

Mémoire de fin d'études

en vue de l'obtention du titre de

Bachelier en Informatique
orientation Développement d'applications

Année académique 2024-2025

Migration vers Angular d'une application ASP.NET visant à faciliter l'ouverture de chantier

TECHNORD

Rue de la Lys, 21 7500 Tournai

Présenté par

Zachary VANVLASSELAER



Mémoire de fin d'études

en vue de l'obtention du titre de

**Bachelier en Informatique
orientation Développement d'applications**

Année académique 2024-2025

Migration vers Angular d'une application ASP.NET visant à faciliter l'ouverture de chantier

TECHNORD

Rue de la Lys, 21 7500 Tournai

Présenté par

Zachary VANVLASSELAER



Je tiens à exprimer ma plus sincère gratitude à **M. Loïc Anciaux**, mon maître de stage et **M. Kevin Putzeys**, le maître de stage Noah. L'encadrement, la bienveillance et la gentillesse qu'ils nous accordés durant tout le stage ont permis de créer un environnement de travail sain et stimulant.

Je souhaite également remercier chaleureusement mon promoteur, **M. Nicolas Simar**, pour son accompagnement lors de ces quinze semaines. Ses retours judicieux et constructifs ont grandement aidé à affiner mon travail.

Un grand merci également à toute **l'équipe de Technord**, qui nous a proposé un accueil convivial et chaleureux durant cette expérience. Leur bonne humeur et leur disponibilité ont été un véritable atout.

Enfin, je remercie **Mlle. Léa Bada**, **Mlle. Tatiana Szimjonka** et **M. Etienne Vanvlasselaer** pour leur relecture de ce document.

Tables des matières

Introduction.....	1
1. Présentation de l'entreprise.....	2
1.1 L'entreprise	2
1.2 L'équipe	3
2. Présentation du TFE.....	3
2.1 Le début.....	3
2.2 Contexte sur l'ouverture de chantier	4
2.3 L'ancienne application.....	5
2.4 La problématique	7
2.5 Le projet.....	8
2.4 Les objectifs	8
3. Méthodologie	9
3.1 SCRUM	9
3.2 Jira	10
3.3 Intégration continue	11
4. Analyse.....	12
4.1 Fonctionnalités.....	12
4.1.1 CRUD et droits	13
4.1.2 Gestion des facteurs	14
4.1.3 Gestion des sous-traitants	18
4.1.4 Encodage et réalisation d'un questionnaire.....	18
4.1.5 Gestion des OVC.....	20
4.1.6 Paramètres utilisateurs.....	21
4.1.7 Fonctionnalités bonus.....	22
4.2 L'architecture logicielle	26
4.3 Les technologies utilisées	27
4.4 Les outils utilisés.....	29
5. Réalisation	30
5.1 La base du projet.....	30
5.2 Le frontend	31
5.3 Le backend	32
5.4 L'abstraction	32

5.5 Les défis techniques.....	33
5.5.1 La navigation.....	34
5.5.2 Les questionnaires	36
5.5.3 Le thème dynamique	38
5.6 La Sécurité	39
5.7 Les performances	40
5.8 Compte-rendu.....	41
Conclusion	42

Introduction

La réalisation d'ouverture de chantier, souvent raccourcie en OVC, est le quotidien de toute entreprise de construction. Un chantier est un lieu potentiellement dangereux, où les accidents peuvent être catastrophiques. Il est donc important de précéder chaque chantier par la rédaction d'un document visant à établir les potentiels risques de celui-ci. Ce document précise l'endroit, le cadre temporel, les tâches à accomplir ainsi que les personnes impliquées afin de déterminer au mieux les mesures à mettre en place pour prévenir tout accident. Une bonne ouverture de chantier permet à une entreprise de se justifier juridiquement en cas d'accident, en prouvant qu'elle a fait son maximum pour mettre en place tous les moyens de préventions nécessaires.

Il y a six ans, à la demande d'un client, Technord a commencé la création d'une application visant à faciliter ce processus de rédaction. Cette application, créée en ASP.NET, permet de gérer chaque facteur intervenant dans un chantier de manière individuelle. Ces facteurs peuvent ensuite être rassemblés dans un formulaire afin d'élaborer une ouverture de chantier, qui peut être imprimée pour être utilisée par l'entreprise. Cette application est vendue sous forme de « package » aux clients de Technord. Elle est accompagnée d'une initialisation des données et d'une personnalisation de l'interface.

Malheureusement, le développement de cette application n'a pas toujours été exemplaire pour des raisons de contraintes de temps. Celle-ci présente des bugs, de longs temps de chargements ainsi qu'une interface graphique peu conviviale. De plus, vu qu'elle a été développée à différents moments par différentes personnes, son code est désordonné. Cela a pour conséquence de rendre difficile l'implémentation ou la correction de fonctionnalités. En outre, son architecture, déjà ancienne au moment de sa création, est maintenant dépassée et il existe de bien meilleures alternatives standard dans le monde du développement d'application.

Pour ces raisons, Technord nous a demandé à Noah, mon coéquipier, et à moi, de recréer cette application en partant de zéro. Cette nouvelle application suivra une architecture frontend – backend classique en essayant de garder la base de données la plus intacte possible afin d'assurer une transition simple entre les deux applications pour les clients. L'API sera implémentée avec .NET Framework API 2, tandis que l'interface graphique sera réalisée à l'aide d'Angular et du framework Kendo.

Les objectifs principaux du projet sont clairs : améliorer le visuel de l'application et mettre en place de bonnes pratiques dans la rédaction du code pour permettre une extensibilité facile. Par ailleurs, la nouvelle application ne devra pas présenter de perte d'utilisabilité pour le client. Ainsi, chaque fonctionnalité devra être ré-implémentée. Il est aussi prévu, en fonction de l'avancement du projet, d'en ajouter des nouvelles afin d'attirer de potentiels nouveaux acheteurs. Au final, ce projet devra améliorer non seulement l'expérience des clients mais aussi celle des développeurs.

Pour éviter que Noah et moi ne fassions le même stage et ne nous gênions l'un l'autre, une distinction claire a été faite entre le travail demandé à chacun. Mon coéquipier s'occupera plutôt de la réalisation du formulaire d'ouverture de chantier, tandis que je m'attarderai sur les facteurs gravitant autour de celui-ci. Chacun de nous interagira de manière égale avec chaque composant de l'architecture.

Dans cet écrit, je vais tout d'abord vous présenter l'environnement dans lequel j'ai passé mes quinze semaines de stages. Je vais ensuite entrer plus en détail dans la problématique du projet, ainsi que décrire les méthodologies utilisées afin de le mener à bien. Les différentes fonctionnalités et l'architecture de l'application seront ensuite analysées. Finalement, je passerai en revue la réalisation du projet : la manière dont il a été implémenté et les obstacles rencontrés lors de mon stage.

1. Présentation de l'entreprise

1.1 L'entreprise

Technord est une entreprise de service spécialisée dans les domaines de l'électricité, de l'automatisation, de l'IoT¹, de l'informatique industrielle et de l'intelligence artificielle. Elle s'est donné comme mission d'apporter des solutions fiables et efficaces sur le long terme à ses clients, venant principalement du monde de l'industrie et de la production. Elle travaille pour d'importants clients comme Spa, Leonidas, Heidelberg Materials...

Elle a été créée en 1945 en tant que société spécialisée dans l'électricité générale et l'électromécanique. Trente ans plus tard, Michel Foucart en est devenu le directeur. C'est lui qui va pousser l'entreprise à se diversifier en créant la société Technord Automation. En 1988, le groupe Technord est créé avec deux filiales situées en France et en Roumanie. En 2010, monsieur Foucart passe le flambeau à son fils, Philippe, qui continue l'expansion du groupe en Europe.

L'année passée, les deux actionnaires actuels de Technord, le directeur et sa sœur, Bénédicte, ont décidé de lancer le programme Technord Actionnariat Salarié (TAS). Celui-ci a pour but de fidéliser leurs collaborateurs et d'augmenter leur rentabilité en ouvrant 20% de leur capital aux travailleurs ayant un CDI. C'est alors 60% des employés et ouvriers qui deviennent actionnaires de l'entreprise.

Comme expliqué sur son site web², les valeurs fondamentales de Technord sont son amour et sa confiance envers le client. En effet, l'entreprise accorde beaucoup d'importance à sa relation avec celui-ci. D'autres lignes directrices sont sa vision, son agilité ainsi que son expertise, sans bien évidemment oublier de garder une touche d'humour et de fun.

Elle est composée de plus de 400 employés à travers différents pays d'Europe. Son siège social se situe à Tournai, en Belgique.

¹ Internet of Things

² <https://www.technord.com>

Son chiffre d'affaires dépasse 80 millions d'euros, dégageant plus d'1 million d'euros de bénéfices par an.

Technord a été une des premières entreprises belges à recevoir la certification B Corp. Cette certification est une label international donné aux entreprises qui suivent des normes rigoureuses en termes de performance sociale et environnementale, de transparence et de responsabilité.

1.2 L'équipe

Pour ma part, je vais évoluer sur le site de Seraing. C'est un lieu important, car il est proche des industries avec lesquelles Technord interagit au quotidien. Le bâtiment est divisé en deux étages, répartissant les deux équipes, les deux spécialités de l'endroit : l'automation et l'informatique industrielle. J'ai intégré cette dernière avec deux autres stagiaires, rejoignant une équipe de huit personnes travaillant main dans la main dans un open-space.

L'équipe est dirigée par Kevin Putzeys, responsable projet, qui s'assure que la réalisation des applications se fait en adéquation avec les attentes des clients. Ce sont lui et Alessandro Missoul, un IPS³, qui vont servir de principaux intermédiaires entre les clients et les développeurs. Certains interagissent cependant directement avec les personnes concernées selon les besoins.

Les deux équipes sont entièrement indépendantes et s'occupent d'objectifs et de clients différents. Malgré cela, il leur arrive de collaborer sur certains projets communs.

Les interactions avec les autres départements sont majoritairement limitées au site de Tournai. C'est là-bas que se situent certaines ressources importantes comme le département RH, le département IT, ainsi que les machines virtuelles. Ce site emploie aussi des développeurs avec beaucoup d'expérience dans des domaines spécifiques. Nous pouvons les contacter en cas de problème difficile à résoudre.

2. Présentation du TFE

2.1 Le début

Tout commence il y a six ans, lorsqu'un client de Technord introduit une demande pour la réalisation d'une application aidant à la réalisation d'ouverture de chantiers. En effet, celles-ci peuvent devenir fastidieuses à mettre en place, au vu de tous les facteurs qui interviennent. Sachant que lors des périodes de rush, certaines entreprises doivent en faire plus d'une centaine par mois, il est facile d'imaginer qu'une digitalisation de ce processus ferait gagner du temps.

Pour répondre à ce besoin, l'équipe d'informatique industrielle de chez Technord a développé une application web en ASP.NET, avec l'aide du framework Telerik, qui s'occupe de faciliter la gestion des facteurs et leur assignation à une ouverture de chantier.

³ Ingénieur projet senior

Durant les années qui vont suivre, d'autres clients vont faire des demandes similaires, ce qui va conduire Technord à leur proposer l'application déjà existante sous forme de « package », offrant en plus de la personnalisation et une initialisation des données. Chaque entreprise possède en effet son thème, ses logos et ses cas d'utilisations bien à elles. L'application est donc toujours légèrement modifiée en fonction du client afin de s'adapter à ses particularités. Ces clients vont aussi demander de nouvelles fonctionnalités, qui vont être petit à petit ajoutées au projet initial.

2.2 Contexte sur l'ouverture de chantier

Mais alors, qu'est-ce qu'une ouverture de chantier et en quoi est-ce important ?

Tout d'abord une ouverture de chantier est un document que chaque entreprise désirant commencer un chantier doit rédiger. Ce document a une grande importance légale. En effet, un chantier présente plusieurs dangers possibles qui, dans le pire des cas, peuvent mener à d'importants dommages physiques ou matériels, donnant parfois lieu à un procès.

L'ouverture de chantier a donc pour but d'identifier à l'avance ces différents risques afin de mettre en place toute une série de moyens de prévention pour les limiter au maximum. Une bonne ouverture de chantier permet donc à l'entreprise de se couvrir au regard de la loi en cas d'accident en montrant qu'elle a tout mis en place pour l'éviter.

Lors d'une ouverture de chantier, l'entreprise concernée doit répondre à plusieurs questions : Qui ? Quand ? Quoi ? Où ? Quels risques ? Quels moyens de prévention ?

Le document mentionne donc la société contractante, ainsi que ses sous-traitants. Il est aussi nécessaire de spécifier quels employés participeront au chantier, ainsi que leur rôle. Les sociétés et leurs travailleurs peuvent posséder des agréments, qui sont des documents prouvant la capacité de quelqu'un à faire quelque chose.

Le document doit aussi spécifier quelles sont les dates de début et de fin du chantier et la date de la réunion d'ouverture de chantier. Celles-ci sont accompagnées de la description des tâches à accomplir aux cours des travaux.

Les différents emplacements sur lesquels se déroule le chantier devront aussi être mentionnés, certains de ceux-ci pouvant engendrer des risques. Par exemple travailler près d'un échafaudage demande de faire attention aux chutes d'objets. D'autres risques sont quant à eux indépendants de l'endroit où se réalise un chantier. L'utilisation d'objets coupants ou le travail générant des risques d'incendie sont des cas courants.

Enfin, des moyens de prévention sont mis en place pour répondre à tous ces risques. Certains de ceux-ci sont des actions à exécuter à un moment donné du chantier. Par exemple, travailler sur ou à proximité d'une route demande de mettre en place des cônes et une signalisation adaptée. D'autres nécessitent qu'au moins une personne travaillant sur le chantier possède une agrément spécifique.

Par ailleurs, une mesure de prévention peut exiger le port d'un équipement de protection individuel, abrégé EPI. Le travail près de l'échafaudage mentionné plus tôt nécessite de porter un casque. Ces EPI devront, eux-aussi, être indiqués sur le document.

De plus, il est parfois nécessaire qu'une vigie soit présente sur le chantier. C'est une personne chargée de surveiller un potentiel risque pendant toute la durée des travaux. Cette personne doit être qualifiée et posséder le permis correspondant au risque présent. Un chantier avec risque d'incendie ne peut pas commencer tant que le permis feu n'est pas accordé et présent sur le poste de travail, une vigie feu peut éventuellement être exigé par le permis.

2.3 L'ancienne application

L'ancienne application permet de gérer chaque facteur intervenant dans une ouverture de chantier de manière individuelle. Elle possède donc des pages dédiées aux sociétés, à leur personnel, aux habilitations, aux certifications, aux permis, aux emplacements, aux risques, à leurs moyens de préventions, aux rôles des intervenants et aux EPI.

L'utilisateur peut ajouter, modifier et supprimer des éléments pour chaque facteur.

Tous ces facteurs sont ensuite utilisables dans un grand formulaire reprenant toutes les informations d'une ouverture de chantier. Ce formulaire permet d'en créer ou d'en modifier, afin de les rendre visibles sur des pages de consultation. Ces dernières prennent deux formes : une liste et un planning.

L'utilisateur peut imprimer une ouverture de chantier afin d'obtenir un document sous format A4 qu'il peut partager. Cette impression contient des annexes en plus des informations de l'OVC. L'application facilite l'association de celles-ci à une ouverture de chantier grâce à un service complémentaire appelé « Auto-scan ». Ce service analyse les documents scannés par une imprimante spécifique et essaye de déterminer l'identifiant d'une OVC mentionné sur ceux-ci. Si l'analyse réussit, le document est directement ajouté à l'ouverture de chantier correspondante. Sinon, il est répertorié sur une page séparée qui permet une association manuelle.

Certaines agrégations demandent à être validées par une personne compétente. Cependant, les clients de Technord peuvent travailler avec des centaines d'entreprises différentes. Pour éviter que cette personne ne doive regarder chaque certification de chaque société et chaque habilitation de chaque membre du personnel une à une, l'application propose des pages résumant les agrégations en demande de validation. Ces pages permettent de les valider ou de les refuser et de visualiser les agrégations expirées.

