oui
Application Android / IOS	Non	Oui	Oui
Application Mac OS / Windows / Linux	Non	Non	Oui

Table. 1 : Récapitulatif de comparaison des outils de gestion de projets

2.3. Critique de l'existant

Chaque entreprise est différente et peut avoir besoin d'une solution logicielle de gestion de projet spécifique, qui sera adaptée à la taille de l'entreprise, au type de clients et d'employés.

Après une analyse de l'existant, nous avons remarqué que les outils de gestion de projets utilisés par les équipes présentent les limites suivantes :

2.3.1. Redmine

- La mise en place est complexe et longue.
- L'interface n'est pas ergonomique, trop de temps sera perdu à la gestion lors d'ajouts massifs.
- Le manque d'un tableau de bord, notamment sur les indicateurs de performance.
- Le paramétrage du système est lourd.
- En fonction de projets, il existe diverses fonctionnalités dont nous n'avons probablement pas besoin.
- Manque de documentation pour certains plug-ins.
- Manque de planification agile et reporting.
- Manque du tableau Kanban.

Redmine reste un point de départ pour organiser la gestion de projet. Pour mettre l'accent sur le collaboratif, la productivité et le pilotage efficace de projet, alors Redmine, dans sa version brute est limité. Il s'agira alors de s'orienter vers une solution propriétaire qui comblera ces lacunes.

2.3.2. Trello

Trello fournit des fonctionnalités qui aident les entreprises à suivre leur collaboration. Cependant, il n'est pas destiné aux projets complexes et n'inclut pas toutes les fonctionnalités dont l'entreprise a besoin. Trello présente les limites suivantes :

- Beaucoup de manques du côté technique. Il apparait trop simpliste pour des projets d'une complexité moyenne à élever.
- Il faut une licence pour pouvoir utiliser la version business class.
- Le Time Tracking n'est actuellement pas disponible.

- L'utilisation est limitée aux projets ponctuels et non à la productivité quotidienne (gestion de tâches en équipe).

2.3.3. Jira

Parmi les concurrents directs de Redmine on trouve Jira qui est le plus connu dans le milieu informatique, mais son coût élevé et sa lourdeur rebute certaines des équipes. Jira présente les limites suivantes :

- Trop lent avec une interface beaucoup plus complexe avec un manque de convivialité.
- Il faut une licence pour pouvoir l'utiliser.
- Les éléments d'intégration comme le Time Tracking sont des plug-ins payants.
- La capacité de personnalisation à tous les niveaux (flux, états…) rend l'administration très complexe.
- Demande d'apprentissage élevée à cause de sa robustesse.
- Son coût reste cher pour les entreprises qui ont un nombre important de collaborateurs par rapport aux autres produits qui se trouvent sur le marché.

À cause de ces lacunes, et puisque le développement nécessite de la concentration, il se peut que le développeur rencontre des problèmes tels que des oublis à cause de la charge du travail, une perte de temps non négligeable, un stresse contre la montre qui est inutile. En général, ces outils ne sont pas très performants en termes d'imputations quotidiennes sur d'autres plateformes professionnelles.

3. Solution proposée

Ce projet ayant été déclenché dans le pôle Front-End au sein de l'agence Smile Maroc dans un cadre de recherche et développement, avec la principale valeur ajoutée sur laquelle il faut absolument se concentrer c'est que le développeur doit passer son temps à développer et non pas perdu dans la gestion de ses activités quotidienne.

Pour remédier aux limites des outils cités dans l'étude de l'existant, nous avons proposé la réalisation d'un système de gestion des tâches et de tracking basé sur ces solutions existantes manipulées quotidiennement, tout en conservant les fonctionnalités importantes et les plus fréquemment utilisées, avec l'ajout de nouvelles fonctionnalités pour répondre mieux aux besoins des équipes et unifié leur manière de travail, on leur offrant une plateforme commune et personnalisée.

Donc notre but est de mettre en place une application permettant la gestion des utilisateurs et des projets, l'application remplit également la mission de gestion collaborative des tâches quotidiennes de chaque collaborateur, et tout ceci dans une ergonomie flat et simple à utiliser, l'application doit être utilisable sur Smartphone, web et aussi bien sur Desktop.

4. Démarche adoptée

Dans cette partie, nous présentons le choix méthodologique et la démarche de travail adopté pour la gestion et la réalisation de notre projet.

4.1 Choix méthodologique

L'organisation est la clé du succès lorsque nous parlons de développement logiciel, et puisque notre projet est déclenché dans ce contexte, nous sommes très conscients de l'importance d'utiliser une méthode efficace qui nous permet de simplifier le processus et d'optimiser au plus le bon cadrage du projet.

La méthode agile Scrum s'impose alors afin qu'on puisse cadrer notre projet et s'assurer de son bon déroulement pour répondre au mieux au besoin des équipes, ainsi, il nous permet d'avoir plus de visibilité sur ce que chaque membre de l'équipe a fait.

4.2 La méthode Agile Scrum

Scrum est une approche légère et agile de gestion de projet qui permet de répondre à des problèmes complexes d'une manière productive et créative afin de fournir des produits de la plus haute valeur possible. Cette méthode met l'accent sur la responsabilité, le travail d'équipe et le progrès itératif vers un objectif bien défini [7].

4.2.1 Principe de Scrum

La méthode Scrum s'appuie sur le découpage d'un projet en boites de temps, nommées « **sprints** ». Les sprints peuvent durer entre une semaine et un mois. Chaque sprint commence par une estimation suivie d'une planification opérationnelle, et se termine par une démonstration de ce qui a été achevé. Avant de démarrer un nouveau sprint, l'équipe réalise une rétrospective. Cette méthode analyse le déroulement du sprint achevé, afin d'améliorer ses pratiques. Le cycle de vie d'un projet Scrum est rythmé par un ensemble de réunions clairement définies et strictement limitées dans le temps.

Voilà un schéma qui résume le processus de la méthodologie Scrum que nous avons adoptée tout au long de notre projet :

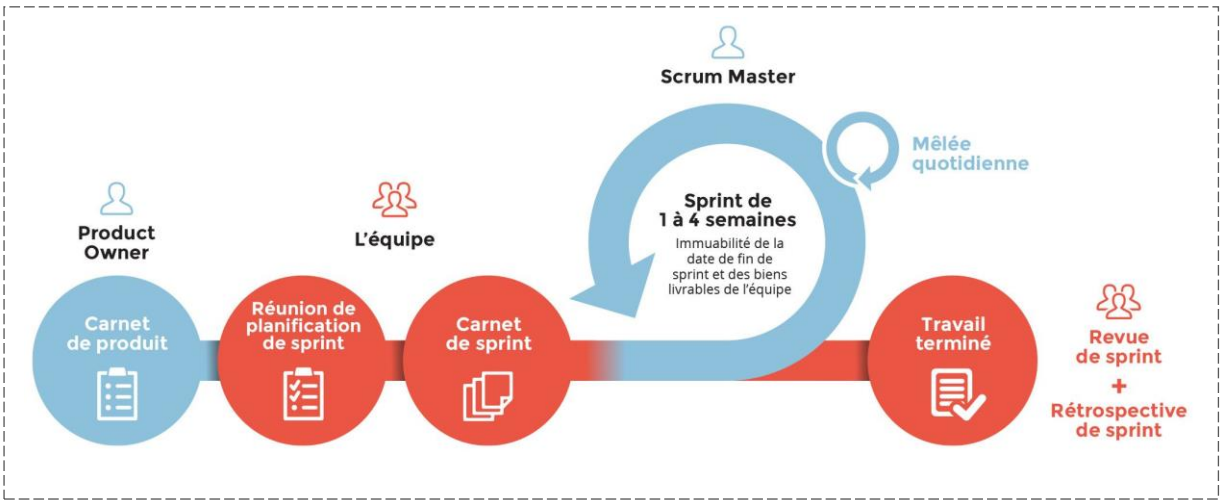


Figure. 12 : Vue globale du processus Scrum

Carnet de produit : La constitution du « **Product backlog** » est réalisée par les membres des équipes Smile ainsi que les chefs et directeurs de projets avant le lancement du premier sprint, ce carnet va être composé de toutes les demandes de fonctionnalités priorisées, ainsi que tous

les éléments nécessitant l'intervention de l'équipe du projet. Tous les éléments inclus dans le backlog Scrum sont classés par priorité indiquant l'ordre de leur réalisation.

Planification des sprints : La réunion de planification du sprint est l'une des étapes les plus importantes d'un projet Scrum, au cours de cette réunion, l'équipe de développement planifie le travail à réaliser au cours du sprint.

Carnet de sprint : Cet extrait du « **Product backlog** » correspond à tout le périmètre qui doit être produit au cours du prochain « **sprint** ». Il peut être présenté sous la forme d'un Scrum Board pour rendre visible toute la gestion de l'itération.

Mêlée quotidienne : Chaque jour du sprint, on assiste à la réunion quotidienne, qui est appelée aussi « Stand up meeting », cet évènement à une durée courte (de 5 à 10 minutes), il s'agit d'une synchronisation de l'équipe de développement, qui nous permet de partager ce que nous avons fait la veille, ce que nous faisons le jour même, ainsi que l'identification de tout problème pouvant gêner le bon déroulement du sprint.

Revue de sprint : Au cours de cette réunion qui a lieu à la fin du sprint, nous présentons les fonctionnalités terminées au cours du sprint et nous recueillons les feedbacks des utilisateurs finaux. C'est également le moment d'anticiper le périmètre des prochains sprints et d'ajuster au besoin la planification de release (nombre de sprints restants).

4.2.2 L'équipe Scrum

L'équipe Scrum se compose d'un « **Product Owner** », de l'équipe de développement et d'un « **Scrum Master** » qui sont tous autoorganisé et inter fonctionnel, c.à.d. ils choisissent la meilleure façon d'accomplir leur travail, plutôt que d'être dirigés par des personnes extérieures de l'équipe. Cette équipe a toutes les compétences nécessaires pour accomplir le travail. Le modèle d'équipe dans Scrum est conçu pour optimiser la flexibilité, la créativité et la productivité.

Scrum définit donc trois rôles particuliers, chacun ayant une fonction bien précise :

Scrum Master : Qui doit maitriser la méthodologie Scrum et s'assurer que cette dernière est correctement appliquée. Il a donc un rôle de coach à la fois auprès du « **Product Owner** », et auprès de l'équipe de développement. Il doit donc faire preuve de pédagogie. Il est également chargé de s'assurer que l'équipe de développement est pleinement productive.

Product Owner : Qui sert d'interface entre l'équipe de développement et les autres parties, il représente la personne chez le client qui est responsable du produit en cours de réalisation, son rôle est de valider les solutions et vérifier si la qualité est acceptable du point de vue de l'utilisateur final. Il doit également indiquer à l'équipe à quoi le produit doit ressembler à la fin.

L'équipe de développement : mets en œuvre les solutions techniques, et réalise les développements. Elle est souvent composée de trois à dix développeurs, elle va travailler de façon incrémentale et livrer une partie du produit final utilisable et testable à la fin de chaque sprint ou itération.

4.3 Planification du projet

L'élaboration d'un plan de projet est l'une des étapes les plus importantes pour garantir la réalisation du projet dans les délais impartis. Il consiste à définir les objectifs, les tâches à réaliser ainsi que les étapes à suivre pour atteindre ses objectifs, c'est l'un des processus les plus importants qui composent la gestion de projet. Un plan bien développé est l'un des facteurs essentiels au succès du projet.

4.3.1 Découpage du projet en sprints

L'approche Scrum propose de commencer par lister les exigences initiales afin de produire le Product Backlog, ainsi, les tâches planifiées ne doivent pas nécessairement contenir toutes les fonctionnalités attendues dès le début du projet, ils vont évoluer durant le projet en parallèle des besoins du client. Nous avons établi en premier lieu un découpage de projet en sprints, puis nous avons réalisé un diagramme de Gantt qui décrit les grands traits des tâches impliquées dans notre projet, ainsi que leur ordre, en fonction d'une échelle de temps.

Tâche	Details
Intégration et planning	o Brainstorming des idées et discussion à propos des différentes problématiques rencontrées. o Finalisation des besoins techniques et spécifiques. o Création d'un cahier de charges à respecter en termes de besoin fonctionnel assez personnalisé. o Choix méthodologique et technique pour le développement de la solution. o Choix des plateformes sur lesquels la solution doit tourner (web, Mobile, Desktop).
Formation technique 1	o Formation en MEAN Stack (MongoDB, ExpressJS, Angular 7, NodeJS) avec l'équipe de développement pour bien cadrer les aspects techniques à prendre en considération au cours du projet.
Formation technique 2	o Formation en Ionic pour la partie Mobile.

Réalisation de la partie Front-End	o Étude des maquettes au sein du pôle Front pour un UI /UX élégant et simple à utiliser. o Initialisation du projet Angular et installation des modules nécessaires. o Initialisation du projet sous Git. o Création des différentes vues (Pages, Components, Dialogs). o Création des différents modèles, services et couches réseau.
Réalisation de la partie Front-End (Suite)	o Initialisation des données fictives sous JSON Server. o Connexion du serveur fictif avec le Front-End pour tester les différentes vues. o Révision et optimisation du code. o Fusion des différentes branches et modules créent. o Finalisation des vues avec le traitement des bugs fixes.
Réalisation de la partie Back-End	o Configuration de l'environnement avec toutes les dépendances nécessaires. o Définition des différentes routes (points finaux d'application). o Création des schémas de données. o Configuration de la base de données. o Sauvegarde et récupération des données. o Gestion de l'authentification et l'autorisation. o Teste de l'API à l'aide de Postman. o Connexion avec le Front-End o Détection et correction des anomalies, et optimisation du code.
Réalisation de la partie Mobile	o Préparez l'environnement de développement avec l'installation des différents modules requis pour le développement avec Ionic (Android Studio, Graddle, Cordova, etc.) o Création des vues inspirées de la partie web. o Création des différents modèles, services et couches réseau. o Connexion avec l'API REST o Tests et vérification des données par rapport aux modèles fournis par le Front. o Détection et correction des anomalies, et optimisation du code.

Réalisation de la partie Desktop	o Création de l'application Desktop

Table. 2 : Liste des tâches et description

4.3.2 Diagramme de Gantt

Le diagramme de Gantt transmet les différentes tâches ainsi que leurs durées de manière visuelle, cela nous donne un aperçu instantané de notre projet, de ses tâches associées et du moment où elles doivent être terminées.

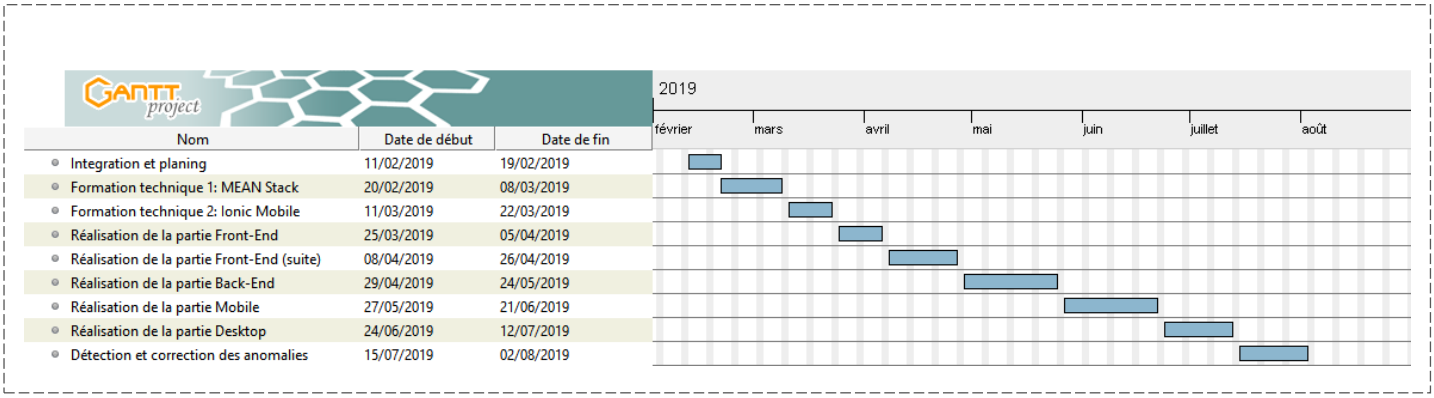


Figure. 13 : Diagramme de Gantt

Conclusion

Dans ce premier chapitre, nous avons commencé par une présentation de l'organisme d'accueil, ses références, et ses activités, après nous avons étudié le fonctionnement actuel de l'entreprise afin de bien identifier les fonctionnalités qu'on doit offrir, et nous avons fini par la présentation de la démarche adoptée pour la gestion de notre projet.

Le chapitre suivant sera consacré à l'analyse et la conception de la solution.

Chapitre 2. Analyse et conception

Introduction

Dans ce chapitre, nous entamons l'analyse et la conception du projet, qui sont considérées comme une étape importante avant la réalisation. Nous commençons tout d'abord par définir le périmètre de notre projet. Ensuite nous passons à l'analyse des besoins dans lesquelles nous allons décrire les futures fonctionnalités de notre solution, ainsi que les acteurs qui vont interagir avec le système. Nous finissons par l'identification des cas d'utilisations principaux de notre projet et la modélisation des besoins par les différents diagrammes UML.

1. Périmètre du projet

La solution étant créée suite au besoin perçue au cours des précédents projets, elle reste limitée à une utilisation interne pour le moment, afin de l'améliorer de plus en plus au fil du temps. Néanmoins, la solution reste compatible pour tout type de travaux ou de gestion des projets et des tâches, du fait que tout projet doit suivre des étapes afin d'aboutir à une solution finale.

Dans notre projet, le challenge étant de créer un package de solutions Web Responsive, Mobile et Desktop, aux différents développeurs, team leads, testeurs, chefs de projets et directeurs de projets. L'application supporte aussi le multilingue (FR - EN) pour permettre à tout type d'individu de l'utiliser avec simplicité en changeant simplement la langue depuis la barre de navigation.

2. Analyse des besoins

Dans cette section, nous commençons par analyser les besoins fondamentaux de notre solution à travers la partie des besoins fonctionnels et non fonctionnels.

2.1. Besoins fonctionnels

L'application disposera de deux volets principaux :

- **Le Back-office**, qui se base sur la création des projets, des tickets et d'utilisateurs, ainsi que l'assignation d'utilisateurs à un certain projet, en plus d'une visualisation de données dans un tableau de bord bien décrit, pour s'assurer que les projets sont sur la bonne voie.

- **Le Front-office** se compose d'un espace **Glisser-Déposer** de tickets selon leurs états d'avancement, il est doté de colonnes **(TODO, IN PROGRESS, DONE, CODE REVIEW/TEST, CANCELLED)** où chacune contient des tickets appropriés à un certain projet. Il y a aussi une liste de projets qui va permettre aux utilisateurs de naviguer entre les projets en cours, pour pouvoir visualiser les différents tickets relatifs à chaque projet.

L'application doit être dotée d'un système de log (journalisation) du temps passé sur chaque ticket avec l'ajout d'un message adéquat. Dans cette vue, le développeur peut générer et exporter l'historique des différentes tâches effectuées en lui indiquant la référence des tickets

assignés ainsi que toutes les modifications associées. Ce log permet d'avoir une traçabilité sur le travail réalisé.

La partie Desktop étant le **Reminder**, c'est un module important qui suit et enregistre le temps passé sur un ticket automatiquement. Il permet de basculer entre les projets et les tickets en cours qui sont assignés à l'utilisateur connecté, ainsi être notifié après chaque laps de temps au cas où le développeur a oublié de vérifier son avancement.

2.2. Besoins non fonctionnels

Pour être plus performante, une application a besoin de répondre à un certain nombre d'exigences techniques, fonctionnelles et ergonomiques. Pour cela, notre solution doit être :

- **Ergonomique** : Tous les standards d'ergonomies doivent être présents : interface utilisateur bien claire et simple dans l'utilisation.
- **Portable** : L'application doit être portable sur tous les environnements logiciels (Windows, Mac OS, Linux, etc.).
- **Fiable** : Elle doit être suffisamment stable et disponible pour pouvoir être consulté avec des temps de réponse performants, un affichage correct et sans bug.
- **Sécurisé** : Tous les champs doivent être validés une fois envoyés par le client (au niveau Front-End).
- **Accessible** : Tout collaborateur assigné à un projet peut accéder à l'application qui doit être certes compatible avec les navigateurs les plus récents.
- **Utilisable** : Un utilisateur doit parvenir à réaliser l'action pour laquelle il a consulté l'application de façon efficace, sans effort inutile et de façon satisfaisante pour lui.
- **Intuitive** : Elle doit présenter l'information utile au moment où l'utilisateur en a besoin, en adoptant une structure compréhensible à l'utilisateur. L'utilisateur doit pouvoir se servir de l'application avec un temps minimum d'apprentissage préalable.
- **Engageante** : L'application doit créer une relation personnelle de confiance avec les collaborateurs, en l'impliquant dans son usage et en devenant un allié indispensable pour lui.

