CGI

Hassou Rayhan
IUT informatique
Clermont-Ferrand
2022-2023

Rapport de stage

Encadrant de stage : Issam Falih
(Professeur de l'IUT)
Tutrice de stage : Geraldine Blanc
(Senior Consultant)

UNIVERSITÉ
Clermont Auvergne

Confidentialité à confirmer

REMERCIEMENT

Je tiens à dédier cette section pour remercier toutes les personnes ayant contribué au bon déroulement de mon stage et qui m'ont aidé et soutenu.

Je voudrais remercier tout d'abord l'entièreté de l'équipe de CGI pour m'avoir accueilli au sein de cette entreprise dynamique, chaleureuse et très professionnelle.

J'aimerai aussi adresser des remerciements à Geraldine BLANC, ma tutrice au sein de l'entreprise, pour m'avoir conseillé lors de ce stage mais aussi Franck GLAZIOU pour m'avoir accompagné pendant mes premieres semaines de stages.

J'aimerai bien évidemment donner mes plus sincères remerciement Emilie RIBOULET et Jonatan Ajus, mes tuteurs de projet, qui m'ont accompagné durant l'intégralité du stage, qui m'ont autant bien conseillé que rassurer, qui ont travaillé avec moi sur mon sujet et qui m'ont permis de passer un excellent stage.

Je souhaite aussi passer des remerciement particulier à Hugo LIVET (alternant en 3eme année de BUT), Nicolas FRANCO (alternant en 3eme année de BUT), Clément DA CUNHA (Développeur) qui m'ont accompagné tout au long du stage, et m'ont renseigné quand j'avais besoin d'aide.

Enfin, je souhaite remercier mon professeur référent, Mr Issam FALIH pour m'avoir suivi et conseillé durant le stage et l'IUT informatique d'Aubière pour les connaissances m'ayant permis le bon déroulement du stage.

INTRODUCTION

Du 19 février au 10 mai 2024 puis du 1 juillet au 2 août 2024, j'ai réalisé mon stage de troisième année de BUT Informatique chez CCI situé à Clermont-Ferrand au 10 rue Eric de Cromières. Ce fut ma 2eme expérience en entreprise suite à mon stage de deuxième année. J'ai pu rencontrer une première fois cette entreprise lors du forum des stages se tenant à l'IUT puis j'ai eu la chance par la suite de faire un entretien et d'obtenir mon stage.

CGI, une entreprise avec une clientèle diversifiée, compte Michelin parmi ses principaux clients. Mon stage a débuté dans le cadre du développement d'une application spécifique, conformément aux exigences détaillées dans le cahier des charges de Michelin.

Tout d'abord, je commencerai par présenter l'établissement et son organisation, puis je détaillerai les objectifs du stage. Suite à cela, je présenterai une partie analyse dans laquelle j'exposerai l'étude et la conception préalable à la réalisation pour ensuite présenter le développement de l'application en lui-même. Enfin, je dresserai un bilan technique et personnel de ce stage.

I. Présentation générale

CGI, une société canadienne établie à Montréal, se spécialise dans les services de consultation en technologie de l'information, l'intégration de systèmes et les solutions informatiques. Serge Godin et André Imbeau l'ont fondée en 1976, et elle est actuellement classée parmi les cinq principales entreprises mondiales de son domaine.

La société opère dans divers secteurs tels que la chimie, la logistique, et son agence à Clermont-Ferrand se concentre principalement sur l'industrie, l'agroalimentaire, ainsi que le secteur public. Il convient de noter que la division Business Consulting est une composante transversale présente dans toutes les agences de CGI.

À l'heure actuelle, CGI affiche un chiffre d'affaires de 12,1 milliards de dollars canadiens, compte 88 500 employés répartis dans le monde entier, et offre ses services à plus de 5 500 clients à travers plus de 400 agences.

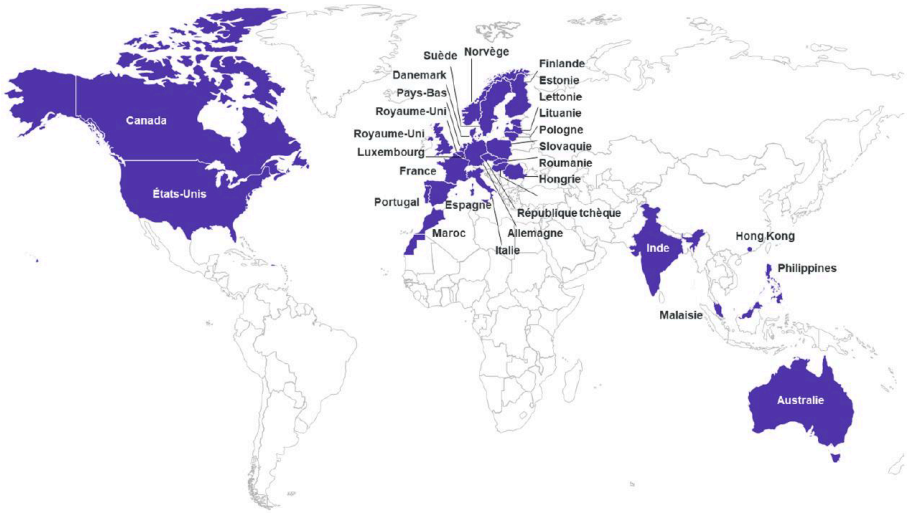

Figure n°1 : Représentation de CGI dans le monde (Source : CGI)

Le premier Figure indique que CGI a une présence étendue à l'échelle mondiale, avec une volonté d'être en proximité géographique avec ses clients pour améliorer leur expérience et faciliter la détection de leurs besoins.

En ce qui concerne l'Europe, elle représente une entité commerciale stratégique à part entière pour CGI, classée en tant que SBU*. Cette région revêt une importance particulière, regroupant un total de 19 000 collaborateurs dans la SBU* Europe de l'Ouest et du Sud. Cette présence européenne se déploie dans 8 pays, organisés en 14 Business Units.

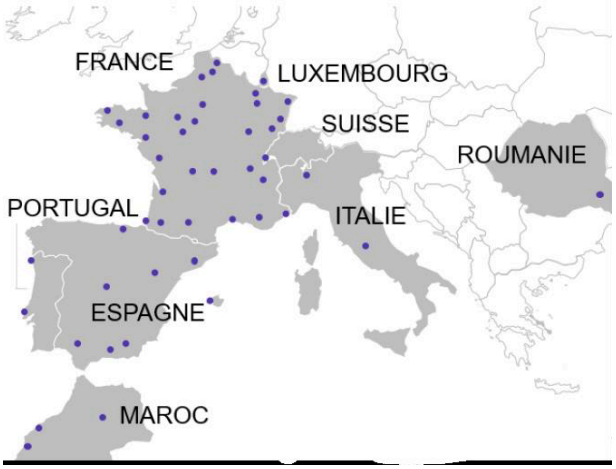

Figure n°2 : Représentation de CGI dans la SBU 3 Europe de l'Ouest et du Sud (Source : CGI)

Nous allons nous focaliser sur la Business Unit (BU) appelée « AURA », étant donné que l'agence de Clermont-Ferrand relève de cette entité. « AURA » désigné « Auvergne-Rhône-Alpes et Suisse ». Voici la structure hiérarchique de la BU AURA. Sur cet organigramme, Lionel Delair occupe le poste de directeur de l'agence de Clermont-Ferrand.