L'application permet de laisser des évaluations aux sociétés qui ont participé à une ouverture de chantier. Un utilisateur peut laisser une note sur différents points, ainsi qu'un commentaire. Ces points sont personnalisables depuis des pages dédiées. Toutes les évaluations sont reprises sur une page de rapport, qui permet de les trier selon les sociétés visées et un intervalle de temps.

D'autres pages de rapport existent, qui permettent de synthétiser des éléments selon les envies de l'utilisateur. Elles fournissent une vue plus globale des habilitations, des certifications et des pointages. Les pointages sont, par ailleurs, également gérés par un autre service.

Afin qu'un utilisateur peu familiarisé avec le site web puisse s'y retrouver, une page de documentation y est présente. Un utilisateur peut ajouter un tutoriel sous différentes formes : des images, des vidéos, des documents PDF. Ceux-ci peuvent ensuite être visionnés par des personnes en recherche d'information.

Plusieurs facteurs d'une ouverture de chantier peuvent posséder des fichiers annexes qui sont, eux-aussi, entièrement gérables depuis le site. Les habilitations, certifications, moyens de préventions et emplacements possèdent des pages spécialisées dans leur gestion.

Par ailleurs, l'application propose une gestion des maintenances et des réservations d'engin. Celles-ci se font à l'aide d'un planning, où un utilisateur peut réserver une ressource.

L'application permet aussi de faciliter l'approvisionnement de produits. Si un utilisateur a besoin d'acheter quelque chose à un fournisseur, il peut créer une fiche d'achat. Cette fiche contient toute une série d'informations essentielles à l'achat comme les références de l'article, la quantité, le fournisseur, etc. Elle doit être ensuite approuvée par un administrateur. Comme pour les agrégations, une page dédiée à la validation existe afin de faciliter le processus.

Un système de questionnaires personnalisables est présent sur le site web. Ils permettent à un membre du personnel de réaliser un test rapide dans le but d'obtenir une habilitation, en répondant à une série de questions. Les entreprises intéressées par cette fonctionnalité créent alors un compte restreint qui donne uniquement accès aux questionnaires. Un responsable sur place s'y connecte via un écran d'accueil dans le bâtiment où travaillent les ouvriers. Ceux-ci peuvent alors s'identifier au moyen de leur nom et de leur prénom afin de réaliser un test pour l'habilitation de leur choix. S'ils réussissent le questionnaire, l'habilitation leur est décernée. Chaque tentative est enregistrée et peut être visionnée sur une page d'historique.

L'application ne permet pas à n'importe qui de se créer un compte. Afin d'avoir accès au site web, il faut qu'un administrateur ajoute un compte à la personne le demandant. L'administrateur, pour le rendre utilisable, doit lui associer des profils, qui sont des collections de droits. En fonction de ceux-ci un utilisateur pourra ou pas faire une action donnée.

Un utilisateur peut cependant modifier ses informations personnelles une fois son compte créé. Il peut changer son nom, son email et son mot de passe mais n'a pas accès à ses droits.

Le site est traduit dans son entièreté en plusieurs langues. Les langues disponibles aux utilisateurs sont gérables dans une page spécialisée. Le contenu des traductions n'est cependant pas modifiable depuis l'interface graphique.

Vu que l'application propose beaucoup de fonctionnalités qui n'intéressent pas spécialement tous les clients, Technord les regroupe par thèmes et propose à chaque client d'en choisir en fonction de leur intérêt. Le coût de l'application est proportionnel à la quantité de fonctionnalités choisies.

2.4 La problématique

L'application a été développée sur plusieurs années, par différentes personnes et avec des technologies déjà relativement anciennes pour l'époque. De plus, en essayant de répondre à certains besoins trop rapidement, elle fut parfois réalisée un peu hâtivement, présentant de ce fait des défauts et devenant moins attirante visuellement, comme le montre la *figure 1*. Le manque de fiabilité s'illustre par des bugs, comme des pop-ups pouvant devenir inaccessibles, ou des temps de chargement excessifs.



Figure 1 : Ancienne page de réalisation d'un test démontrant le peu d'intérêt porté au visuel

Par ailleurs, le processus de création s'étant fait de manière dispersée, le code de l'application est désordonné, rendant l'ajout et la correction de fonctionnalités plus difficile que ce que ça ne devrait l'être. Une bonne illustration de ceci est une demande de correction d'un bug sur la page des permis qui a été introduite lors de mon stage. Cette correction était normalement simple à implémenter et aurait pu être faite en l'espace de deux heures. Malheureusement, le code de cette page n'était pas clairement organisé et tenter d'en changer une partie engendrait des problèmes sur une autre. Il a donc fallu plusieurs jours à la personne chargée de faire cette correction pour la mettre en place.

En outre, l'architecture de l'application, mise en avant dans la *figure 2*, n'est plus à jour par rapport aux standards actuels. Effectivement, le code qui s'occupe de l'UI et celui qui

se charge du business model sont mélangés dans une seule application monolithe en ASP.NET. Cette dernière interagit directement avec la base de données.

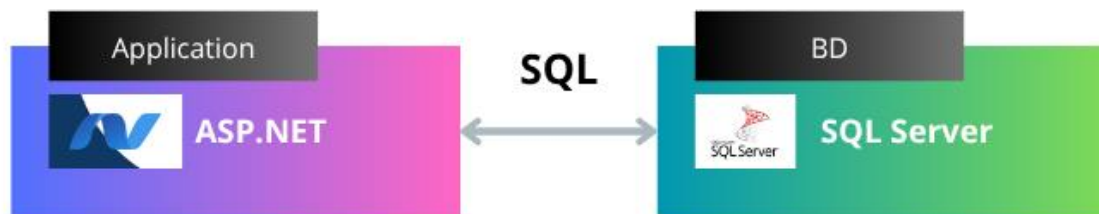


Figure 2 : Architecture de l'ancienne application

Cette manière de fonctionner limite la réutilisabilité ainsi que l'extensibilité de l'application. Qui plus est, elle rend difficile le développement en parallèle et le test de fonctionnalités précises.

Pour toutes ces raisons, l'équipe de Technord a décidé de tout recommencer à partir de zéro, avec de nouvelles technologies et une interface grandement améliorée.

2.5 Le projet

Technord nous a demandé, à mon coéquipier et à moi, de migrer l'entièreté de l'ancienne solution monolithique vers une application composée de deux services comme il est d'usage actuellement : un frontend et un backend.

Le frontend permettra à l'utilisateur d'interagir avec les fonctionnalités de l'application et sera réalisé en Angular à l'aide du framework Kendo.

Le backend servira d'intermédiaire entre l'interface visuelle et la base de données et s'occupera du traitement des requêtes. Il sera réalisé en .NET Framework avec le langage de programmation C#. Il se basera en grande partie sur le code existant, étant donné que la logique métier ne changera pas énormément. Il faudra cependant s'assurer que celui-ci soit correct et efficace, et le modifier si ce n'est pas le cas.

La base de données restera majoritairement inchangée pour permettre une installation simple de la nouvelle solution chez le client.

2.4 Les objectifs

D'abord, il est très important que la totalité des fonctionnalités de l'ancienne solution soit disponibles sur la nouvelle. Le but de la migration est de fournir une amélioration au client. Il est donc hors de question que certaines fonctionnalités manquent à l'appel. Dès lors, l'application sera testée rigoureusement afin de ne pas créer une perte de productions chez les clients.

Ensuite, comme déjà mentionné, la nouvelle application sera un renouvellement visuel. La barre sera donc grandement relevée en ce qui concerne le design de l'interface. La navigation va être entièrement revue afin de fournir une expérience plus intuitive aux utilisateurs. Les pages de gestion vont suivre une structure similaire afin de garder une cohérence dans l'application. Chacune possèdera un tableau central reprenant une liste

d'éléments. Ce tableau sera entouré de boutons permettant de réaliser les différentes actions de la page.

Dans l'ancienne application, pour ajouter ou modifier un élément à un tableau, un formulaire apparaissait directement dans ce dernier. Il a été constaté que ce design n'était pas simple d'utilisation et qu'il engendrait des bugs. La nouvelle application propose une approche nouvelle : les pop-ups. Il est essentiel que ceux-ci restent simples pour ne pas gêner la lisibilité de l'application. Si un formulaire présente trop d'information pour être incorporé à un pop-up, il sera déplacé dans une page à part.

Enfin, il devra être possible de facilement ajouter de nouvelles fonctionnalités au projet, que ça soit dans un futur lointain ou non. Le code devra donc permettre une extensibilité simple. Pour cela, il devra être clair, concis et correctement organisé. Technord va d'ailleurs proposer certains ajouts comme tâches bonus aux stagiaires en fonction de l'avancement du stage. La logique commune devra être généralisée et maintenue en un seul endroit pour permettre des modifications globales simples et rapides.

3. Méthodologie

3.1 SCRUM

Technord utilise la méthodologie SCRUM afin de mener à bien ses projets. Ces derniers sont divisés en plusieurs petites tâches réalisables par une seule personne qui composeront ce que l'on appelle le « Backlog ». La réalisation d'un projet se fait durant des sprints, qui sont des unités de temps arbitraires choisies au début du projet. Dans notre cas, chaque sprint dure deux semaines, ce qui a divisé le stage en sept sprints. La première semaine n'est pas prise en compte, ayant majoritairement servi à tout mettre en place, à analyser la solution existante et à nous familiariser avec l'équipe.

C'est durant une réunion appelée le « Sprint Planning », faite à chaque début de sprint, que l'on attribue les tâches à chacun. Les deux maîtres de stage proposent des tâches intéressantes à faire pour le prochain sprint et les deux stagiaires donnent leur avis sur leur faisabilité. Il est important d'attribuer à chaque tâche un certain nombre de « Story Points », qui sont, elles aussi, des unités arbitraires représentant le temps estimé que prendra sa réalisation.

À la fin d'un sprint, les membres de l'équipe organisent une autre réunion, nommée « Sprint Review », dans laquelle chacun pourra exprimer son ressenti par rapport au travail réalisé. Les points difficiles seront passés en revue ainsi que les raisons expliquant ces difficultés. Les « Story Points » seront aussi comparés à la véritable durée de chaque tâche. Tout ce processus permet d'améliorer le prochain « Sprint Planning », afin d'estimer de manière plus précise le travail qui pourra être accompli.

Technord ne suit cependant pas SCRUM à la lettre. En effet, durant notre stage, nous avons fusionné le « Sprint Planning » et le « Sprint Review » en une seule réunion afin de gagner du temps. De plus, des meetings journaliers sont prévus dans cette méthodologie

afin que chacun détaille son travail. Ceux-ci n'ont pas été mis en place, étant donné que le projet a été développé par seulement deux personnes.

3.2 Jira

En parallèle à cette méthodologie, Technord utilise Jira, une application très populaire permettant la gestion des projets, des incidents et des bugs. Elle se base sur un principe de ticket, correspondant à une tâche SCRUM, et propose plusieurs manières d'organiser ceux-ci.

Tout d'abord, avant même notre arrivée chez Technord, un employé de l'entreprise a passé l'ancienne application en revue afin de créer la majorité des tickets que nous avons dû réaliser au cours de notre stage. Ces tickets ne sont pas restés immuables et il nous est arrivé à plusieurs reprises de les modifier ou d'en ajouter de nouveaux selon les aléas du développement.

Ensuite, à chaque « Sprint Planning », les maîtres de stage ajoutent les tickets au sprint créé sur Jira. Ainsi, ceux-ci sont affichés sur la page « Tableaux » de Jira, qui est la page où Noah et moi allons passer la majorité de notre temps. Celle-ci permet de répartir les tickets du sprint sous forme de différentes colonnes intitulées qui représentent l'état d'avancement du ticket.

Les différentes étapes que suit un ticket sont les suivantes : « Attente d'information », « À faire », « En cours », « À corriger », « En attente de déploiement », « À valider » et « Validé ». Chaque ticket ne passe pas forcément par chaque colonne, mais cette organisation permet d'établir un principe bien pratique : plus un ticket est avancé dans le processus, plus il est important. C'est pour cette raison que « À corriger » est après « À faire », étant donné qu'il est préférable de d'abord faire des corrections avant de se lancer dans quelque chose de nouveau, pour ne pas se disperser. Je vais maintenant détailler rapidement chaque étape.

« Attente d'information » signifie qu'il manque quelque chose au développeur pour pouvoir réaliser la tâche. Cela peut être, évidemment, une information, mais aussi plus rarement lorsqu'il faut attendre la réalisation d'une autre tâche. La majeure utilité de cette colonne est pour les tickets nécessitant de poser des questions aux maîtres de stage, qui ne sont pas disponibles sur le moment même.

« À faire » est relativement simple et directe : ce sont les tickets qui ont été ajoutés au sprint lors du « Sprint Planning » qui n'ont pas encore été réalisés.

« En cours » représente les tickets commencés mais pas encore finis, ce qui permet de tenir au courant nos maîtres de stages de notre avancement.

« À corriger » est pour les tickets qui ont été réalisés, mais qui présentent un manque de fonctionnalité ou des bugs. Ce sont principalement les maîtres de stages qui vont déplacer les tickets dans cette colonne, mais il va nous arriver à Noah et moi de faire de même de plus en plus souvent au cours du stage.

« En attente de déploiement » contient les tickets qui sont finis mais pas encore déployés sur la machine de test. En effet, nos maîtres de stages ne vérifient pas directement nos tickets sur notre environnement de développement. À la place, Noah et moi nous sommes vus assigner une machine virtuelle sur laquelle on déploie notre application comme on la déploierait chez un vrai client. Cela permet aux développeurs de travailler en toute sérénité, sans avoir peur de casser quelque chose en phase de test. Ce système s'appelle l'intégration continue, que je détaillerai dans le chapitre suivant.

« À valider » est la colonne sur laquelle se concentrent principalement nos maîtres de stages. Ce sont les tickets qu'ils vont tester et déplacer dans une autre colonne selon leur degré de satisfaction.

« Validé », finalement, contient tous les tickets qui ont été validés par nos maîtres de stages.

Ce système de différentes étapes n'a pas été mis en place tout de suite, ou en tout cas pas d'une manière aussi détaillée. Nous l'avons implémenté pour répondre à un problème qui s'est posé durant les premières semaines du stage : un manque de test. En effet, Noah et moi étant habitués aux environnements de développement des travaux de groupes pour l'école, nous ne prenions pas assez au sérieux toute la phase de test des fonctionnalités. Dès lors, peu de nos tickets passaient les standards de nos maîtres de stages.

Afin de régler cette problématique, nous avons donc mis en place ce système de colonnes qui présente trois phases de test différentes au cours du cycle de vie d'un ticket. La première phase est lorsque l'on finit une tâche. Noah et moi nous assurons qu'elle fonctionne en respectant nos attentes avant de passer le ticket correspondant en « En attente de déploiement ». Ensuite, lorsque l'on déploie la solution sur la machine virtuelle, on passe tous les tickets de cette colonne en revue de manière plus poussée avant de les faire passer à l'étape suivante. Cela permet une arrivée de tickets relativement raffinés aux maîtres de stages, ce qui rend l'évolution des tickets plus linéaire. Enfin, comme mentionné précédemment, les tickets sont testés une dernière fois par les maîtres de stages, rendant la découverte d'un bug après coup très rare.

3.3 Intégration continue

Comme mentionné plus tôt, nous utilisons le principe d'intégration continue afin de mener à bien notre projet de stage. Cependant, nous avons un peu dévié de l'idée de base, qui est de fournir des versions provisoires de l'application au client tout au long du développement afin qu'il puisse donner ses retours. En effet, ce n'est pas à lui que nous avons présenté directement l'avancement de l'application puisqu'il avait une application déjà existante et fonctionnelle. Apporter autant de fonctionnalités que cette application nous a déjà pris énormément de temps, et les tester individuellement n'aurait rien apporté.

Ce sont nos maîtres de stage qui se sont chargés des retours sur ce que nous avons réalisé en déplaçant les tickets dans les colonnes adéquates et en écrivant des commentaires pour nous guider. Monsieur Putzeys, le maître de stage de Noah et gestionnaire de projet, est la personne en charge de l'ancienne application d'ouverture de chantier depuis plus de cinq ans. Il va aussi souvent à la rencontre des clients, ce qui le rend très apte à connaître leurs attentes et les choses qui vont les bloquer. C'est donc lui qui a joué le rôle du client tout au long de notre stage.