3. Identification des acteurs

Dans le cadre de notre analyse, cinq acteurs principaux sont définis :

- **Superviseur** : Cet acteur doit avoir une vue globale en lecture seule, il peut visualiser le Frontoffice et le Backoffice pour voir ce que les développeurs ou les CP (Chef de projets) font comme, mais il ne peut en aucun cas modifier l'avancement du projet ou bien l'état des tickets.
- **Chef de projet** : C'est l'acteur qui a la main sur toutes les fonctionnalités offertes par notre solution. Il doit avoir un CRUD (Create Read Update Delete) sur les utilisateurs de l'application, les projets et les tickets côté Backoffice, et peut aussi interagir avec le Frontoffice par la modification de l'avancement des tickets, l'ajout des commentaires sur ces tickets, les assigner à des développeurs, la modification de leurs états, etc.
- **Team Lead** : C'est le responsable d'équipe qui possède uniquement l'accès au Frontoffice. Il peut assigner des tickets aux développeurs, faire avancer leurs états, exporter un log sur un projet, etc.

- **Développeur** : Il a les mêmes rôles qu'un Team Lead, mais sans avoir la possibilité de changer l'état d'un ticket en **CANCELLED**, ou bien d'assigner un ticket à un développeur, qui sont des fonctionnalités valables seulement pour les Team Leads et les Chefs de projet.
- **Testeur** : Il a les mêmes rôles qu'un développeur, mais il peut seulement modifier les tickets en **REVIEW/TEST** en y ajoutant des commentaires pour que les développeurs puissent fixer les bugs ou faire des évolutions.

4. Modélisation des besoins fonctionnels

Dans cette section, nous présentons les diagrammes des cas d'utilisation. Cette phase représente la vue fonctionnelle de l'architecture du système. Dans ce qui suit, nous identifions les cas d'utilisation principaux de notre projet. Cette identification sera par la suite suivie d'une description détaillée de chacun d'eux.

4.1. Diagrammes des cas d'utilisation

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel.

4.1.1. Chef de projet

Nous avons commencé par le diagramme de cas d'utilisation du chef de projet puisqu'il englobe toutes les fonctionnalités offertes par le système. La figure 14 présente les cas d'utilisation qui mettent en évidence les fonctionnalités de cet acteur :

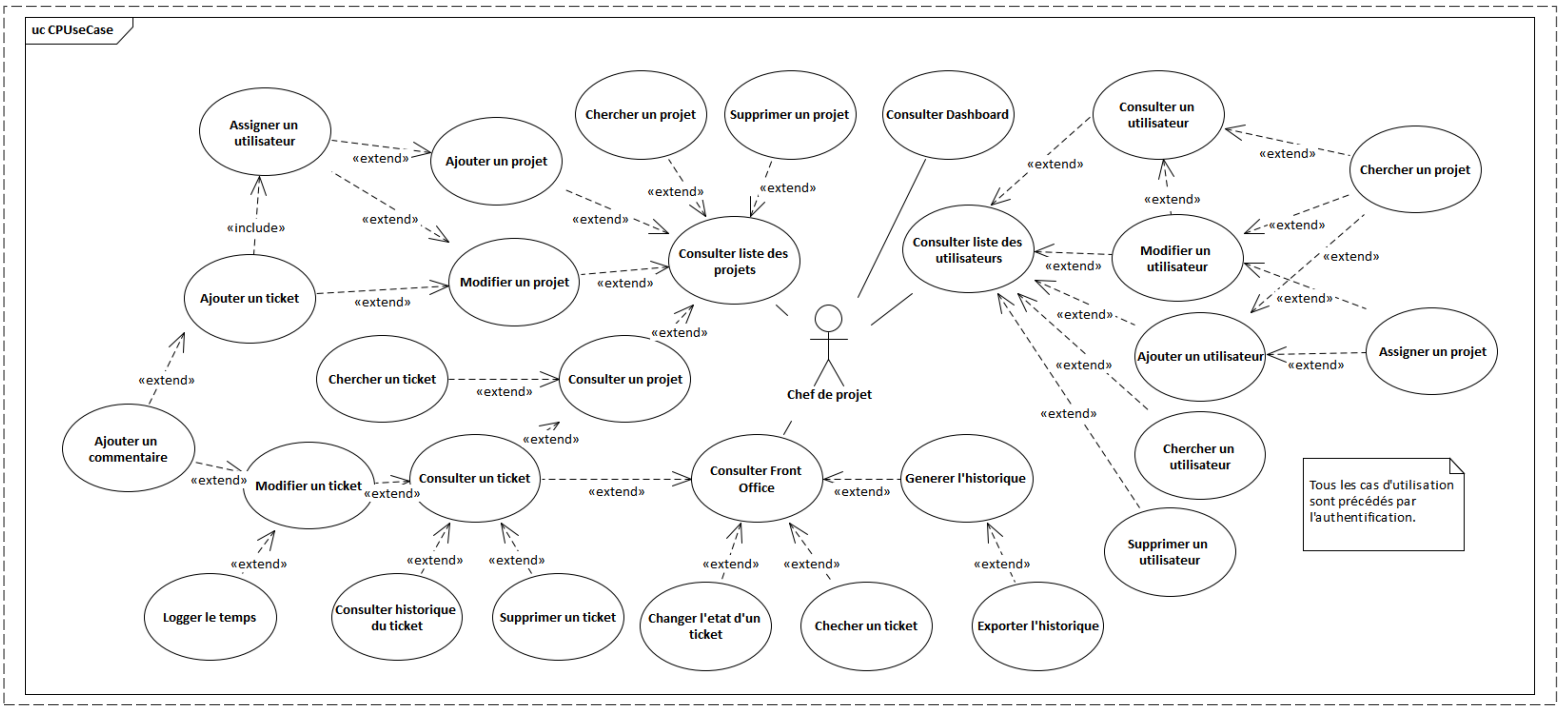


Figure. 14 : Diagramme de cas d'utilisation du chef de projet

4.1.2. Superviseur

La figure 15 présente les cas d'utilisation qui mettent en évidence les fonctionnalités du superviseur :

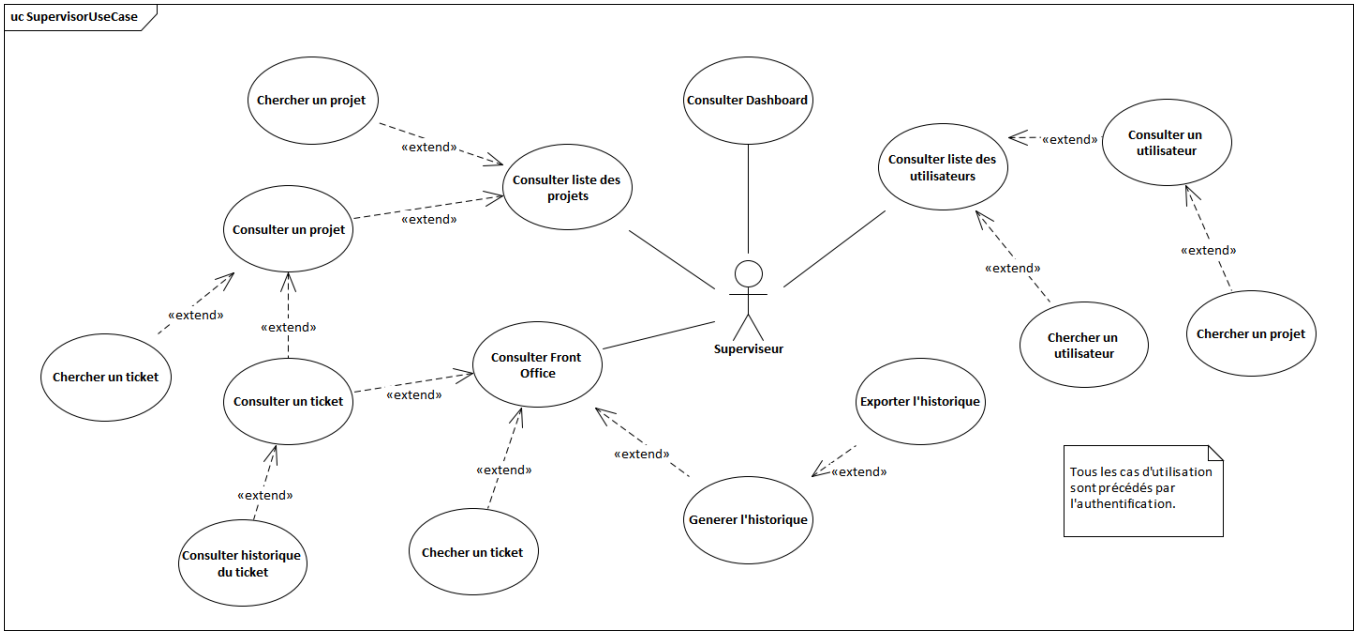


Figure. 15 : Diagramme de cas d'utilisation du superviseur

4.1.3. Développeur

La figure 16 présente les cas d'utilisation qui mettent en évidence les fonctionnalités du développeur :

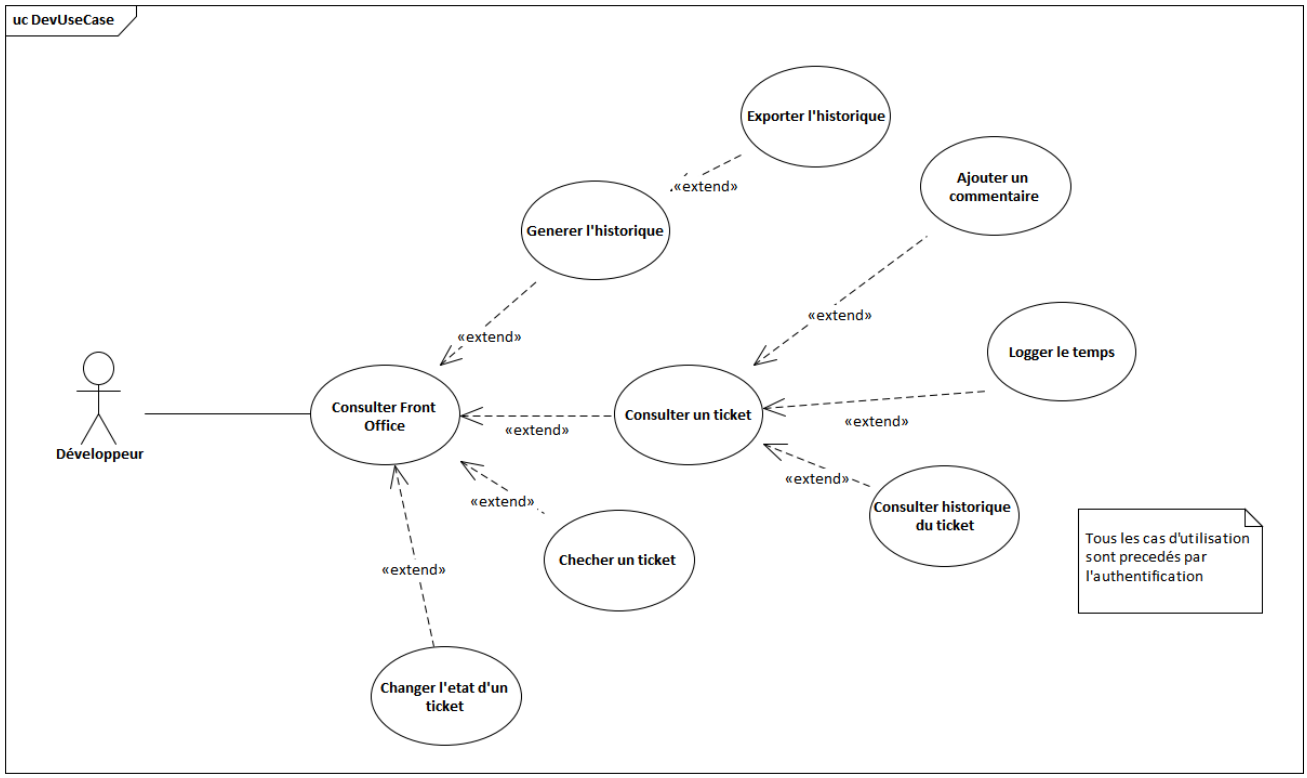


Figure. 16 : Diagramme de cas d'utilisation du développeur

4.2. Description des cas d'utilisation

Pour bien comprendre les différentes fonctionnalités mentionnées dans les diagrammes des cas d'utilisations illustrés dans les figures 14, 15 et 16, nous allons décrire les services les plus importants :

- **Gestions des projets**

Dans le tableau 3, nous décrivons les cas d'utilisations relatives à la gestion des projets :

Cas d'utilisation	Acteurs	Description
Consulter la liste des projets	Chef de projet Superviseur	Ces deux utilisateurs peuvent consulter la liste des projets dans un tableau qui contient le nom des projets créés.
Consulter un projet	Chef de projet Superviseur	À partir de la liste des projets, les acteurs peuvent accéder à l'ensemble des détails du projet, et cette consultation inclut l'accès aux tickets relatifs à ce projet.
Chercher un projet	Chef de projet Superviseur	Depuis la liste des projets, les utilisateurs qui ont accès au Backoffice peuvent effectuer des recherches sur l'ensemble des projets.
Modifier un projet	Chef de projet	Le chef de projet peut modifier un projet à partir de la liste ou bien après la consultation du projet. Il peut modifier toutes les informations relatives à ce projet avec la possibilité d'ajouter un ticket à ce projet depuis cette vue.
Supprimer un projet	Chef de projet	Le chef de projet peut supprimer un projet seulement depuis la liste des projets.
Ajouter un projet	Chef de projet	À partir de la liste des projets, le chef de projet peut ajouter un nouveau projet avec un nom, une date de début et une date de fin. Le chef de projet peut affecter des utilisateurs à ce projet au moment de la création.
Assigner un utilisateur	Chef de projet	Cette assignation peut être faite soit au moment de la création ou bien la modification d'un projet.

Générer l'historique	Tous les acteurs	Depuis la navigation, tous les utilisateurs de l'application peuvent générer un Log (un historique) des différentes tâches effectuées sur un projet en indiquant le nom de projet et un intervalle de temps. Ce log est exporté en format Excel.

Table. 3 : Description des cas d'utilisation relatifs aux projets

- **Gestions des tickets**

Dans le tableau 4, nous décrivons les cas d'utilisations relatives à la gestion des tickets :

Cas d'utilisation	Acteurs	Description
Consulter un ticket	Tous les acteurs	La consultation d'un ticket est accessible pour le chef de projet et le superviseur depuis le modal qui contient les informations d'un projet. Cette consultation et accessible aussi depuis le front-office pour les autres acteurs qui n'ont pas accès au backoffice.
Chercher un ticket	Tous les acteurs	Après la consultation d'un projet, les utilisateurs qui ont accès au Backoffice peuvent effectuer des recherches sur l'ensemble des tickets par référence ou par titre. La recherche est accessible aussi depuis le front-office.
Modifier un ticket	Chef de projet Team Lead Développeur	Le chef de projet peut modifier toutes les données d'un ticket depuis le modal. Le team Lead peut modifier l'assignation du ticket à un développeur ainsi que l'état du ticket qui peut être modifié par le développeur.
Supprimer un ticket	Chef de projet	Le chef de projet peut supprimer un ticket depuis son modal.
Ajouter un ticket	Chef de projet	À partir du modal d'un projet, le CP peut ajouter un nouveau ticket avec un titre et une description, il doit assigner ce ticket à un utilisateur au moment de la création.
Loger le temps	Chef de projet Team Lead Développeur	Au moment de la modification d'un ticket, l'utilisateur peut logger le temps passé sur ce

	Testeur	ticket avec l'ajouter d'un message qui décrit la tâche effectuée.
Changer l'état d'un ticket	Chef de projet Team Lead Développeur	L'état d'un ticket peut être changé après la consultation et la modification du ticket, ou bien à partir le front-office par glisser-déposer.

Table. 4 : Description des cas d'utilisation relatifs aux tickets

- **Gestions des utilisateurs**

Dans le tableau 5, nous décrivons les cas d'utilisations relatives à la gestion des utilisateurs :

Cas d'utilisation	Acteurs	Description
Consulter la liste des utilisateurs	Chef de projet Superviseur	Ces deux acteurs peuvent consulter la liste des utilisateurs dans un tableau qui contient le nom des utilisateurs créés par le chef de projet.
Consulter un utilisateur	Chef de projet Superviseur	À partir de la liste des utilisateurs, les acteurs peuvent accéder à l'ensemble des détails d'un utilisateur.
Chercher un utilisateur	Chef de projet Superviseur	Depuis la liste des utilisateurs, les acteurs qui ont accès au Backoffice peuvent effectuer des recherches sur l'ensemble des projets.
Modifier un utilisateur	Chef de projet	Le chef de projet peut modifier un utilisateur à partir de la liste ou bien après la consultation de l'utilisateur. Il peut modifier toutes les informations relatives à cet utilisateur.
Supprimer un utilisateur	Chef de projet	L'acteur peut supprimer un utilisateur seulement depuis la liste des utilisateurs.
Ajouter un utilisateur	Chef de projet	À partir de la liste des utilisateurs, le chef de projet peut ajouter un nouvel utilisateur en renseignant les champs nécessaires. Il peut **assigner des projets** à cet utilisateur au moment de la création.
Assigner un projet	Chef de projet	Cette assignation peut être faite soit au moment de la création ou bien la modification d'un utilisateur.

Table. 5 : Description des cas d'utilisation relatifs aux utilisateurs

4.3. Diagrammes de séquences

La description des cas d'utilisation présentée ci-dessus permet de lever l'ambigüité et rendre clairs les cas d'utilisation. Cependant, le texte présente certaines limites puisqu'il est difficile de montrer la succession des enchainements. Il est donc recommandé de compléter la description textuelle par des diagrammes de séquence.

Les diagrammes de séquence constituent des références utiles, car ils décrivent comment et dans quel ordre plusieurs objets fonctionnent ensemble. Ils permettent de :

- Représenter les détails d'un cas d'utilisation UML.
- Modéliser le déroulement logique d'une procédure, fonction ou opération complexe.
- Schématiser et comprendre le fonctionnement détaillé des scénarios

Dans cette section, nous présentons les diagrammes de séquences pour des scénarios susceptibles d'avoir lieu dans notre système, afin de mettre en évidence l'aspect dynamique de l'application.

4.3.1. Diagramme de séquence relatif aux utilisateurs

Dans cette partie, nous présentons le diagramme de séquence relatif à l'authentification [cf. Figure 17].

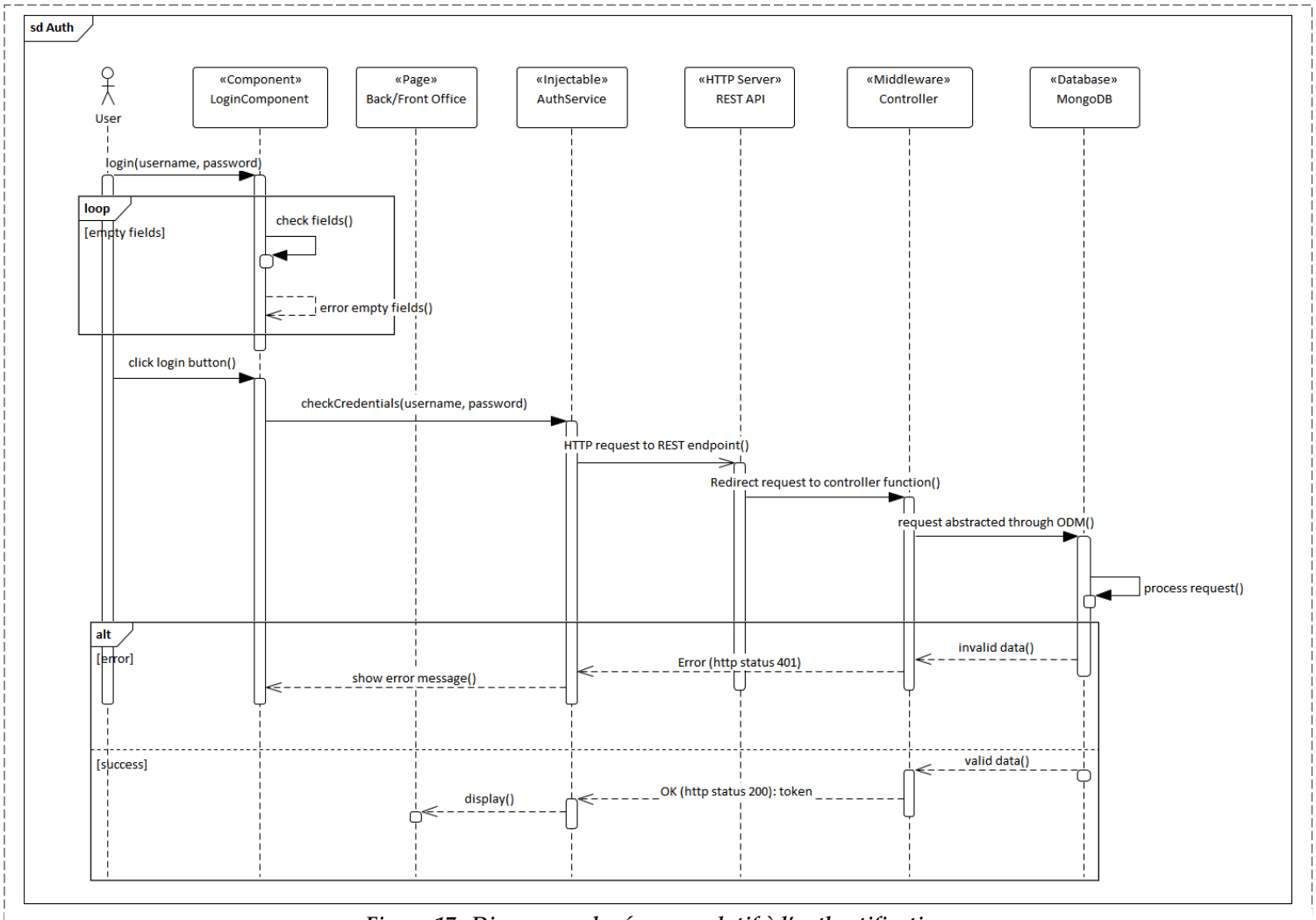


Figure. 17 : Diagramme de séquence relatif à l'authentification.

