versusmind

UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

Développement d'une plateforme de recueil de consentements RGPD

Rapport de stage ST50 - A2019

Nicolas BALLET

Département Génie Informatique
Filière libre

Entreprise Versusmind

30 avenue du Rhin
67000 Strasbourg
versusmind.eu

Tuteur en entreprise
Philippe Didiergeorges

Suiveur UTBM
Vincent Hilaire

utbm
université de technologie
Belfort-Montbéliard

Remerciements

Je tiens tout d'abord à remercier Versusmind et l'UTBM pour m'avoir donné l'opportunité d'effectuer ce stage.

Philippe Didiergeorges pour m'avoir suivi et guidé durant mon stage.

Rémi Benoit et Jacques Lorentz pour m'avoir aidé et épaulé au sein de l'équipe durant ma formation.

Francois Simond et Ahmed Zahri, qui ne faisaient pas partie de mon équipe et qui m'ont tout de même apportés un grand support.

Christophe Cote-Collisson pour ses relectures et ses conseils tout au long de mon stage.

Ma famille et mes proches qui ont pu m'aider à la rédaction de mon rapport avec leurs relectures et leurs conseils, mais aussi pour m'avoir soutenu durant ce stage.

Table des matières

Chapitre 1

Introduction

Du 2 Septembre 2019 au 7 Février 2020, j'ai effectué un stage au sein de l'agence Strasbourgeoise de l'entreprise Versusmind.
Versusmind est un cabinet d'architecture numérique spécialisé dans l'expertise et l'accompagnement des entreprises et de leurs solutions digitales.
Je me suis intéressé à cette entreprise car je désirais appréhender la gestion du développement en équipe dans un cadre professionnel. Aussi, le métier de consultant représente bien les valeurs qui m'ont poussé à choisir la filière libre.
Au cours de ce stage j'ai pu m'intéresser au développement assisté par les services clouds ainsi qu'aux méthodes Agiles.
Mon stage dans l'équipe de développement Central Consent Manager (CCM) a consisté en l'extension, l'amélioration et l'optimisation de la plateforme.
Afin de présenter le travail que j'ai fourni, je vais commencer par la présentation du groupe Versusmind, ses valeurs et marques, puis je vais expliciter ma mission et le contexte dans lequel elle s'inscrit, pour enfin conclure sur mon expérience au sein de ce stage.

Chapitre 2

Présentation de l'Entreprise

2.1 Le groupe Versusmind

FIGURE 2.1 – Logo de Versusmind

Versusmind est une entreprise de services du numérique (ou ESN anciennement SSI) fondée en 2006 à Nancy par Benoît Koch. Ce cabinet d'architecture numérique est polyvalent et se spécialise en conseil en systèmes, logiciels informatiques, IA, IOT, Sitecore, systèmes embarqués, infrastructure réseau et base de données. Ce stage rentre dans un contexte de diversification de ses offres, notamment autour du RGPD.

Dans une dynamique de croissance depuis sa fondation (50% d'expansion annuelle), le groupe Versusmind emploie environs 200 collaborateurs répartis sur cinq pôles : le siège Nancy, Pariz, Metz, Strasbourg, et au Luxembourg. Ainsi le groupe peut couvrir localement la région Grand Est. Après une vague de recrutement en 2018 avec 100 collaborateurs cherchés, il exprime un désir de croissance externe en signant cette année, le 28 janvier 2020, l'acquisition de la société de conseil indépendante parisienne AFERSYS.