3.4 Gestion du code

Afin de travailler en équipe sur un code commun, nous avons utilisé Git à l'aide de la plateforme Azure Repos créée par Microsoft. Noah et moi implémentions nos fonctionnalités sur la branche qui nous était assignée. Ensuite, plusieurs fois par semaines, lors des déploiements sur la machine virtuelle, nous créions des « Merge Request » afin de rassembler le code sur la branche principale. Si des conflits étaient présents, nous nous rassemblions derrière un même ordinateur afin de les résoudre ensemble. Cela nous permettait d'éviter d'effacer les progrès de l'autre par accident.

Technord ne possède pas de standard de test unitaire pour son code. Les responsables projets sont plus intéressés par le bon fonctionnement des fonctionnalités. Les seuls mécanismes de test mis en place au cours du stage ont donc été ceux du système de colonnes.

Nos maîtres de stages nous ont laissé beaucoup de liberté dans notre travail. Si une analyse ou une recherche documentaire devait être faite, l'étudiant pouvait choisir de la réaliser comme il le voulait. En cas de problème lors de l'implémentation d'une fonctionnalité, nous étions libres d'utiliser les outils que nous voulions afin de rechercher une solution. Si le problème persistait, on pouvait demander de l'aide à l'un de nos maîtres de stage.

Il nous est aussi arrivé plusieurs fois de faire des mini-réunions à quatre, maîtres de stage et étudiants, dans l'open-space si une fonctionnalité demandait de faire des changements conséquents dans la logique ou l'architecture de l'application.

4. Analyse

4.1 Fonctionnalités

Au vu de la taille conséquente de cette application d'ouverture de chantier, Technord a décidé de remettre le travail à deux personnes. L'HELMo ne voulant pas que deux étudiants aient exactement le même stage, les maîtres de stage se sont concertés afin de diviser les fonctionnalités développées par Noah et moi de la manière la plus équilibrée possible. Il a donc été décidé que mon collègue s'occuperait plutôt de la partie centrale de l'application, l'ouverture de chantier en elle-même, tandis que je serais chargé de gérer tous les éléments gravitant autour de cette dernière, comme illustrer dans la *figure 3*.

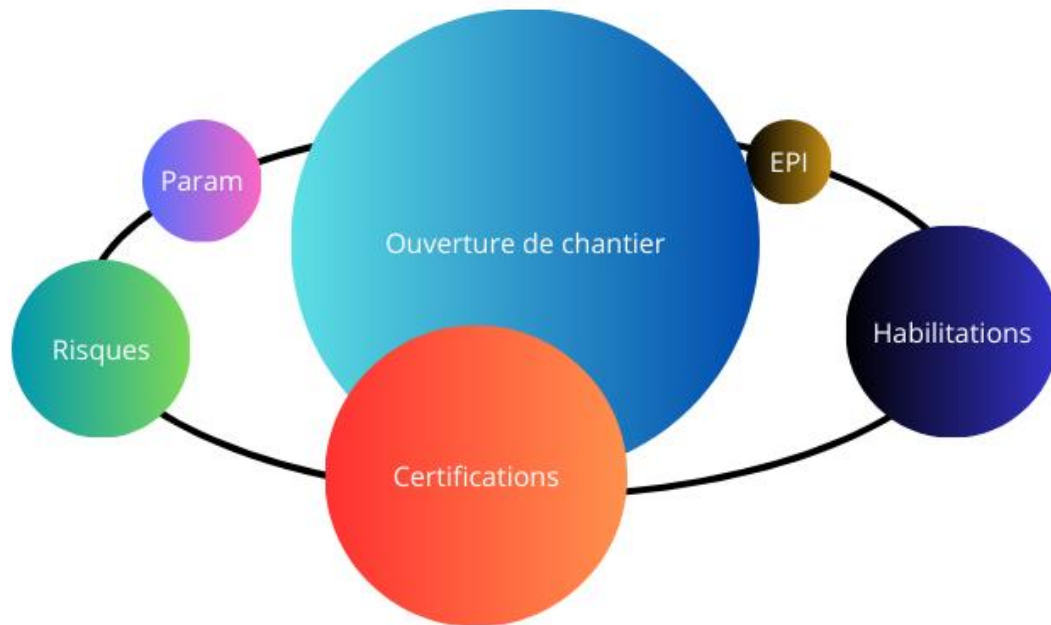


Figure 3 : Illustration de la répartition du travail

Les tâches qui m'ont été attribuées peuvent être divisées en plusieurs sections : la gestion des facteurs, la gestion des sous-traitants, l'encodage et réalisation des questionnaires, une petite partie de la gestion des OVC et la gestion des paramètres utilisateurs. De plus, ayant été rapide dans la réalisation de mon travail, j'ai eu l'occasion de participer à la création de fonctionnalités bonus. Je vais expliquer chacune de ces sections en détail à l'aide de « User Stories » et de commentaires.

4.1.1 CRUD et droits

Avant de rentrer dans l'analyse de ces sections, je vais faire une parenthèse permettant de leur donner plus de contexte. Des fonctionnalités très récurrentes dans toute l'application sont les opérations CRUD⁴. Dans notre cas, elles sont un peu plus poussées que leur définition initiale. En effet, l'application se base sur un système de « Soft Delete » pour la majorité de ses données. Elles ne sont donc pas supprimées définitivement de la base de données et le système les marque comme inactives à la place. Cela permet d'avoir une persistance des données et d'éviter tout accident malheureux.

Dès que les opérations CRUD seront mentionnées dans ce document, elles comprendront en plus la gestion des éléments supprimés, sauf mention explicite. Leurs « User Stories » se présentent de la manière suivante :

- En tant qu'utilisateur, je souhaite pouvoir ajouter un élément
- En tant qu'utilisateur, je souhaite pouvoir voir tous les éléments à ma disposition.
- En tant qu'utilisateur, je souhaite pouvoir modifier un élément.

⁴ Create Read Update Delete : Créer, lire, mettre à jour et supprimer

- En tant qu'utilisateur, je souhaite pouvoir supprimer un élément.
- En tant qu'utilisateur, je souhaite pouvoir voir les éléments supprimés afin de pouvoir en restaurer un.
- En tant qu'utilisateur, je souhaite pouvoir restaurer un élément afin de pouvoir l'utiliser de nouveau.

Il est aussi important de noter que l'application dans son entièreté repose sur un système de droits. Tout le monde n'a pas le droit de faire ce qu'il veut. Chaque action a un droit qui lui est associé et seules les personnes ayant ce droit peuvent la réaliser. Dans la majorité des cas, le fait de pouvoir voir les éléments supprimés et donc les restaurer est réservé aux administrateurs, qui sont les seuls possédant les droits nécessaires.

Cependant, l'application permet aussi de gérer les droits associés à chaque d'utilisateur. Il est donc tout à fait possible qu'un utilisateur lambda puisse restaurer un élément si un administrateur le décide. Aucune action n'est donc réellement réservée aux administrateurs, à l'exception de celles concernant la personnalisation de l'application que j'expliquerai plus tard. Je vais dès lors généraliser tout au long de ce chapitre la personne faisant l'action à « utilisateur » dans les « User Stories ».

4.1.2 Gestion des facteurs

Comme mentionné plus tôt, une ouverture de chantier dépend de beaucoup de facteurs différents, susceptibles de varier. Il est donc très important de laisser la possibilité aux utilisateurs de gérer ceux-ci. Chaque facteur et chaque élément qui peut leur être associé a donc une page désignée permettant au minimum les opérations CRUD.

Bien évidemment, la gestion des facteurs ne se limite pas à ces simples opérations, la plupart des pages proposant des fonctionnalités supplémentaires. Je vais donc détailler celles-ci et contextualiser davantage chaque facteur.

La gestion des paramètres n'est pas vraiment ce à quoi on s'attend lorsque l'on parle de paramètres d'application (*figure 4⁵*). C'est plutôt une liste d'options possibles que l'on peut choisir pour une famille donnée de paramètres (*figure 5*). Cette fonctionnalité permet donc de définir par exemple toutes les tâches ou toutes les familles de risque.

⁵ Source : <https://blog.prototypr.io/settings-ui-design-inspiration-85b23aafb3e6>

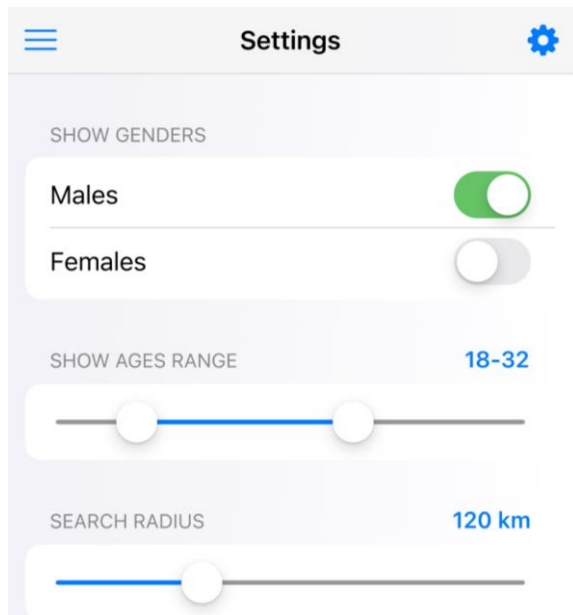


Figure 4 : Illustration d'une page de paramètre habituelle

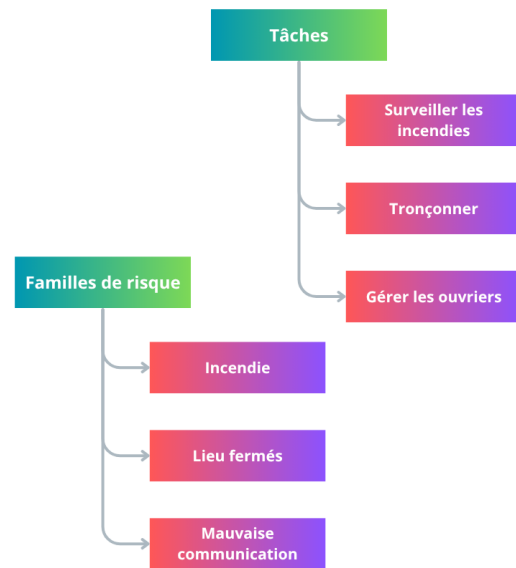


Figure 5 : Représentation des paramètres sur notre application

Pour cette page, on peut ajouter comme « User Story » :

- En tant qu'utilisateur, je souhaite pouvoir changer la famille de paramètre actuelle afin que les opérations que je fais sur les paramètres s'appliquent à celle-ci.

La gestion des types d'habilitation et des types de certification sont similaires en tout point, mise à part la chose à laquelle elles s'appliquent. Il ne s'agit pas ici de venir gérer directement les agrégations d'une personne ou d'une société, mais plutôt leur type. Une agrégation d'un certain type héritera de toutes ses propriétés. Ces propriétés spécifient notamment si une agrégation doit être validée ou non ainsi que sa durée de validité. Cette durée peut prendre deux formes : une date de fin fixe ou une durée en mois. Toute modification d'un type doit se refléter dans les agrégations héritant de lui. Voici donc les « User Stories » à ajouter :

- En tant qu'utilisateur, lorsque je décide qu'un type d'agrégation ne demande plus de validation, je souhaite que chaque agrégation ayant hérité ou héritant de ce type soit automatiquement validée.
- En tant qu'utilisateur, lorsque je décide qu'un type d'agrégation doit maintenant être validée, je souhaite que chaque agrégation existante ayant hérité de ce type reste validée, mais que les nouvelles agrégations demandent une validation.
- En tant qu'utilisateur, lorsque j'ajoute une fin de validité à un type d'agrégation, je souhaite que chaque agrégation héritant ou ayant hérité de ce type change sa fin de validité en accordance et la rende impossible à modifier.
- En tant qu'utilisateur, lorsque je retire une fin de validité à un type d'agrégation, je souhaite que chaque agrégation existante ayant hérité de ce type garde sa fin de validité mais que celle-ci puisse maintenant être changée.

La gestion des risques et des moyens de prévention est la seule exception à « une page par facteur » étant donné qu'ils sont intrinsèquement liés. En effet, un risque implique automatiquement des moyens de prévention, vu qu'il ne serait pas cohérent de commencer une ouverture de chantier avec un risque auquel on ne peut pas remédier. Il est aussi possible de lier des habilitations, des certifications et des permis à un moyen de prévention. Ainsi, lorsqu'un risque est présent sur une ouverture de chantier, chaque moyen de prévention associé à ce risque devra voir ses agrégations correspondre à celles d'un membre du personnel ou d'une société. Je vais donner un exemple afin de mieux illustrer mes propos. Imaginons un scénario simple avec un risque unique illustré par la *figure 6* :



Figure 6 : Agrégations liées à un risque

Si une ouverture de chantier présente le risque « Chute d'objets », il est alors impératif qu'au moins une des personnes travaillant dessus ait l'habilitation « Levage / Elingage » et qu'une société participant au chantier possède la certification « Balisage ». En ce qui concerne les permis, ils sont associés à une ouverture de chantier de manière globale et journalière.

Par ailleurs, chaque risque est lié à une famille de risques, permettant de mieux les organiser. De plus, on peut associer des fichiers à un moyen de prévention. Les « User Stories » suivantes doivent donc être prises en compte :

- En tant qu'utilisateur, je souhaite pouvoir changer la famille de risque actuelle afin de pouvoir gérer les risques venant de cette famille.
- En tant qu'utilisateur, je souhaite pouvoir associer des habilitations, des certifications ou des permis à un moyen de prévention afin que ceux-ci soient pris en compte dans une ouverture de chantier présentant le risque auquel le moyen de prévention est associé.

La gestion des emplacements et la gestion des permis se différencient des précédentes avec leur représentation sous forme d'arbre. Chaque emplacement ou permis peut posséder un emplacement ou permis enfant. Leurs pages nécessitent donc quelques adaptations pour que l'utilisateur puisse les utiliser en toute facilité :

- En tant qu'utilisateur, je souhaite pouvoir ajouter un permis/emplacement enfant à un autre afin de créer une structure sous forme d'arbre.

- En tant qu'utilisateur, je souhaite pouvoir voir les éléments enfant d'un permis/emplacement afin de visualiser la structure.
- En tant qu'utilisateur, je souhaite pouvoir étendre l'entièreté de la structure afin de pouvoir voir tous les permis/emplacement enfants.
- En tant qu'utilisateur, je souhaite pouvoir réduire toute la structure afin de ne voir que les permis/emplacements du premier niveau.

On peut aussi associer des moyens de prévention à un emplacement. Ainsi, l'ajout d'un emplacement à une ouverture de chantier lui lie tous ses moyens de prévention, de la même manière qu'un risque le fait. La page permettant cette association est un peu particulière. En effet, lorsqu'on ajoute un moyen de prévention à un emplacement, on ne fait que les lier ensemble. Ainsi, à l'inverse de la majorité des pages, supprimer ce lien l'efface définitivement de la base de données. Il n'y a donc pas d'opération de restauration possible ni de visualisation des éléments supprimés.

Par ailleurs, il est aussi possible d'attacher des fichiers à un emplacement depuis une page dédiée.

Les gestions des catégories et des types d'évaluation ressemblent un peu à celles des risques et des moyens de prévention dans leur lien fort. Ces gestions ont été séparées en deux pages par soucis de clarté mais auraient très bien pu se faire en une seule. Elles permettent de définir les critères d'une évaluation donnée à une société ayant participé à une OVC. Un critère est représenté par un type d'évaluation, qui est lui-même associé à une catégorie d'évaluation, ajoutant la « User Story » suivante :

- En tant qu'utilisateur, je souhaite pouvoir voir les types d'évaluations associés à une catégorie d'évaluation.