4.3.2. Diagrammes de séquence relatifs aux projets

Une fois authentifié, le chef de projet peut créer un nouveau projet à travers le modal de création à partir de l'interface de consultation de la liste des projets.

Le chef de projet doit remplir tous les champs nécessaires à la création du projet pour pouvoir activer le bouton de sauvegarde. Sinon, des messages d'erreur s'affichent au-dessous des champs obligatoires non remplis ou bien ceux qui sont mal remplis.

Le scénario de création d'un nouveau projet est décrit dans le diagramme de séquence montré dans la figure 18 :

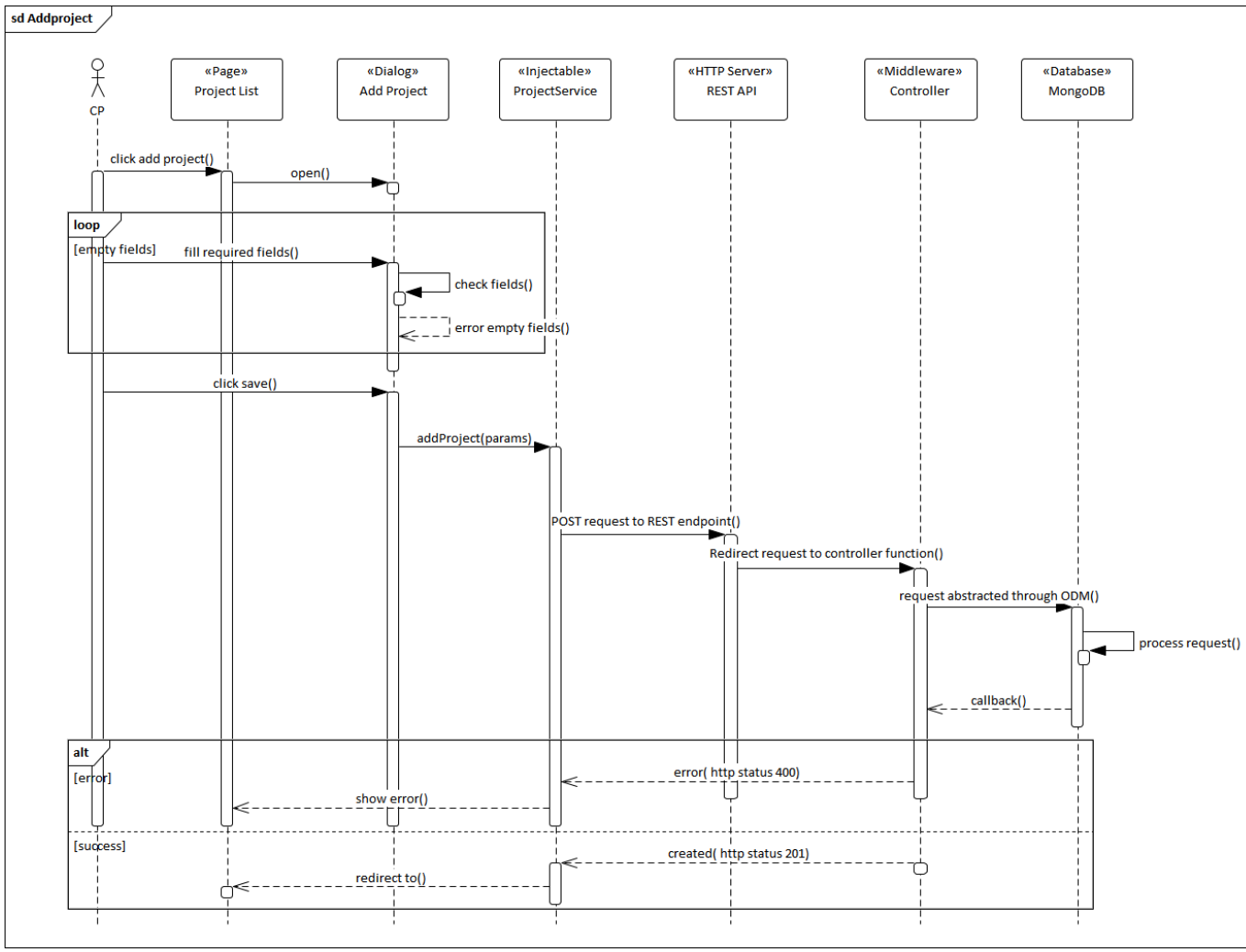


Figure. 18 : Diagramme de séquence relatif à la création d'un nouveau projet

Le scénario de modification d'un projet ainsi que la création d'un nouveau ticket est décrit dans le diagramme de séquence montré dans la figure 19 :

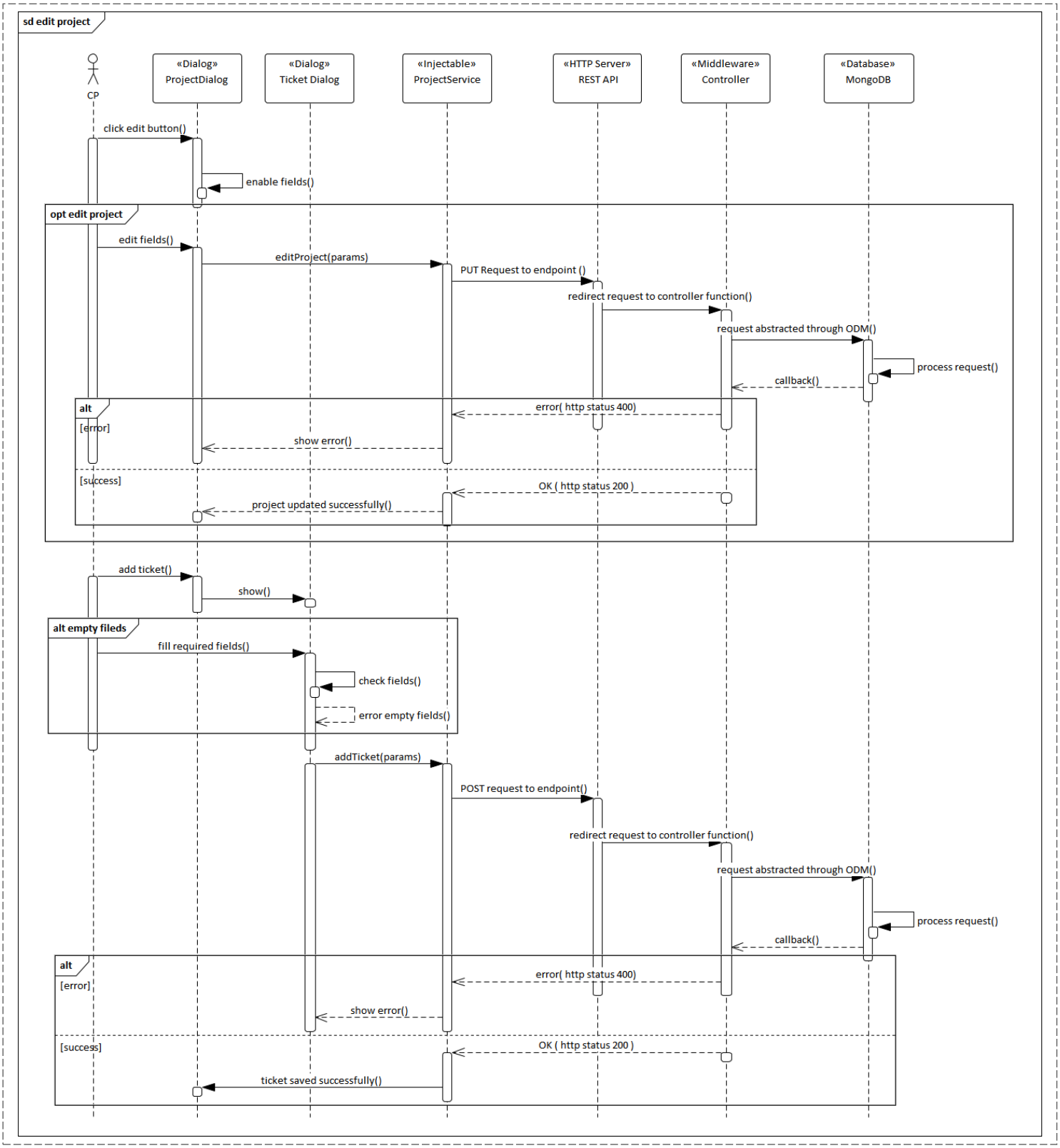


Figure. 19 : Diagramme de séquence relatif à la modification d'un projet

4.4. Diagramme de classes

Les diagrammes de classes sont l'un des types de diagrammes UML les plus utiles. Ils décrivent clairement la structure d'un système particulier en modélisant ses classes, ses attributs et les relations entre ses objets. Ce diagramme aide dans la phase de développement d'applications logicielles. La figure 20 représente le diagramme de classes de notre application.

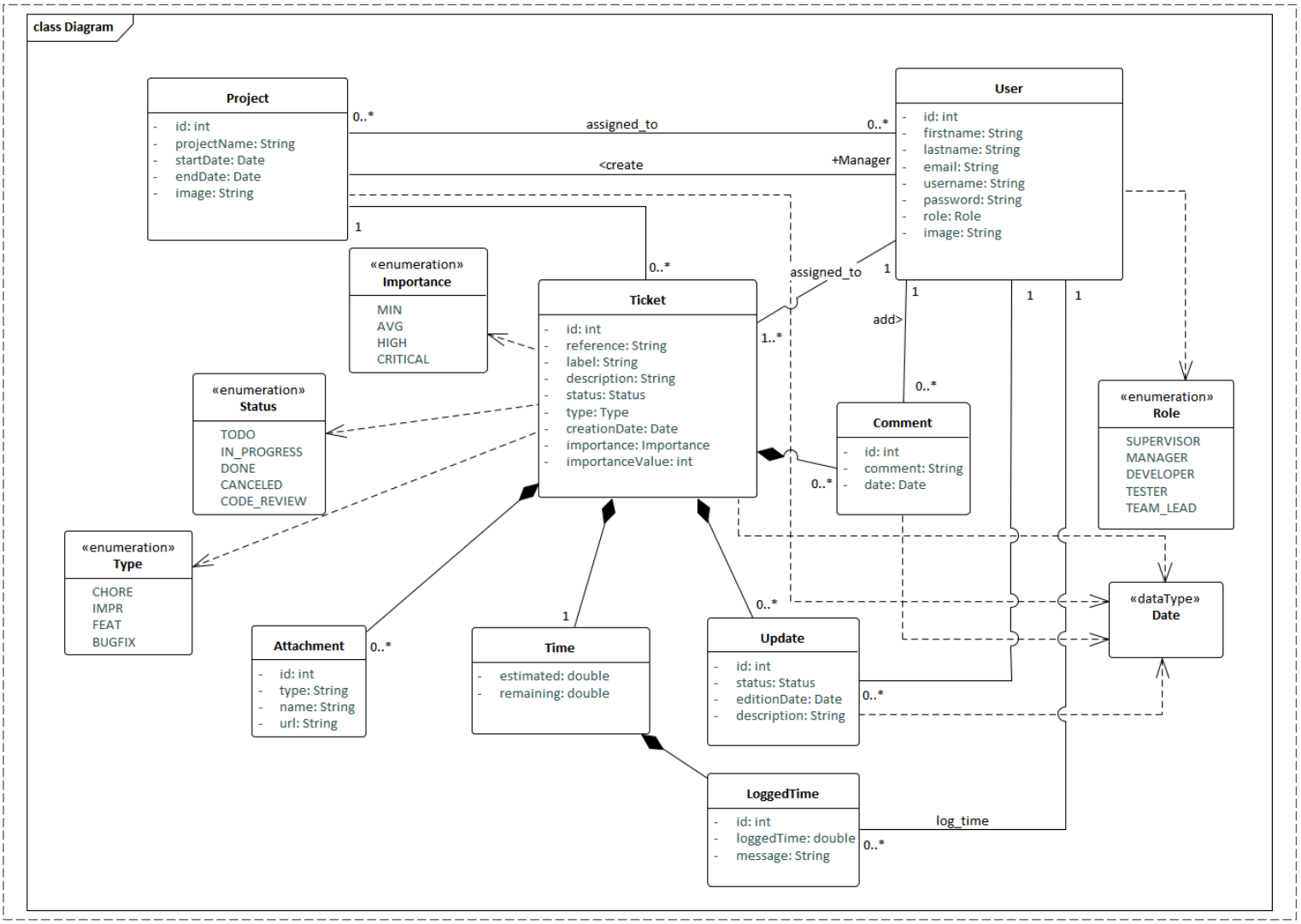


Figure. 20 : Diagramme de classes

4.5. Description des classes et leurs relations

- « **Ticket** » est la classe principale qui contient toutes les informations relatives à un ticket (référence, statut, type, etc.)
- Lors de sa création, un ticket doit être assigné à un seul utilisateur et à un seul projet.
- Les commentaires, les attachements, le temps et les mises à jour font partie du ticket et ne peuvent pas exister sans ticket.
- Chaque ticket a un statut qui décrit son état, il peut être dans les états suivants :
    - **To do :** c'est l'état par défaut que chaque ticket prend lors de sa création, dans ce cas, la tâche n'est pas encore réalisée.
    - **In progress :** ce statut indique que la tâche est en cours de réalisation.

    - **Done :** ce statut indique que la tâche relative à ce ticket est réalisée avec succès.
    - **Code review :** indique que la tâche est réalisée et le code est en cours de révision par le Team Lead si nécessaire.
    - **Cancelled :** la révision du code est terminée et la tâche relative au ticket est finie. Il se peut qu'un ticket prend ce statut sans avoir passé par le Code review. Il se peut aussi que la tâche soit annulée lors de sa réalisation.
- Chaque ticket a un type qui peut être :
    - **Feature :** réalisation d'une nouvelle fonctionnalité dans un projet.
    - **Bugfix :** correction d'un bug à l'origine d'un dysfonctionnement.
    - **Improvement :** amélioration d'une fonctionnalité déjà réalisée.
    - **Chore :** modification relative à la structure du projet ou bien une configuration dans l'environnement du projet.
- Chaque ticket doit avoir un degré d'importance :
    - **Minimal : u**tilisé souvent avec des tâches d'amélioration d'une fonctionnalité.
    - **Average** : utilisé souvent avec des tâches de type Feature qui prennent moins de temps lors de la réalisation.
    - **High :** utilisé souvent avec des tâches de type Feature qui peuvent prendre plus de temps lors de la réalisation.
    - **Critical :** utilisé lors de l'apparition d'un Bug qui doit être corrigé rapidement.
- La classe « **User** » représente un utilisateur de l'application qui peut avoir comme rôle :
    - **Chef de projet :** qui a la main sur toutes les fonctionnalités offertes par l'application.
    - **Superviseur :** qui doit avoir une vue globale en lecture seulement.
    - **Développeur, Testeur et Team Lead :** qui ont l'accès seulement au Front-Office de l'application et ceux qui chargent de la réalisation des différentes tâches.
- Un utilisateur peut être affecté à plusieurs projets et à plusieurs tickets en parallèle.
- Seul le manageur peut créer un projet.
- Un utilisateur peut ajouter des commentaires, faire des mises à jour et loger le temps sur un ticket.
- La classe « **Projet** » désigne un projet créer par le chef de projet, ce projet peut avoir plusieurs tickets et peut être assigné à plusieurs utilisateurs (Développeur, Testeur, Team Lead).

Conclusion

Ce chapitre a été consacré à l'analyse et la conception de la solution proposée, à travers la présentation des différents acteurs du système en les découpant en un ensemble des domaines fonctionnels que nous avons pu traduire en des diagrammes de cas d'utilisation illustrant les besoins fonctionnels de chaque acteur. Par la suite, nous avons élaboré les diagrammes de séquences et les diagrammes de classes afin de délimiter le cadre de notre travail et de préparer un terrain favorable pour la prochaine étape. Dans le chapitre suivant, nous allons nous intéresser à l'étude technique concernant notre projet.

Chapitre 3. Présentation technique

Introduction

Dans ce chapitre, nous allons présenter l'environnement de développent avec les outils que nous avons manipulés, ainsi que l'architecture technique globale de l'application, ainsi

1. Environnement de développement

1.1. La pile MEAN

MEAN est une pile logicielle JavaScript gratuite et à code source ouvert permettant de créer des applications web. MEAN est un acronyme de MongoDB, Express.js, Angular et Node.js.

Parmi les avantages les plus importants de cette pile :

- Permets au développeur d'écrire l'intégralité du code en JavaScript du client au serveur.
- Supporte l'architecture MVC (Model View Controller).
- Les composants MEAN sont open source, qui signifient que la pile est mise à jour régulièrement.
- En plus de cela, elle est facile et flexible à comprendre et à utiliser.

Les plateformes de grande entreprise se tournent vers MEAN comme : PayPal, Yahoo, Netflix, Uber, LinkedIn.

1.1.1 Architecture de la pile

MEAN est une pile logicielle simple et facile à utiliser, car elle réunit des technologies écrites dans un seul langage pour une exécution côté serveur et côté client. Le fonctionnement de la pile est illustré dans la figure ci-dessous :

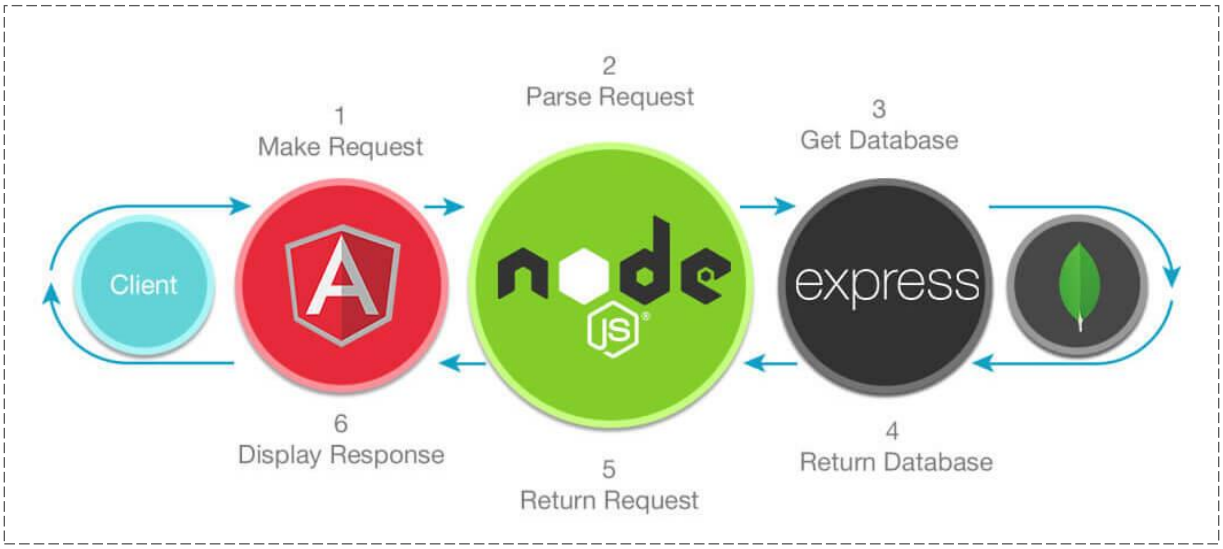


Figure. 21 : Architecture de la pile MEAN

L'explication de ces technologies sera exposée dans les parties qui suivent.

1.2. Partie Front-End

1.2.1. TypeScript

TypeScript devient de plus en plus populaire dans l'environnement Front-End. C'est un langage gratuit et open source développé et maintenu par Microsoft depuis octobre 2012. C'est une surcouche d'ECMAScript permettant l'ajout optionnel de typage statique [8].

Il n'existe actuellement pas de réel runtime TypeScript. Il faut utiliser un compilateur pour transpiler le code TypeScript en code ECMAScript valide que l'on peut ensuite exécuter sur le runtime JavaScript de notre choix : Browser pour le Front-End ou bien Node.js pour le Back-End.