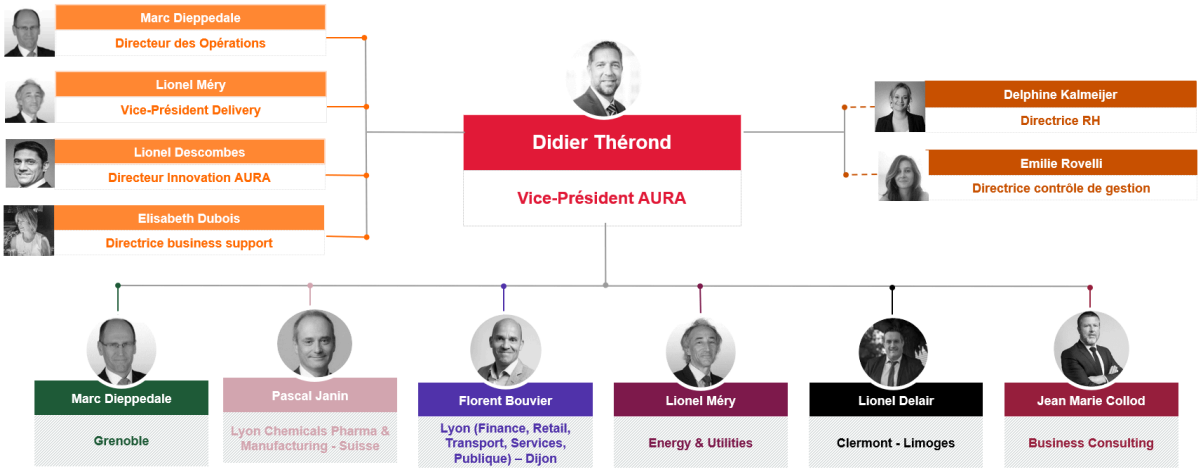

Figure n°3 : Organigramme de la BU4 « Auvergne-Rhône-Alpes et Suisse » (Source : CGI)

* SBU : « Strategic Business Unit » ou « Unité d'Affaires Stratégiques » (UAS) en français.

* BU : Business Units – entités responsables de la relation client.

II. Présentation de l’agence de Clermont-Ferrand

Situé au 10 rue eric de cromières, l’agence de Clermont-Ferrand accueille des employés de différents secteurs que ce soit le développement, le déploiement, la TMA et le support.

Figure n°3 : Agence de CGI

Pour cette agence, voici l'organigramme représentant la hiérarchie et les différents postes occupés par ces personnes. On retrouve les termes « VPCS » et « VPCE » sur le diagramme. VPCS désigne les Vice-Présidents Service Conseil et VPCE équivaut à « Vice-Présidents Service Expert ».
Dans le cadre de mon stage, je suis dans le verticale “Production System”.

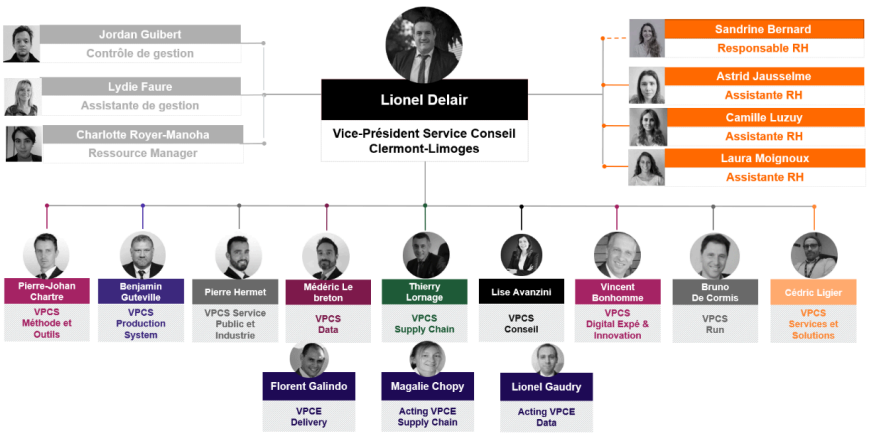

Figure n°4 : Organigramme du secteur Clermont-Ferrand/Limoges (Source : CGI)

III. Présentation de Michelin

La société Michelin a été établie en 1889 par les frères André et Edouard Michelin à Clermont-Ferrand, se consacrant à la fabrication et à la commercialisation de pneumatiques. Elle est célèbre pour son personnage emblématique Bibendum. En 1900, Michelin a lancé son premier guide Michelin. Au fil des années, Michelin a adopté une approche paternaliste envers ses employés et a joué un rôle significatif pendant les deux guerres mondiales.

Michelin a établi une collaboration fructueuse avec CGI en matière de développement d'applications. Cette collaboration s'étend à divers domaines au sein de l'entreprise, englobant des applications cruciales pour les opérations des usines ainsi que des solutions internes destinées aux bureaux.

CGI, en tant que fournisseur de services-conseils en technologie de l'information, joue un rôle clé dans la conception et la mise en œuvre d'applications sur mesure répondant aux besoins spécifiques de Michelin. Ces applications sont conçues pour optimiser les processus de fabrication dans les usines de Michelin, améliorer l'efficacité opérationnelle et répondre aux exigences internes au sein des bureaux de l'entreprise.

Grâce à cette collaboration, Michelin bénéficie de solutions technologiques adaptées et innovantes, renforçant ainsi ses capacités opérationnelles à tous les niveaux. La combinaison de l'expertise de CGI en matière de technologie de l'information et des besoins spécifiques de Michelin contribue à maintenir un environnement informatique efficace et performant au sein de l'entreprise.

Figure n°4 : Siège social de Michelin à Clermont-Ferrand

PRESENTATION SYNTHETIQUE DU STAGE

I. Existant et objectif du projet

Michelin détient actuellement plusieurs applications opérationnelles sur divers sites, avec d'autres en cours de développement. Une étape cruciale dans le cycle de vie de ces applications est le passage du mode développement au mode support, et vice versa. Une application prête à être déployée sur un site va passer une série de tests effectués par l'équipe support afin de valider sa conformité. Chaque application à plusieurs critères à valider. Ainsi, lorsqu'une application passe en mode support et qu'un déployeur la déploie sur un site, celui-ci sait qu'en cas de problèmes techniques il doit s'adresser à l'équipe support et non au déployeur.

Dans ce contexte, on m'a confié la mission de concevoir une application visant à répertorier l'ensemble des applications de Michelin et à faciliter la gestion de leur statut (mode développement ou mode support).

Cette application offre une interface permettant d'ajouter de nouvelles applications en cours de développement. Elle propose des champs détaillés pour saisir des informations telles que le nom de l'application, sa description et son statut actuel. Pour simplifier la gestion opérationnelle, l'application intègre également un mécanisme permettant de basculer une application en mode support en suivant une liste de critères prédéfinis.

En résumé, l'objectif principal de cette application est de fournir une solution centralisée pour suivre et gérer le statut de toutes les applications de Michelin, en facilitant le processus de transition entre les phases de développement et de support. Elle vise à optimiser la gestion interne tout en assurant une adaptation fluide aux besoins évolutifs de l'entreprise.

Partant de zéro, ma première étape consiste à élaborer un cahier des charges initial que je proposerai à Michelin. Ce Figure détaillera les exigences et les fonctionnalités attendues pour l'application. Une fois le cahier des charges validé par Michelin, je procéderai au développement de l'application, suivie par des phases de tests approfondis pour garantir la fiabilité et la performance du produit final. Enfin, une fois que l'application aura passé avec succès les tests, je serai responsable du déploiement efficace de l'outil au sein de l'environnement opérationnel de Michelin. L'ensemble du processus sera réalisé en étroite collaboration avec Michelin, afin de m'assurer que l'application répond parfaitement à leurs besoins spécifiques.

II. Gestion de projet

Pour ce projet, j'ai opté pour une gestion Scrum. Scrum est une méthode agile qui permet de structurer la gestion d'un projet en le subdivisant en plusieurs phases distinctes, chacune durant quelques jours, et ayant des objectifs spécifiques.Chaque phase est nommée sprint et dure 2 semaines chacune.