La gestion des EPI et la gestion des rôles intervenants restent particulièrement simples, excepté l'association d'une image à un EPI, ce qui donne la « User Story » qui suit :

- En tant qu'utilisateur, je souhaite pouvoir visualiser l'image que j'ai associée à un EPI

Toutes les gestions mentionnées dans ce chapitre ont un point commun : l'association à un site. En effet, une entreprise possédant l'application peut être composée de plusieurs sites, chacun ayant besoin de réaliser des ouvertures de chantier. Cependant, chaque site ne fonctionne pas forcément de la même manière et n'utilise donc pas forcément les mêmes facteurs. Il est donc fondamental de permettre à un utilisateur de dire quel facteur est présent sur quel site. Sur l'ancienne application, un utilisateur ne pouvait associer que deux valeurs possibles pour le site d'un facteur : le site de l'utilisateur qui s'est connecté ou « global » pour tous les sites. Cette gestion simple fonctionne correctement lorsque le client possède peu de sites, mais pose problème si ce nombre devient trop grand. La nouvelle application étant prévue en partie pour un client ayant jusqu'à cinquante sites différents, il a été essentiel de revoir partiellement ce système.

Il faut savoir qu'il existe plusieurs niveaux de droits sur l'application, dont deux qui nous intéressent ici : les administrateurs locaux et les administrateurs globaux. Dans la nouvelle application, les premiers gèrent le site de la même manière qu'auparavant, tandis que les seconds peuvent voir et gérer les facteurs pour n'importe quel site.

4.1.3 Gestion des sous-traitants

Plusieurs sociétés peuvent participer à une ouverture de chantier, en étant la société contractante ou une des sociétés sous-traitantes. Il faut donc être capable de gérer toutes les sociétés avec lesquelles le client est habitué à travailler, ainsi que leur personnel. Chaque société et chaque membre du personnel peuvent posséder des agréments, qui peuvent elles-mêmes posséder des annexes. Ces liens sont montrés dans la figure 7.

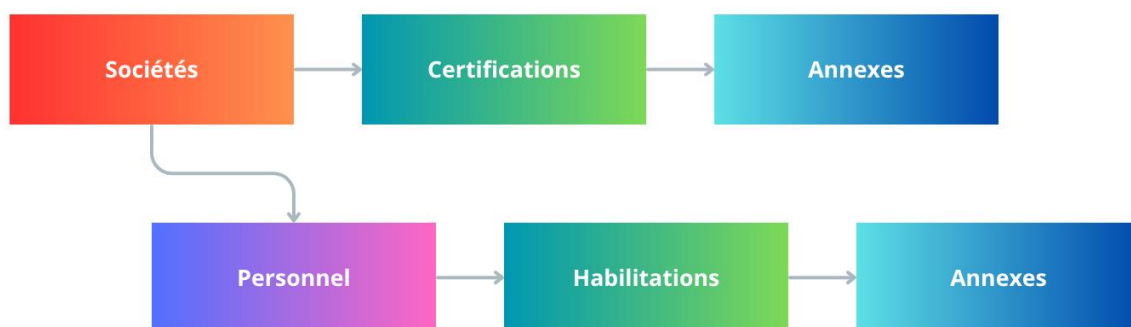


Figure 7 : Liens entre les sociétés et les éléments qui en dépendent

Il doit être possible d'effectuer les opérations CRUD présentées plus tôt sur chacun des éléments du schéma. De plus, lors de l'ajout ou de la modification d'une agrégation, certaines données dépendent de son type, comme montré par ces « User Stories » :

- En tant qu'utilisateur, lorsque j'ajoute ou modifie une agrégation et que son type possède une fin de validité, je souhaite que cette dernière soit reflétée dans l'agrégation actuelle et ne soit pas modifiable.
- En tant qu'utilisateur, lorsque j'ajoute une agrégation, je souhaite que sa validité corresponde à la propriété « validation nécessaire » de son type.
- En tant qu'utilisateur, lorsque je modifie une agrégation, pour laquelle je n'ai pas le droit de changer la validité et qu'elle demande une validation, je souhaite qu'elle soit invalidée.

4.1.4 Encodage et réalisation d'un questionnaire

Une autre fonctionnalité importante proposée par l'application est celle des questionnaires. Il doit être possible pour un utilisateur de s'identifier en tant que membre d'une société, qu'il en fasse partie ou non, et de répondre à un questionnaire de son choix. Plusieurs « User Stories » doivent être ainsi prises en compte :

- En tant qu'utilisateur membre d'une société sous-traitante, je souhaite m'identifier à l'aide de mon nom et prénom afin de répondre à un questionnaire.

- En tant qu'utilisateur ne faisant pas partie du système, je souhaite m'identifier à l'aide de mon nom et prénom afin de répondre à un questionnaire et, qu'à la fin de celui-ci, je sois ajouté comme membre du personnel de la société de mon choix.
- En tant qu'utilisateur, je souhaite pouvoir visionner un document explicatif au début du questionnaire et confirmer ma lecture effective de celui-ci afin de continuer le test.
- En tant qu'utilisateur, je souhaite pouvoir visionner une image ou une vidéo associée à une question afin d'avoir plus de contexte sur celle-ci.
- En tant qu'utilisateur, je souhaite pouvoir répondre à chaque question du test afin d'essayer de le réussir.
- En tant qu'utilisateur, je souhaite pouvoir valider un questionnaire lorsque je l'ai fini afin de visualiser ma note et de potentiellement obtenir l'habilitation. .
- En tant qu'utilisateur, si je rate un questionnaire, je souhaite pouvoir de nouveau répondre aux questions que j'ai ratées.

Il est essentiel de noter que, pour qu'une habilitation soit décernée à la personne répondant au questionnaire, celui-ci obtienne une note de 100%. Il existe pour l'instant trois types de questions possibles : à choix unique, à choix multiple et libre. Les réponses à ces questions sont évaluées automatiquement par le système et ne nécessitent pas l'intervention d'une tierce personne. Les questions libres sont considérées comme toujours correctes et sont utiles lorsqu'elles sont mises en lien avec l'historique.

En effet, chaque tentative sur un questionnaire est enregistrée dans le système. Les réponses de l'utilisateur y sont détaillées. Il existe ainsi une page dédiée à l'historique qui permet de visionner ces tentatives, engendrant les « User Stories » suivantes :

- En tant qu'utilisateur, je souhaite pouvoir avoir une vue d'ensemble sur toutes les tentatives de questionnaire afin de chercher celles qui m'intéressent
- En tant qu'utilisateur, je souhaite afficher les détails d'une tentative afin de voir les réponses de l'utilisateur, notamment aux questions libres.

Il est aussi nécessaire de considérer l'encodage d'un questionnaire. La structure initiale de ce dernier est relativement simple. Chaque questionnaire est composé d'une ou plusieurs questions, contenant une ou plusieurs réponses prédéfinies (sauf pour les questions libres). Il faut donc être capable de faire les opérations CRUD sur chacun de ces trois composants.

Cependant, avec l'arrivée de l'historique, une question s'est posée : que faire si le questionnaire est modifié après avoir déjà reçu des réponses ? Que se passe-t-il lorsque quelqu'un modifie la liste de réponses possibles à une question après que j'y aie répondu précédemment ? L'ancienne implémentation du système d'historique n'avait que partiellement pris ce problème en compte et il était donc possible d'afficher des données erronées.

Vu que la réalisation de ces fonctionnalités est arrivée plus ou moins au milieu du stage, l'équipe de Technord m'a fait confiance pour la création d'un tout nouveau système. On

m'a donc demandé de repartir de zéro et de construire une architecture permettant de créer plusieurs versions immuables d'un questionnaire ainsi que de traduire ces versions en plusieurs langues.

Dorénavant, dans la nouvelle architecture, lorsque l'utilisateur veut créer un questionnaire, il doit d'abord créer une version, à partir de rien ou d'une version déjà existante. Cette version devient alors un « brouillon », que l'utilisateur peut modifier à sa guise. Il peut aussi y ajouter des langues, ce qui l'oblige à traduire tout le questionnaire. Lorsque l'utilisateur a fini la création de sa version, il la valide, ce qui la rend impossible à modifier. Ce système implique les « User Stories » suivante.

- En tant qu'utilisateur, je souhaite créer une nouvelle version « brouillon » d'un questionnaire afin de modifier celui-ci.
- En tant qu'utilisateur, lorsque je crée une nouvelle version à partir d'une autre, je souhaite qu'elle en soit une copie.
- En tant qu'utilisateur, je souhaite pouvoir réaliser les opérations CRUD sur les questions d'une version « brouillon » et leurs réponses.
- En tant qu'utilisateur, je souhaite pouvoir changer le document PDF associé à une version « brouillon ».
- En tant qu'utilisateur, je souhaite pouvoir valider une version afin de la rendre disponible.
- En tant qu'utilisateur, je souhaite annuler la création de la version actuelle afin de la supprimer.
- En tant qu'utilisateur, je souhaite choisir la version actuelle du questionnaire, afin que ça soit la version utilisable pour passer un test.
- En tant qu'utilisateur, je souhaite pouvoir ajouter une langue au questionnaire afin de le traduire.
- En tant qu'utilisateur, je souhaite pouvoir choisir la langue du questionnaire que je veux modifier afin que les changements que j'apporte aux énoncés ou aux descriptions des questions/réponses ne s'appliquent qu'à cette langue.

Il est important de noter que certains attributs d'une question ou d'une réponse sont dépendants de la langue, tandis que d'autres non. Des informations comme la validité, l'ordre ou le type sont communs aux questions/réponses pour toutes les langues, tandis que les énoncés et les descriptions sont spécifiques à une association langue – question/réponse.

4.1.5 Gestion des OVC

Afin d'alléger un peu la charge de travail de Noah, j'ai également travaillé sur des parties de la gestion des ouvertures de chantier qui étaient directement en lien avec la gestion des facteurs. J'ai ainsi réalisé les pages s'occupant des permis et des évaluations liés à une OVC, ainsi que celle gérant les « Auto-scan » ratés.

Pour les évaluations, il est possible de choisir une des sociétés ayant participé à l'ouverture de chantier et de laisser une note pour chaque critère défini dans la gestion des facteurs. Les « User Stories » suivantes sont à prendre en compte :

- En tant qu'utilisateur, je souhaite pouvoir sélectionner une société ayant participé à l'ouverture de chantier afin d'afficher toutes les évaluations qui lui ont été associées lors de l'OVC.
- En tant qu'utilisateur, je souhaite pouvoir effectuer les opérations CRUD sur les évaluations d'une société.
- En tant qu'utilisateur, je souhaite que si je modifie une évaluation, sa date soit mise à jour.

La modification d'une évaluation est unique en son genre. En effet, pour éditer une évaluation, un utilisateur doit non seulement posséder le droit nécessaire, mais aussi en être le créateur.

Pour les permis, un utilisateur peut lier un permis de 1^{er} niveau (sans parent) à une ouverture de chantier pour une durée d'un jour. Il peut ensuite spécifier quels permis enfants sont utilisés durant cette journée en leur associant une vigie. Les « User Stories » de cette fonctionnalité sont les suivantes :

- En tant qu'utilisateur, je souhaite pouvoir réaliser les opérations CRUD sur les permis liés à une OVC.
- En tant qu'utilisateur, je souhaite pouvoir réaliser les opérations CRUD sur les vigies liées à un permis
- En tant qu'utilisateur, je souhaite pouvoir dupliquer un permis déjà lié à une OVC ainsi que ses vigies afin de l'assigner à un autre jour.
- En tant qu'utilisateur, je souhaite pouvoir imprimer un permis lié à une ouverture de chantier.

Comme pour les moyens de prévention liés à un emplacement, les opérations CRUD mentionnées dans ces « User Stories » ne comprennent pas la restauration et la visualisation des éléments supprimés.

Pour les « Auto-scan », il ne m'a pas été demandé de modifier le service à part qui s'occupe d'analyser les documents scannés. Je me suis uniquement occupé de permettre aux utilisateurs d'assigner un fichier dont l'analyse a échoué à une ouverture de chantier, ce qui donne les « User Stories » suivante :

- En tant qu'utilisateur, je souhaite pouvoir visualiser chaque fichier dont l'association à une OVC a échoué, ainsi que la cause de l'échec.
- En tant qu'utilisateur, je souhaite assigner un fichier à une ouverture de chantier, afin qu'il soit disponible dans les annexes de cette dernière et disparaisse de la liste des échecs.
- En tant qu'utilisateur, je souhaite pouvoir supprimer un fichier de la liste des échecs

4.1.6 Paramètres utilisateurs

Un utilisateur peut modifier ses informations personnelles et son mot de passe. Ces opérations se font sur la même page mais avec un questionnaire différent pour éviter à l'utilisateur de devoir réécrire son mot de passe à chaque fois qu'il modifie son profil. Un changement de mot de passe demande à l'utilisateur d'écrire le nouveau deux fois. Les « User Stories » suivantes sont à prendre en compte :

- En tant qu'utilisateur, je souhaite pouvoir modifier mon nom, mon prénom, mon e-mail et mon numéro d'identification afin de mettre à jour ces informations
- En tant qu'utilisateur, je souhaite changer mon mot de passe afin de pouvoir l'utiliser lorsque je me connecte.

Un utilisateur peut aussi changer sa langue via un pop-up spécifique. Ce changement est persistant et permet de changer la traduction de tous les textes de l'application.

Une fonctionnalité supplémentaire qui n'a pas pu être aboutie est la signature électronique. Un utilisateur pourrait enregistrer une signature qu'il dessine à l'aide de sa souris ou d'un stylo et l'utiliser pour signer des documents. Cette fonctionnalité n'a pas pu être entièrement implémentée durant le période de stage car les cas d'utilisations exactes n'étaient pas déterminés. Il m'a quand même été demandé de mettre en place le système d'association de signature à un utilisateur. Celle-ci se fait donc sur la page de modification des informations personnelles via un pop-up annexe.

4.1.7 Fonctionnalités bonus

Ayant fini le scope initial de mon stage plus vite que prévu, j'ai eu l'opportunité de travailler sur des fonctionnalités bonus, n'existant pas dans l'ancien système. Cette partie du travail a été le moment le plus agréable pour moi. Vu que rien n'existait déjà, j'ai dû réfléchir moi-même à la manière d'implémenter ce qui m'était demandé et c'est quelque chose qui me plaît beaucoup.

Ces fonctionnalités sont orientées autour de deux grands thèmes : la personnalisation du site et l'authentification.

Comme mentionné précédemment, ce projet est vendu aux clients sous forme de « package » qui contient la personnalisation du site en fonction du client. Cette personnalisation devait être faite manuellement par les développeurs, rendant le travail parfois fastidieux. Mon job a donc été de rendre cette personnalisation plus facile à mettre en place, en laissant un administrateur Technord modifier le site directement depuis l'interface graphique. Les pages de personnalisation sont réservées uniquement aux développeurs Technord, aucun droit n'y donne donc accès.

Tout d'abord, j'ai créé une page de modification du thème du site. Chaque entreprise a sa palette de couleurs bien à elle et ses différents logos. Il est donc important que l'application reflète cette charte graphique. J'ai donc fait en sorte que les couleurs ainsi que les images du site soient dynamiques, donnant les « User Stories » suivantes :

- En tant qu'administrateur Technord, je souhaite pouvoir visualiser une liste de toutes les couleurs du site afin de choisir celle je veux modifier.

- En tant qu'administrateur Technord, je souhaite pouvoir visualiser une liste de toutes les images modifiables du site afin de choisir celle que je veux modifier.
- En tant qu'administrateur Technord, je souhaite avoir une « Preview » des modifications que j'apporte aux couleurs et aux images afin de me faire une idée du résultat sans nécessairement devoir les appliquer.
- En tant qu'administrateur Technord, je souhaite pouvoir changer une couleur ou une image et que ce changement se reflète dans la « Preview ».
- En tant qu'administrateur Technord, je souhaite pouvoir appliquer mes changements à l'entière du site afin que ceux-ci soient visibles par tous les utilisateurs.
- En tant qu'administrateur Technord, je souhaite pouvoir importer et exporter un jeu de couleurs afin de synchroniser deux applications différentes facilement.