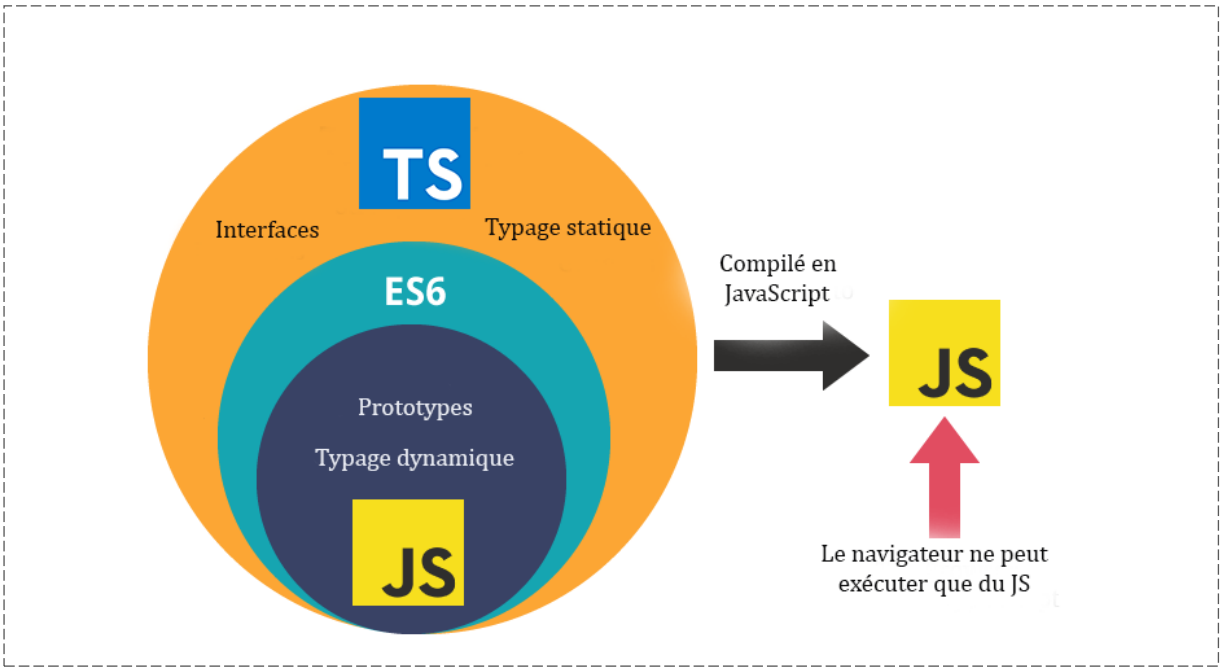


Figure. 22 : Architecture TypeScript

C'est un parfait compromis entre la flexibilité d'un langage dynamiquement typé et la rigueur d'un langage statiquement typé sans tomber dans la lourdeur syntaxique associée. TypeScript est caractérisé par :

Une vraie Programmation orientée objet : Le nouveau standard ES6 a ajouté de nombreux nouveaux outils à JavaScript, comme la syntaxe des classes, l'import/export de modules, une portée de bloc pour les variables, etc. Cela rapproche JavaScript d'autres langages comme Java, C# ou Python.

Un typage statique : JavaScript est un langage permissif. Or dans une application, on ne peut pas se permettre la moindre erreur, sinon l'application plante. Un typage statique des données devient donc indispensable, en évitant tout problème.

Une transpilation ES6 : Nous devons transformer le code TypeScript en JavaScript standard pour les navigateurs, mais dans le développement Front-End d'aujourd'hui, nous devons en premier lieu transformer l'ES6+ en ES5 pour la compatibilité (ce qu'on va voir avec Babel). Une étape de Build est donc nécessaire dans tous les cas. TypeScript se charge de cette transpilation.

Largement adopté : TypeScript est soutenu par Microsoft, Google et d'autres grands acteurs du web. Plusieurs outils JavaScript utilisent TypeScript, et des Frameworks majeurs comme Angular (depuis sa version 2+) sont intégralement en TypeScript.

1.2.2. Angular

Angular est un Framework moderne entièrement construit en TypeScript. À partir d'Angular 2, l'équipe de développement de Google a fait une refonte complète de leur ancienne structure AngularJS, en instaurant la programmation orientée objet avec l'utilisation du langage Typescript qui représente une surcouche de JavaScript qui le sécurise et le simplifie.

L'architecture d'Angular est divisée en différentes parties telles que Composant, Modèle, Directive, Service et Module [9].

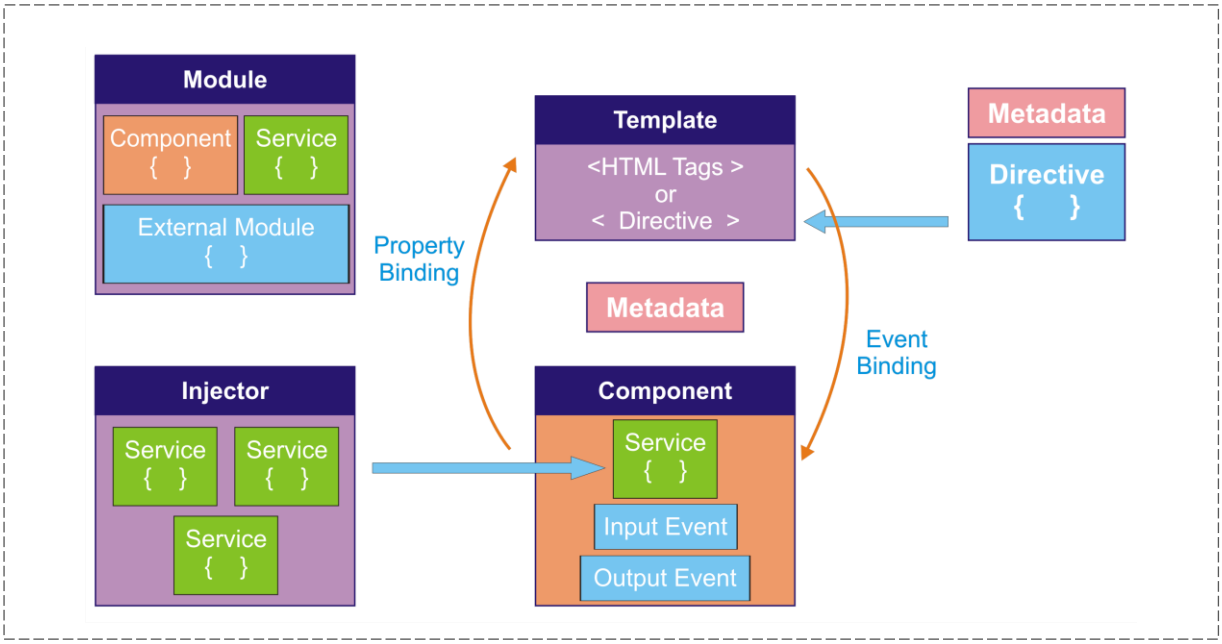


Figure. 23 : Architecture d'une application Angular

Module : Les applications sont modulaires et Angular possède son propre système de modularité appelé **NgModule**. Chaque application a au moins une classe qui représente le module racine. Pour utiliser un composant dans une application, on doit le déclarer dans le module associé.

Component : Le composant est le bloc de construction de base de l'interface graphique qui traite la vue de l'application et qui contient la logique de base de la page. Nous devons écrire la logique d'application à l'intérieur de la classe qui est utilisée par la vue.

Template : Ressemble à un code HTML normal, à quelques différences telles que les directives, évènements, interpolation, liaison de données et autres balises de composant.

Data Binding : Ou liaison des données, Angular prend en charge la liaison de données pour la coordination des parties du Template avec les parties du composant. Il existe deux types de cette liaison : Event Binding ou Property Binding.

Dependency Injection (DI) : l'injection de dépendance est un modèle de conception logicielle dans lequel les objets sont transmis en tant que dépendances. Cela nous aide à supprimer les

dépendances codées en dur et à les rendre configurables. En utilisant DI, nous pouvons rendre les composants maintenables, réutilisables et testables.

1.2.3. Material Design

Le Material design est un langage visuel et interactif créé par Google pour unifier le style graphique de ses applications et de ses plateformes. Il est aussi un guide pour concevoir une interface graphique (Design system) [9].

Les composants de Material Design permettent de créer des interfaces graphiques (User Interface) professionnelles dotées de puissantes fonctionnalités de modularité, de thème et de personnalisation. Il existe plusieurs façons d'implémenter Material design dans un flux de travail. Angular Material est le cas dans un projet basé sur Angular.

Angular Material fournit une série de modules, directives, classes CSS, qui permettent de créer une interface moderne conforme au système de Material Design. Le module offre divers composants pour faciliter le processus de développement, tels que :

- Form controls (input, select, checkbox, date picker, sliders etc.)
- Navigation patterns (menus, sidenav, toolbar)
- Layout components (grids, cards, tabs, lists)
- Buttons
- Indicator (progress bars, spinners)

1.2.4. Sass (CSS Preprocessor)

Un préprocesseur CSS est un langage de script qui étend le CSS en permettant aux développeurs d'écrire du code dans un langage, puis de le compiler en CSS. Sass est peut-être le préprocesseur le plus populaire, mais Less et Stylus sont d'autres exemples courants [10].

Sass (Syntactically Awesome Style Sheets) est une extension de CSS qui vous permet d'utiliser des éléments tels que des variables, des règles imbriquées, des importations, etc. Cela aide également à garder les choses organisées et de créer des feuilles de style plus rapidement.

Quelques lignes répétées de CSS peuvent ne pas sembler être un gros problème pour des petits projets, mais lorsque nous sommes en charge de la gestion de grands projets, les choses peuvent devenir assez désordonnées en cas de négligence. Les préprocesseurs CSS tels que Sass nous aident à réduire ce risque et constituent donc une exigence commune.

1.3. Partie Back-End

1.3.1. JavaScript

Utilisé dans le développement des applications web et mobiles actuels, le JavaScript peut à la fois être utilisé côté client, c'est-à-dire interprété par le navigateur web, et côté serveur avec l'utilisation de Node.js [11].

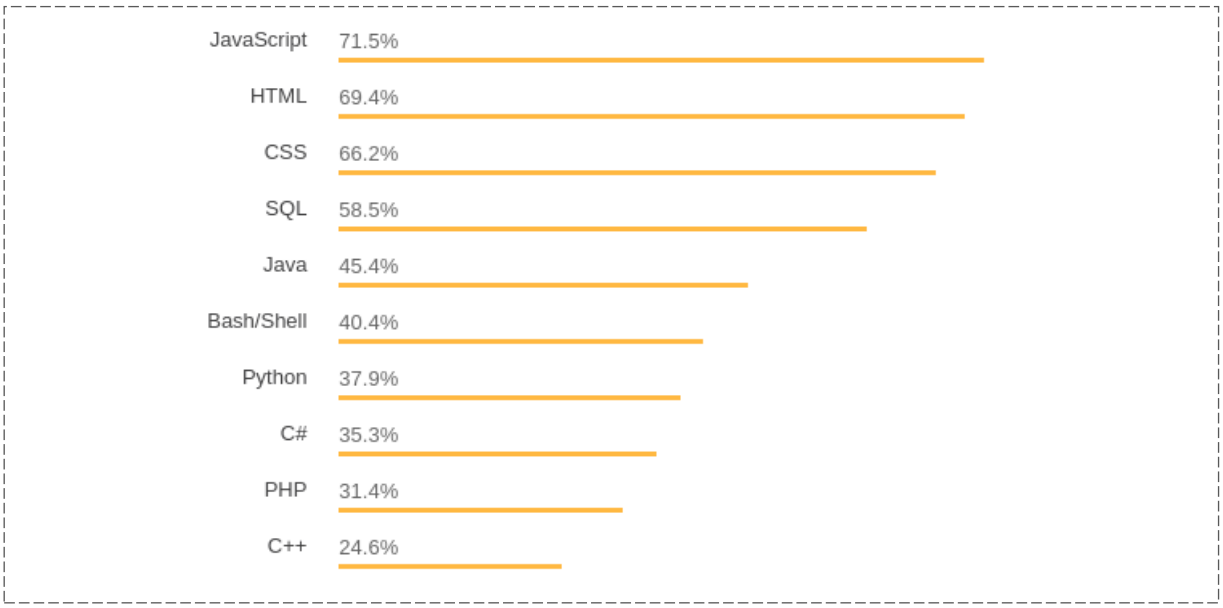


Figure. 24 : Langages les plus utilisés dans l'année 2018 d'après Stackoverflow

Voici un aperçu des multiples possibilités en JavaScript :

Applications web Front-End : En JavaScript pure ou avec des Frameworks comme Angular, React.js ou Vue.js on peut créer des applications web complexes, rapides et performantes.

Serveur HTTP et Back-End : Grâce à Node.js, on peut créer un serveur web ultra réactif, et aussi unifier les compétences frontend et backend autour de JavaScript.

Application Full-Stack : En utilisant un Framework comme Angular, React.js ou Vue.js côté client, et en le combinant à Node.js côté serveur, on peut aujourd'hui créer un site web entièrement en JavaScript, tout en bénéficiant d'un chargement rapide et du référencement naturel.

Applications mobiles natives et hybrides : Grâce aux outils comme Ionic, Cordova, NativeScript ou React Native, les applications web auront aussi accès aux fonctionnalités natives des téléphones et des tablettes.

Logiciels Desktop : Grâce à des outils comme Electron.js, il est également possible de créer des applications de bureau.

Comprendre l'utilité et savoir utiliser JavaScript est donc devenue incontournable, que ce soit pour concevoir des applications dynamiques ou bien pour l'utiliser côté serveur.

1.3.2. Node.js

Node.js est une plateforme logicielle libre et évènementielle basée sur le moteur V8 de Google

Chrome qui permet de développer des applications en utilisant du JavaScript. Il se distingue des autres plateformes grâce à une approche non bloquante permettant d'effectuer des entrées/sorties (I/O) de manière asynchrone [12].

Node.js n'est pas un Framework. Ce n'est pas un outil qui va nous aider à mettre en place une application web rapidement avec peu de code. C'est un outil plus bas niveau qui nous permettra de communiquer avec le système à travers différentes librairies.

Voici les raisons pour lesquelles nous avons choisi cette technologie :

Super adapté aux SPA (Single Page Application) : Node.js est un système single thread non bloquant. En effet, si un client fait une requête d'un fichier au serveur alors il lance cette requête sans attendre le résultat, et si un autre client vient faire une autre requête, Node.js est tout de suite capable de traiter cette requête également. Au final, ça rend les choses super rapides.

Et pour les SPA, la complexité du code réside souvent du côté client. Il n'y a pas généralement d'opérations complexes du côté serveur, les clients font pas mal de requêtes au serveur pour récupérer les données et le serveur se contente de procurer ces données pour alimenter l'application.

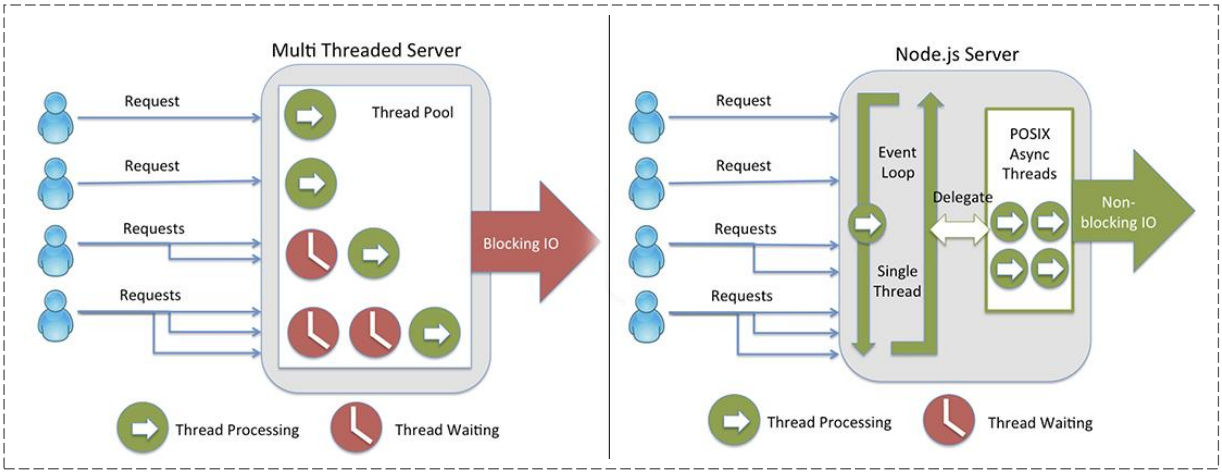


Figure. 25 : Différence entre un serveur Multithread et Node.js

Il pourra récupérer le résultat de la tâche quand elle sera finie. Ceci permet de gérer plus de concurrences en évitant les phases d'attentes.

L'écosystème Node.js : Il dispose de son gestionnaire de paquet officiel NPM qui permettra de télécharger et de partager des librairies. Il dispose d'une communauté très importante et d'un très grand nombre de paquets.

1.3.3. Express.js

C'est rarement qu'on utilise Node.js sans la présence de Express.js. D'après le site officiel du Framework, Express est une Infrastructure web minimaliste, souple et rapide pour Node.js. Il est écrit en JavaScript et hébergé dans l'environnement d'exécution Node.js.

Une application Express n'est ni plus ni moins qu'une succession d'appels de fonctions middleware. Les fonctions de middleware sont des fonctions qui peuvent accéder à l'objet Requête (req), l'objet Réponse (res) et à la fonction middleware suivante (next) dans le cycle demande-réponse de l'application [13].

Les fonctions middleware effectuent les tâches suivantes :

- Exécuter tout type de code.
- Apporter des modifications aux objets de demande et de réponse.
- Terminer le cycle de demande-réponse.
- Appeler la fonction middleware suivant dans la pile.

Si la fonction middleware en cours ne termine pas le cycle de demande-réponse, elle doit appeler la fonction **next()** pour transmettre le contrôle à la fonction middleware suivant, sinon, la demande restera bloquée.

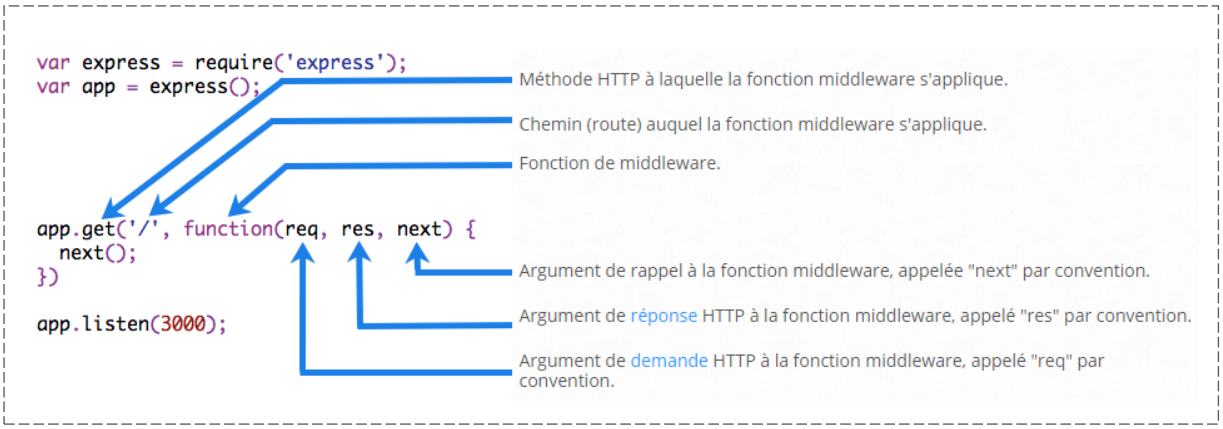


Figure. 26 : Les éléments d'un appel de fonction middleware

1.3.4. MongoDB

Dans un système de base de données relationnelle, les données sont stockées par ligne dans des tables. Et il est souvent nécessaire de faire des jointures sur plusieurs tables afin de tirer des informations assez pertinentes de la base [14].

Dans MongoDB, les données sont modélisées sous forme de document sous un style JSON, on ne parle plus de tables, ni d'enregistrements, mais de collections et de documents. Ce système de gestion de données nous évite ainsi de faire des jointures de tables, car toutes les informations propres à un certain donnée sont stockées dans un même document.

Les documents sont les unités de base dans une base MongoDB. Ils sont équivalents aux objets JSON et sont comparables aux enregistrements d'une table dans une base de données relationnelle.

Les collections sont des regroupements de documents MongoDB. Une collection est l'équivalent d'une table créée dans tout autre RDBMS. Elle existe dans une seule base de données et n'impose aucune structure.

En plus d'être une base de données NoSQL, MongoDB possède quelques qualités qui nous ont poussés à le choisir comme système de gestion de base de données pour notre application, certaines de ses principales caractéristiques sont listées ci-dessous :

- Accélère le développement d'applications et réduit la complexité des déploiements.
- Facile à installer et à configurer.
- Utilise un BSON (format JSON) pour stocker les données.
- Facile de mapper les objets de document sur le code d'application grâce à des Drivers et des ODM.
- C'est gratuit et open source.

1.3.5. Mongoose

Mongoose est une bibliothèque ODM (Object Data Modeling) pour MongoDB et Node.js. Il gère les relations entre les données et fournit la validation du schéma. Il est utilisé aussi pour la conversion entre les objets du code et la représentation de ces objets dans la base de données [14].