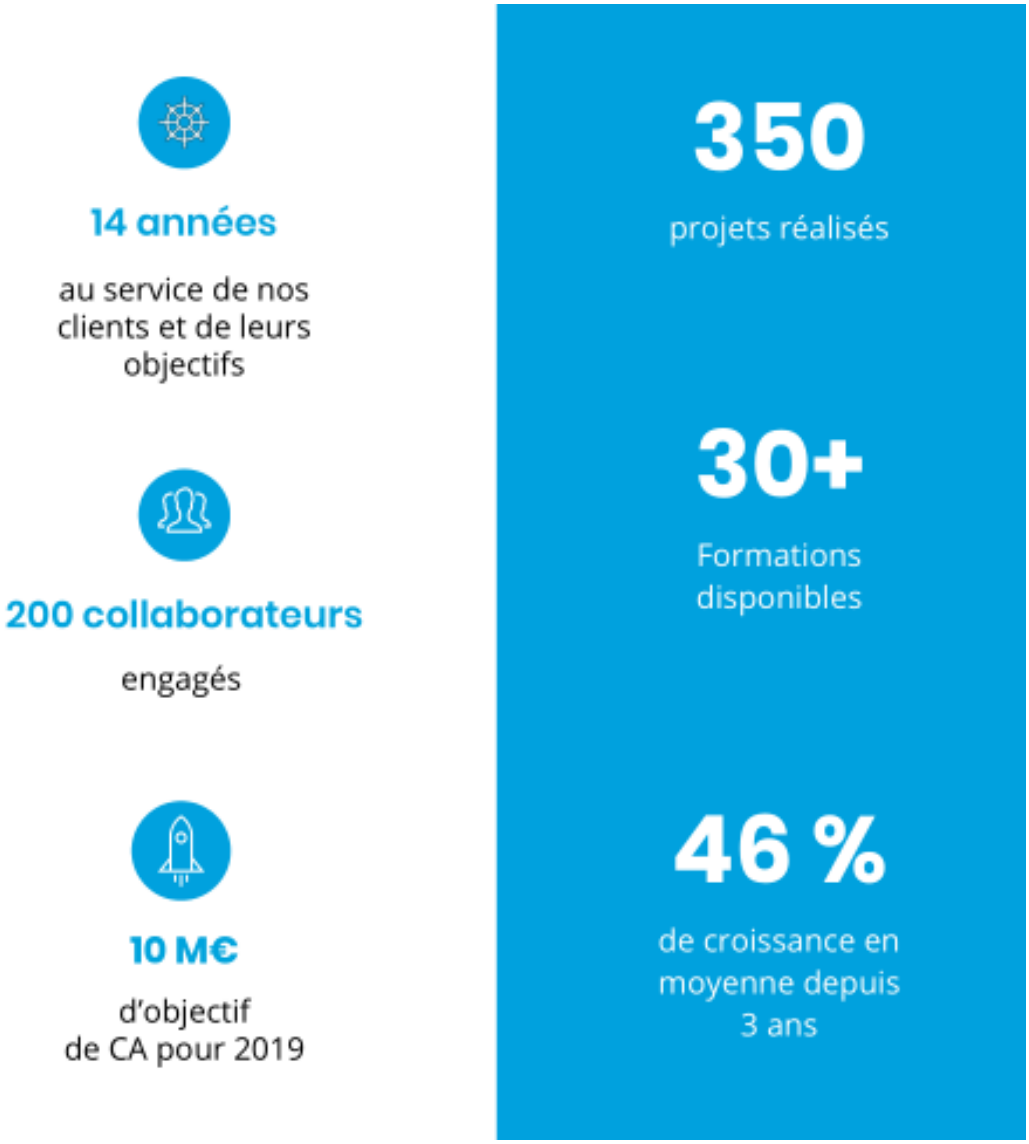

FIGURE 2.2 – Versusmind en quelques chiffres

2.2 Les marques

FIGURE 2.3 – Logos des différentes marques du groupe Versusmind

Versusmind : Cette marque assiste la transition digitale des entreprises off et online, multi et cross canaux, afin de les aider à concevoir et personnaliser leur image de marque grâce à Sitecore. L'intérêt de Sitecore est sa base de données d'expérience qui encadre l'intégralité des interactions entre un client et l'entreprise. VersusXperience, propose alors aux entreprises clientes de partir de leur contexte, d'élaborer un plan d'action et des objectifs SMART, d'être accompagnées par des consultants certifiés et de valoriser l'expérience utilisateur.

VersusInstitute : Ce marque est centré sur la formation des professionnels sur les dernières technologies du web et du digital. Les formateurs sont des consultants internes de Versusmind, « praticiens passionnés » de ces technologies. Les entreprises peuvent participer à des formations inter- entreprises, intra-entreprises, ou sur mesure, mais 70% du temps basées sur la méthode agile, en trois Sprints : Le projet du client, les solutions des consultants, puis la finalisation du projet (avec évaluation de l'acquisition). VersusInstitute participe aussi à un Agile Tour national en proposant des conférences et en participant à des salons et évènements externes.

VersusConsulting : Conseil en stratégie opérationnelle autour du RGPD. Avec cette marque, le client a l'occasion de mettre en commun sa connaissance du métier et l'expertise technique et juridique des consultants de Versusmind. Le panel des offres de Versusconsulting est composé d'Audits (de cinq jours environ), d'accompagnement de mise en conformité (en six étapes), de mise à disposition de Délégué à la Protection des données (DPO)
Et dans le cadre de l'accompagnement des entreprises, Versusmind propose la solution Central Consent Manager. Central Consent Manager ou CCM, est un système de gestion des consentements centralisé.