Ensuite, j'ai mis au point une page de gestion des langues et des traductions de l'application. Les traductions fonctionnent avec un système de clé. Ces clés sont utilisées dans le code dès qu'un texte est présent. L'application remplace cette clé par la traduction associée en fonction de la langue actuelle. Il existait déjà un système de gestions des langues rudimentaires dans l'ancien site, mais celui ne permettait que d'activer ou de désactiver une langue. Pour traduire une nouvelle langue du site, il était donc nécessaire qu'un développeur insère toutes les traductions dans la base de données manuellement. J'ai donc ajouté à l'existant une page de gestions de traductions. Cette page affiche chaque clé existante dans le site et les traductions associées pour une langue donnée. Les « User Stories » liés à ces fonctionnalités sont donc :

- En tant qu'administrateur Technord, je souhaite voir une liste de toutes les clés du site ainsi que leur traduction dans la langue choisie.
- En tant qu'administrateur Technord, je souhaite pouvoir filtrer les clés selon leur traduction et ajouter des langues aidantes afin de faciliter le processus.
- En tant qu'administrateur Technord, je souhaite pouvoir modifier une traduction afin qu'elle soit visible par tous les utilisateurs du site.
- En tant qu'administrateur Technord, je souhaite pouvoir importer et exporter une liste de traductions afin de synchroniser deux applications différentes facilement.

Par ailleurs, lorsqu'un client achète l'application, il n'est pas obligé de prendre toutes ses fonctionnalités. L'application doit donc être capable de gérer l'accès du client à certaines fonctionnalités en fonction de son achat. Ceci est implémenté avec un système de licence, qui est une clé chiffrée déterminant les accès de tous les utilisateurs. En effet, pour certains clients, la base de données est déployée localement. Il ne faut donc pas que les informations sur les fonctionnalités autorisées soient stockées en clair. Pour remédier à cela, l'application formate et chiffre ces informations pour les représenter sous forme de suite de caractère, la licence. A l'initialisation de l'application, celle-ci déchiffre la licence stockée en base de données afin de savoir ce que l'utilisateur peut faire.

Dans l'ancien système, cela était fait avec une application annexe élaborée uniquement pour créer une licence, avant de l'envoyer au client. Pour faciliter ce procédé, j'ai permis à un administrateur Technord de directement modifier la licence depuis l'interface graphique avec les « User Stories » suivantes :

- En tant qu'administrateur Technord, je souhaite pouvoir ajouter ou enlever des fonctionnalités associées à la licence de l'application.
- En tant qu'administrateur Technord, je souhaite pouvoir donner une date de fin de validité à la licence de l'application afin que plus aucune fonctionnalité ne soit accessible après celle-ci.
- En tant qu'administrateur Technord, je souhaite pouvoir appliquer mes changements afin que ceux-ci fassent effet pour tout utilisateur du site.

Enfin, j'ai ajouté une page d'import massif. En effet, chaque client possède des données qui lui sont spécifiques. Ils n'ont pas tous les mêmes emplacements, types d'habilitation, familles de risques, etc. Les données de ce style doivent être présentes dès le premier jour de l'utilisation de l'application et sont fastidieuses à ajouter manuellement. La page d'import massif permet donc d'ajouter une grande quantité de données à l'aide d'un fichier Excel pouvant facilement être complété. Cette page est donc composée des « User Stories » :

- En tant qu'administrateur Technord, je souhaite pouvoir choisir un type de données à importer afin d'en voir les spécificités.
- En tant qu'administrateur Technord, je souhaite pouvoir télécharger le template d'un fichier Excel à remplir afin d'ajouter des données.
- En tant qu'administrateur Technord, je souhaite avoir une liste détaillant chaque colonne du template Excel afin de mieux comprendre quelles données je dois y placer.
- En tant qu'administrateur Technord, lorsque j'ai soumis un fichier Excel dans le but d'ajouter des données, je souhaite avoir une « Preview » me montrant les données qui vont être ajoutées.
- En tant qu'administrateur Technord, je souhaite pouvoir appliquer les changements montrés dans la « Preview ».
- En tant qu'administrateur Technord, si l'application des changements échoue car certaines données sont erronées, je souhaite avoir un rapport détaillé reprenant chaque problème rencontré.

L'ancien projet proposait une seule méthode d'authentification, les utilisateurs locaux. Ceux-ci sont ajoutés par les administrateurs du site et peuvent se connecter à l'aide de leur identifiant et de leur mot de passe. Cependant, certains nouveaux clients ont demandé des moyens d'authentification un peu plus avancés, notamment le SSO⁶ et le MFA⁷.

⁶ Single Sign-On : Permet à un utilisateur de se connecter à plusieurs applications avec les mêmes identifiants

⁷ Multi-Factor Authentication : Demande à l'utilisateur d'utiliser plus qu'un mot de passe pour se connecter, comme un code envoyé à un numéro de téléphone

Pour le SSO, beaucoup d'entreprises possèdent un « Active Directory », ou AD, qui recense les membres de leur personnel et leurs données. Ces membres sont organisés hiérarchiquement sous forme de groupes. L'idée derrière cette nouvelle méthode d'authentification est de permettre à un utilisateur de se connecter avec ses identifiants repris dans l'« Active Directory ». Il devrait aussi être possible d'associer les groupes de l'AD à des profils locaux de l'application, ce qui permet de définir les droits de ces utilisateurs spéciaux. Dans le cadre de mon stage, je me suis concentré dans l'implémentation de l'authentification SSO à l'aide de Microsoft Entra ID, qui est le service de gestion d'identité et d'accès dans le Cloud de Microsoft. On peut cependant imaginer que Technord développera dans le futur une compatibilité avec d'autres systèmes du même style. J'ai donc implémenté les « User Stories » suivantes :

- En tant qu'utilisateur, lorsque que je modifie un profil, je souhaite avoir une liste de tous les groupes présents dans l' « Active Directory » associé au site afin de les associer au profil actuel.
- En tant qu'utilisateur, je souhaite que si un anonyme essaie de se connecter au site avec ses identifiants AD et qu'aucun de ses groupes n'est associé à un profile, sa connexion soit refusée
- En tant qu'anonyme, je souhaite pouvoir utiliser mes identifiants de l'AD afin de me connecter au site.
- En tant qu'anonyme, je souhaite que si je me connecte pour la première fois en utilisant mes identifiants de l' « Active Directory », un compte me soit créé avec les données qui y sont présentes.

Il est important de noter qu'un utilisateur venant de l'extérieur ne peut pas modifier ses données personnelles, malgré qu'elles soient stockées dans la base de données.

Pour le MFA, il s'agit de créer un système un peu plus complexe que de demander à chaque utilisateur de se connecter avec cette méthode d'authentification. La vraie intention derrière cette fonctionnalité est de faciliter la connexion de personnes externes à l'entreprise du client. Le MFA sert alors de sécurité supplémentaire pour ce genre d'utilisateur. Un processus d'invitation a alors été imaginé. Un administrateur du site peut créer des invitations et y associer un numéro de téléphone et des profils. Sur la page de connexion de l'application, un anonyme peut alors décider d'accepter une invitation et de se créer lui-même un compte. Les profils de l'invitation lui seront alors attribués et l'invitation sera effacée du système. Cette fonctionnalité demande les « User Stories » qui suivent :

- En tant qu'utilisateur, je souhaite pouvoir faire les opérations CRUD sur les invitations d'utilisateur.
- En tant qu'anonyme, je souhaite pouvoir essayer d'accepter une invitation en entrant mon numéro de téléphone dans un formulaire.
- En tant qu'anonyme, lorsque j'ai rentré mon numéro et qu'une invitation correspondante est trouvée, je souhaite recevoir un code par SMS afin d'ensuite l'entrer dans le site pour prouver mon identité.
- En tant qu'anonyme, je souhaite pouvoir créer un compte à l'aide d'un formulaire une fois identifié.

Ces utilisateurs spéciaux devront alors à chaque fois s'authentifier à l'aide du MFA, à l'inverse des utilisateurs créés localement.

4.2 L'architecture logicielle

L'architecture initiale de l'application, mise en avant dans la *figure 8*, est celle de toute application basée sur un frontend et un backend. Elle est simple, car toutes les données utilisées par l'application sont internes à Technord. De plus, elle ne possède pas de service généralisé pour des opérations comme l'authentification ou la gestion des fichiers.



Figure 8 : Architecture initiale de la nouvelle application

Le frontend de l'application, son interface graphique, est une « Single Page Application », aussi appelée SPA. C'est une application web qui, au lieu de demander chaque page au serveur, charge l'entièreté de l'application en une requête et réécrit dynamiquement la page actuelle. Les interactions entre l'utilisateur et l'application sont donc beaucoup plus rapides, puisque qu'aucun temps de chargement n'est nécessaire.

Mais, comment fait l'application pour interagir avec la base de données ? Elle envoie des requêtes HTTP au backend qui est une API REST, ce qui veut dire qu'elle est sans état. Chaque requête doit contenir toutes les informations nécessaires à sa réalisation. En fonction de la demande, le backend effectue des opérations afin de renvoyer une réponse HTTP au frontend. La majorité du temps, cela demande au backend d'envoyer des requêtes à la base de données. Ces requêtes utilisent chez Technord des procédures stockées, qui seront détaillées au chapitre suivant.

Pour illustrer ceci, je vais donner un exemple simple. Imaginons que l'utilisateur veut voir les membres supprimés du personnel d'une société. Le frontend envoie alors une requête HTTP contenant l'ID de la société ainsi qu'un paramètre spécifiant la volonté de voir les éléments supprimés. Le backend réceptionne cette requête et vérifie que l'utilisateur a bien le droit de voir ces éléments. Si c'est le cas, il appelle une procédure stockée qui renvoie les informations voulues sous forme de requête SQL au backend. Celui-ci à son tour envoie une réponse HTTP au frontend qui l'interprète et affiche les éléments à l'utilisateur.

Dans le cadre de l'ajout de nouveaux moyens d'authentification, mentionnés précédemment, j'ai dû légèrement modifier cette architecture afin d'inclure des interactions avec les services Microsoft (*figure 9* et *figure 10*).

Pour l'authentification SSO, je dois d'abord rediriger l'utilisateur vers le service d'authentification Microsoft afin que celui-ci se connecte et que le service me renvoie un « Access Token » sous forme de réponse HTTP. Le frontend envoie alors une

demande d'authentification au backend qui contacte l'API Graph de Microsoft. Cette API permet d'aller rechercher les informations concernant un utilisateur. Dans le cadre de l'application, ces informations sont les données de ce dernier (ID, nom, e-mail, prénom, nom de famille, etc.) ainsi que les groupes auxquels il appartient. Le reste de l'opération se déroule comme une requête classique expliquée plus tôt dans ce chapitre.

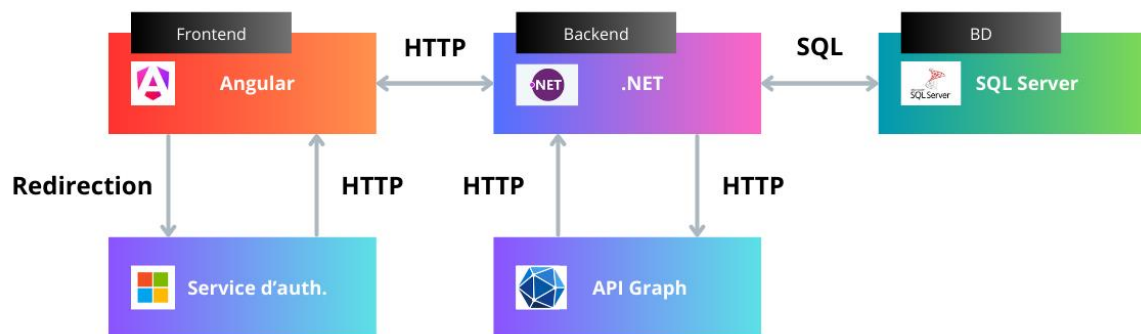


Figure 9 : Architecture de la nouvelle application en lien avec les services d'authentification Microsoft

Pour le système d'invitation, lorsque qu'une authentification par MFA est nécessaire, le backend contacte les services de communication de Microsoft afin que ceux-ci envoient un message avec le code secret à l'utilisateur. Ce dernier peut ainsi donner le code au frontend qui envoie une deuxième requête au backend. Celui-ci vérifie alors l'identité de l'utilisateur et continue le processus d'authentification.

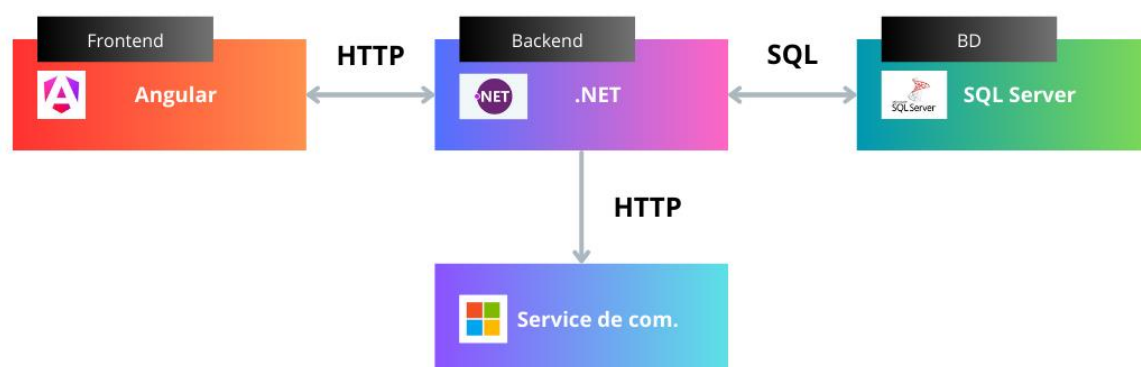


Figure 10 : Architecture de la nouvelle application en lien avec les services de communication Microsoft

4.3 Les technologies utilisées

Pour la réalisation du frontend de l'application, nous avons utilisé du Javascript avec le framework Angular. Javascript a été étudié durant le cursus dans l'unité d'apprentissage UE20 – Langages de script dynamique.

Angular, quant à lui, est un framework spécialisé dans la création de SPA. Il a été développé par Google et est une réécriture complète de son prédécesseur : AngularJS. Il est basé sur un système de composants qui permettent de créer des fragments d'interface graphique réutilisables. Ces composants sont divisés en deux parties principales : l'HTML, qui décrit le contenu de la page et le Typescript qui gère la logique derrière le composant. Pour permettre la communication entre ces deux parties,

Angular supporte la liaison de données bidirectionnelle. D'autres aides au développeur sont comprises dans ce framework telles que les Directives⁸, la navigation et une CLI⁹.

L'adoption d'Angular comme nouveau standard pour le développement frontend est relativement récente pour Technord. La principale raison derrière ce choix est la fiabilité de cette technologie, largement utilisée depuis 2012-2013. C'est aussi le framework Javascript avec lequel l'équipe Technord était le plus familiarisé avant l'adoption. Lors de celui-ci, l'entreprise a offert une formation à tous ses employés, permettant de mettre à jour ceux qui ne connaissaient pas encore la technologie.

Afin d'accélérer le développement et de garder une expérience visuelle similaire à travers toutes ses applications, Technord utilise aussi le framework Kendo. Celui-ci est une librairie d'éléments graphiques prêts à l'emploi proposant de nombreuses fonctionnalités. Le lien fort entre Kendo et Technord existe depuis l'époque des applications ASP.NET. La librairie étant compatible avec Angular, l'entreprise a décidé de continuer de l'utiliser.

Dans la même idée qu'Angular pour le frontend, Technord possède un standard pour la réalisation du backend : .NET framework Web API 2. Cette technologie utilise du C# comme langage de programmation, vu dans l'unité d'apprentissage *UE36 – Programmation avancée*.

Web API 2 est donc un framework créé par Microsoft permettant de créer des services HTTP REST. Il fait partie de la plateforme ASP.NET mais se concentre spécifiquement sur la création d'API. Il permet de spécifier des « routes » qui sont des URL spécifiques que l'on peut appeler à l'aide de requêtes HTTP. Le corps d'une réponse contient les informations demandées à l'API et peut se présenter sous différents formats. Dans notre cas, ce fut du JSON. Pour la même raison qu'Angular, Technord a choisi cette technologie pour sa robustesse, étant populaire depuis 2013-2014. Elle propose aussi une authentification améliorée facilement utilisable : OAuth. Cette technologie commence cependant à se faire vieille, avec son remplacement en .NET Core publié en 2017. Technord est donc dans une phase de migration, se dirigeant vers cette nouvelle technologie qui n'a malheureusement pas pu être utilisée pour ce projet.