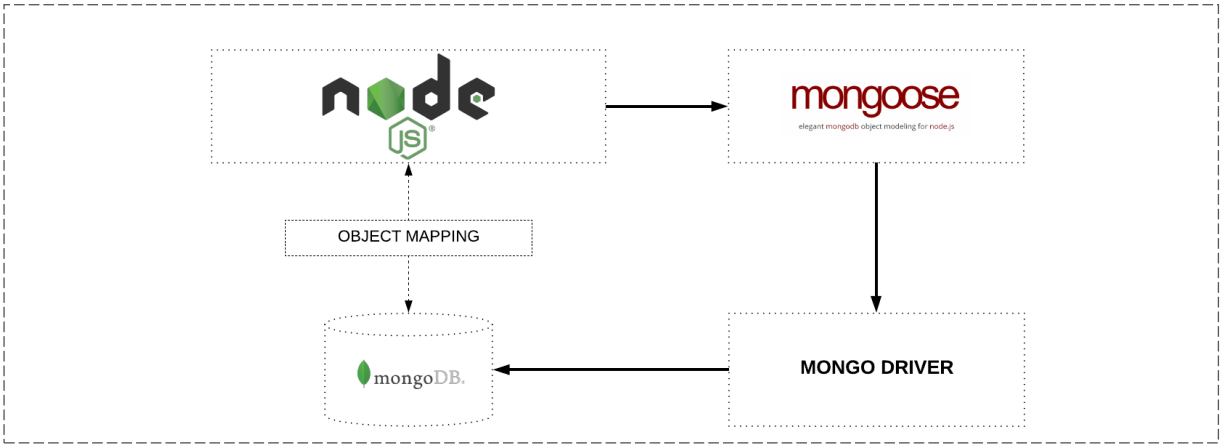


Figure. 27 : Utilisation de Mongoose avec Node.js et MongoDB

1.3.6. JWT

Afin de pouvoir utiliser l'application, il faut au préalable s'y être authentifié. L'étape d'authentification consiste à fournir le couple **username/password** à la plateforme, qui en retour fournit un jeton (JWT). Ce jeton peut être considéré comme une clé d'accès qui doit être fournie par l'utilisateur à chaque usage d'une fonctionnalité protégée de l'application. Un certain nombre de mécanismes existent pour sécuriser des applications web. Celui retenu pour notre projet est JSON web Token.

Le principe est simple, après s'être authentifié, le serveur génère un hash qui servira de signature pendant une certaine durée. Ce token va ensuite transiter dans chaque requête entre le client et le serveur. Cela rend le processus d'authentification est plus efficace et sécurisé sans utilisation de cookie ou de session serveur.

Le diagramme suivant [cf. Figure 28] présente l'accès à des ressources protégé par un client à l'aide de JWT [15] :

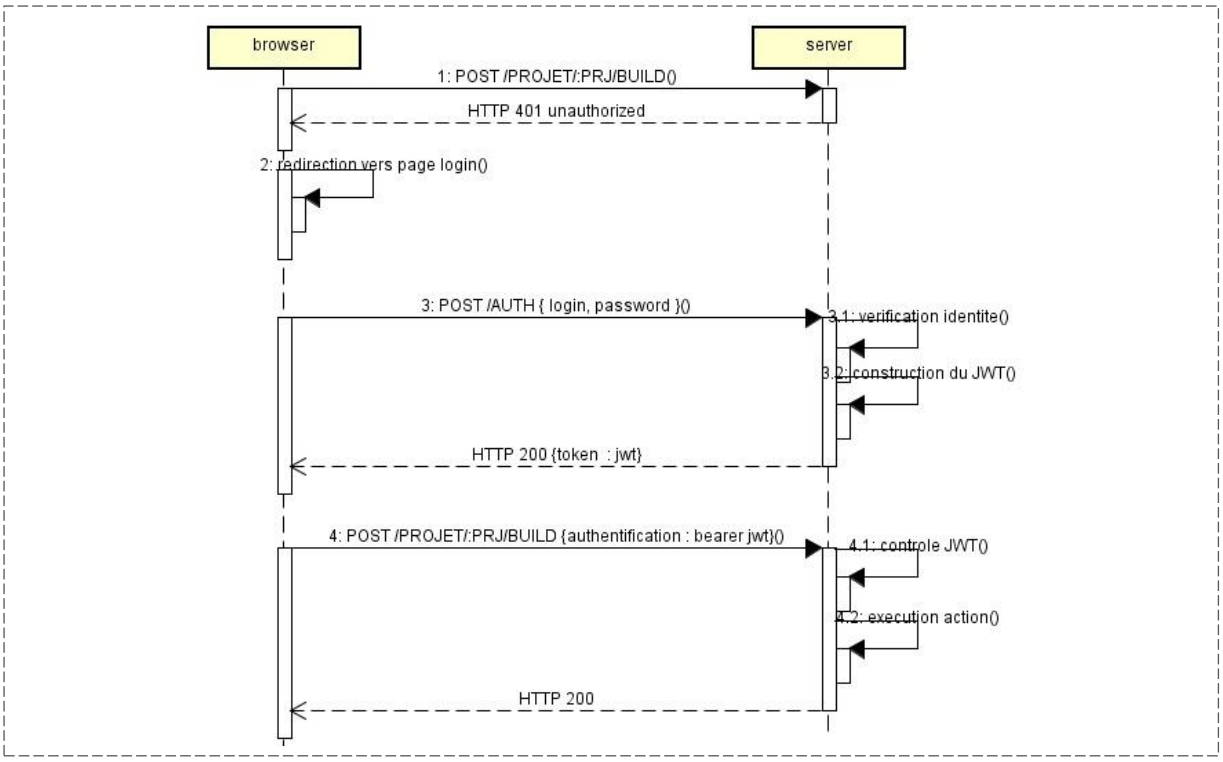


Figure. 28 : L'accès à des ressources protégées à l'aide de JWT

D'après la figure [cf. Figure 28], l'accès aux ressources à l'aide de JWT se déroulent comme suit :

1 : L'utilisateur accède à une ressource protégée, le serveur indique que l'accès à cette ressource n'est pas autorisé.

2 : L'utilisateur est redirigé vers la page de login.

3 : L'utilisateur fournit son login /mot de passe.

3.1 : le serveur vérifie le login et le mot de passe de l'utilisateur.

3.2 : Si c'est OK, il construit un jeton JWT et le transmet en réponse de la requête à l'utilisateur pour être stocké dans le navigateur.

4 : L'utilisateur accède à une ressource protégée et fournit le JWT.

4.1 : Le serveur contrôle le JWT reçu.

4.2 : Le serveur exécute l'action souhaitée par l'utilisateur.

1.4. Partie Mobile

1.4.1. Ionic

Ionic est un mélange d'outils et de technologies pour développer des applications mobiles hybrides rapidement et facilement à l'aide de HTML, CSS et JavaScript. Il s'appuie sur Angular pour la partie application web du Framework et sur Cordova pour la partie construction des applications natives [16].

Ce Framework open source permet de développer une application déployable sur plusieurs environnements et systèmes tels que Android ou iOS ou Windows Phone.

Le développement d'une application Ionic a de nombreux avantages. C'est une application hybride, à la différence des applications natives qui doivent être développées pour chaque système d'exploitation, elle ne requiert qu'un seul développement pour être accessible sur tous les systèmes.

Les applications natives sont beaucoup plus longues à mettre en place, mais elles sont d'une excellente qualité, leurs performances permettent une exploitation optimale des fonctionnalités des smartphones. Leurs designs remarquables et la fluidité des animations en font une expérience utilisateur plus agréable.

Le Framework Ionic permet de réunir l'ensemble de ces avantages en diminuant le coût de développement de manière significative tout en bénéficiant d'une qualité et d'une performance similaires à celles des applications natives.

1.5. Partie Desktop

1.5.1. Electron.js

Le Framework Electron.js nous permet de développer des applications de bureau multiplateformes en utilisant JavaScript, HTML et CSS. Il est basé sur Node.js pour l'accès au système de fichiers et Chromium pour le rendu des interfaces graphiques. Electron.js est développé par Github et utilisé par l'éditeur Atom, mais également de nombreuses autres applications, parmi ces dernières, on peut citer :

- Visual Studio Code, l'éditeur de code open source développé par Microsoft.
- Slack, l'application de messagerie pour les équipes.
- L'application bureau de WordPress.
- L'application de bureau de GitHub [17].

1.6. Outils et plateformes

1.6.1. Angular CLI

Angular CLI (Command Line Interface) est un outil permettant de créer, construire, générer et tester nos applications et librairies Angular.

Sans Angular CLI, la création et la construction d'une application Angular nécessite l'utilisation et la maitrise de nombreux outils : typescript, webpack, karma, protractor, etc.

1.6.2. Ionic CLI

Le CLI de Ionic propose plusieurs outils pour simplifier le développement mobile. La commande « ionic serve » lance l'application dans un navigateur. L'une des options permet d'afficher en parallèle le visuel de l'application des différentes plateformes dans le navigateur. Une autre fonctionnalité très intéressante est le rechargement automatique de l'application dès qu'un fichier est sauvegardé. Ainsi, il n'est plus nécessaire de recompiler et de revenir sur une page précise pour chaque modification du programme. La commande « ionic » permet de compiler et de lancer l'application sur un appareil mobile ou dans un émulateur. L'application sera lancée de préférence sur un appareil mobile.

1.6.3. Webpack

Webpack est un outil qui est aujourd'hui incontournable dès lors que l'on travaille sur des projets JavaScript complexes, et surtout avec l'utilisation des Frameworks JS moderne comme Angular. Cet outil va nous permettre de morceler notre code sous forme de modules qui seront ensuite fusionnés en un seul fichier par Webpack. Il dispose, en plus, d'un système de "loaders" qui vont permettre d'inclure de nouveaux types de fichiers ou d'appliquer des transformations spécifiques, comme une transformation ES2015 en ES6 [18].

1.6.4. Babel

Babel est un transpileur ECMAScript 6. Il transforme le code ES6 en code ES5. Ce qui signifie que nous pouvons commencer à utiliser les nouvelles fonctionnalités du langage JavaScript dans notre application sans devoir attendre leur support par les navigateurs.

Babel sait compiler et traduire les éléments suivants :

- Les raccourcis de fonction.
- Les fonctions asynchrones.
- Les classes.
- La déstructuration.
- Les décorateurs.
- Déclaration des variables.

1.6.5. NPM

NPM (Node Package Manager) comme son nom l'indique est le "package manageur" officiel de l'univers JavaScript. Il est installé automatiquement lors de l'installation de Node.js [19].

NPM permet de :

- Créer un module JavaScript et le publier.
- Installer les dépendances d'un module à partir d'un fichier de description de dépendances.
- Mettre à disposition des développeurs un point d'entrée pour "build", "debug", "deploy" ou "test" votre application.

1.6.6. Git

Git est un outil de versioning et de développement collaboratif. Il est devenu le standard du domaine grâce à son fonctionnement décentralisé, la facilité de gestion des branches et l'automatisation grâce à une multitude d'outils comme github, gitlab, git flow, etc.

Par défaut, les projets Angular sont initialisés sur un "repository" Git, mais on peut utiliser un autre système du contrôle de version [20].

1.6.7. JSON Server

JSON Server est un package NPM pour lequel nous pouvons créer une API fictive, et tout ce dont nous avons besoin est un fichier JSON qui sera utilisé comme notre backend REST.

Les technologies Front-End ont connu une croissance exponentielle depuis la création de Node.js et de puissants Frameworks comme Angular. Il existe des possibilités illimitées pour rendre notre configuration de développement plus efficace et plus robuste avec les outils appropriés. Dans notre processus de développement, nous avons décidé d'utiliser JSON Server pour préparer des données fictives afin de tester notre Application Angular avant de commencer le développement de l'API, cela rend le processus de développement rapide et efficace.

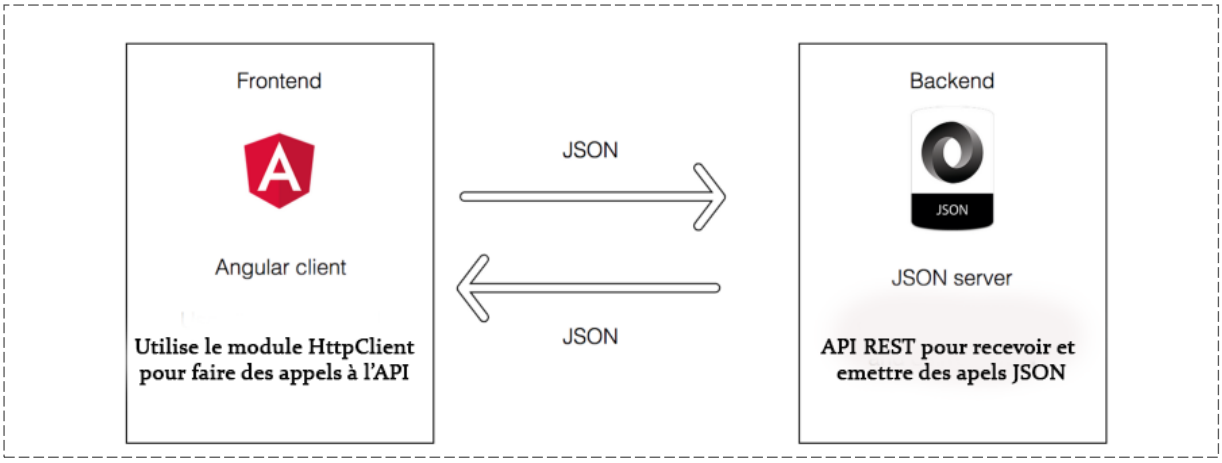


Figure. 29 : Communication entre Angular et JSON Server

1.6.8. Postman

Postman est un outil puissant pour tester et exécuter des API. Il est devenu très populaire pour tester les Microservices, notamment grâce à sa simplicité et ses fonctionnalités très spécialisées. Une fois Postman installé, nous disposons d'un environnement graphique complet pour gérer l'ensemble des interactions avec notre API.

Après avoir défini un environnement « **Smile Tracker** » et une variable d'environnement « tracker.smile.fr » de valeur « **localhost:3000** », on peut envoyer notre requête GET paramétrée ici pour nous retourner tous les utilisateurs enregistrés dans la base de données.

La figure [cf. Figure 30] suivante présente le résultat de cette requête :

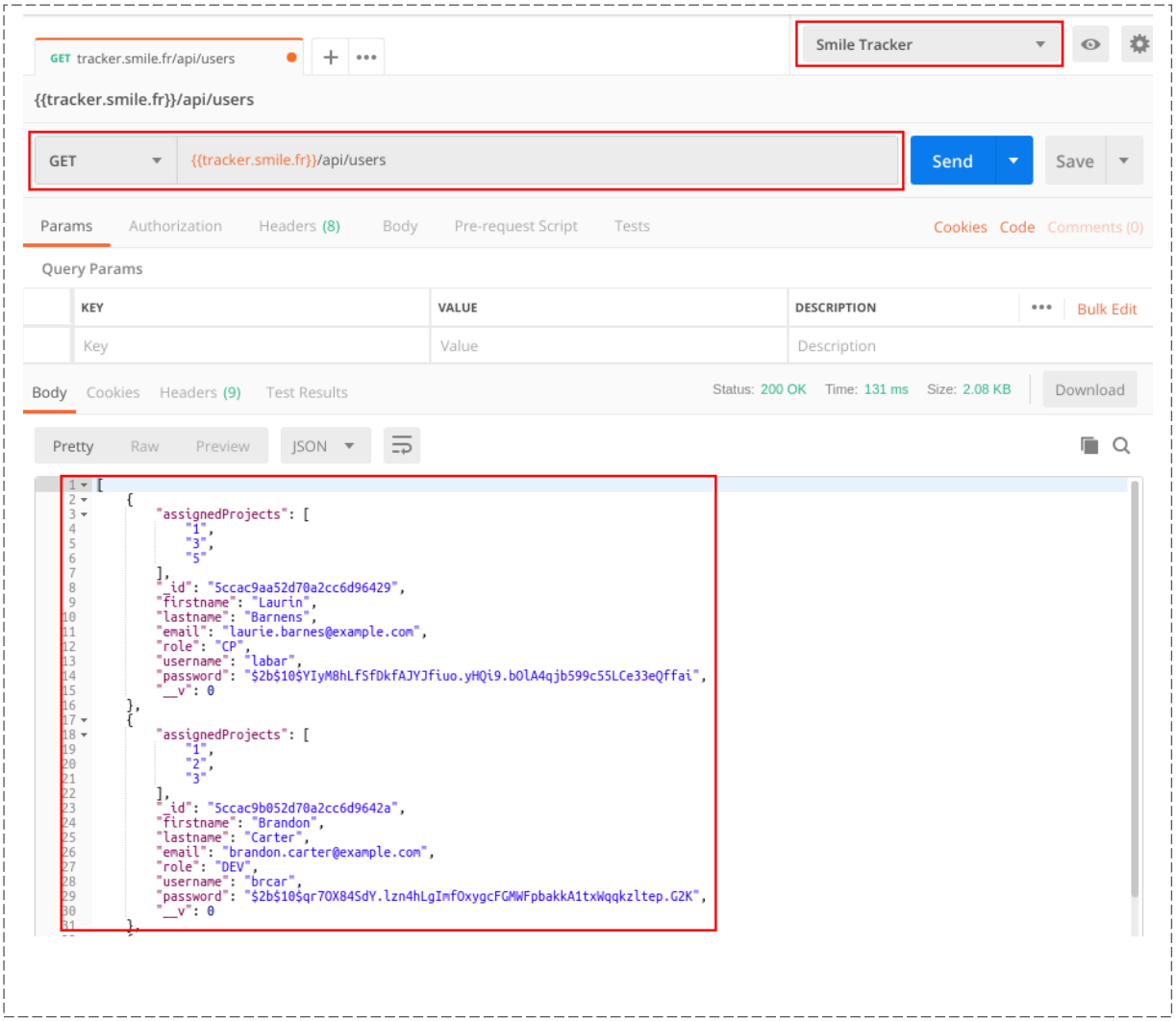


Figure. 30 : L'envoi d'une requête GET depuis Postman.

1.6.9. GitLab

GitLab est un gestionnaire web de référentiel git incluant un wiki et des fonctionnalités de suivi de problèmes. GitLab fournit une gestion centralisée des référentiels Git, permettant aux utilisateurs d'avoir le contrôle complet de leurs référentiels ou projets [21].

Écrit en Ruby, le gestionnaire inclut les contrôles d'accès granulaires, les revues de code, le suivi de problèmes, les flux d'activités, les wikis, et intégrations continuent. En décembre 2016, il regroupe 1400 contributeurs Open Source et est utilisé par les grandes entreprises comme Sony, IBM, CERN, NASA, etc.

1.6.10. VSCode

Visual Studio Code est un éditeur de code open source, gratuit et multiplateforme (Windows, Mac et Linux), développé par Microsoft, à ne pas confondre avec Visual Studio, l'IDE propriétaire de Microsoft. VSC est développé avec Electron.js et exploite des fonctionnalités d'édition avancées du projet Monaco Editor. Principalement conçu pour le développement d'application avec JavaScript, TypeScript et Node.js, l'éditeur peut s'adapter à d'autres types de langages grâce à un système d'extension bien fourni [22].

1.6.11. Slack

L'acronyme de « Searchable Log of All Conversation and Knowledge » est une plateforme collaborative qui nous permet de mieux travailler en équipe. Slack nous aidera tout au long de nos projets, de la réunion de lancement à la conclusion du projet en passant par les discussions autour du budget [23].

2. Architecture de l'application

Le schéma de la figure [cf. Figure 31] illustre l'architecture fonctionnelle que nous avons conçue en utilisant les technologies citées précédemment, ainsi que l'interaction des parties frontend (mobile, web) et Desktop avec le backend via une API REST.

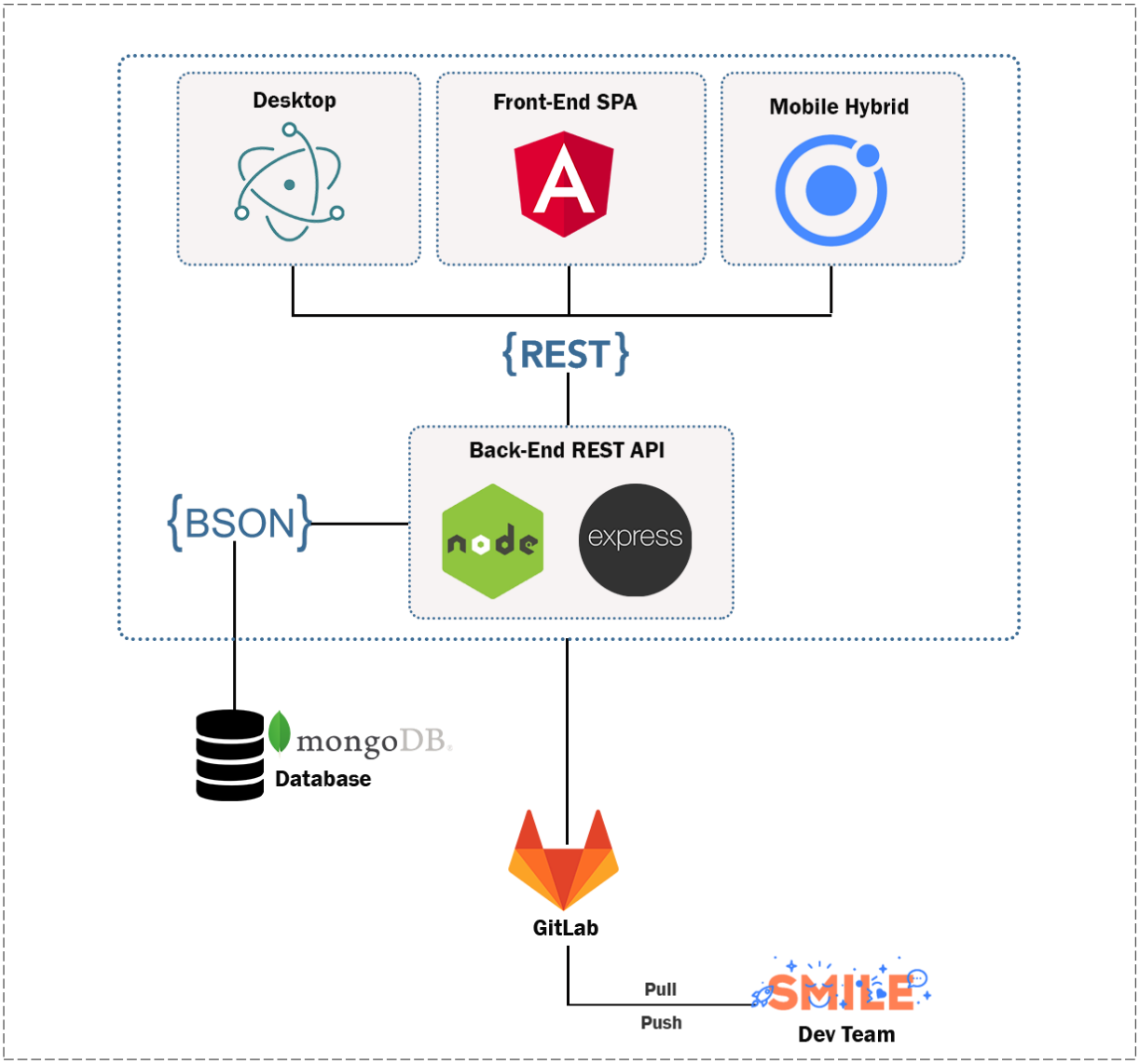


Figure. 31 : Architecture technique de l'application

Conclusion

Dans ce chapitre, nous avons effectué dans un premier temps une étude technique du projet en présentant l'architecture de la pile MEAN, après nous avons présenté les grandes parties de l'application avec les technologies et Frameworks utilisés dans chacune de ces parties. Et nous avons terminé par la présentation de l'architecture technique globale de la solution.