Cette approche agile favorise une gestion flexible et itérative du projet. Chaque sprint se concentre sur des objectifs clairement définis, ce qui permet une adaptation rapide aux changements éventuels tout en assurant une progression constante. La méthode du sprint favorise une collaboration efficace au sein de l'équipe de développement, encourageant la transparence et la communication continue. Elle offre également la possibilité d'ajuster les priorités en fonction des retours obtenus après chaque itération, assurant ainsi une meilleure satisfaction des besoins de Michelin tout au long du processus de développement de l'application.

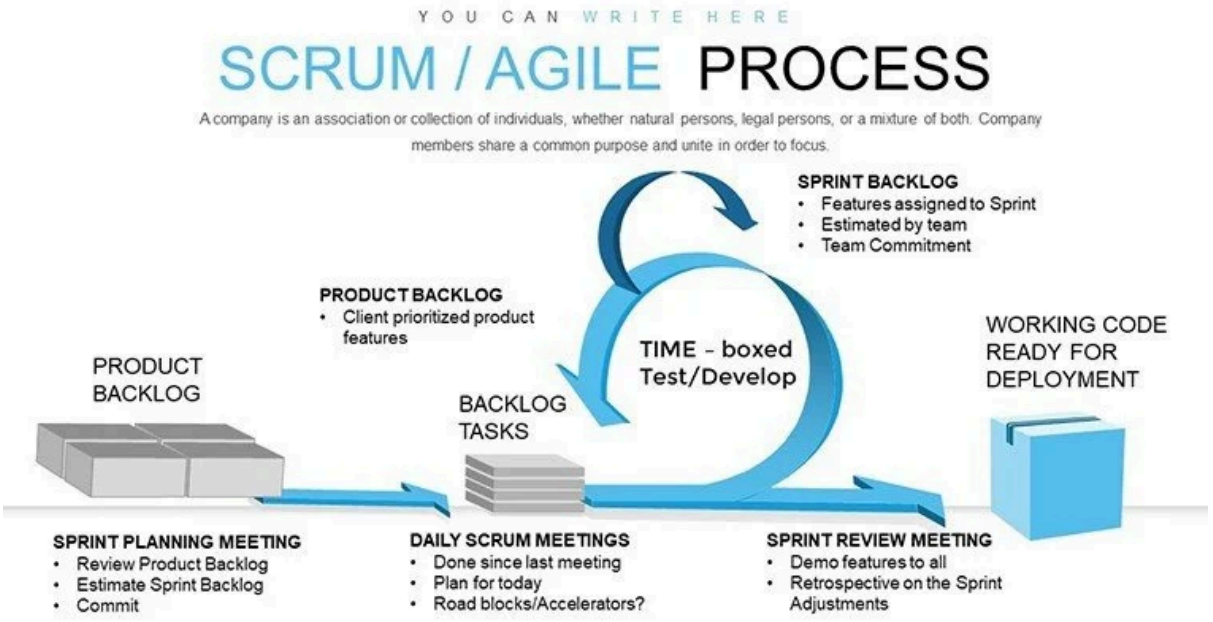

Figure n°4 : Les différentes étapes de la méthode scrum

Néanmoins, étant seul sur le projet dans un premier temps, je ne vais pas utiliser tous les composants de cette méthode. Pour ce projet, je n'utiliserai qu'un sprint backlog afin de planifier les tâches à réaliser pendant mes sprints et un sprint review pour suivre mon avancement et identifier les difficultés rencontrées.

parler du role de chaque membre

parler du kanban (trello)

Avant d'entamer le développement de toute application, il est crucial d'effectuer une analyse approfondie du sujet. Cette étape est essentielle pour bien comprendre le contexte, s'approprier les enjeux, et concevoir efficacement les différentes parties du projet. L'objectif ultime est d'optimiser le temps consacré au projet et de livrer une application de qualité. Dans cette section, nous explorerons donc l'analyse que j'ai réalisée sur le sujet, la réflexion qui en a découlé, et nous examinerons les différents schémas de conception élaborés en réponse à cette analyse.

I. Diagramme de cas d'utilisation

Dans un premier temps, j'ai analysé le diagramme de cas d'utilisation fourni par Michelin pour comprendre un peu mieux le sujet et en extraire les fonctionnalités attendues.

Pour rappel, un diagramme de cas d'utilisation, dans le cadre d'UML, représente les fonctionnalités et exigences d'un système. Il inclut les acteurs (entités externes au système), les cas d'utilisation (fonctions ou actions du système), le système lui-même (définissant le périmètre des cas d'utilisation), et éventuellement des paquets pour regrouper ces éléments.

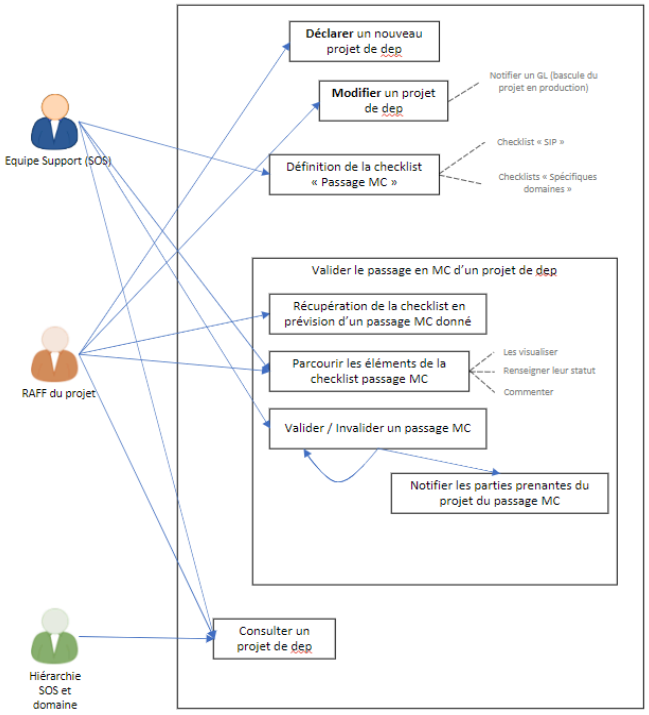

Figure n°5 : Diagramme de cas d'utilisation

Analysons plus en détail :

Il est clair que l'équipe de support assume la responsabilité de basculer les projets en mode support, entraînant ainsi la nécessité de gérer des rôles et des autorisations distincts au sein de l'application. Cette gestion des utilisateurs introduit une complexité supplémentaire dans le projet.

En outre, il est observé qu'il existe une checklist à suivre pour effectuer la transition d'une application en mode support. Cela sous-entend également la nécessité de gérer l'ajout, la suppression et la modification de ces checklists.

Afin de faciliter cette gestion et d'améliorer l'efficacité du processus, il serait judicieux de mettre en place des procédures claires et une communication transparente entre les membres de l'équipe de support, les RAFF du projet et la hiérarchie SOS et domaine. De plus, l'automatisation des tâches répétitives liées à la gestion des utilisateurs et des checklists pourrait être envisagée pour minimiser les erreurs humaines et optimiser le temps et les ressources.

Après avoir réalisé mon analyse, j'ai consigné les différentes fonctionnalités à implémenter. La prochaine étape consiste à comprendre le fonctionnement de ces fonctions. Pour ce faire, j'ai employé des diagrammes de séquences.