Lors de mon arrivée dans l'entreprise, j'ai intégré l'équipe de développement de CCM en cours de saison de refonte graphique, le projet avait été lancé un an plus tôt et n'était pas encore proposé à des clients. L'équipe était composée de deux à trois développeurs

stables (moi inclus) auxquels s'ajoutaient des développeurs temporairement, en attente de projet.
Dans une équipe de développeurs qui évolue beaucoup, mon rôle a été d'aider au développement de la plateforme CCM, à sa mise en production et à améliorer ses performances.

2.3 L'Environnement de travail

Versusmind s'applique à créer une culture d'entreprise autour des valeurs d'échange, du partage, d'entraide et de la curiosité. La « Versusfamily » communique fréquemment en ligne sur plusieurs applications : lumapps, par mails, et sur Microsoft Teams.

Les consultants ont la possibilité de publier et de présenter des guides et conseils techniques sur des nouvelles technologies au travers de conférences mensuelles appelées Symposium (accessibles en ligne sous forme de vidéo), de témoignages vidéo #PowerUp d'employés sur différents postes (disponibles sur la chaine youtube de Versusmind) et d'articles techniques sur le blog « Les Billets de Versusmind ». Les développeurs peuvent aussi participer à des évènements moins formels mais toujours techniques comme des Afterwork ou des Hackatons internes d'un weekend, environs tous les six mois.

En plus de cela, l'agence strasbourgeoise de Versusmind s'applique à rendre les relations entre collègues conviviales via des activités de team building. Comme un évènement Coding Dojo après le travail, consistant en la résolution d'un problème ludique en équipe proposée par un maitre de séance. Dans cette même direction, ludique et amicale, les consultants peuvent se retrouver régulièrement au sein de soirées jeux de société ou plus généralement « soirées Versusmind » dans les locaux même de l'agence. Enfin, une fois par mois sont organisés des Bretzmiam, un repas invitant tous les employés même ceux qui sont mobilisés chez des clients.

Cet environnement de travail a eu beaucoup de poids lorsqu'il m'a fallut choisir mon stage de fin d'études.

Chapitre 3

Méthodologie de travail agile

J'ai été intégré durant mon stage à une équipe utilisant la méthodologie Scrum, qui fait partie des méthodes de gestion de projet agiles. Les methodes agiles visent à supprimer ou au moins à réduire l'effet tunnel d'une méthode de gestion classique par exemple le Cycle en V.
Le développement est découpé en cycles (de trois semaines dans le cas présent) appelés "Sprint".
Les Sprints sont regroupés en saisons afin de représenter un objectif général. Par exemple, quand j'ai démarré mon stage, le but de la saison en cours était de terminer la refonte graphique. Au lieu d'un cahier des charges donné au début du projet, nous allons découper en User Stories tout au long du projet avec l'aide du client.
Cela permet de rester concentré sur les aspects importants et de ne pas développer de fonctionnalités qui ne seront jamais utilisées.
Un Scrum Master est assigné au projet, son rôle est d'aider l'équipe à bien suivre un fonctionnement agile ainsi qu'a s'organiser. Il est aussi là pour guider le client dans la rédaction des User Stories. Le Projet Owner est un membre à part entière de l'équipe Scrum dont la responsabilité principale est de définir un produit qui apportera le maximum de valeur métier aux utilisateurs dans le temps et le budget impartis au projet.