Chapitre 4. Réalisation

Introduction

Dans ce chapitre nous décrivons la phase de réalisation de l'application. Nous exposons d'abord l'environnent matériel et logiciel de notre projet. Puis nous présentons à travers des captures d'écrans notre application réalisée.

1. Environnement du projet

1.1. Environnement matériel

Pour la réalisation de notre projet dans des conditions favorables, il faut que nous disposions des ressources plus performantes. Durant le déroulement du projet, nous avons utilisé deux machines, un PC Dell qui tourne sur Ubuntu 18.04, et un MacBook Pro utilisé pendant le développement ainsi que le build de l'application mobile sous Android et IOS en même temps. Nous avons utilisé aussi des Smartphones Android et IOS pour le test de l'application mobile sur différentes plateformes.

La configuration de l'environnement matériel mis en place pour le développement du projet est présentée dans le tableau suivant :

Matériel	Configuration
Dell OptiPlex 7060	Intel Core i7 8700 3.2 GHz - 8 GB - 1 TB
MacBook Pro	Intel Core i7 8700 2.6 GHz - 16 GB - 512 GB

Table. 6 : Environnement matériel du projet

1.2. Environnement logiciel

L'environnement logiciel utilisé pour réaliser notre projet est structuré dans le tableau ci-dessous :

Outils	Version	Description
TypeScript	**3.2.2**	Langage compilé et typé qui génère du JavaScript pur.
Angular	**7.2.4**	Plateforme et Framwork qui permet de créer des applications coté client en HTML et en TypeScript.

Node.js	**11.9.0**	Environnement de bas niveau permettant l'exécution de JavaScript côté serveur.
Nodemon	**1.18.11**	Service de redémarrage automatique d'un serveur Node
Express.js	**4.16.4**	Infrastructure web minimaliste, souple et rapide pour Node.js.
MongoDB	**3.4.20**	Système de gestion de base de données NoSQL orientée documents.
Mongoose	**5.5.3**	Bibliothèque ODM (Object Data Modeling) pour MongoDB et Node.js.
Angular CLI	**7.3.1**	Interface de ligne de commande de Angular.
Angular Material	**7.3.2**	Module Angular permet de créer des interfaces modernes conformes au système de Material Design.
Ionc	**3.9.5**	Framework qui permet de créer des applications mobiles multiplateformes en utilisant des technologies web.
Electron.js	**4.2.2**	Framework qui permet de développer des applications multiplateformes de bureau avec des technologies web.
Babel	**7.4.3**	Compilateur JavaScript libre et open source.
npm	**6.5.0**	Gestionnaire de paquets officiel de Node.js.
Git	**2.21.0**	Logiciel de gestion de versions décentralisé.
Postman	**7.0.6**	Outil pratique pour tester une API REST.
Entreprise Architect	**14.1**	Logiciel de modélisation et de conception UML.
VSCode	**1.32**	Éditeur de code extensible développé par Microsoft.

Table. 7 : Environnement logiciel du projet

2. Présentation des interfaces graphiques

Dans cette partie, nous détaillerons la solution finale obtenue. Ainsi, nous présentons notre application à travers des captures écrans correspondantes au travail réalisé.

2.1. Interface d'authentification

La figure ci-dessous [cf. Figure 32] illustre l'interface d'accès à l'application. Elle implémente la fonction qui consiste à vérifier l'identité de l'utilisateur par un nom d'utilisateur et un mot de passe, afin de lui autoriser l'accès et lui offrir les services dédiés à son profil.

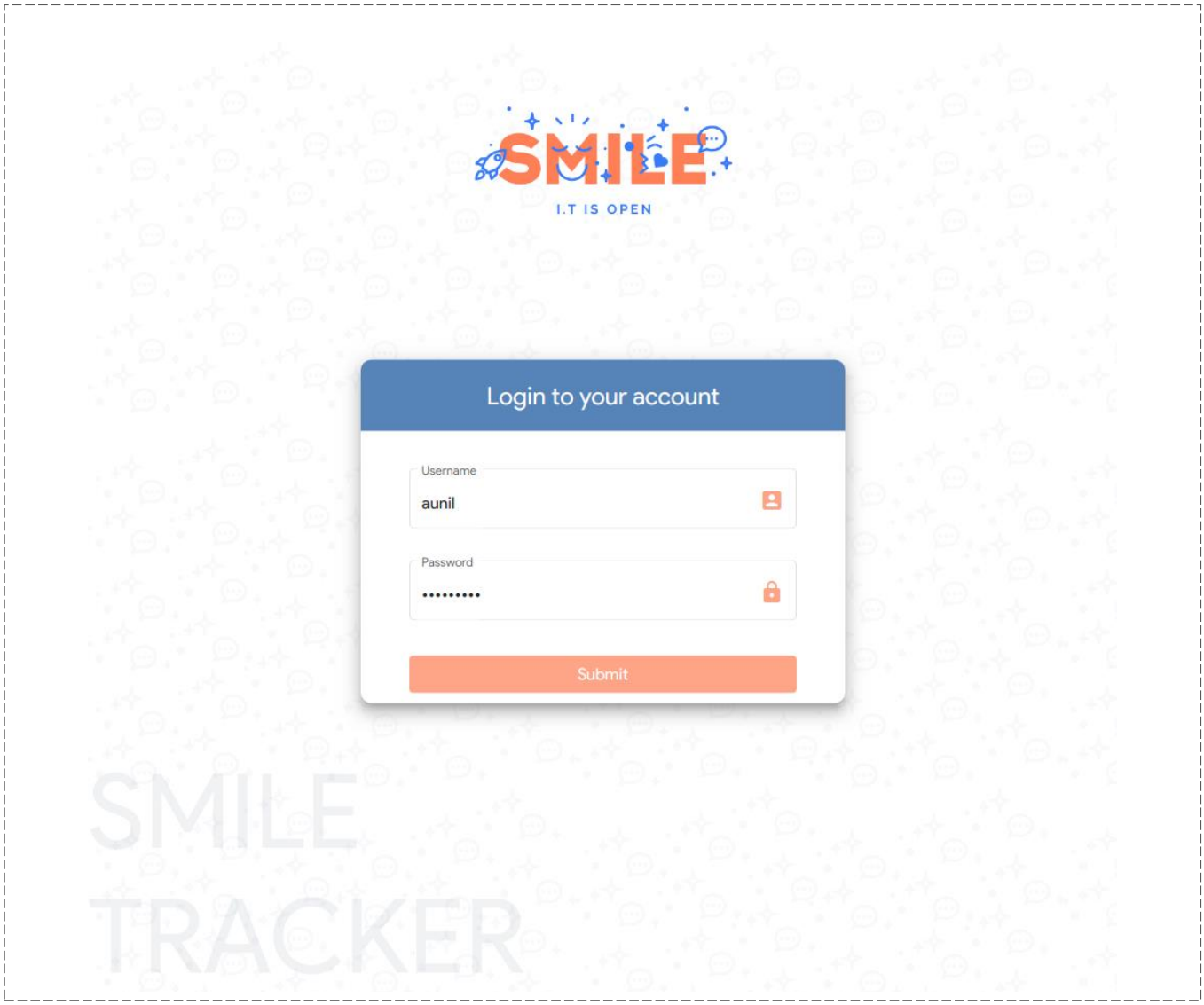


Figure. 32 : Interface d'authentification

2.2. Partie Back Office

Si l'utilisateur connecté est un Chef de projet ou un Superviseur, il sera redirigé directement au Backoffice de l'application.

2.2.1. Consultation de la liste des utilisateurs

La figure [cf. Figure 33] présente l'onglet de la liste des utilisateurs, où le chef de projet est en mesure de consulter les détails de tous les utilisateurs, comme il a la main aussi de les gérer soit par l'ajout, en cliquant sur le bouton « **Add user** », soit par la modification ou par la suppression, en cliquant sur les buttons à droite du tableau. Il peut aussi chercher un utilisateur dans la liste depuis la barre de recherche située en haut du tableau. Cet onglet offre aussi la possibilité de consulter les détails d'un utilisateur ainsi que les projets qui lui sont assignés, en cliquant sur le nom de l'utilisateur.

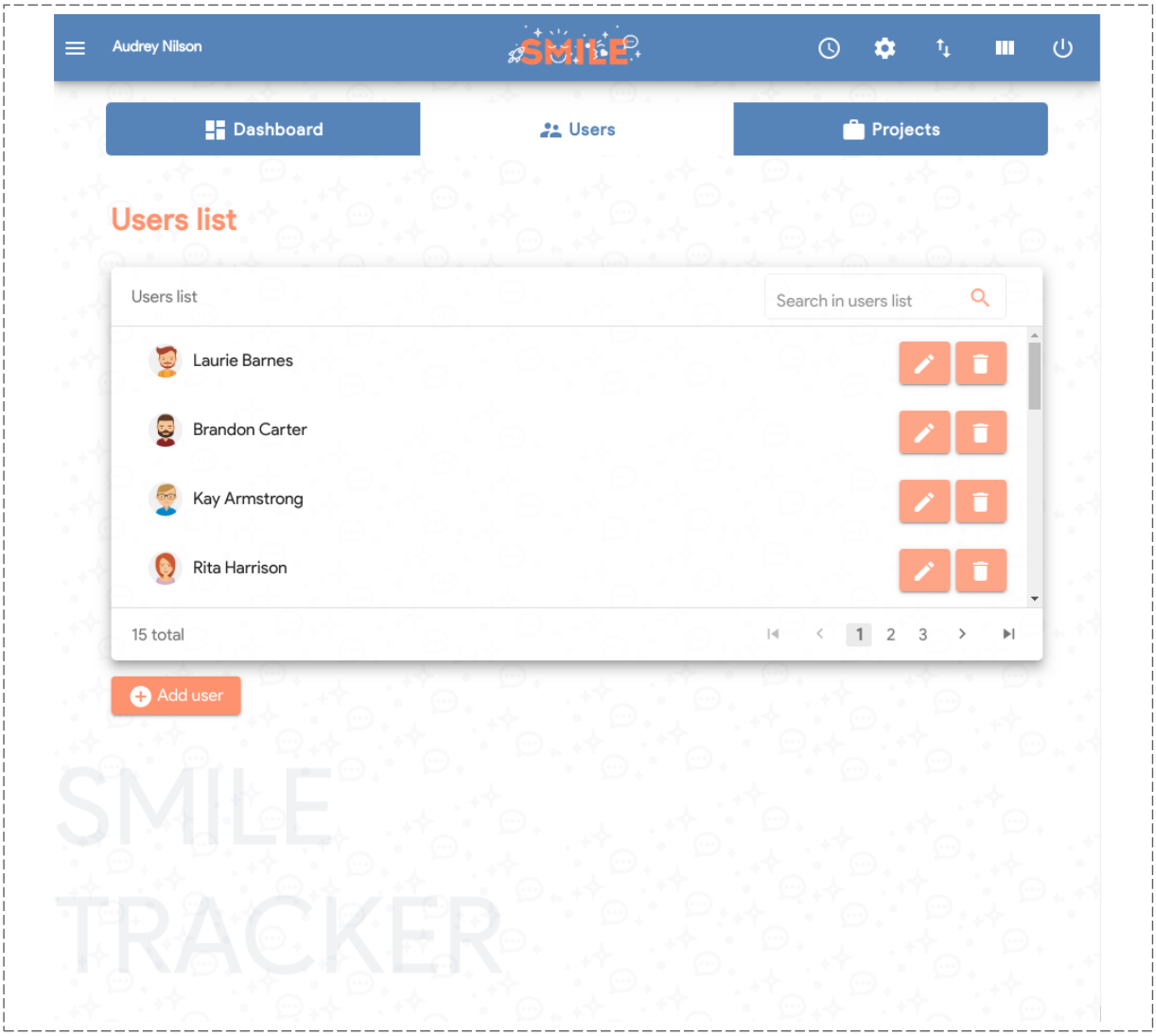


Figure. 33 : Interface de consultation de la liste des utilisateurs

2.2.2. Consultation d'un utilisateur

En cliquant sur le nom d'un utilisateur, un modal apparait en affichant toutes les informations qui concernent cet utilisateur ainsi que les projets qui lui sont assignés. À partir de cette interface, le chef de projet peut modifier les données relatives à cet utilisateur en cliquant sur l'icône de modification en haut du modal.

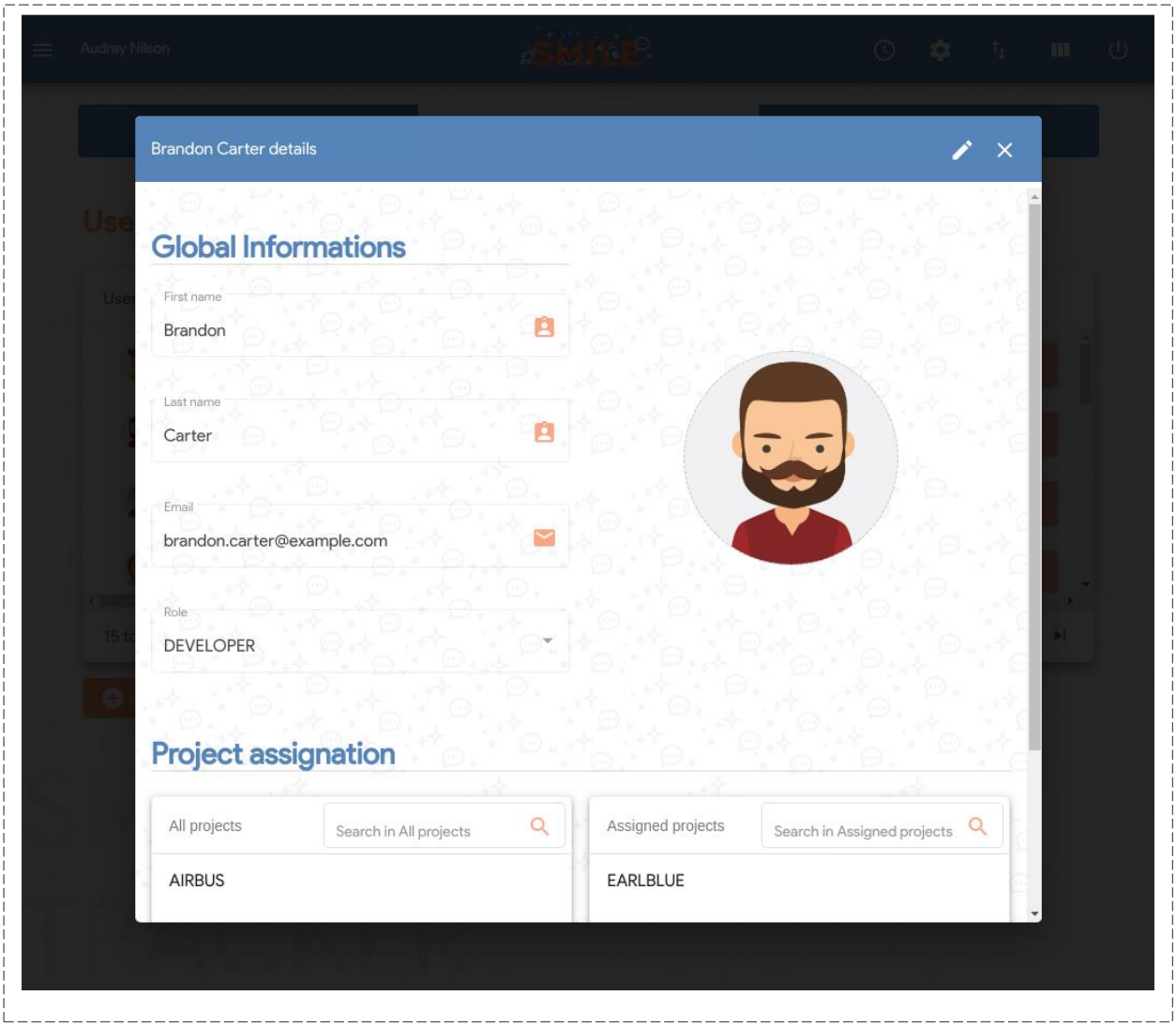


Figure. 34 : Interface de consultation d'un utilisateur

2.2.3. L'ajout d'un nouveau compte utilisateur

La figure [cf. Figure 35] montre le modal d'ajout d'un nouvel utilisateur, où le chef de projet doit saisir les différentes informations concernant un utilisateur. Le bouton « Save » sera activé une fois les champs obligatoires sont remplis.

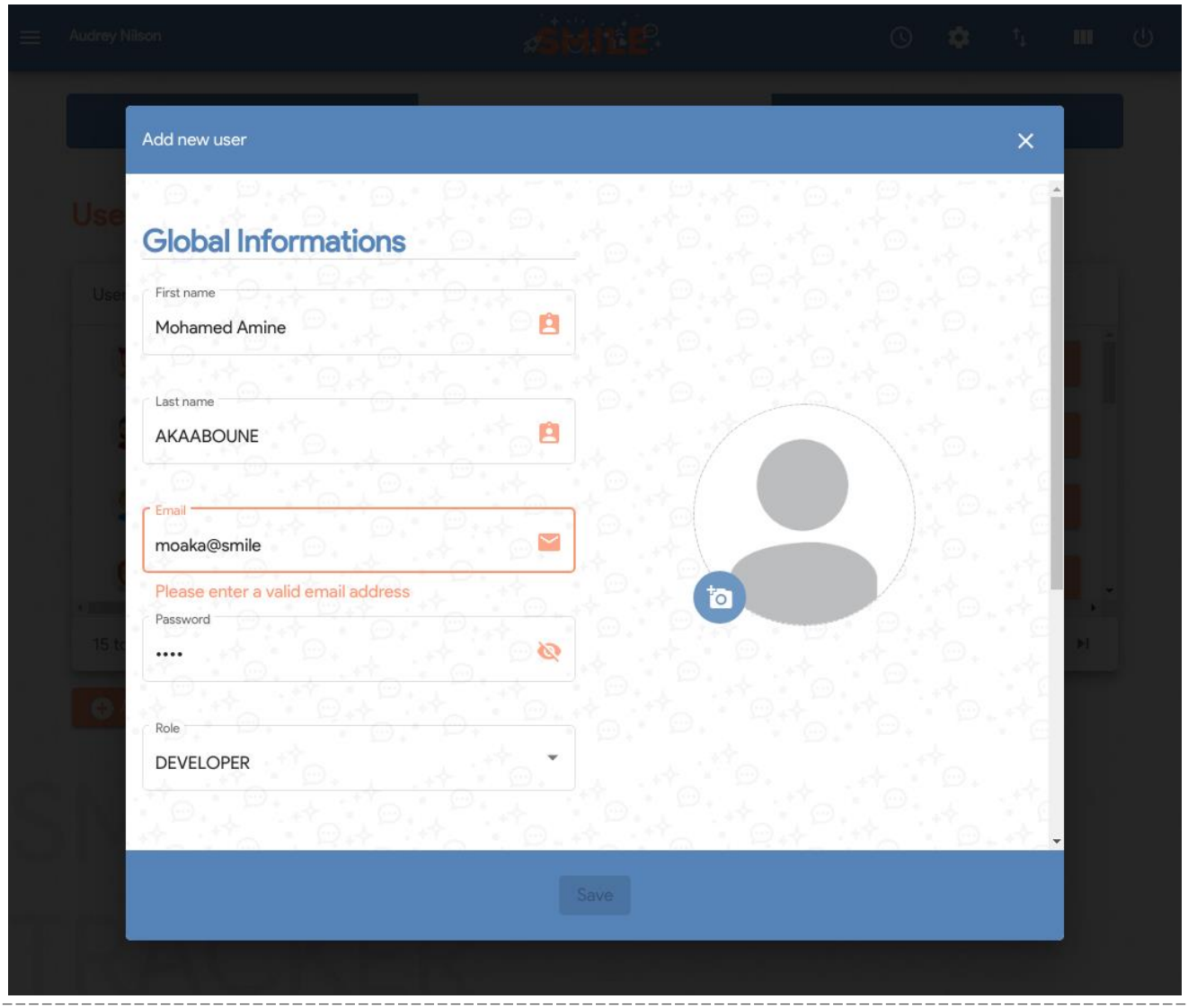


Figure. 35 : Interface d'ajout d'un nouveau compte utilisateur

L'assignation d'un projet à un utilisateur peut être faite soit au moment de la création ou bien dans la modification

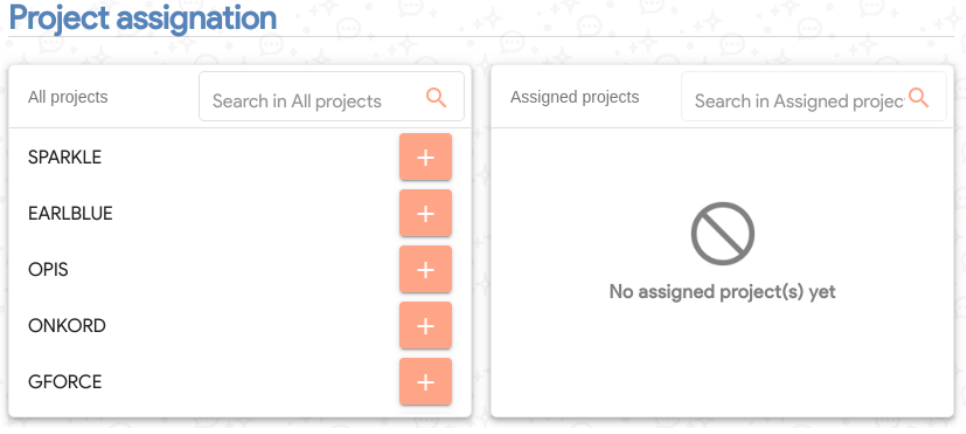


Figure. 36 : Assignation d'un projet à un utilisateur

2.2.4. Consultation de la liste des projets

La figure ci-dessous [cf. Figure 37] présente la liste des projets. Le chef de projet est en mesure de consulter les détails de tous les projets, comme il a la main aussi de les gérer soit par l'ajout, en cliquant sur le bouton « **Add project** », soit par la modification ou par la suppression, en cliquant sur les buttons à droite du tableau. Il peut aussi chercher un projet dans la liste depuis la barre de recherche située en haut du tableau. Cet onglet offre aussi la possibilité de consulter les détails d'un projet, les utilisateurs qui lui sont assignés, ainsi que les tickets qui lui sont associés en cliquant sur le nom du projet.