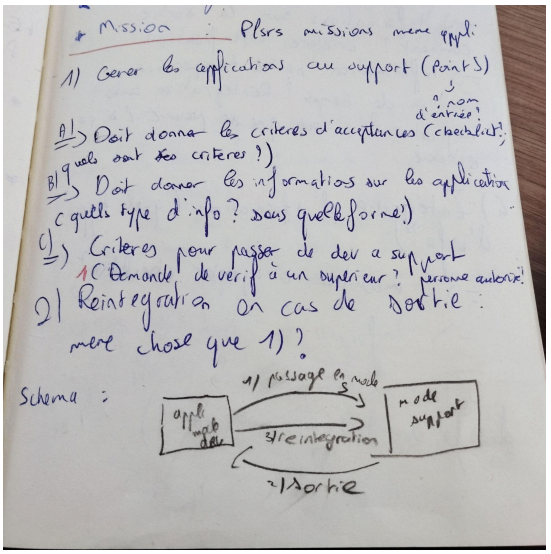

Figure n°6 : Extrait des notes prises après analyse.

Dans un second temps, j'ai eu l'occasion de créer mes propres diagrammes de cas d'utilisation, mais avant ça, j'ai réalisé des users stories afin de mieux déterminer quels sont les cas à détailler.

Une user story est une description des fonctionnalités d'un projet web, écrite du point de vue de l'utilisateur final. Elle est utilisée dans la méthode agile pour définir comment un travail apportera de la valeur ajoutée au client. Son objectif est de décrire tout le contenu d'une fonctionnalité de manière à répondre au mieux aux besoins de l'utilisateur. La user story met l'utilisateur et ses problématiques au centre de la réflexion pour garantir une réponse efficace à son besoin.

Elle se décompose en trois parties : En tant que (permet de définir l'utilisateur), je souhaite (permet de définir la fonctionnalité), afin de déterminer l'utilité de la fonction et le besoin réel).

```
En tant que membre de l'équipe support,
je souhaite pouvoir mettre un commentaire lorsque je passe un critère en "validé avec réserve"
afin de pouvoir exprimer les éléments à changer.

En tant que membre de l'équipe support,
je souhaite pouvoir accéder à un historique complet des modifications apportées aux checklists
afin de pouvoir verifié à tout moment si les changements sont correctes.

En tant que membre de l'équipe support,
je souhaite pouvoir effectuer des recherches avancées dans les checklists
afin de trouver rapidement des informations spécifiques.
```

Figure n°7 : Exemples d'user stories

Ces users stories serviront aussi comme dit précédemment dans la phase de développement. A chaque sprint nous déciderons avec mes tuteurs des users stories à développer car une user story est égale à une fonctionnalité. Pour faciliter cette étape, j'ai classé les user stories par ordre d'importance en me basant sur mon expérience et les discussions avec le client.

A partir des users stories, j'ai déterminé quels cas méritent un diagrammes de cas d'utilisation pour mieux comprendre. Voici quelques exemples :

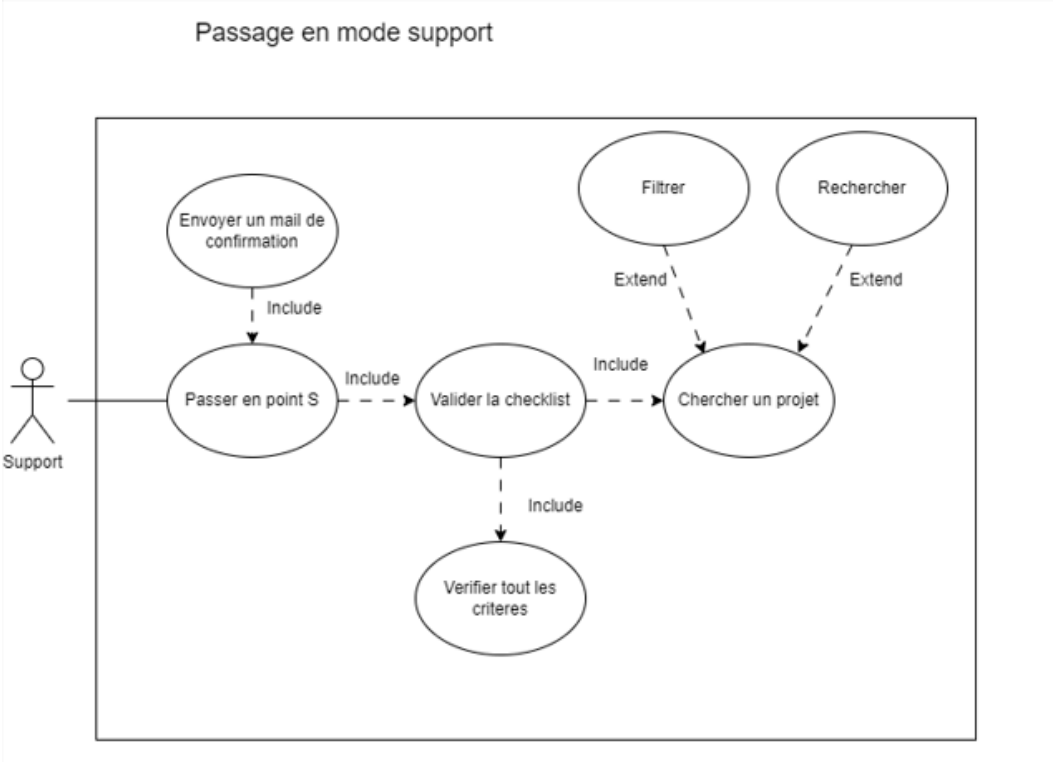

Figure n°? : Diagramme de cas d'utilisation pour le passage en mode support

Le premier diagramme représente le cas d'utilisation pour le passage en mode support. Dans ce cas, l'acteur est un membre de l'équipe de support. Pour passer une application en mode support (point S), l'acteur doit effectuer plusieurs actions obligatoires, indiquées par l'utilisation de l'indicateur **Include**. Ainsi, pour passer une application en mode support, un membre de l'équipe de support doit obligatoirement valider la checklist. Pour valider une checklist, l'utilisateur a également deux actions obligatoires : chercher le projet pour accéder à la checklist et vérifier tous les critères de la checklist.

De plus, l'utilisateur a la possibilité d'envoyer un mail de confirmation, mais cette action doit obligatoirement être effectuée après le passage en mode support.

On peut également constater que l'utilisateur dispose d'actions non obligatoires, représentées par l'indicateur Extend associé à l'action "chercher un projet". Cela signifie que lorsqu'il recherche un projet, l'utilisateur a la possibilité de filtrer les résultats ou de faire une recherche spécifique, mais il n'est pas obligé de le fa

II. Diagrammes de séquence

Les diagrammes de séquence, intégrés dans le langage de modélisation UML, sont une méthode populaire pour représenter de manière dynamique les interactions entre différents éléments tels que objets, processus et lignes de vie. Ils mettent l'accent sur la chronologie des événements, montrant comment ces entités interagissent en échangeant des messages pour accomplir une fonction spécifique avant la fin de leur existence. Ces diagrammes sont essentiels pour comprendre les scénarios temporels et les relations entre les composants d'un système, offrant ainsi une vision claire et détaillée des interactions dans un format graphique facile à interpréter.

Ici, je vais vous présenter le diagramme que j'ai réalisé pour la fonctionnalité "passage en mode support".
Pour commencer, regardons la fonction passage en mode support.
Cette fonction concerne 2 pages, la page d'application et la page de validation. L'utilisateur choisit une application à passer en mode support. Après avoir choisi l'application, l'utilisateur va vérifier toutes les conditions pour le passage en mode support. Si les critères ne sont pas vérifiés, on peut penser à un ajout à une sorte de liste d'attente, répertoriant les applications ayant fait l'objet d'une demande de passage en support sans y aboutir. Au contraire, on pourrait envoyer une notification au site sur lequel l'application est déployée afin de leur indiquer que l'application est bien passée en mode support. Ce diagramme m'a aussi permis de soulever de nombreuses questions dont je n'avais pas de questions ce qui me permettra par la suite d'avoir un meilleur échange avec le client et mieux comprendre son besoin.