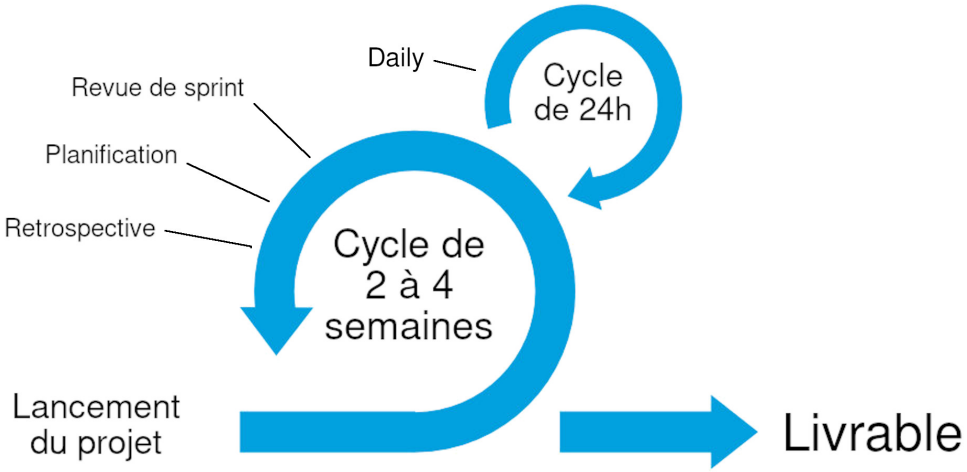

FIGURE 3.1 – Cycle de développement Scrum

3.1 Réunions journalières (Daily)

Chaque matin nous avons une petite réunion qui ne doit pas dépasser 15min afin d'informer rapidement le reste de l'équipe de l'avancement des différentes taches, de discuter brièvement des différents problèmes rencontrés mais aussi de choisir ensemble la priorité des différentes taches qu'il reste à faire.
Cela permet de toujours rester conscient de l'état du projet et d'avoir une vision d'ensemble du travail accompli et du travail restant.

3.2 Réunions inter-Sprints

Entre chaque Sprint, une série de réunions nous permet de rester dans la bonne direction concernant le développement du projet.

3.2.1 Revue de Sprint

Le but de la revue de Sprint est de montrer à toutes les parties prenantes (Project Owner, Scrum Master, développeurs) l'avancement du projet pendant le dernier Sprint.
L'équipe de développeurs décrit le travail accompli et après une démonstration du résultat, on discute afin de peut-être corriger la direction à prendre pour le prochain Sprint.

3.2.2 Planification

S'en suit la planification, elle se fait avec le Scrum Master ainsi que les développeurs.
On va y sélectionner les User Stories à implémenter durant le prochain Sprint. Cela

constituera ce qu'on appelle "L'objectif de Sprint".
Sous forme d'un nombre on va se mettre d'accord sur une complexité pour chaque User Stories. Cela permet de vérifier qu'on en à bien la même compréhension. Si je propose 1 (facile) et qu'un de mes collègue propose 4 (moyen), c'est que l'un de nous deux à mal compris ce qui est attendu.
Ensuite on découpe chaque User Stories en taches concrètes et enfin on estime le temps de travail sur chaque taches.

3.2.3 Rétrospective

Et enfin, encore une fois avec le Scrum Master ainsi que les développeurs, on discute de comment s'est déroulé le dernier Sprint, des pratiques qu'on pourrait mettre en place ou améliorer, mais aussi de comment on l'a vécu et ressenti.
Cela s'inscrit dans un objectif d'amélioration de l'environnement de travail et de l'efficacité.

Chapitre 4

Développement de la plateforme CCM

4.1 Contexte

Adopté par l'Union Européenne en avril 2016, la date d'entrée en vigueur du Réglement Général sur la Protection des Données (RGPD) est le 25 mai 2018. Celui-ci oblige les entreprises à identifier les données personnelles en leur possession ainsi que leurs modalités de traitement et de protection et a pour objectifs de :

1. Uniformiser la réglementation au niveau européen
2. Responsabiliser les entreprises
3. Renforcer les droits des personnes

Le non-respect du RGPD peut mener à des sanctions financières importantes ainsi qu'a des sanctions administratives ayant un fort impact sur le fonctionnement de l'entreprise. Il est donc important pour les entreprises d'être en conformité avec le RGPD.

C'est pourquoi Central Consent Manager permet aux entreprises de se libérer d'un poids en s'assurant que ses utilisateurs ont bien consenti à l'utilisation de leurs données personnelles en centralisant l'enregistrement et la vérification d'authenticité des consentements qui pouvaient être sur des applications et plateformes diverses auparavant. L'application CCM permet au client de gérer l'administration des consentements grâce à trois fonctionnalités : la consigne de l'ensemble des informations dans un registre, la gestion des traitements et la gestion des consentements.

L'application étant déjà existante lors de mon arrivée, les missions qui m'ont été confiées ont étés : la participation à une refonte graphique, l'ajout de fonctionnalités, mais aussi l'optimisation de l'application CCM.