Enfin, pour gérer sa base de données, Technord utilise SQL Server, qui est un système de gestion de bases de données relationnelles.

SQL Server a été créé par Microsoft et est particulièrement adapté aux systèmes de grandes envergures pour les entreprises. De plus, il est facilement utilisable en lien avec d'autres technologies Microsoft, tels que .NET framework.

Afin d'interagir avec cette base de données, nous ne créons pas de requête SQL directement dans le backend ni n'utilisons d'ORM¹⁰. À la place, des procédures stockées,

⁸ Permettent l'ajout de nouveaux attributs aux balises HTML offrant des fonctionnalités personnalisables.

⁹ Command Line Interface : Outils utilisable depuis la console de commandes.

¹⁰ Object-Relational Mapper : Outil permettant de faciliter les interactions avec les bases de données en « mappant » les tables à des objets.

aussi appelées SP, sont utilisées. Ce sujet n'a pas été abordé aux cours, mais présente de nombreux avantages. Il s'agit de scripts créés par les développeurs qui sont enregistrés directement dans la base de données. Pour faire une requête vers celle-ci, le développeur appelle la SP par son nom et lui donne des arguments. Cette dernière s'exécute alors et retourne une réponse. Les procédures stockées permettent tout d'abord de créer un standard, réutilisé dans chacune d'elles. Pour Technord, chaque SP possède des lignes de codes destinées à attraper et répertorier les erreurs dans une table spécifique, ce qui permet une gestion des problèmes bien plus efficace. Ensuite, vu que la requête ne doit pas être envoyée par le réseau, cela diminue beaucoup la charge qui lui est imposée, améliorant les performances globales de l'application. Enfin, une fois la procédure créée, elle peut être utilisée par différents systèmes, ne nécessitant donc pas de réécrire la requête complètement à chaque fois.

4.4 Les outils utilisés

Comme déjà mis en avant dans ce document, Technord est un fervent utilisateur de technologies Microsoft. En effet, celles-ci offrent un grand nombre de services et de fonctionnalités différents, avec des interactions facilitées. C'est donc logique que cette appréciation s'illustre aussi dans les outils utilisés par les développeurs.

Pour la gestion du code en équipe, Technord utilise Azure Repos. C'est un service permettant la collaboration des développeurs basés sur Git comme Github ou GitLab. Ces outils ont été présentés et utilisés lors du cursus.

L'édition de code se fait à l'aide de deux applications différentes chez Technord, en fonction du langage de programmation.

Pour la réalisation du frontend, nous avons utilisé Visual Studio Code. Ce logiciel est un éditeur de texte enrichi supportant une énorme quantité de langages différents. Il est largement utilisé autant par les particuliers que par les grandes entreprises notamment grâce à sa gratuité. Il est open-source, amélioré en continu et permet une grande personnalisation. C'est donc le choix idéal pour Technord afin de développer des applications en Angular. Ce développement est d'autant plus facilité par les plugins communautaires, proposés publiquement et qui enrichissent les fonctionnalités de l'éditeur.

Le backend, quant à lui, a été réalisé avec Visual Studio. C'est un IDE¹¹ plus lourd que son petit frère présenté juste avant, mais qui propose des fonctionnalités plus avancées. Les avantages que propose ce logiciel pour Technord sont ses templates intégrés pour les projets Web API, ainsi que le système de comparaison de schéma. Ce dernier est très utile pour les projets communs et a été utilisé tout au long du développement de l'application. Il permet de mettre en évidence les différences entre les schémas de deux bases de données et de les synchroniser. Comme Technord se base sur l'utilisation de procédures stockées, Noah et moi avons dû souvent modifier nos BD. Cet outil a donc permis d'importer facilement les changements effectués par l'un vers la base de données de l'autre.

¹¹ Integrated Development Environment : Application aidant un développeur à créer des applications

Afin de gérer nos BD et modifier les SP, nous avons utilisé SQL Server Management Studio, aussi appelé SSMS. C'est un outil dédié à la gestion des bases de données SQL Server que Technord utilise pour tous ses projets. Il permet aussi de gérer l'aspect administration de ces BD, ce qui s'avère utile dans les projets où certains participants ne devraient pas avoir tous les accès.

Seuls quelques outils utilisés par l'entreprise ne proviennent pas de Microsoft. La majorité du temps, cela est dû à un concurrent plus populaire ou une meilleure offre.

Pour le déploiement d'applications sur des machines virtuelles, nous avons utilisé Remote Desktop Manager, développé par Devolutions. Il permet de gérer les connections à distance et propose une gestion des authentifications sécurisée et efficace que les alternatives ont du mal à concurrencer.

La gestion du projet s'est faite avec Jira au vu de son énorme popularité auprès des entreprises de développement d'applications et des particuliers.

5. Réalisation

5.1 La base du projet

Noah et moi n'avons pas dû réaliser le projet à partir de zéro. Technord a mis en place une série d'aides pour nous faciliter la tâche.

Tout d'abord, ils nous ont donné accès au code de l'ancien projet. Cela nous a permis de nous familiariser avec la manière dont Technord développe ses produits. De plus, comme mentionné plus haut, le projet avait pour but de recréer une interface visuelle améliorée, ce qui veut dire que l'ancien code gérant le business model était toujours réutilisable. Bien évidemment, nous n'avons pas copié l'existant de A à Z. Nous nous sommes assurés à chaque fois que ce code était efficace, propre et conforme aux nouvelles pratiques. Si ce n'était pas le cas, nous le modifiions ou le réécrivions entièrement selon les besoins.

Par ailleurs, Technord a fourni à chaque stagiaire une copie de la base de données d'un client, ce qui a apporté plusieurs avantages. Vu que chaque stagiaire avait sa propre base de données, il était libre de l'utiliser comme il le souhaite sans avoir peur d'entraver l'autre. De plus, les données existantes dans ces copies nous ont donné une idée de ce qui était attendu lors de la création d'une page. Il était aussi prévu avant que le projet commence de garder cette base de données majoritairement intacte pour que la transition d'un projet à l'autre se fasse en douceur. Il a cependant été très vite évident que cela ne serait pas possible au vu du grand nombre de changements à apporter à celle-ci. L'ancienne base de données nous a donc servi de base pour la structure des tables et des procédures stockées.

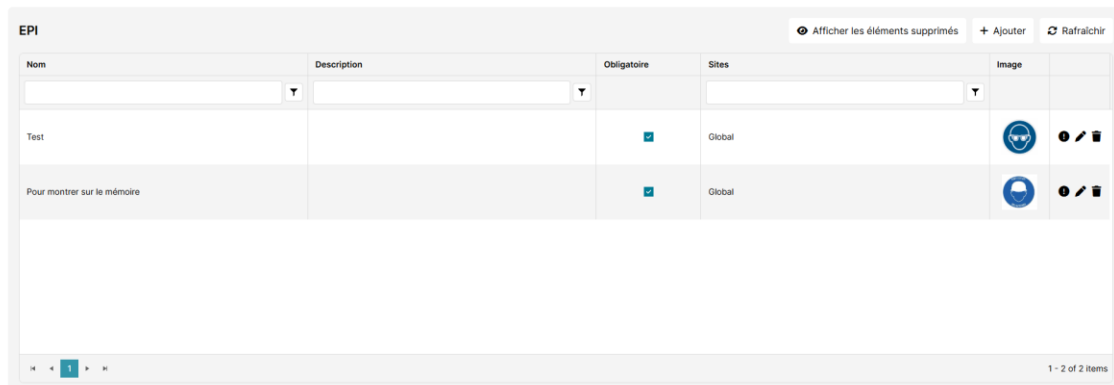
Ensuite, un des employés a été chargé de nous réaliser une maquette montrant le design de la future application. Il a donc créé un document Figma relativement avancé qui nous montrait en détail la navigation du site, le layout de plusieurs pages, ainsi que la direction artistique du projet. Nous nous sommes beaucoup basés sur ce travail, notamment au début du stage. On nous a néanmoins laissé une grande liberté en nous permettant d'adapter ce qui avait été fait selon les besoins, toujours avec l'accord de nos maîtres de stage.

Enfin, des templates, des débuts de projets, nous ont été fournis pour le frontend et le backend. Ces derniers montraient la structure que le code devait suivre et donnaient des exemples sur la création de pages.

5.2 Le frontend

Le développement de l'interface graphique à l'aide d'Angular s'est fait à l'aide de composants réutilisables dans toute l'application.

La plupart des pages de l'application suivent la même architecture : un tableau central reprenant des éléments entourés de boutons permettant diverses actions, comme illustré dans la *figure 11*. C'est notamment le cas de toutes les pages de gestion. Ce tableau central est réalisé à l'aide du composant « Grid » du framework Kendo. Celui-ci permet une grande flexibilité dans la manière d'afficher les données pour chaque colonne indépendamment des autres. Cela m'a permis d'implémenter des fonctionnalités comme des previews de vidéos. Les actions que proposent ces pages sont réalisables à l'aide de pop-up affichant un formulaire. La gestion de la validité de ces formulaires se fait à l'aide d'une logique sur mesure qui permet d'afficher des messages d'erreur personnalisés venant de l'API.





Nom	Description	Obligatoire	Sites	Image
Test		<input checked="" type="checkbox"/>	Global	
Pour montrer sur le mémoire		<input checked="" type="checkbox"/>	Global	

Figure 11 : Page classique de la nouvelle application

D'autres pages, comme celle de la réalisation d'un questionnaire (*figure 12*), ne suivent pas cette généralisation et demandent une architecture spéciale. J'ai alors eu plus de liberté artistique pour ces pages. Celles-ci sont réalisées avec un mélange de composants Kendo et de composants personnels

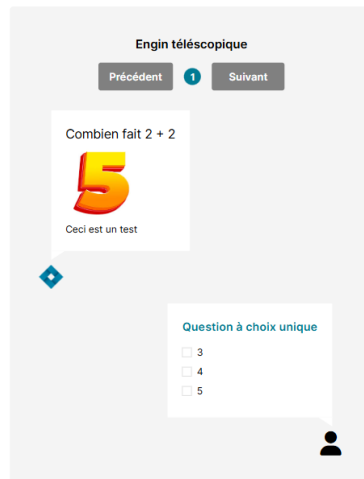


Figure 12 : Nouveau schéma de base de données des questionnaires

Les styles utilisés sur plusieurs pages sont créés et maintenus dans des fichiers CSS communs, appliqués à tout le projet. Cela permet une cohérence globale de l'application et facilite la modification d'un élément commun à plusieurs endroits.

5.3 Le backend

L'API est composé d'un ensemble de contrôleurs qui sont des classes C# regroupant des endpoints selon une thématique. La majorité de ces endpoints sont relativement simples. Ils font appel à une procédure stockée en base de données et renvoient éventuellement un résultat. Ces appels sont facilités par la librairie Technord, mettant à disposition une série de méthodes toutes faites permettant d'interagir avec les SP en mentionnant uniquement leur nom et d'éventuels paramètres.

Il arrive cependant que certains endpoints intègrent de la logique supplémentaire si nécessaire. Par exemple, à la fin d'un questionnaire, lorsqu'un utilisateur soumet ses réponses, le backend s'occupe de calculer son résultat et, en fonction de celui-ci, attribue ou non l'habilitation à l'utilisateur.

La logique native de gestion des erreurs a cependant dû être un peu modifiée. En effet, le standard chez Technord pour signaler un problème dans une procédure stockée est de retourner un entier différent de zéro. Cependant, la librairie Technord ne prenait pas en compte ce résultat et lançait une erreur seulement en cas d'inadéquation entre le code du backend et celui de la base de données. Pour pallier ce problème, j'ai mis en place une surcouche qui, en fonction de la valeur retournée par la procédure, émet un message d'erreur adéquat.

5.4 L'abstraction

Vu que la plupart des pages de l'application demandent d'implémenter les opérations CRUD, celles-ci ont pu être généralisées dans le code. Que ça soit dans le frontend ou le backend, j'ai créé une classe abstraite qui implémente les fonctionnalités nécessaires au bon fonctionnement de ces opérations. Cela m'a permis de ne pas toujours devoir réécrire le même code et d'ainsi gagner beaucoup de temps.

Prenons comme exemple la gestion des EPI. Le composant Angular et le contrôleur .NET Framework responsables de cette gestion héritent tous les deux d'une classe généralisant les opérations CRUD.

On peut observer dans la *figure 13* que cet héritage se définit dans la partie TypeScript du composant. Cela me permet d'appeler les fonctions de la classe parent dans l'HTML, entourées dans les *figures 14 et 15*.

```
@Component({
  selector: 'app-manage-general',
  templateUrl: './permit.manage.epi.component.html',
  styleUrls: ['./permit.manage.epi.component.scss']
})
export class PermitManageEPIComponent extends ListViewWithFilesAndSitesBase implements OnInit {
```

Figure 13 : Définition du composant Angular de gestion des EPI

```
<div class="actions">
  <i class="fa-solid fa-circle-exclamation" (click)="redirectToPreventions(dataItem)" [ngClass]='{"text-primary": dataItem.HasPreventions}'></i>
  <i *ngIf="canEditGlobal(dataItem)" class="fa-solid fa-pen" (click)="edit(0, dataItem)"></i>
  <i *ngIf="canDeleteGlobal(dataItem)" class="fa-solid" [ngClass]=dataItem.Enable ? 'fa-trash' : 'fa-trash-arrow-up' (click)="remove(0, dataItem)"></i>
</div>
```

Figure 14 : Définition des boutons permettant de modifier et de supprimer un élément

```
<div class="interact">
  <div (click)="see()" *ngIf="canSeeDeleted">
    <i class="fa-solid" [ngClass]="canSeeDeletedItems[0] ? 'fa-eye-slash' : 'fa-eye'"></i>
    <p class="m1-2">{{canSeeDeletedItems[0] ? 'PAGE.HIDE.DELETED.ITEMS' : 'PAGE.SHOW.DELETED.ITEMS'}} | translate}</p>
  </div>
  <div (click)="add(0)" *ngIf="canAdd">
    <i class="fa-solid fa-plus"></i>
    <p class="m1-2">{{'PAGE.ADD' | translate}}</p>
  </div>
  <div (click)="refresh()">
    <i class="fa-solid fa-arrows-rotate"></i>
    <p class="m1-2">{{'PAGE.REFRESH' | translate}}</p>
  </div>
</div>
```

Figure 15 : Définition des boutons permettant de voir les éléments supprimés, d'en ajouter et de rafraîchir la page

La *figure 16* montre la définition du contrôleur gérant les EPI, héritant d'une classe permettant de simplifier la définition des « routes ». Le contrôleur n'a plus qu'à implémenter les appels à la base de données correspondant à la requête (*figure 17*).

```
1 référence | Vanvlasselaer Zachary, il y a 14 jours | 2 auteurs, 11 modifications
public class EpiController : CUDFileControllerBase<int, EpiDAO, EpiEditDAO>
```

Figure 16 : Définition du contrôleur .NET Framework gérant les EPI

```

2 références | Vanvlasselaer Zachary, il y a 77 jours | 1 auteur, 2 modifications
protected override ActionResult Add(EpiDAO dao)
{
    return DA_Epi.ADD(dao.Nom, dao.Description, dao.Picture, dao.Required, dao.Sites);
}

3 références | Vanvlasselaer Zachary, il y a 77 jours | 1 auteur, 1 modification
protected override ActionResult SetActif(int id, bool yes)
{
    return DA_Epi.SET_ACTIF(id, yes);
}

2 références | Vanvlasselaer Zachary, il y a 16 jours | 1 auteur, 1 modification
protected override ActionResult<List<OldFile>> UpdateWithFiles(EpiEditDAO dao)
{
    return DA_Epi.UPDATE(dao.ID, dao.Nom, dao.Description, dao.Picture, dao.Required, dao.Sites);
}

```

Figure 17 : Définition des URL d'ajouter, de suppression, de restauration et de modification d'un EPI.

5.5 Les défis techniques

Ce projet m'a mis au défi à plusieurs occasions. Il a parfois été nécessaire de passer par plusieurs itérations ou de faire des recherches documentaires. Je vais expliquer ici les problèmes les plus intéressants selon moi.