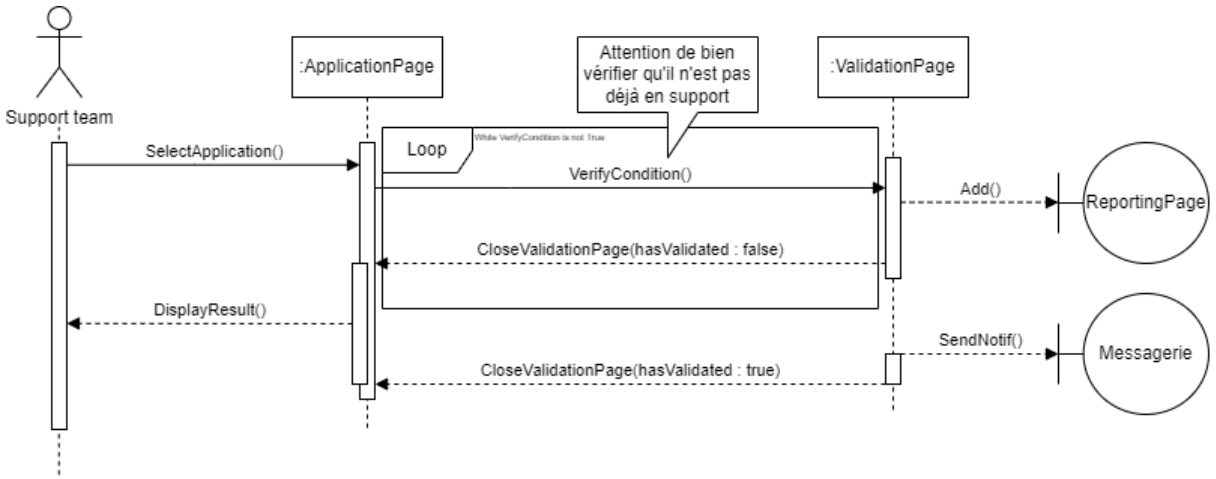

Figure n°? : Diagramme de séquence "passage en mode support"

III. Diagramme d’activité

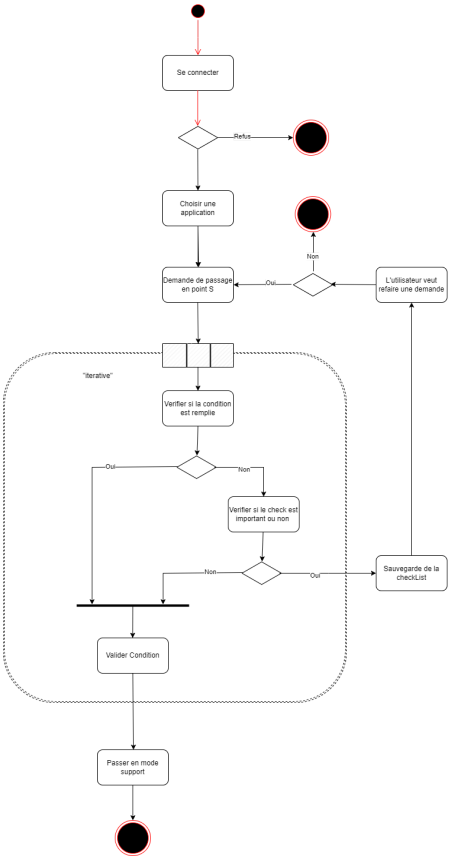

Figure n°? : Diagramme d'activité “passage en mode support”

Les diagrammes d'activités, les diagrammes de cas d'utilisation et les diagrammes d'états-transitions sont des outils comportementaux qui décrivent les interactions et le fonctionnement d'un système, facilitant ainsi la compréhension et la communication des exigences fonctionnelles. Dans mon cas, j'ai utilisé ces diagrammes pour comprendre le processus de passage d'un projet en mode support. Initialement, l'utilisateur doit se connecter pour des raisons de sécurité, afin de limiter l'accès aux seules personnes autorisées de CGI ou Michelin. Ensuite, il sélectionne un projet et demande un passage en point S, déclenchant une évaluation itérative des critères. À chaque étape, nous vérifions la validation du critère, passant au suivant s'il est validé, sinon, nous vérifions s'il est essentiel. Si ce n'est pas le cas, nous passons au critère suivant, mais sinon, nous refusons le passage en point S. À ce stade, l'utilisateur a la possibilité de refaire une demande de point S ou non. Ce processus est sujet à des évolutions et des améliorations, mais il a été conçu pour m'aider à saisir le besoin de manière approfondie et à identifier les classes impliquées.

IV. Conception de la base de données

A) Premiere version

Après avoir appréhendé le fonctionnement attendu de l'application, j'ai pu progresser vers l'étape suivante, à savoir la phase de conception. En priorité, j'ai travaillé sur ce qui m'a semblé le plus crucial, à savoir l'organisation des données et les relations entre elles. Pour ce faire, j'ai élaboré un Modèle Conceptuel de Données (MCD) afin de définir les entités et les interactions entre celles-ci.