4.2 Fonctionnement de l'application

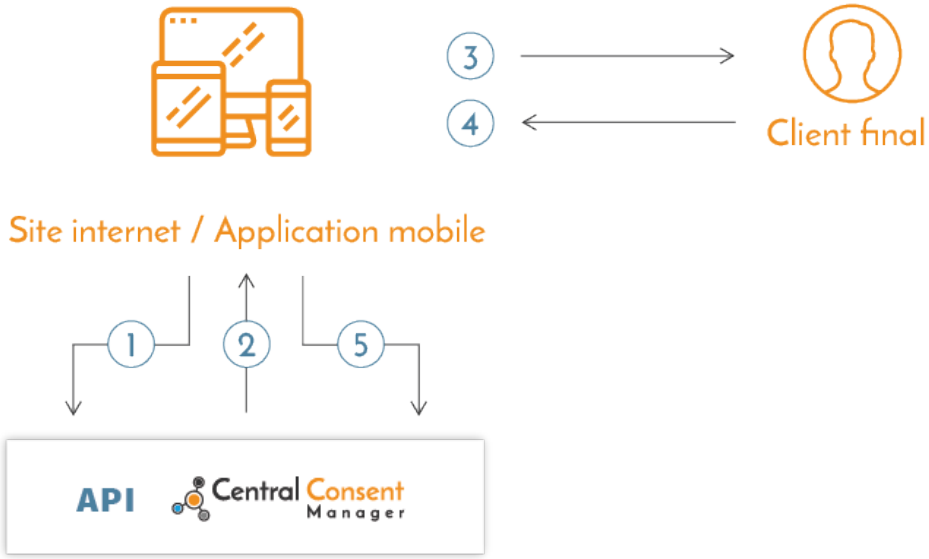

FIGURE 4.1 – Fonctionnement de l'application CCM

Ci-dessus est schématisé le processus d'enregistrement d'un consentement au sein de la plateforme CCM.

1. Requête sur le service de mentions pour un traitement
2. Envoi des mentions d'informations
3. Demande de consentement (sous forme d'e-mail ou autre)
4. Confirmation
5. Enregistrement du consentement

4.3 Architecture de l'application CCM

Avant d'entrer dans le détail de chaque partie de l'application, voici une version simplifiée de l'architecture de l'application :

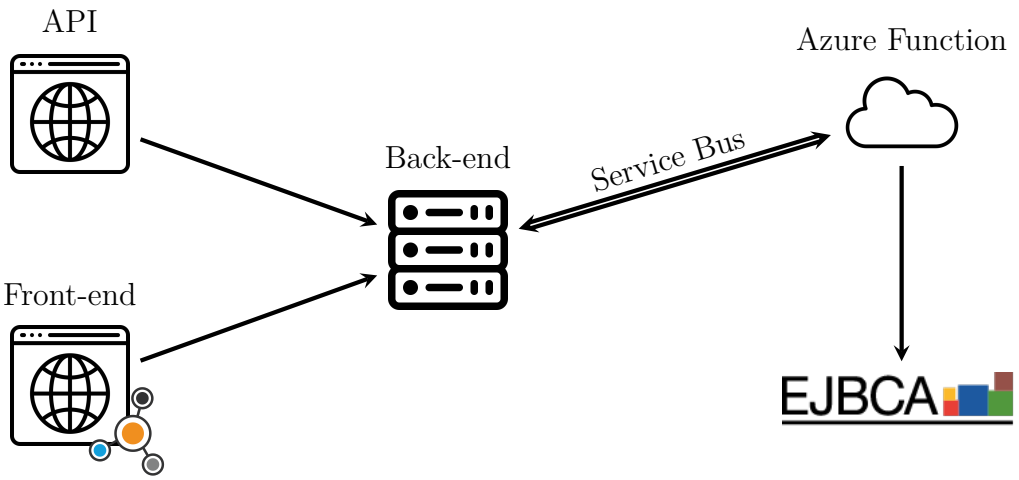

FIGURE 4.2 – Architecture globale simplifiée

Voici un rapide résumé des personnes destinées à utiliser l'application :

- Un administrateur Versusmind, qui pourra, via le Back-Office, accéder aux contrats souscrits avec les entreprises. Il n'a pas besoin de compétences techniques.
- Un administrateur client, qui pourra, via le Back-Office, ajouter des traitements et des consentements. Il n'a pas besoin d'avoir de compétences techniques.
- Les développeurs clients, qui accèdent à l'API du Back-end afin d'intégrer CCM dans leur solution déjà existante.
- Les utilisateurs finaux, qui donneront leurs consentements au travers de la plateforme.