5.5.1 La navigation

Un des éléments importants de l'application et la première chose sur laquelle je me suis attardé est la navigation. Effectivement, l'application possède beaucoup de pages différentes qui sont organisées sous forme d'arbre. Visuellement, cela se représente avec trois barres de navigation distinctes, illustrées sur la *figure 18*.

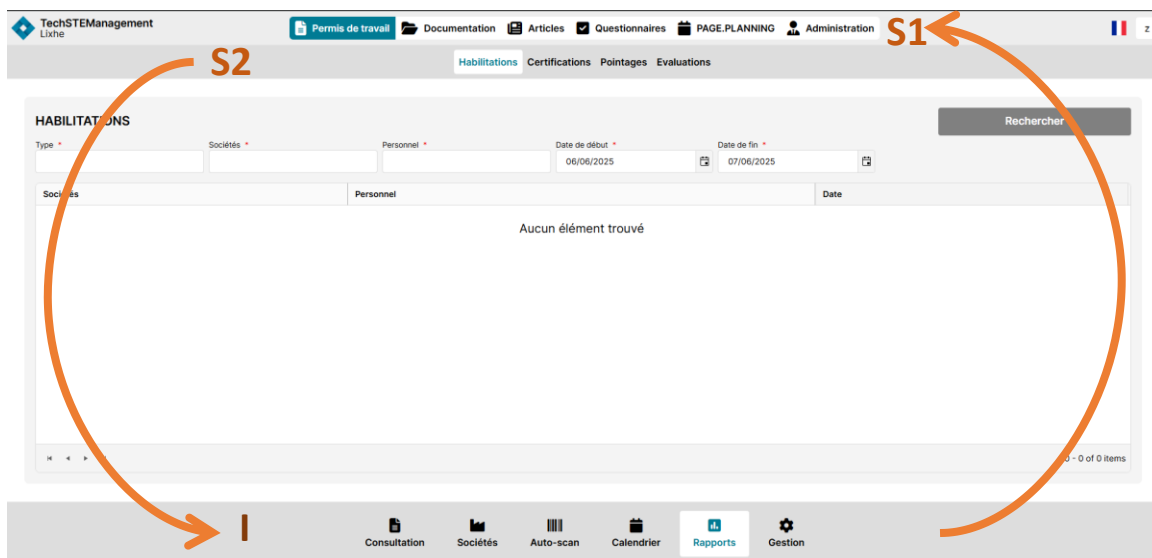


Figure 18 : Définition des URL d'ajouter, de suppression, de restauration et de modification d'un EPI.

La deuxième barre de navigation supérieure (S2) dépend de la barre de navigation inférieure (I), qui dépend elle-même de la première barre de navigation supérieure (S1). Il est donc nécessaire d'avoir une logique de navigation robuste qui s'assure d'afficher les bons éléments dans la navigation, peu importe la page actuelle.

Il est aussi important de noter que certaines pages, notamment celles qui dépendent d'un paramètre, ne doivent pas être visibles dans la navigation. Par exemple, la page montrant les habilitations d'un sous-traitant ne doit pas être accessible depuis une barre de navigation, sauf lorsque l'utilisateur clique sur un sous-traitant dans la page du personnel.

De plus, de la même manière que toute action est soumise à un droit, toute page est elle-même soumise à un droit d'accès spécifique. Cette règle présente elle aussi des exceptions, vu que les pages de personnalisation de l'application ne peuvent être utilisées que par des administrateurs Technord. Aucun droit ne doit y donner accès.

L'accès aux pages est aussi soumis à la licence actuelle. Effectivement, selon ce que le client a acheté, il n'aura pas forcément accès à toutes les fonctionnalités du site et donc aux pages qui en dépendent.

Angular propose nativement une gestion de la navigation, mais celle-ci n'est pas assez poussée pour me permettre de mettre en place toutes ces règles. J'ai donc été obligé de créer une surcouche à cette gestion native.

Tout d'abord, j'ai défini toutes les informations dont avait besoin une page pour rentrer dans le système dans une classe nommée « *TreeNode* », montrée dans la *figure 19*.

```
export class TreeNode {
  path: string;
  name: string;
  faImage?: string;
  children?: TreeNode[];
  //Influence la navigation
  rights?: string[];
  //Influence la navigation
  licence?: string[];
  //Influence la navigation
  minRightLevel? : number;
  filterFunc? : Function;
  //N'influence PAS la navigation
  guards?: any[];
  //Cache l'élément de la navigation mais
  //est toujours accessible
  hidden?: boolean;
  //Si true, l'élément ne peut pas être une
  // route par défaut de la navigation
  //Utile pour les page demandant des
  //query params pour fonctionner notamment
  cannotBeDefaultedTo? : boolean;
  component?: any;
  data? : any;
}
```

Cette classe contient le nom, le chemin et la représentation imagée de chaque élément de navigation. Si cet élément est une feuille de l'arbre, et donc une page, la classe possède un composant. Sinon, elle reprend tous ses éléments enfants. Plusieurs propriétés de la classe permettent de gérer l'accessibilité de la page : les droits, la licence, le niveau de droit et une fonction de filtrage personnalisable. Il est possible aussi d'une spécifier qu'une page est cachée de la navigation mais accessible ou qu'elle n'est pas utilisable comme page par défaut.

Figure 19 : Définition de la classe *TreeNode*

Noah et moi avons ensuite créé un arbre contenant chaque page du site. Cet arbre n'est cependant pas utilisable directement par le système de navigation d'Angular. En effet, celui-ci suppose que chaque élément de l'arbre est une page alors que dans notre cas, seules les feuilles le sont. J'ai donc implémenté un algorithme, illustré dans la *figure 20* qui aplatit notre arbre afin de créer une liste reprenant chaque feuille, précédée de ses composants parents.

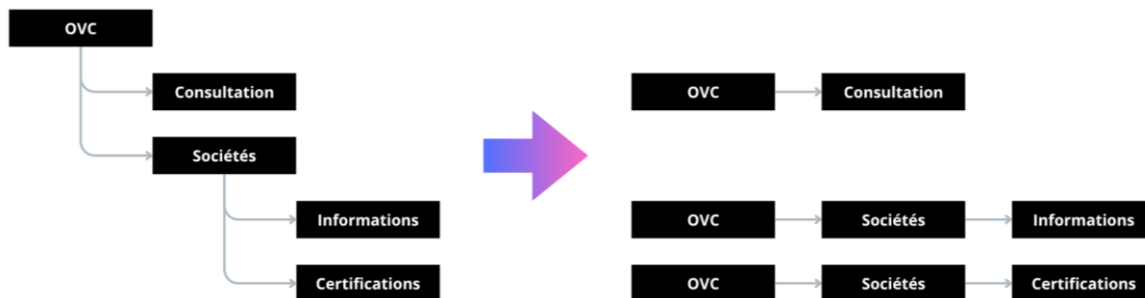


Figure 20 : Avant/Après utilisation de l'algorithme de l'aplatissement

Par ailleurs, afin que la navigation soit claire pour l'utilisateur, il est important qu'il soit clairement renseigné sur sa position dans les barres de navigation. Pour que cette information soit toujours fiable, elle est déterminée à partir de l'URL de la page actuelle. Un algorithme identifie d'où provient la page en traversant l'arbre section par section sur base de l'URL.

5.5.2 Les questionnaires

Comme expliqué dans le chapitre quatre, j'ai dû recréer entièrement le système des questionnaires. Les deux notions importantes de ce nouveau système étaient le fait qu'une version ne puisse plus être modifiée une fois validée et qu'une version puisse être traduite en plusieurs langues.

Pour m'assurer de cela, j'ai mis en place un schéma de base de données assez complexe que je vais présenter dans la *figure 21*. Dans ce schéma, « App. à » signifie « Appartient à » et « Réf. » veut dire « Référence ». Dans les deux cas, cela implique que la table de laquelle part la flèche possède une clé étrangère référençant la table vers laquelle la flèche pointe.

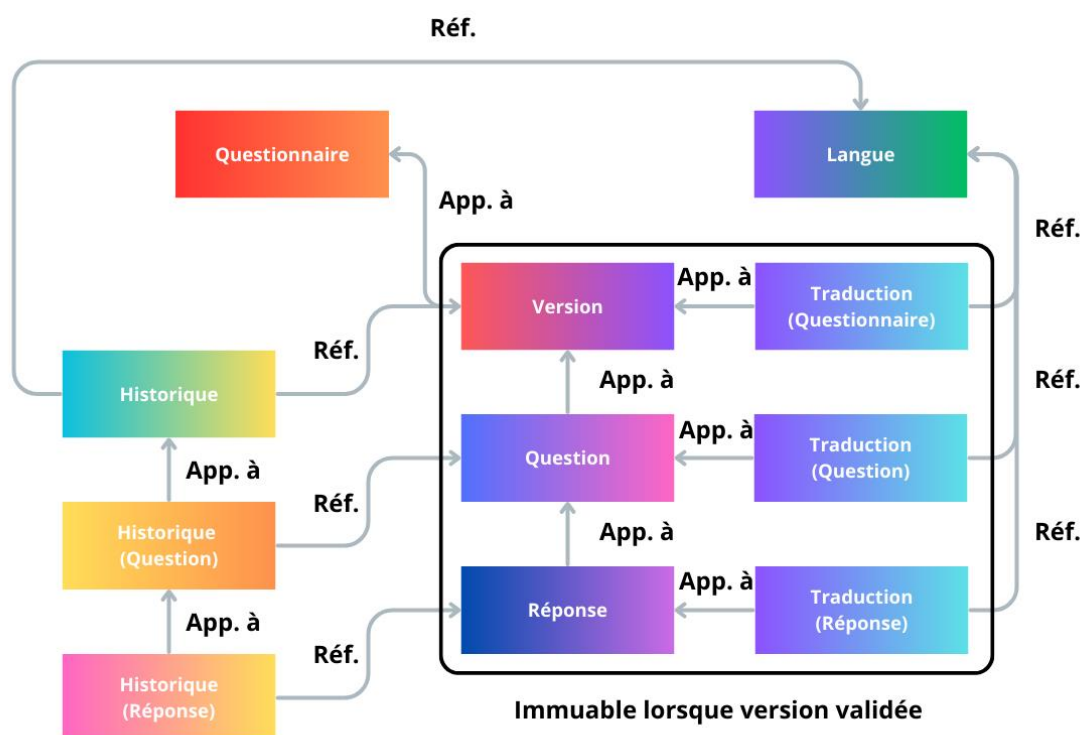


Figure 21 : Nouveau schéma de base de données des questionnaires

Une fois une version validée, la table « Version » et toutes les tables la référençant directement ou indirectement deviennent immuables, à l'exception des tables dédiées à l'historique. Une tentative sur une version d'un questionnaire ne peut être faite que lorsque celle-ci est choisie comme « courante » par un utilisateur. De plus, une entrée dans l'historique se fait uniquement après une tentative. Dès lors, on peut être certain de ne plus avoir de données incohérentes dans l'historique.

Par ailleurs, lorsque quelqu'un veut faire une tentative, on lui demande de spécifier la langue dans laquelle il souhaite la réaliser. On va alors chercher uniquement les traductions référençant cette langue dans la base de données.

Il existe maintenant beaucoup plus d'informations à afficher et d'actions à effectuer sur la page de gestion d'un questionnaire. Le layout de cette page est donc unique en son genre pour l'application et a demandé plusieurs itérations avant d'obtenir un résultat qui convenait à tout le monde.

Le premier design affichait ses informations sous forme de carrés de couleur en dessous du titre du tableau. Chaque action possible était représentée par un bouton avec du texte. Il a vite été constaté que ces carrés prenaient trop de place et qu'il fallait diminuer leur taille.

Le texte à l'intérieur des boutons a alors été remplacé par des petites icônes et les informations ont été alignées à l'horizontale, donnant un aspect plus rectangulaire comme montré la *figure 22*.



Figure 22 : 2^{ème} itération de l'affichage des informations d'un questionnaire

Un autre défaut a alors été relevé : le design est peu intuitif. L'application est prévue pour des opérateurs, des personnes n'étant pas toujours familières avec le domaine de l'informatique. Il faut donc une utilisation simple, ce qui n'était de toute évidence pas le cas car mon responsable projet n'a pas immédiatement compris l'utilité de chaque bouton. Le souci venait principalement de l'état de validation du questionnaire. En effet, l'information semble répétée trois fois : à côté de la sélection de la version, dans le rectangle « Actif » et dans le rectangle « Valide ». Il s'agit en réalité de trois informations différentes : si la version est celle actuelle, si le questionnaire est actif et s'il est validable. Pour résoudre ce problème de clarté, deux ajustements ont été mis en place. Premièrement, le rectangle déterminant la validité du questionnaire n'est affiché que si celui-ci ne l'est pas encore. Il ne sert effectivement à rien de savoir si un questionnaire est validable s'il a déjà été validé. Deuxièmement, le bouton « Actif » et le bouton de la version actuelle ont été fusionnés. Ainsi, un questionnaire est actif seulement s'il possède une version courante. Pour désactiver un questionnaire, il suffit de ne pas lui en assigner une. Par ailleurs, des images ont été ajoutées à côté des versions lors de leur sélection pour signaler leur état de brouillon ou d'actualité. Ceci donne le design final de la page des questionnaires, mis en avant dans la *figure 23*.

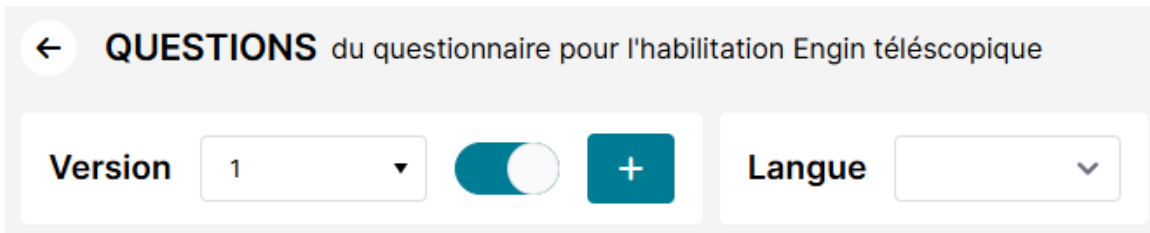


Figure 23 : Itération finale de l'affichage des informations d'un questionnaire

5.5.3 Le thème dynamique

Un autre challenge important rencontré lors de la réalisation du projet a été la création du thème dynamique. Cette fonctionnalité a failli ne pas voir le jour à cause de certaines limitations de l'environnement de développement.

Initialement, le thème du projet était situé dans un fichier CSS situé à la racine de celui-ci. Ce fichier contenait une liste de variables, chacune correspondant à une couleur utilisée à divers endroits de l'application.

Le but de la fonctionnalité était de pouvoir modifier ces variables de manière globale, c'est-à-dire que les modifications s'appliqueraient à tous les utilisateurs.

La première solution imaginée consistait à venir directement lire et modifier ce fichier CSS. Malheureusement, Angular limite les opérations possibles sur les fichiers de

l'application pour des raisons de sécurité. Il est donc impossible d'éditer un fichier contenu directement dans le projet.

Vu qu'il n'est pas possible d'enregistrer des changements de manière définitive dans le frontend, le thème devait donc venir du backend. Ceci engendrait un autre problème : une requête vers l'API n'est pas instantanée. Un changement des couleurs de l'application pendant une session serait particulièrement désagréable pour un utilisateur. À ce moment, la faisabilité de cette fonctionnalité a commencé à être remise en cause.

Cependant, après quelques recherches, j'ai découvert qu'il était possible de demander à Angular d'exécuter une fonction avant d'initialiser l'application. Qui plus est, si cette fonction est asynchrone, Angular attend la fin de son exécution avant d'afficher quoi que ce soit à l'utilisateur.

Après ce bon début, il fallait ensuite s'occuper de l'implémentation de la sauvegarde du thème. Pour cela, j'ai déplacé les variables CSS dans un fichier JSON, qui, lui, est lisible par le frontend. Ce dernier envoie alors les couleurs au backend, qui s'occupe de changer leurs valeurs en fonction du thème stocké en base de données et les renvoie à l'interface graphique. Cette dernière utilise alors un algorithme pour transformer le fichier JSON en CSS valide et l'enveloppe dans une balise de style qu'elle met à la racine de l'HTML. Pour appliquer des changements au thème, il suffit alors de faire une requête à l'API qui va changer les données de la BD.