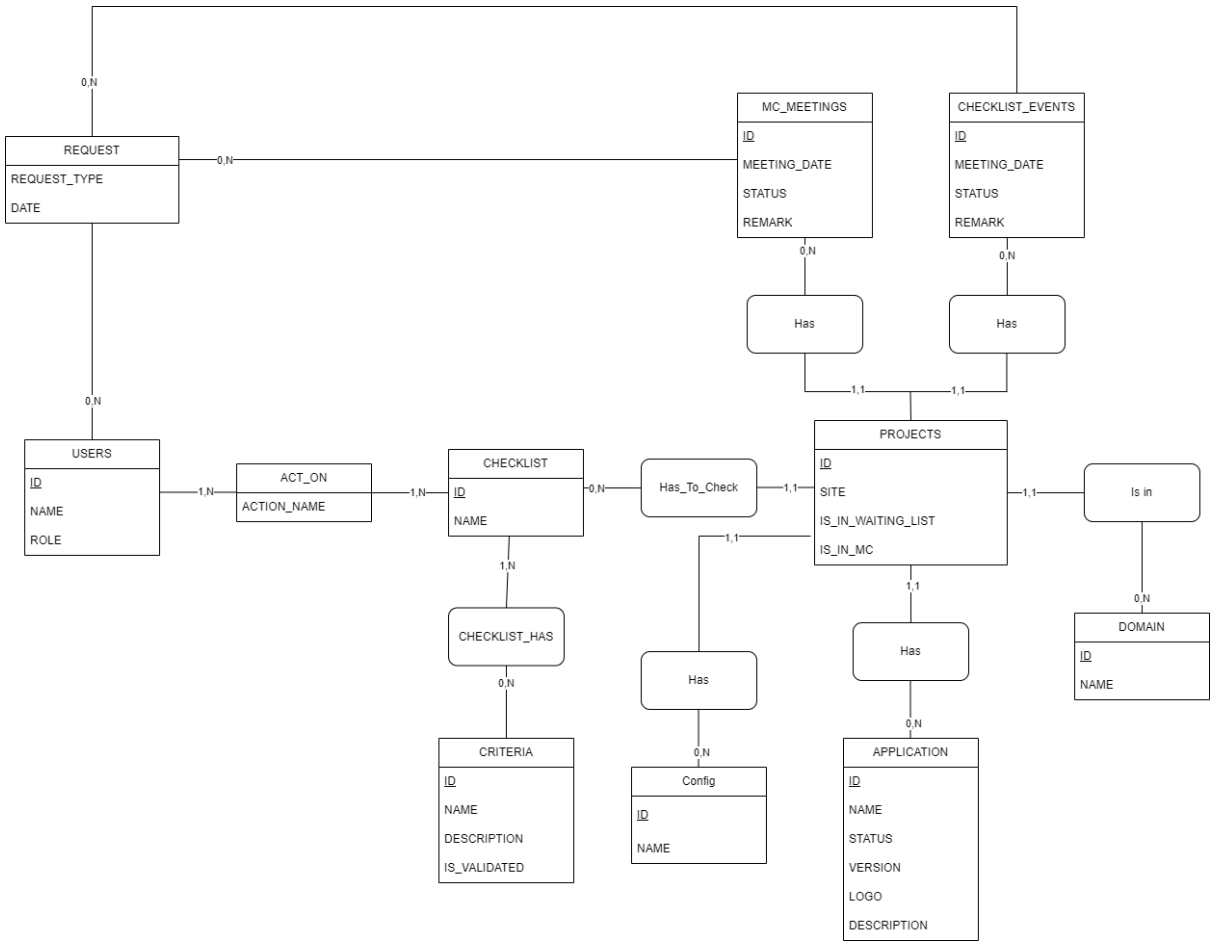

Figure n°? : Première version du Modèle Conceptuel de Données (MCD)

MCD* : Dans la méthodologie Merise destinée à créer des bases de données, le **MCD (Modèle Conceptuel des Données)** est une représentation graphique de haut niveau qui permet facilement et simplement de comprendre comment les différents éléments sont liés entre eux.

Voici une explication des différentes entités et leurs relations :

Projets : Correspond à la mise en production d'une application sur un site de données. Un projet a donc une relation avec une application, un site, une checklist à vérifier avant de le passer en mode support, la configuration utilisée sur le site, ainsi que les classes Mc_Meetings/CheckLists_Events, que nous présenterons ultérieurement.

Application : Correspond aux applications développées. Chacune a une version, car une application peut être déployée sur différents sites avec des versions différentes. Ainsi, une application possédant deux versions distinctes est enregistrée comme deux applications distinctes dans la base de données.

CheckList : Correspond aux checklist que les applications devront valider pour pouvoir passer en mode support. Chaque checkList est relié à un ou plusieurs critère (table : Criteria) qui viennent de 3 provenances différentes : La SIP, le domaine et l'application en elle-même.

Users : Les utilisateurs de l'application sont inclus dans le MCD car j'ai anticipé la gestion des rôles et des restrictions de fonctionnalités en fonction du type d'utilisateur, comme illustré dans le cas d'utilisation de la page ?. Les utilisateurs ont la possibilité d'interagir avec les checklists (modification, ajout, suppression) et de déclencher des Mc_Meetings et Checklist_Events.

Mc_Meetings / CheckList_Events : Ces classes étaient incluses dans le sujet que l'on m'a initialement donné. Cependant, mes tuteurs étaient absents lors des deux premières semaines, ce qui a entraîné un manque considérable d'informations pour moi. En conséquence, je n'ai pas compris pleinement pourquoi ces deux classes étaient si similaires et leur utilité m'échappait. J'ai supposé qu'elles représentaient le passage en point S et j'ai donc décidé de les inclure dans mon travail.

Après avoir élaboré mon Modèle Conceptuel de Données (MCD), j'ai procédé à la création d'un Modèle de Relations de Données (MRD). Les tables de la base de données seront codées conformément à ce MRD. Pour les relations de type 1,1, les clés primaires de la table liée deviennent des clés étrangères pour la table principale. Par exemple, dans la relation 1,1 entre les tables "Projects" et "Application", la clé primaire d'Application devient une clé étrangère dans la table Projects.

En revanche, pour les relations de type 0,N/1,N, la liaison entre les deux entités devient une table intermédiaire. Cette table intermédiaire utilise les clés primaires des deux tables qu'elle relie en tant que clés primaire et étrangère. Cette approche facilite la représentation des relations entre entités dans la base de données.

Par exemple, la relation Act_On entre User et CheckList devient une table à part entière.

Ainsi, en respectant ces principes, le MRD sert de guide pour la création de la structure de la base de données, garantissant une représentation précise et cohérente des relations entre les différentes entités du système.

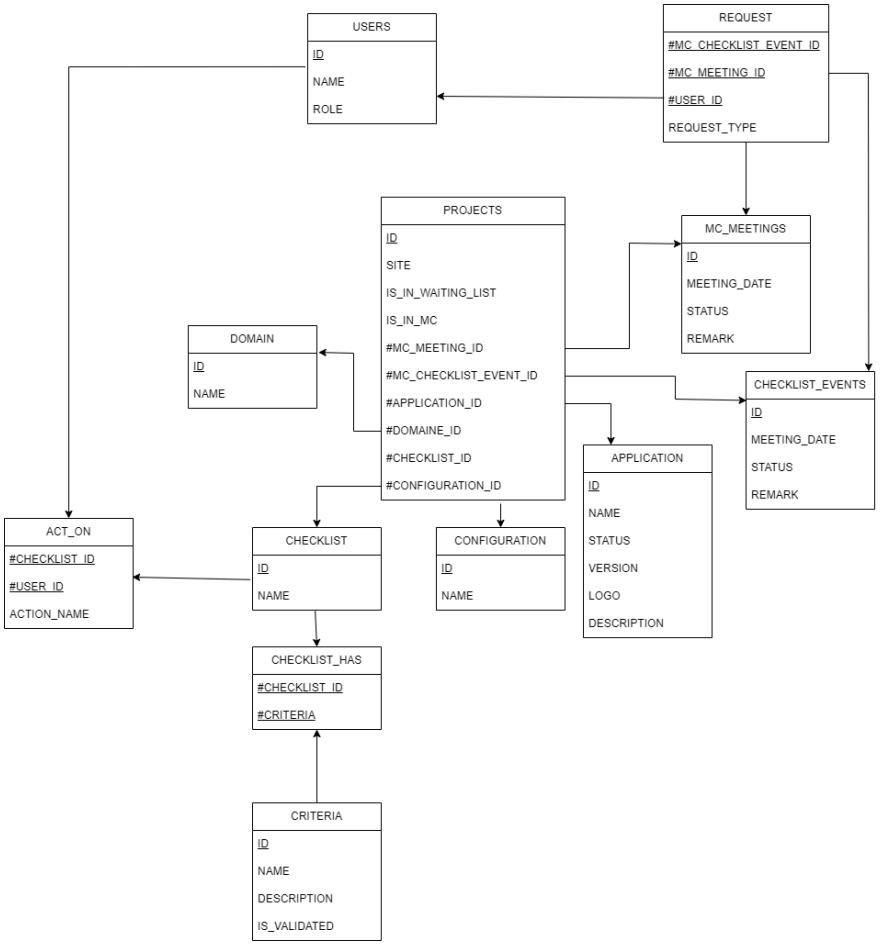

Figure n°? : Première version du Modèle Relation de Données (MRD)

B) Deuxième version

En n'ayant pas accès à l'intégralité du sujet au départ, j'ai conçu ma solution de manière flexible, anticipant la possibilité de la modifier et de l'adapter facilement. Après plusieurs discussions avec mes tuteurs et une meilleure compréhension du sujet, j'ai pu créer cette version finale du MLD.