Vous pourrez retrouver en annexe A une liste plus détaillée des rôles de l'application. Tout d'abord je vais rapidement expliquer l'architecture du projet. La solution CCM est composée de différents éléments :

- Une partie Font-end écrite en HTML [1]/CSS [2]/Typescript [3] avec le framework Angular, c'est le back-office d'administration, il sert les différentes pages web à l'utilisateur.
- Une partie Back-end écrite en Java avec le framework Spring Boot, c'est le cœur de l'application, elle peut recevoir des requêtes via le Front-end ou directement par API
- La gestion du chiffrement est en deux parties
  - Une instance EJBCA qui prend le rôle d'autorité de certification
  - Des Fonctions Azure écrites en Java qui chiffrent et vérifient les consentements. Elles communiquent avec le Back-end via des Service Bus Azure (de simples files d'attentes de messages JSON)
- Une instance Redis [4] afin de garder du cache des différentes requêtes faites à la

1. HyperText Markup Language : langage utilisé pour afficher du contenu dans une page web
2. Cascading Style Sheet : Langage utilisé afin de créer une mise en page sur le HTML
3. Typescript est un sur-ensemble du Javascript, utilisé pour dynamiser des pages web
4. Base de données rapide utilisant la mémoire vive afin de stocker les données

base de données.

4.3.1 Outils et environnement de travail

Les différentes parties du système sont hébergées sur plateforme cloud Microsoft Azure.
Nous avons aussi utilisé Azure DevOps afin d'informatiser les notions de Sprints et de User Stories mais aussi pour faciliter la gestion du code source, jouer les jeux de tests et automatiser le déploiement des nouvelles versions.
Voici deux captures d'Azure DevOpes, l'une montrant le tableau de l'ensemble des taches, et l'autre montrant des taches associées à une User Story sur la page d'un Sprint.

FIGURE 4.3 – Capture du dashboard d'Azure DevOps

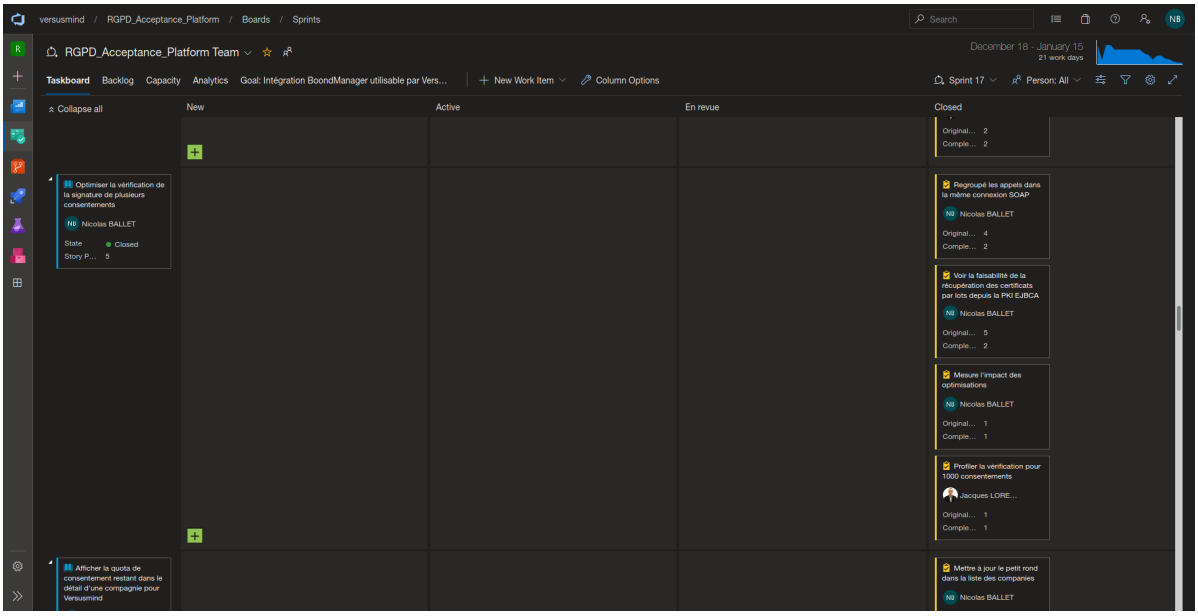

FIGURE 4.4 – Capture d'un Sprint d'Azure DevOps

Concernant mon poste de travail, j'ai eu l'agréable surprise de pouvoir travailler avec l'environnement de mon choix et l'IDE[5] ou éditeur de texte de mon choix. J'ai donc pu installer la distribution Arch Linux ainsi que mes configurations habituelles et cela a grandement participé au confort au quotidien. J'ai développé en utilisant NeoVim (qui est une réécriture de l'éditeur Vim) couplé à un ensemble de modules et notamment le module Conquer of Completion (Coc) qui permet d'utiliser de multiples serveurs de langages dérivés des implémentations utilisées par Visual Studio Code. J'ai donc pu profiter d'une autocomplétion intelligente identique à celle de mes collègues de travail qui utilisent Visual Studio Code.