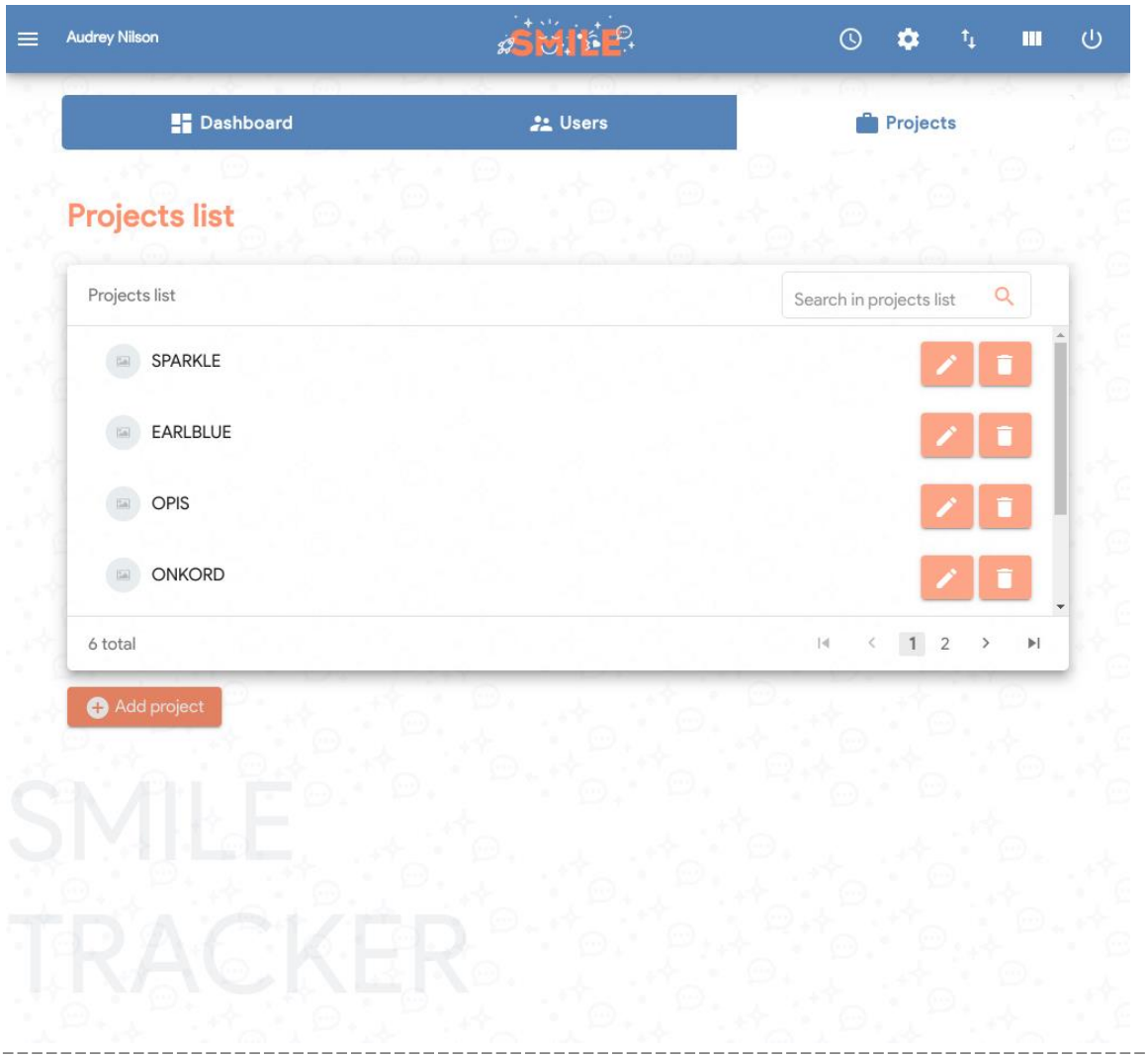


Figure. 37 : Interface de consultation de la liste des projets

2.2.5. Modification d'un projet

La figure ci-dessous expose le modal de modification d'un projet, il comprend trois parties, la première contient les informations générales sur ce projet (nom, date de début et de fin). La deuxième contient la liste de tous les utilisateurs ainsi que ceux qui sont assignés à ce projet.

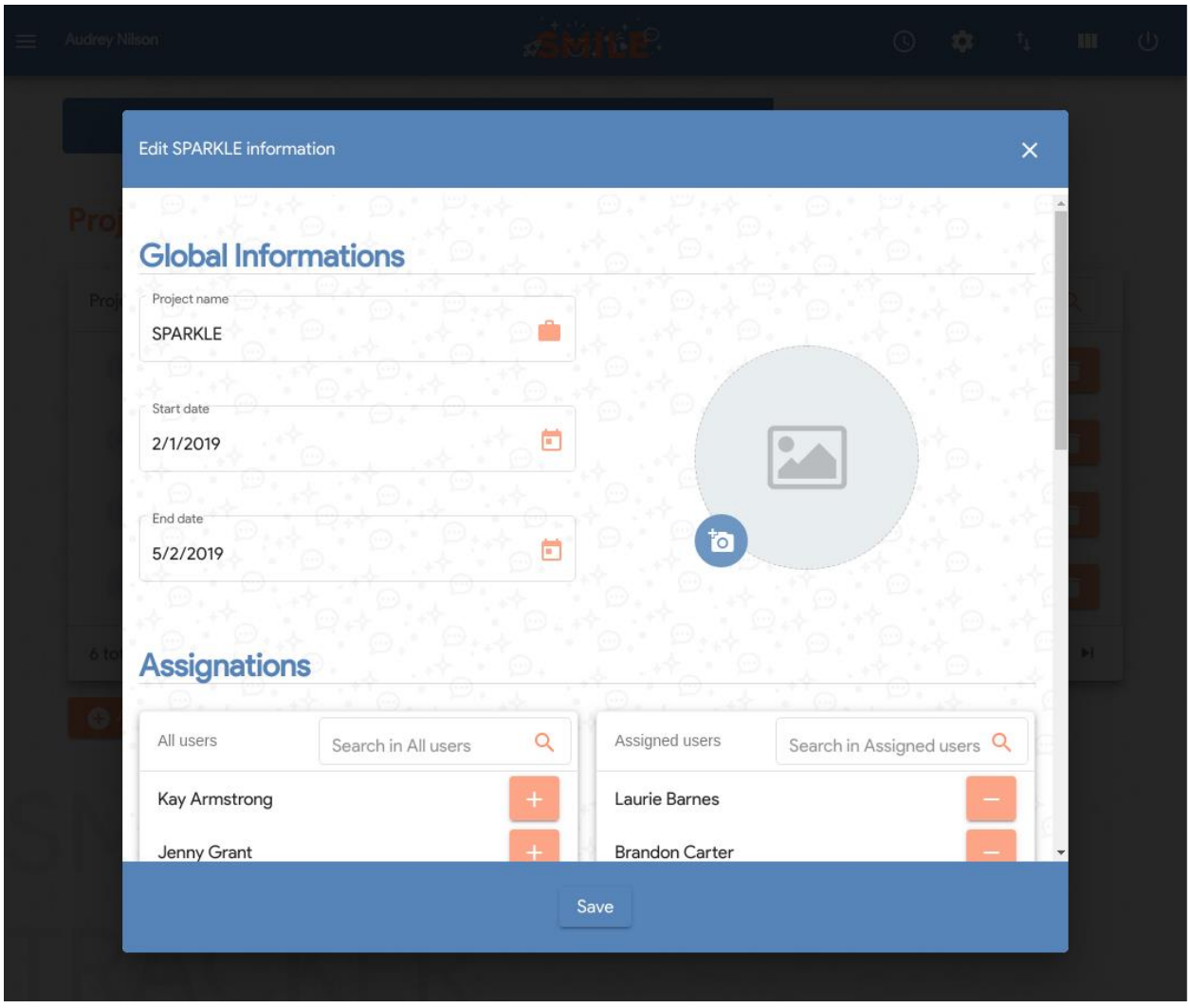


Figure. 38 : Interface de modification d'un projet

La troisième comprend les tickets associés à ce projet. À partir de cette interface [cf. Figure 38], le chef de projet peut consulter un ticket ou bien de créer un nouveau en cliquant sur le bouton « **Add ticket** ».

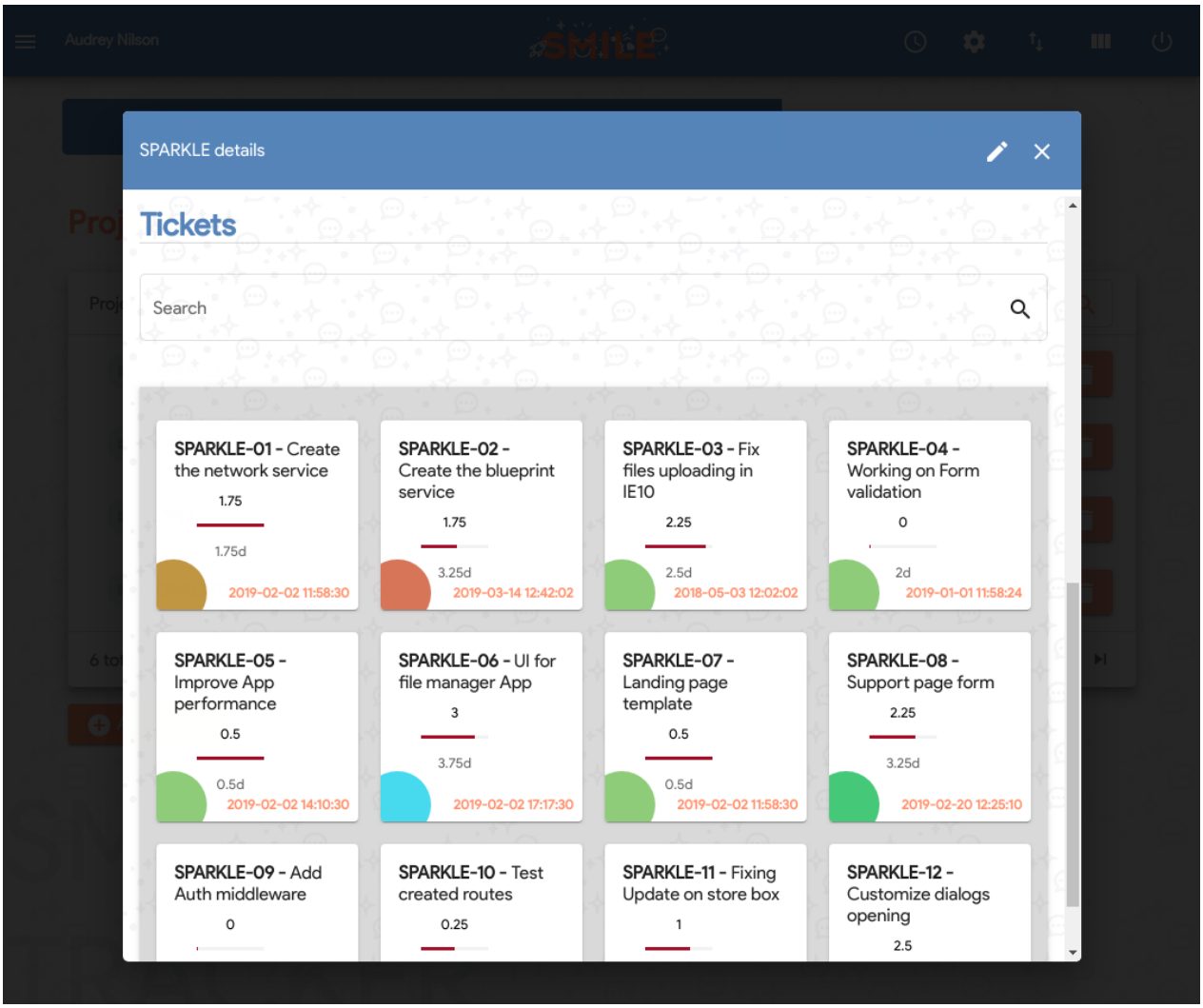


Figure. 39 : Liste des tickets associés au projet

2.2.6. L'ajout d'un nouveau ticket

La figure [cf. Figure 40] présente le modal d'ajout d'un ticket, il comprend quatre parties. La première contient les informations générales sur le ticket, le statut du ticket et en « **To Do** » par défaut, la référence est générée automatiquement et ne peut pas être modifiée. La deuxième concerne le timesheeting. À sa création, le temps restant prend automatiquement la valeur du temps estimé qui a comme valeur initiale 0,125. Dans la troisième partie, le chef de projet peut ajouter des attachements contenant des informations nécessaires pour le traitement du ticket. Il peut aussi ajouter des commentaires.

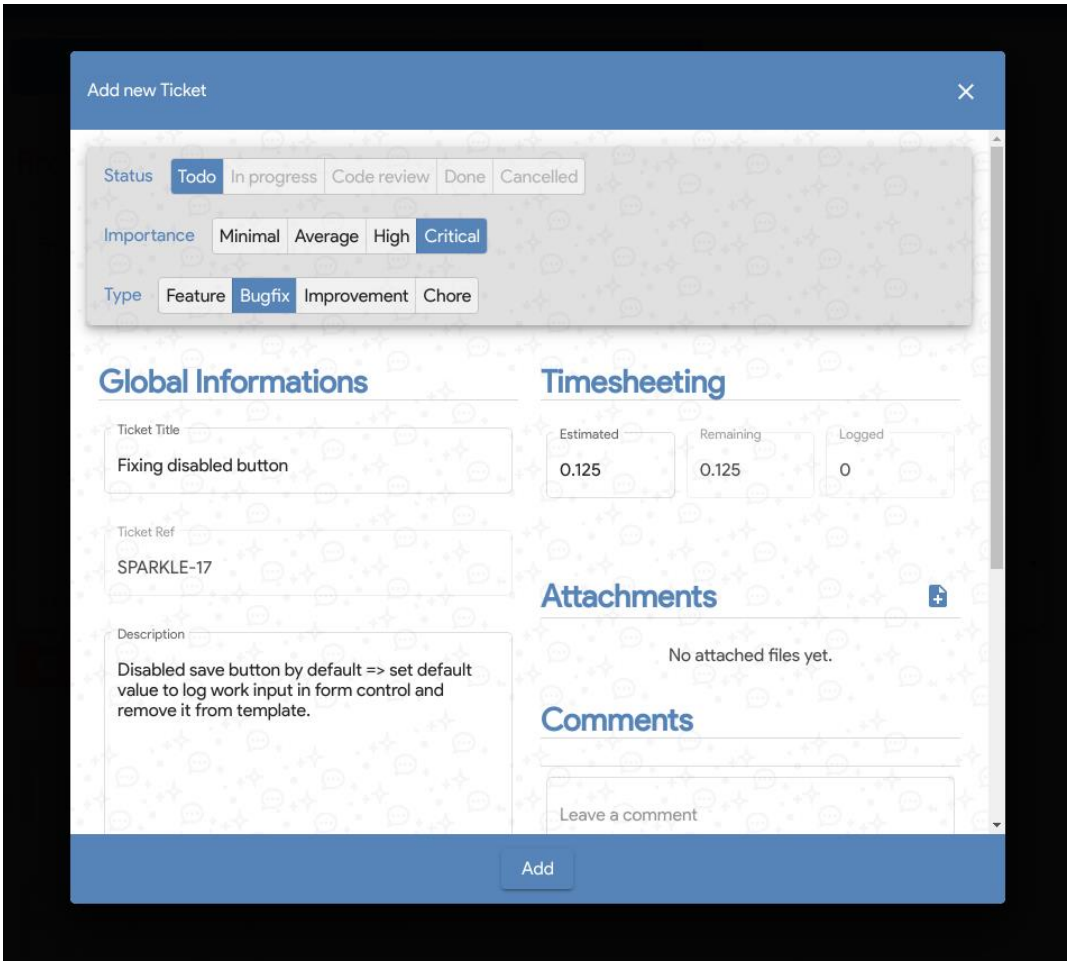


Figure. 40 : Interface d'ajout d'un nouveau ticket

2.3. Partie Front Office

Si l'utilisateur connecté est un Team Lead, Développeur ou Testeur, il sera redirigé directement au Frontoffice de l'application. Cette interface [cf. Figure 40] est composée des éléments suivants :

La barre de navigation : Cette barre et la même pour tous les utilisateurs sauf pour la navigation entre le Frontoffice et le backoffice. Si l'utilisateur connecté a l'accès au backoffice, un bouton sera affiché pour accéder à cette page. L'utilisateur peut aussi exporter l'historique des mises à jour des tâches d'un projet.

Liste des projets : À partir de cette liste, l'utilisateur peut sélectionner un projet pour qu'il puisse consulter les tickets qui lui sont associés ainsi que les utilisateurs qui lui sont affectés.

Barre de recherche : Où l'utilisateur peut chercher un ticket par référence ou bien par titre.

Espace Glisser-Déposer : Il est doté de colonnes **(TODO, IN PROGRESS, DONE, CODE REVIEW/TEST, CANCELED)** où chacune contient des tickets appropriés à un certain projet selon leurs états d'avancement.