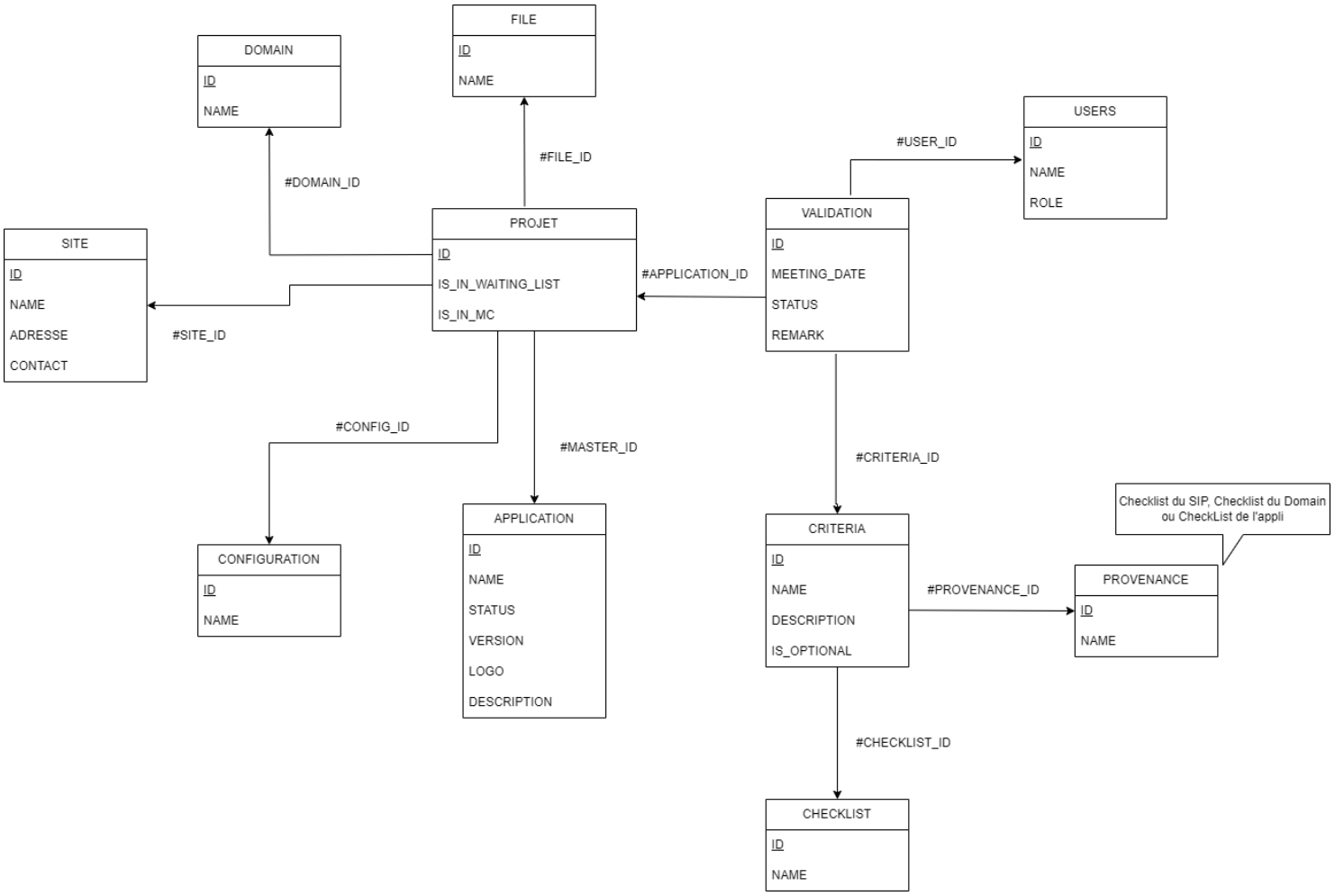

Figure n°? : Première version du Modèle Relation de Données (MRD)

La différence majeure avec l'ancienne version est la disparition des classes Mc Meetings et CheckList Events finalement inutiles car représentant les réunions de passages en point S, il n'est donc pas nécessaire de les enregistrer en base de données surtout en deux fois. Ces classes sont remplacées par une classe validation qui permet de répertorier les mises en production permettant tout de même un suivi en cas de doute. Cette version à aussi permis de mettre en valeur la problématique de la provenance des checklist car en effet chose que je ne savais pas lors de ma première conception est que les critères peuvent venir de 3 sources différentes : Le SIP, le domaine du projet et de l'application (donc les développeurs).

V. Définition des vues

Après avoir finalisé la conception de la base de données, j'ai entrepris la création de maquettes pour avoir une vision initiale de l'interface de l'application et pour identifier les éventuels éléments manquants.

La première maquette que j'ai réalisée est celle de la page d'accueil de l'application, car elle constitue la porte d'entrée principale et permet de définir les parcours vers les autres sections. En termes de design, j'ai puisé mon inspiration dans les applications Blazor pour le menu, car j'estime que leur structure fonctionne très bien. Pour le reste de l'interface, je n'ai pas trouvé d'applications similaires à celle que je développe, ce qui m'a conduit à procéder par essais successifs pour obtenir un design à la fois simple, intuitif et esthétique. Après plusieurs itérations, j'en suis arrivé à ce résultat où la majeure partie de la page est occupée par une liste déroulante des applications, avec des filtres situés au-dessus pour les mettre en évidence. En ce qui concerne les projets, j'ai inclus uniquement les informations pertinentes telles que le statut, la région, la zone et le domaine, accompagnées d'une pastille de couleur permettant de visualiser rapidement le statut de l'application.

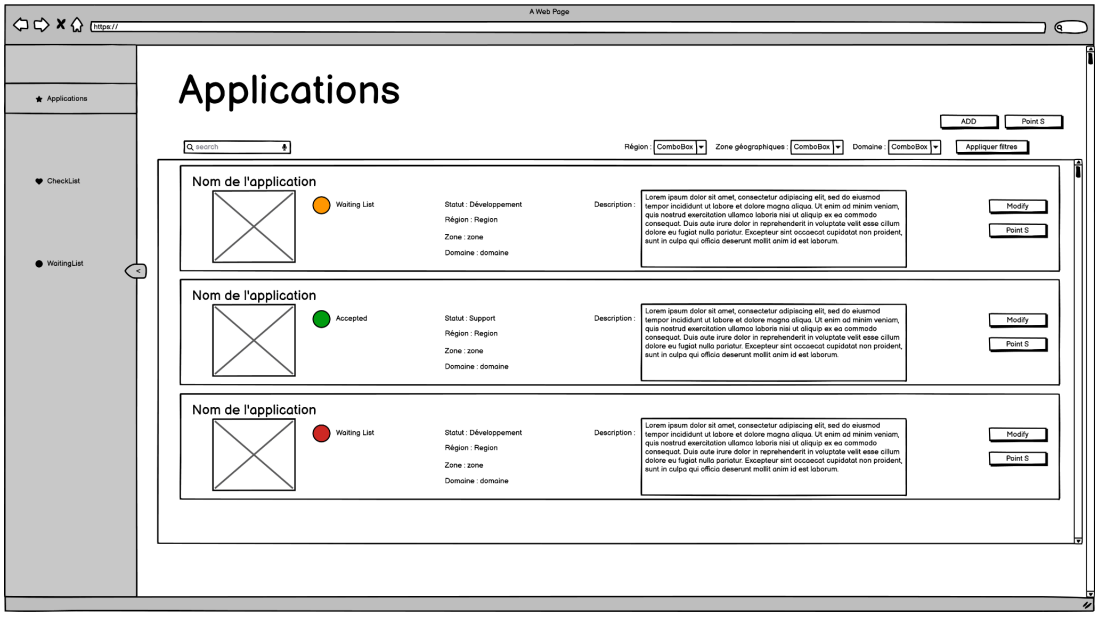

Figure n°? : Première version de la maquette pour la page “Applications”

Pour la page "Waiting List", j'ai adopté le même principe que pour la page d'application, en supprimant les informations superflues telles que la description et l'étiquette de statut (étant donné que les applications dans la liste d'attente ont automatiquement le statut "En attente").