4.3.2 Développement du Front-end

Utilisation du framework Angular

J'ai commencé mon stage en me formant sur l'utilisation du framework Angular sur lequel je suis maintenant à l'aise.
Angular est un framework front-end destiné à faciliter la création de pages web en utilisant le pattern modèle-vue-contrôleur.
L'interface web se découpe en une multitude de composants imbriqués les uns dans les autres. Ces composants utilisent des services afin de communiquer avec le back-end, et affichent des données à l'écran au travers de templates HTML. La mise en place d'un nouveau composant nécessite de définir un certain nombre d'éléments :

5. Integrated Development Environment : Environment de développement tout-en-un, exemple : Visual Studio

— Les données dont le composant a besoin : ses entrées
— Les événements produits par le composant : ses sorties
— Les différents services et autres composants dont dépend le composant : ses dépendances
— Son design visuel : son template

Mais certaines parties sont plus spécifiques. Comme par exemple la gestion des traductions et la gestion de l'authentification. Dans notre projet il existe deux fichiers au format JSON associant des clés de traductions à des valeurs. Chacun des fichiers comportant les traductions d'une langue précise, par exemple :
En anglais :

```
{
    "pageTitle": "Consents"
}
```

En Francais :

```
{
    "pageTitle": "Consentements"
}
```

Et nous pourrons utiliser la clé **pageTitle** afin d'automatiquement afficher la bonne traduction en fonction de la langue choisie.

Concernant l'authentification, il s'agit d'un service qui va écouter l'ensemble des retours HTTP faits au back-end, et si l'un d'entre eux retourne un code d'erreur 401 (signifiant que l'utilisateur n'est pas authentifié), le service s'occupera d'enregistrer la page courante, de rediriger l'utilisateur sur la page d'authentification pour enfin le faire revenir sur la page où il était.

J'ai pu ensuite aider à la fermeture d'une saison de refonte graphique en apportant mes connaissances et mon expérience en intégration web à l'équipe.
J'ai aussi participé à l'amélioration et à l'enrichissement de l'interface sur toute le reste de mon stage. En modifiant des composants que ce soit pour fixer un comportement non voulu ou pour ajouter des fonctionnalités. Mais aussi en ajoutant de nouveaux composants avec l'aide de l'équipe UX, chargée de livrer des maquettes de designs d'interfaces et d'expérience utilisateur.

Refonte graphique

Tout au long de ma formation, j'ai pu faire de la veille technologique et guider l'équipe vers une restructuration de l'architecture CSS plus maintenable et plus facile à étendre.
Cela se concrétise par la rédaction d'un guide de bonnes pratiques que les développeurs

peuvent consulter lors de la modification, l'extension ou la création d'un composant graphique. Afin de rédiger ce guide, j'ai du aussi me baser sur l'architecture existante afin de ne pas viser un but parfait mais inatteignable à cause de la dette technique actuelle. J'aborde dans ce guide (accessible en annexe B), la notion de précision en CSS afin d'aider les développeurs à mieux comprendre les conflits qu'ils peuvent rencontrer et d'éviter l'utilisation abusive de la directive " !important". Je propose d'utiliser la convention de nommage BEM, afin de mieux gèrer la précision et la modulatiré de nos règles CSS. J'apporte aussi quelques notes sur l'utilisation de préprocesseurs CSS et de dépendances externes tels que Angular Matérial dans notre cas.
La rédaction m'a été confiée suite à de multiples phases de relecture de code à plusieurs où mes collègues se sont apercu que j'appréciais le développement de mise en page web. Après plusieurs jours de rédaction, mise au points je l'ai soumis à mes collègues le 13 Janvier, il a pu être relu et discuté par l'équipe et après quelques clarifications a été validé dans la journée.

Voici plusieurs captures d'écran prises à la suite de la refonte graphique, mon guide ayant servi à son développement :