Une question subsistait : est-ce que toutes ces opérations ne ralentiraient pas trop le démarrage de l'application ? Après quelques tests, il s'est avéré que ce ralentissement était de l'ordre de la centaine de milliseconde, ce qui n'est pas suffisant pour faire une différence notable. De plus, vu qu'un thème peut être exporté, l'appel API peut être désactivé à tout moment. Il suffit alors de remplacer les valeurs de base des couleurs par celles de son choix.

5.6 La Sécurité

Un point important à améliorer sur la nouvelle application était l'aspect sécurité. Cela était d'autant plus nécessaire avec la nouvelle architecture backend – frontend. Le responsable du projet a décidé d'engager une société pour qu'elle réalise un « Pentest¹² » sur l'application dans les semaines qui suivent la fin du stage afin de s'assurer que tout soit en ordre.

Je me suis donc penché sur cet aspect très complexe qu'est la cybersécurité avec les quelques connaissances que j'ai acquises pendant mon cursus.

La première chose que j'ai passée en revue est l'injection d'HTML directement dans la page. Ce genre d'opération peut mener à des vulnérabilités face aux attaques XSS¹³. J'ai

¹² Test de pénétration : Simulation autorisée de cyber-attaque sur un logiciel dans le but de trouver de potentiels failles de sécurité

¹³ Cross-Site Scripting : Attaque où un script malicieux est injecté dans une application

donc remplacé ces injections par du data binding lorsque c'était possible. Si la fonctionnalité ne pouvait pas être implémentée autrement, comme pour le thème dynamique, j'ai fait en sorte que l'HTML soit assaini avant d'être utilisé.

Vu que l'application permet à l'utilisateur d'uploader des fichiers, il était impératif de s'assurer que ceux-ci soient conformes à ce que l'on attend. Pour cela, j'ai implémenté plusieurs vérifications dans le frontend et le backend. Tout d'abord, l'extension du fichier est vérifiée. Elle doit correspondre à ce qui est attendu par la fenêtre de sélection de fichier native et au type MIME. Ensuite, la taille de fichier ne peut pas dépasser une certaine valeur, définie dans la configuration de l'application et dépendante du type de fichier. Enfin, je regarde les « Magic Bits » du fichier (*figure 24*¹⁴). En effet, chaque type de fichier possède une signature sous forme de suite de bits avec un décalage. Cette dernière est constante et rend le fichier invalide si modifiée.

Hex signature ↕	ISO 8859-1 ↕	Offset ↕	Extension ↕	Description ↕
23 21	#!	0		Script or data to be passed to the program following the shebang (#!)[1]
02 00 5a 57 52 54 00 00 00 00 00 00 00 00 00 00	STXNULZWRTNULNULNULNULNULNULNULNULNULNUL	0	cwk	Claris Works word processing doc
00 00 02 00 06 04 06 00 08 00 00 00 00 00	NULNULSTXNULACKEOACKNUL BS NULNULNULNULNUL	0	wk1	Lotus 1-2-3 spreadsheet (v1) file

Figure 24 : Exemple de « Magic Bits »

Ma dernière tâche concernant la sécurité était de vérifier les endpoints du backend. En effet, ceux-ci peuvent être contactés par n'importe qui avec un minimum de connaissances en informatique. De plus, avec tous les outils de développement à la disposition du public, aucune information venant du frontend n'est digne de confiance. Pour chaque endpoint, il faut considérer que toute information venant de l'extérieur est potentiellement erronée. J'ai donc implémenté une validation des données entrantes pour tous les endpoints acceptant des requêtes POST ou PUT. De plus, l'identité de la personne émettant la requête est vérifiée dans la plupart des cas. Le backend s'assure qu'il est authentifié, qu'il possède les droits nécessaires pour effectuer une action donnée et que la licence actuelle de l'application permet de faire cette action.

5.7 Les performances

L'aspect performance n'a pas été un point fondamental de cette migration, bien qu'il n'ait pas non plus été négligé. Noah et moi avons suivi un principe instauré par nos maîtres de stage tout au long du développement : l'utilisateur ne doit pas attendre trop longtemps pour une requête. C'est une définition assez vague qui laisse pas mal de marge de manœuvre, mais qui permet de donner une idée globale de la vitesse attendue de l'application.

¹⁴ Source : https://en.wikipedia.org/wiki/List_of_file_signatures

Il a été constaté seulement une poignée de fois qu'un chargement était trop long. À chaque fois, c'était dû à une requête GET, qui allait chercher trop d'éléments en base de données. Nous avons donc implémenté une pagination SQL dans ces cas très précis.

La pagination SQL déplace la responsabilité des éléments à afficher du frontend à la base de données. En effet, dans la majorité des pages de l'application, l'interface graphique reçoit tous les éléments possiblement affichables et s'occupe de gérer lesquels sont montrés en fonction de la page actuelle choisie par l'utilisateur. Lorsque la base de données obtient cette responsabilité, le frontend envoie une requête à l'API avec un nombre d'éléments et un décalage. Ce nombre d'éléments est de l'ordre de la vingtaine, ce qui est beaucoup plus petit que les possibles milliers d'entrées dans la base de données. La requête GET au chargement de la page est donc beaucoup plus rapide, avec en contrepartie un léger délai lors du changement de page.

5.8 Compte-rendu

Au final, le stage est considéré comme un succès par nos maîtres de stages. Toutes les fonctionnalités de l'ancienne application ont pu être répliquées à l'exception de la gestion des articles, qui est un élément optionnel à l'utilisation du site web. Ce nouveau projet est fondé sur une base beaucoup plus solide qui prend en compte la sécurité et l'expansion possible des fonctionnalités. Le code est aussi maintenant plus clair dans son ensemble.

Un des points importants de ce stage était la refonte graphique. Notre responsable projet en est très satisfait, ce dernier proposant même d'utiliser certains de nos composants graphiques dans d'autres applications. Cette amélioration est illustrée par les *figures 25 et 26*.



Figure 25 : Ancienne page de réponse à un questionnaire

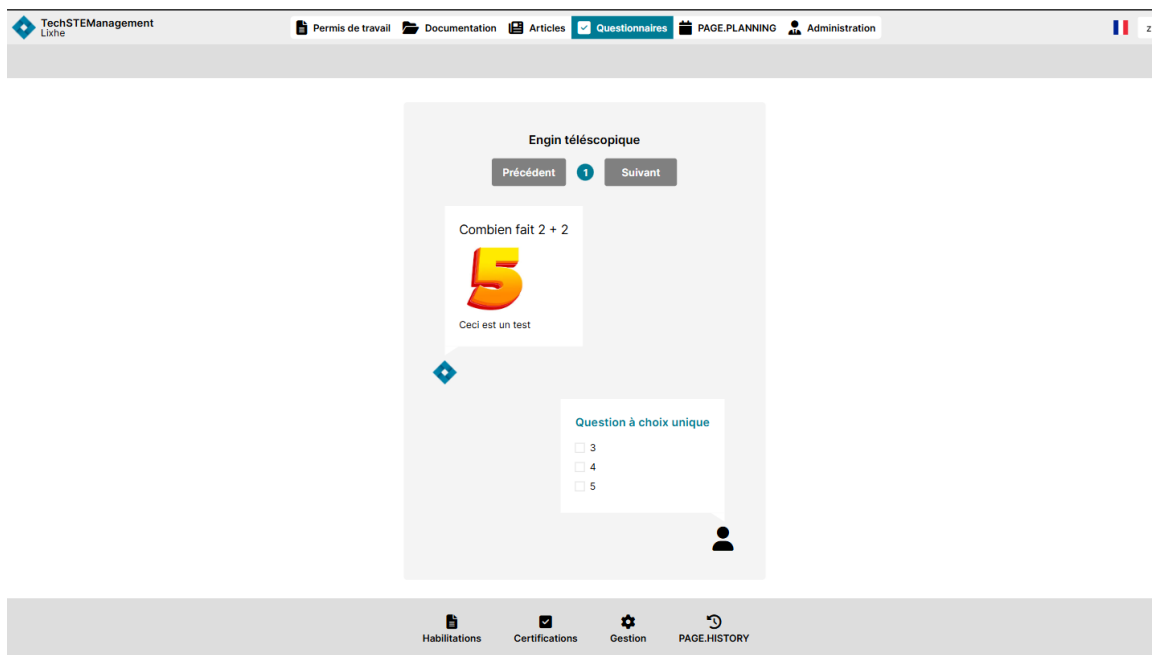


Figure 26 : Nouvelle page de réponse à un questionnaire

Les temps de chargement ont été grandement diminués, ne dépassant jamais la seconde. L'application dans son ensemble est plus interactive et plus agréable à l'emploi.

Les fonctionnalités bonus apportées aux projets auront aussi leur impact. Elles permettront de faciliter et d'accélérer la personnalisation de l'application en fonction du client. En outre, les nouvelles méthodes d'authentification auront leur importance aux yeux des nouveaux clients potentiels.

Par ailleurs, à la fin du stage, le projet a été présenté à un client de longue date de l'ancienne application. Celui-ci a exprimé son enthousiasme quant à l'arrivée de cette nouvelle version.

Conclusion

Aux termes de ces quinze semaines de stage, je peux dire avec fierté que j'ai complété tous les objectifs de ce projet, et même plus. Le projet est un franc succès et les maîtres de stage sont heureux du travail accompli.

Dans son ensemble, l'application reprend toutes les fonctionnalités prévues initialement à l'exception d'une seule : la gestion des articles. Cette fonctionnalité est optionnelle et n'intervient pas directement dans le processus d'ouverture de chantier

La nouvelle interface graphique est un gros point fort du projet. Noah et moi avons suivi la direction artistique proposée par Technord, tout en prenant des libertés dans la réalisation des éléments graphiques. Nous avons mis en place un standard pour chaque page, ce qui permet d'avoir une cohérence globale dans le visuel de l'application. Par

ailleurs, les éléments souvent utilisés ont été généralisé en composant réutilisable. Cela a permis de créer une interface modulaire et intuitive qui est une nette amélioration par rapport à l'ancienne version. Cette interface a notamment beaucoup plu à notre maître de stage, qui a même proposé de réutiliser certains de nos composants dans d'autres applications.

Le code de l'API, bien que restant majoritairement similaire à celui de l'ancienne application, a été dépoussiéré. Certaines parties ont été optimisées et d'autres ont été modifiées afin de suivre de bonnes pratiques de programmation.

La base de données n'a malheureusement pas pu être conservée à l'identique. Durant le stage, l'équipe s'est aperçue que certaines de ses parties pouvaient être améliorées. Cela s'illustre notamment par la refonte entière des tables de questionnaires. L'idée de ne pas modifier la base de données a alors été abandonnée et un historique des changements a été mis en place pour faciliter la transition.

De mon côté, j'ai implémenté les tâches qui m'ont été attribuée plus rapidement que prévu. J'ai effectivement terminé la majorité du scope initial au bout d'environ neuf semaines, le reste ayant été mis en place plus tard pour différentes raisons. Cette rapidité est grandement due à l'établissement de code généralisant les fonctionnalités qui m'étaient attribuées. En effet, la plupart des pages de l'application présentaient des similarités, notamment dans la gestion des opérations CRUD. Ces similarités ont pu être exploitées dans des classes abstraites, qui s'occupaient de gérer la logique commune. J'ai dès lors gagné beaucoup de temps lors de la réalisation des pages en ne réécrivant pas du code très similaire.

Cela m'a donné l'opportunité de travailler sur différentes fonctionnalités bonus, qui étaient considérées comme des améliorations optionnelles de l'application. Celles-ci touchaient à des domaines plus complexes du développement d'application, comme l'authentification, les thèmes dynamiques et la gestion des traductions. C'est la partie du stage qui m'a le plus emballé. En effet, ces fonctionnalités étaient moins faciles à mettre en place et très peu de code existait pour m'orienter. J'ai ainsi dû entreprendre une démarche de recherche et de réflexion pour chacune d'entre-elles. Cette difficulté ajoutée touche à ce qui me fait aimer la programmation : la résolution de problèmes. J'ai dû moi-même imaginer la logique du code, ainsi que l'architecture la supportant. J'ai eu l'occasion d'interagir avec les API publiques de Microsoft, ce qui a été une expérience extrêmement enrichissante pour moi.

La réalisation de ces tâches additionnelles a permis deux choses : une personnalisation de l'application facilitée et une authentification diversifiée. Les modifications apportées à l'application pour chaque client, autrefois fastidieuses, sont maintenant plus rapides à mettre en place et entièrement réalisables depuis l'interface graphique. En deuxième lieu, les différentes méthodes d'authentification ajoutées à l'application plairont à des clients avec une organisation hiérarchique de ses employés déjà bien établie.

Pour ce qui est des perspectives du projet, il a un bel avenir devant lui. Technord a décidé d'assigner des employés au projet à la suite du stage, afin de le fiabiliser et de

corriger les derniers petits problèmes restants. Ce processus durera un ou deux mois, avant que la migration ne soit complétée cet été. À partir d'août, Technord proposera à ses clients encore sous garantie de migrer leur application sans aucun frais. Par ailleurs, c'est déjà la nouvelle application qui est présentée aux nouveaux clients potentiels, certains exprimant leur impatience pour la fin du développement de celle-ci.

Personnellement, j'ai eu l'occasion de beaucoup apprendre, tant dans le code qu'en dehors. J'ai appris au fur et à mesure du stage comment créer une application en Angular tout en suivant de bonnes pratiques. Avec le recul et à la lumière de mes nouvelles connaissances, je pense que certaines parties du projet auraient pu être mieux implémentées, mais je reste particulièrement satisfait de la réalisation de l'interface graphique dans son ensemble.

L'utilisation des procédures stockées afin d'interagir avec la base de données a été une nouvelle expérience pour moi. J'ai redécouvert le SQL sous un autre jour. Petit à petit, j'ai appris toute une série de fonctionnalités proposées par ce langage et j'ai pris confiance en mes capacités. Lorsque l'on m'a demandé de recréer la partie de la base de données qui gérait les questionnaires, j'étais suffisamment à l'aise avec la technologie pour implémenter une architecture propre et efficace. Bien que la réalisation de certaines procédures stockées m'ait parfois mis au défi, j'ai apprécié travailler sur cette partie du projet. Je pense avoir réalisé quelque chose de pratique et d'extensible.

Mon stage m'a permis d'avoir un aperçu de la manière dont les projets sont gérés au niveau professionnel. Ces quinze semaines ont poussé la gestion de projet plus loin que durant toutes les années du cursus. J'ai ainsi gagné un savoir-faire qui me sera utile lors de l'entièreté de ma carrière.

Les critères de satisfaction des tickets étaient aussi beaucoup plus sévères que pour les projets de l'école. Ce n'était maintenant plus acceptable de terminer un ticket sans vraiment l'avoir testé. Nos maîtres de stage avaient des attentes et il était hors de question de laisser passer un ticket qui n'y correspondait pas. Cette élévation des critères de satisfaction m'a permis de remettre en cause la manière dont je développais. J'ai appris que correctement tester une fonctionnalité était tout aussi important que de l'implémenter.

Cette expérience dans le milieu professionnel m'a aussi enseigné beaucoup de choses en dehors du développement d'application. J'ai pu être témoin de comment la hiérarchie d'une entreprise s'organise et des interactions entre ses différentes parties prenantes. J'ai intégré une équipe gérée par quelqu'un avec une dizaine d'années d'expérience, ce qui m'a permis d'apprendre beaucoup de choses sur la gestion d'un projet, mais aussi sur les personnes qui y participent.

Au final, ce stage a été pour moi une expérience positive dans tous les domaines. Je suis content du travail accompli et des relations que j'ai créées. Ces quinze semaines ont enrichi mon arsenal de compétences et m'ont préparé à ce qui m'attend dans le milieu professionnel. Une chose est sûre, je suis prêt pour le début de ma carrière.

L'intelligence artificielle n'a été utilisée qu'à des fins de recherche d'informations dans le cadre de l'écriture de ce mémoire. Aucun texte présent dans ce document n'a été généré ou modifié par une intelligence artificielle.