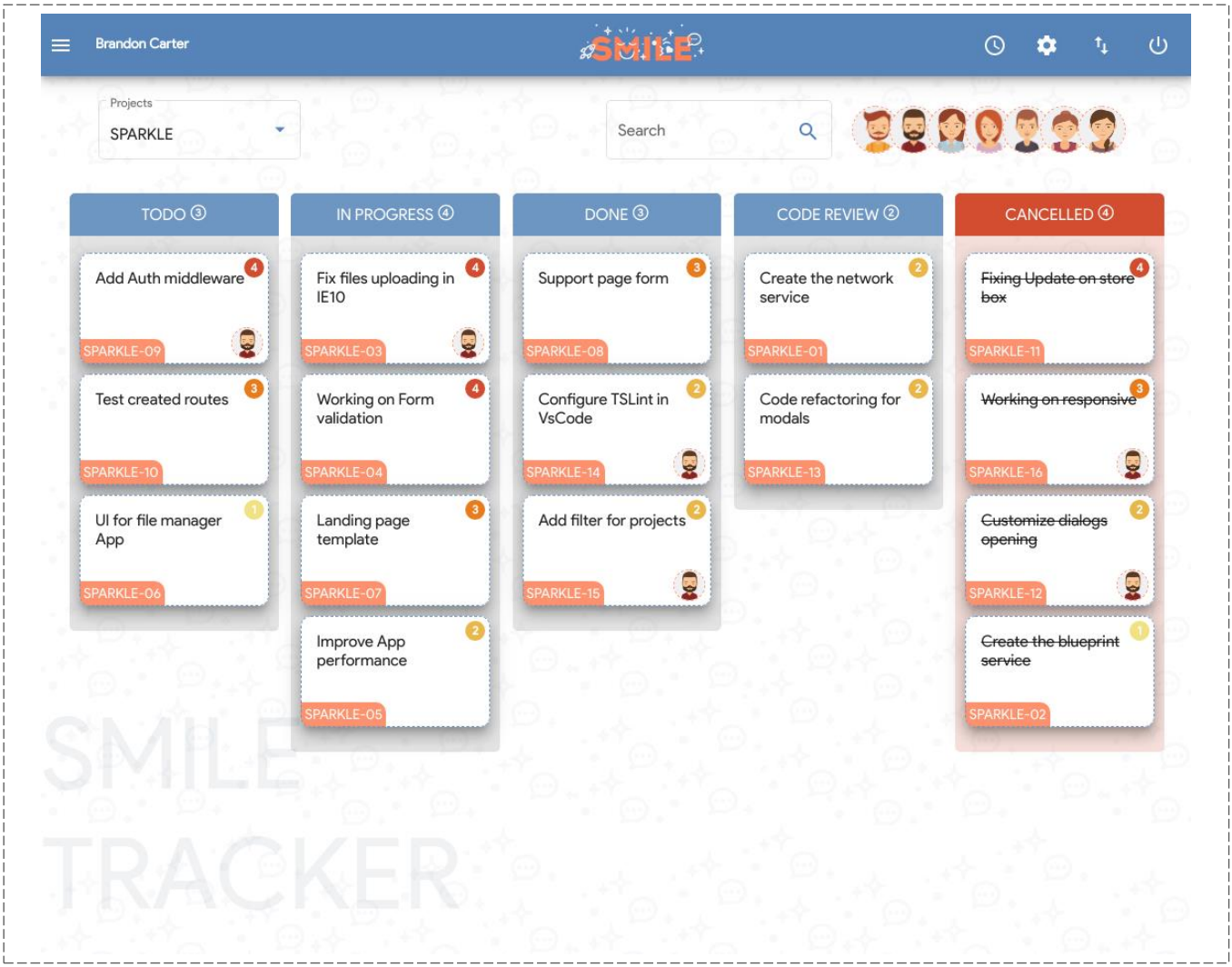


Figure. 41 : Interface du Frontoffice

2.3.1 Consultation d'un ticket

La consultation d'un ticket est différente depuis le FrontOffice [cf. Figure 41]. L'utilisateur assigné à ce ticket peut loger le temps passé sur cette tâche, ajouter un commentaire et voir l'historique de la mise à jour de ce ticket depuis sa création.

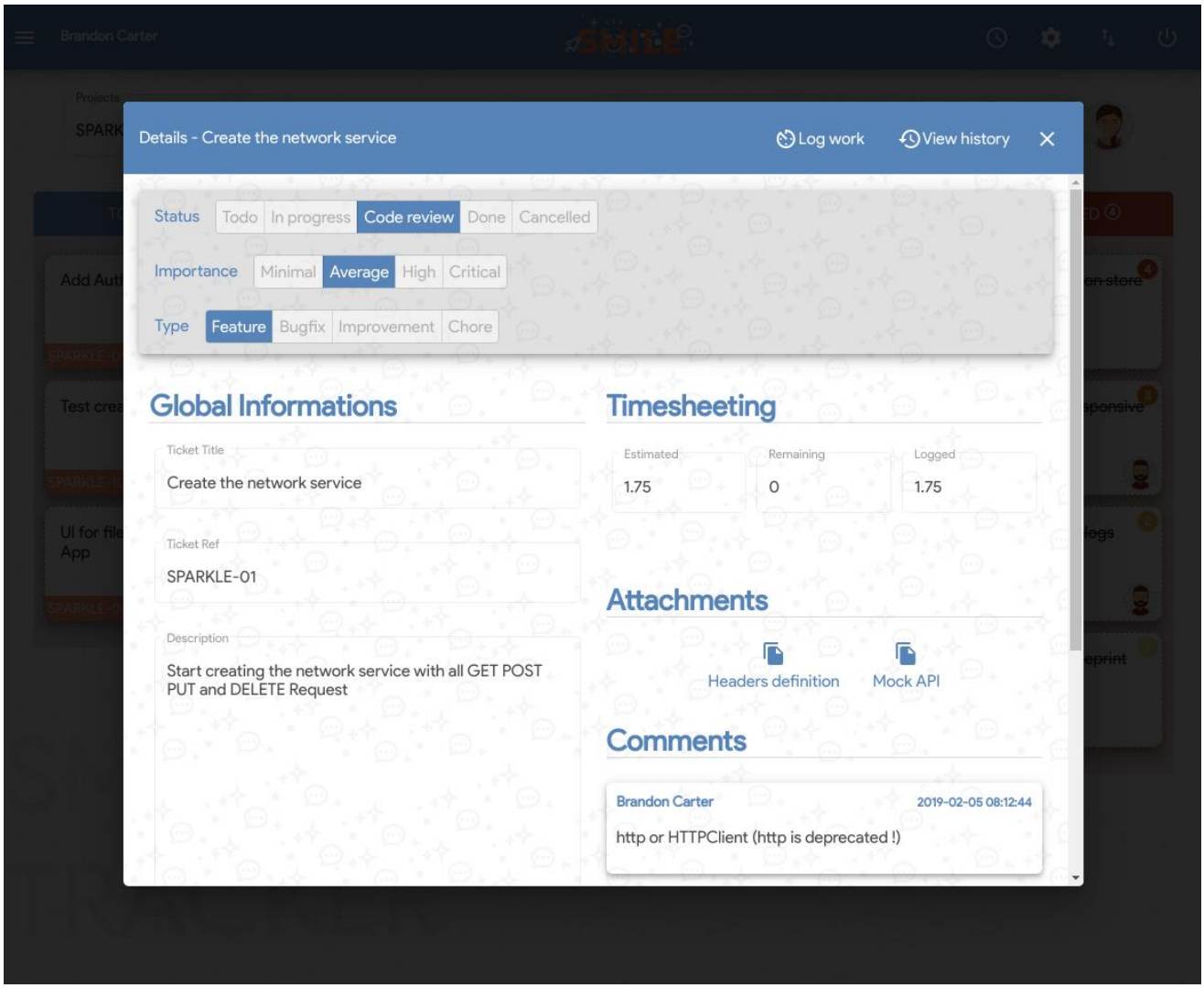


Figure. 42 : Interface de consultation d'un ticket

2.3.2 Log du temps

La figure [cf. Figure 43] présente le modal qui va permettre aux utilisateurs de loggé le temps passé sur une tâche. La valeur minimale est 0.125, qui représente une heure. L'utilisateur doit saisir aussi un message qui décrit brièvement le travail réalisé.

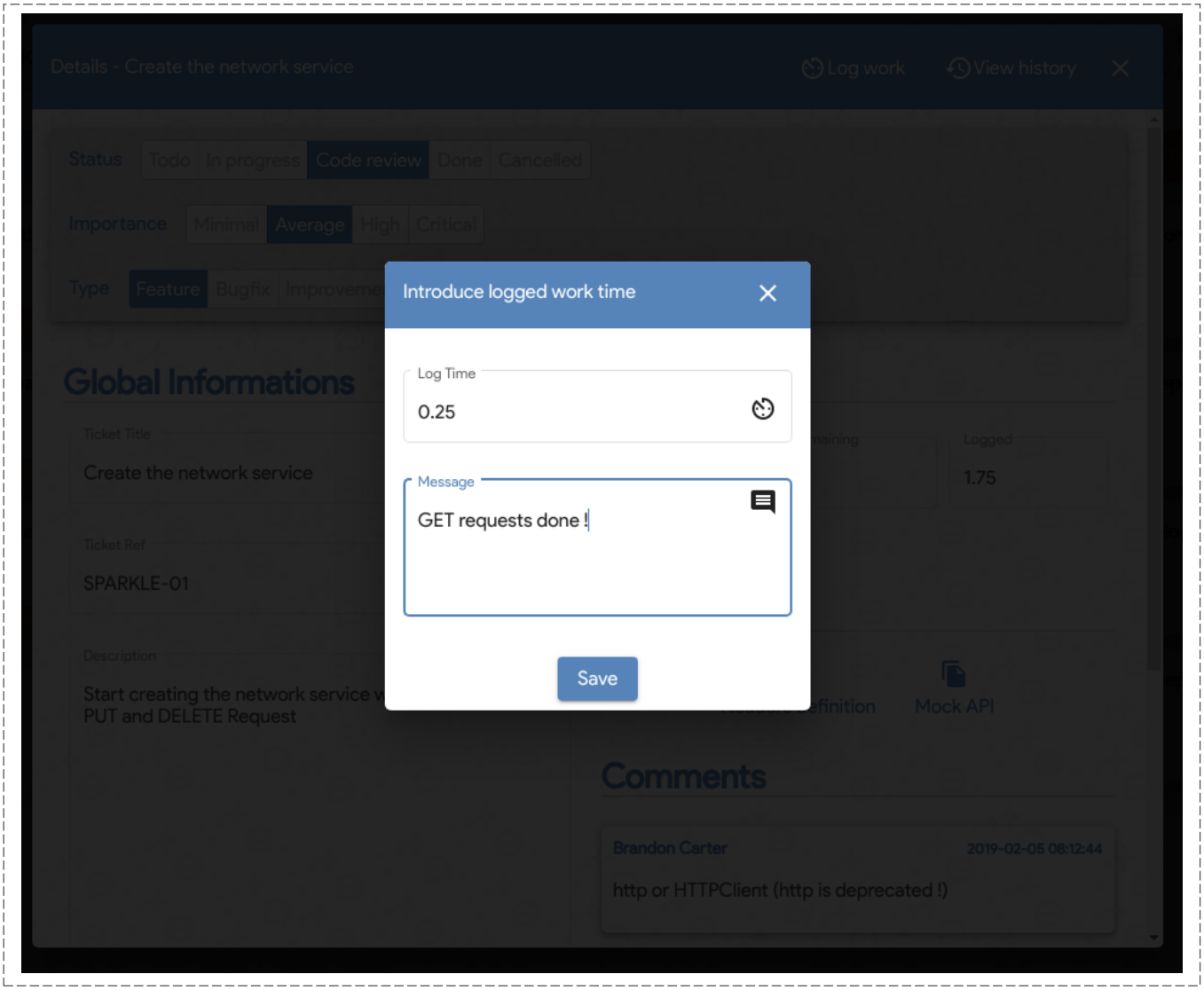


Figure. 43 : Interface de Log du temps

2.3.3 Génération de l'historique des tickets

Tous les utilisateurs de l'application peuvent générer un historique des différentes tâches effectuées sur un projet depuis la barre de navigation, en indiquant le nom du projet et un intervalle de temps. L'utilisateur peut après exporter ce log en format Excel [cf. Figure 44].

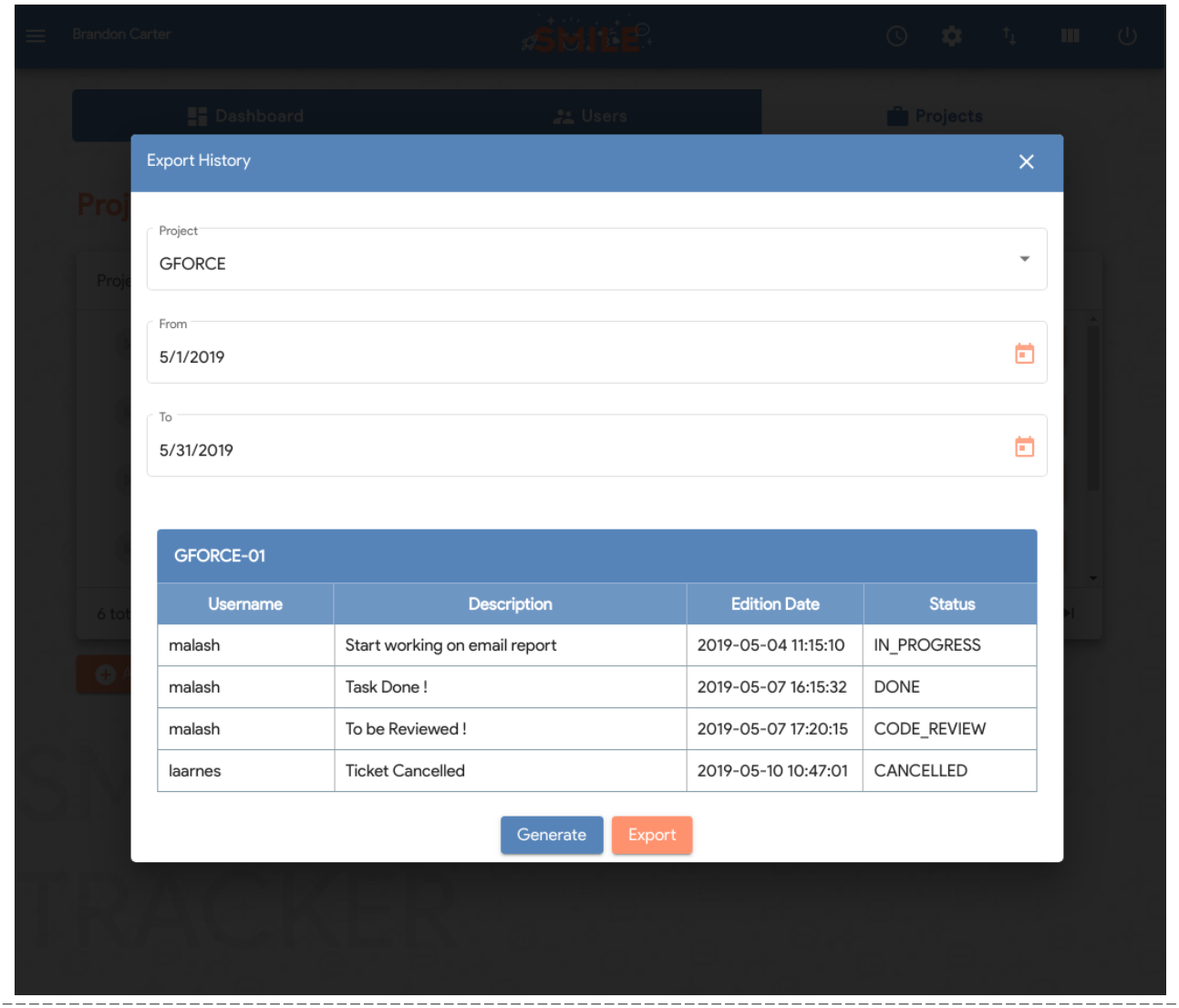


Figure. 44 : Interface de génération de l'historique des tickets

Conclusion

Dans ce dernier chapitre, nous avons décrit l'environnement matériel et logiciel, ainsi que les principales interfaces Backoffice et Frontoffice de notre application.

Conclusion générale

Dans le cadre de ce projet de fin d'études, nous étions amenés à la conception et le développement d'une application multiplateformes de gestion des tâches et de tracking. Le présent rapport couvre l'étude, la conception et l'implémentation de la solution en appliquant la méthodologie Agile Scrum.

Pour réaliser ce travail, nous avons commencé dans un premier temps par la présentation du contexte général de notre projet en étudiant le fonctionnement des outils existants où nous avons réalisé une étude comparative entre ces outils, avant de procéder à l'analyse et la spécification des besoins de notre application. Cette analyse a servi dans la conception et l'implémentation de la nouvelle solution.

Les difficultés rencontrées lors de ce travail concernent principalement l'estimation du temps nécessaire à la réalisation des différentes tâches et parties de l'application, à savoir la partie mobile et Desktop qui a plus d'importance.

Ce travail nous a permis d'approfondir nos connaissances dans les bonnes pratiques de programmation, ainsi que les modernes technologies utilisées dans le domaine. Au-delà de l'aspect technique, ce projet a été abordé à fin d'assurer une formidable expérience pour notre parcours, grâce aux contacts des membres du pôle Front-End et Mobile qui nous ont fait bénéficier de leur expérience. En effet, prendre en charge un tel projet dans un cadre professionnel nous permet assurément de développer notre esprit d'analyse, de réflexion et de décision.

Comme perspectives à ce travail, nous proposons l'amélioration de quelques fonctionnalités dans la partie Front-End de l'application et l'ajout de nouvelles fonctionnalités à savoir le multilingue, ainsi que la réalisation de la partie Mobile et Desktop dans les prochains sprints.

Bibliographie et Webographie

[1] C. Laros, Bonnes pratiques du web. [En ligne]. Disponible :https://www.smile.eu/fr/livres-blancslivres-blancs/bonnes-pratiques-du-web.

[2] https://www.smile.eu Site officiel de Smile, consulté en Avril 2019.

[3] https://stileex.xyz/redmine Présentation de l'outil de gestion de projet Redmine, consulté en Avril 2019.

[4] http://myentreprise.com/presentation-de-trello/ Présentation de Trello, consulté en Avril 2019.

[5] https://www.atlassian.com/agile/kanban/boards Introduction à la méthodologie Kanban, consulté en Avril 2019.

[6] https://les-tilleuls.coop/en/blog/article/introduction-a-jira-software Introduction à Jira, consulté en Avril 2019.

[7] https://agiliste.fr/introduction-methodes-agiles/ Introduction aux méthodes agiles et Scrum, consulté en Avril 2019.

[8] https://guide-angular.wishtack.io/typescript TypeScript - Le Guide Angular par Wishtack, consulté en Mai 2019.

[9] https://next.angular.io/docs Documentation officiel de Angular 7, consulté en Mars 2019.

[10] https://www.anthedesign.fr/developpement-web/preprocesseur-css-less-sass/ Préprocesseur CSS, consulté en Avril 2019.

[11] https://www.clevertoday.com/fr/javascript/ Pourquoi le JavaScript, consulté en Mai 2019.

[12] https://www.grafikart.fr/tutoriels/nodejs-intro-792 Qu'est-ce que NodeJS, consulté en Mai 2019.

[13] https://expressjs.com/fr/guide/using-middleware.html Utilisation du Middleware, consulté en Mai 2019.

[14] S. Holmes, Mongoose for Application Development. Birmingham, Mumbai : Packt Publishing, 2013. [En ligne]. Disponible : https://www.packtpub.com/web development/mongoose-application-development.

[15] https://www.supinfo.com/articles/single/179-introduction-ionic-framework Introduction à Ionic Framework, consulté en Mai 2019.

[16] https://confituregeek.com/2016/02/26/authentification-et-autorisation-avec-jwt/ Authentification et Autorisation avec JWT, consulté en Mai 2019.

[17] https://www.developpez.com/actu/164714/Faut-il-utiliser-Electron-pour-le-developpement-d-applications-de-bureau-Quels-sont-ses-avantages-et-inconvenients/ Utilisation de ElectronJS, consulté Mail 2019.

[18] https://www.grafikart.fr/formations/webpack Formation Comprendre Webpack, consulté en Mai 2019.

[19] https://guide-angular.wishtack.io/tools/npm NPM - Le guide Angular, consulté en Mai 2019.

[20] https://guide-angular.wishtack.io/tools/git Git - Le guide Angular, consulté en Mai 2019.

[21] https://assessment-tools.ca.com/tools/continuous-delivery-tools/fr/continuousintegration Définition de l'outil d'intégration continue, consulté en Mai 2018.

[22] https://edutechwiki.unige.ch/fr/Visual_studio_code Visual Studio Code, consulté en Mai 2018.

[23] https://slack.com/intl/fr-ma/ A propos le produit, consulté en Mai 2019.

Application multiplateforme pour la gestion et le tracking des tickets

Résumé

Pour rationaliser les processus et garder les équipes au courant, l'entreprise d'aujourd'hui vise à fournir à ses collaborateurs une plateforme centralisée où ils pourront avoir une vue d'ensemble claire de toutes leurs activités et de leurs communications au sein de l'équipe. C'est dans ce sens que s'inscrit notre projet au sein de Smile Maroc, qui consiste à mettre en place une solution de gestion des tâches et de tracking, qui est basé sur les outils manipulés quotidiennement par des équipes offshores et locales, avec l'ajout de nouvelles fonctionnalités pour répondre mieux aux besoins des équipes, et unifié leur manière de travail, on lui offrant une plateforme commune et personnalisée. Pour ce faire, nous avons commencé par l'étude de fonctionnement des outils existants utilisés par les équipes de développement pour bien analyser les besoins fonctionnels, puis nous avons réalisé une étude comparative entre ces outils, afin de mettre en œuvre la nouvelle solution. Et pour assurer l'agilité et la réactivité, nous avons choisi Scrum comme méthodologie de gestion.

Mots clés : MEAN Stack, JavaScript, Front-End, Gestion des tickets, Suivi du temps.

Multiplatform application for ticket management and tracking

Abstract

To rationalize processes and keep teams informed, today's company aims to provide employees with a centralized platform where they can have a clear overview of all their activities and communications within the team. In this context, our project consists in setting up a personalized solution of task management and tracking, which is based on the tools used daily by offshore and local teams, by the addition of new functionalities to better meet the needs of teams, and unified their way of working, offering them a common and personalized platform. To complete this mission, we began by studying the existing tools used by the development teams to properly analyze the functional needs, and then carried out a comparative study between these tools in order to implement the new solution. And to ensure agility and responsiveness, we chose Scrum as a management methodology.

Keywords : MEAN Stack, JavaScript, Front-End, Tasks management, Time Tracking.

Master Systèmes Intelligents & Réseaux
Département d'Informatique
Faculté des Sciences et Techniques de Fès

A.U. 2018 - 2019