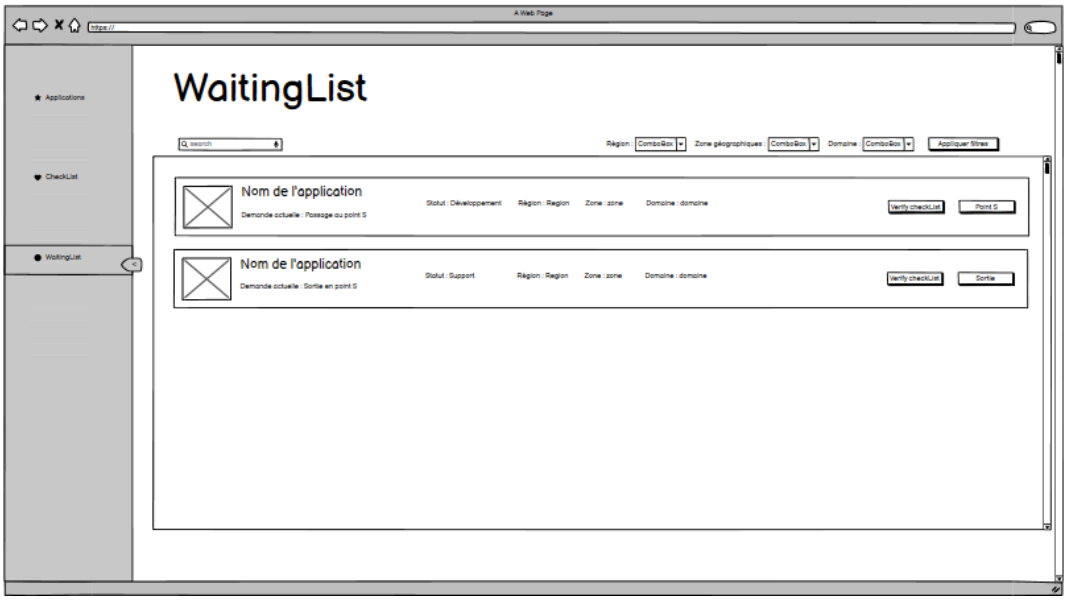

Figure n°? : Première version de la maquette pour la page “WaitingList”

Une fois une application sélectionnée, on peut voir sa checklist. La checklist est représentée sous la forme d’un tableau car je trouve ça très clair et connu de tous donc très intuitif d’utilisation avec en haut bien en évidence les boutons d’options tels que la modification, la suppression et l’ajout.

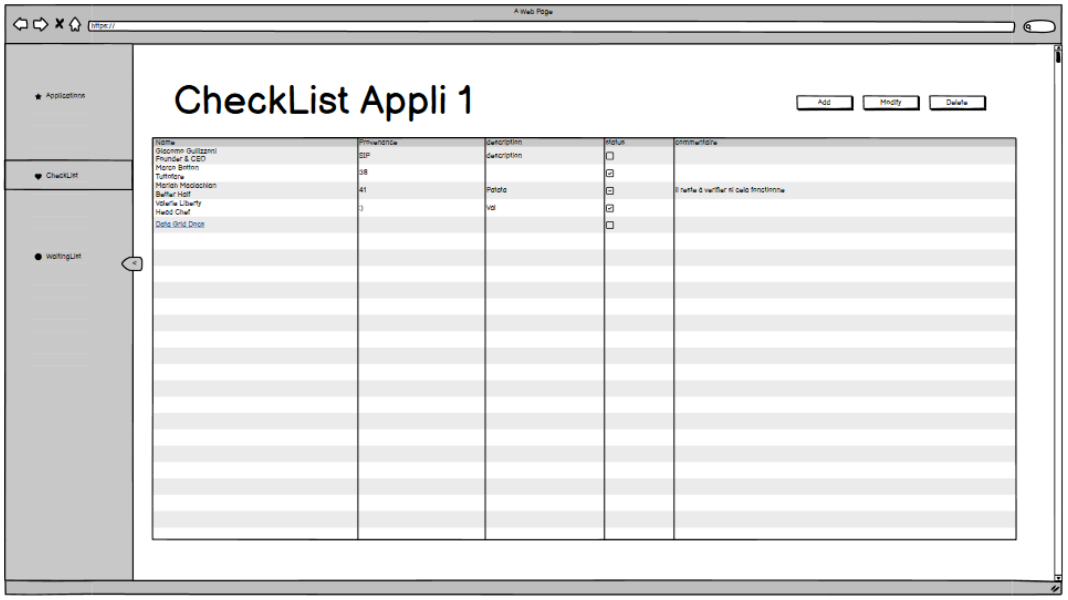

Figure n°? : Première version de la maquette pour la page “CheckList détails”

La dernière page concerne le passage en Point S. J'ai réalisé d'autres maquettes pour des pages plus spécifiques comme l'ajout et la modification, qui sont très conventionnelles. Cette page permet à l'équipe de support de vérifier les critères d'un projet et de les valider ou non. Elle est divisée en deux parties : une pour les critères à valider et une pour les critères déjà validés (bien qu'ils aient été validés, nous souhaitons pouvoir les modifier en cas de changement). Pour cette conception, je me suis inspiré des messages dans Teams, en intégrant des sections repliables pour une meilleure lisibilité.

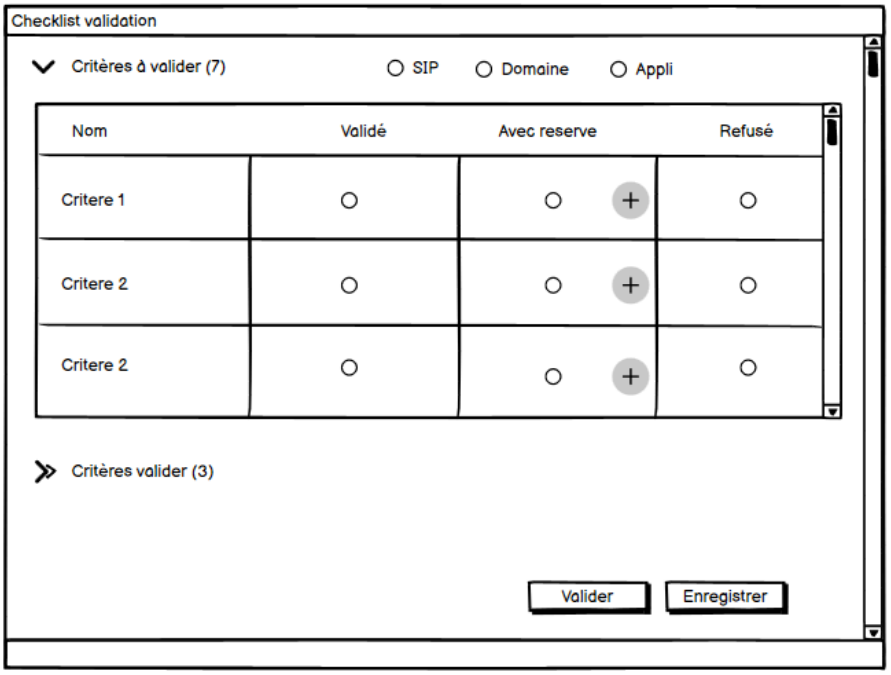

Figure n°? : Première version de la maquette pour la page “CheckList détails”

DEVELOPPEMENT

1) choix technologique entre power app et blazor :

- Présentation 2 technologie
- Avantage et inconvénient des 2 (faire un tableau)
- Pourquoi on à choisi power apps

2) Phase 1 du développement : la v0 : - fonctionnalité à faire
    - Fonctionnement des formules
    - Base de données et liaison avec celle ci
    - Problèmes rencontrés

3) Arrivé du stagiaire + v1 : fonctionnalité en plus
    - nouvelle difficultés

4) retour en entreprise

5) bilan technique

6) bilan global