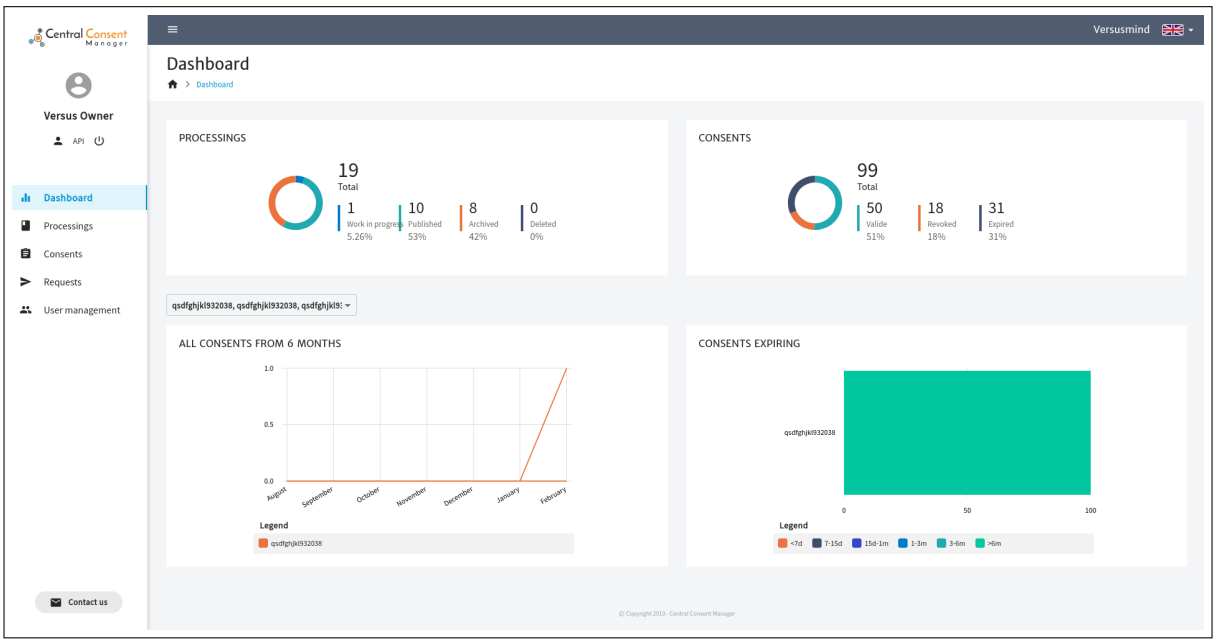

FIGURE 4.5 – Page d'accueil de l'application

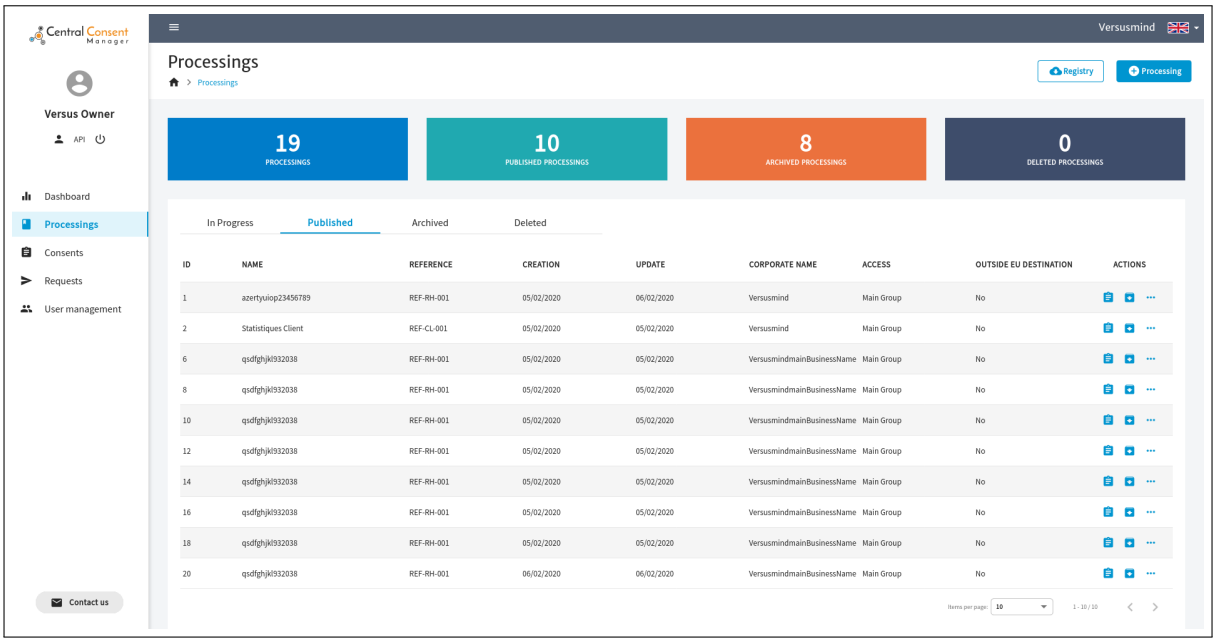

FIGURE 4.6 – Liste des traitements

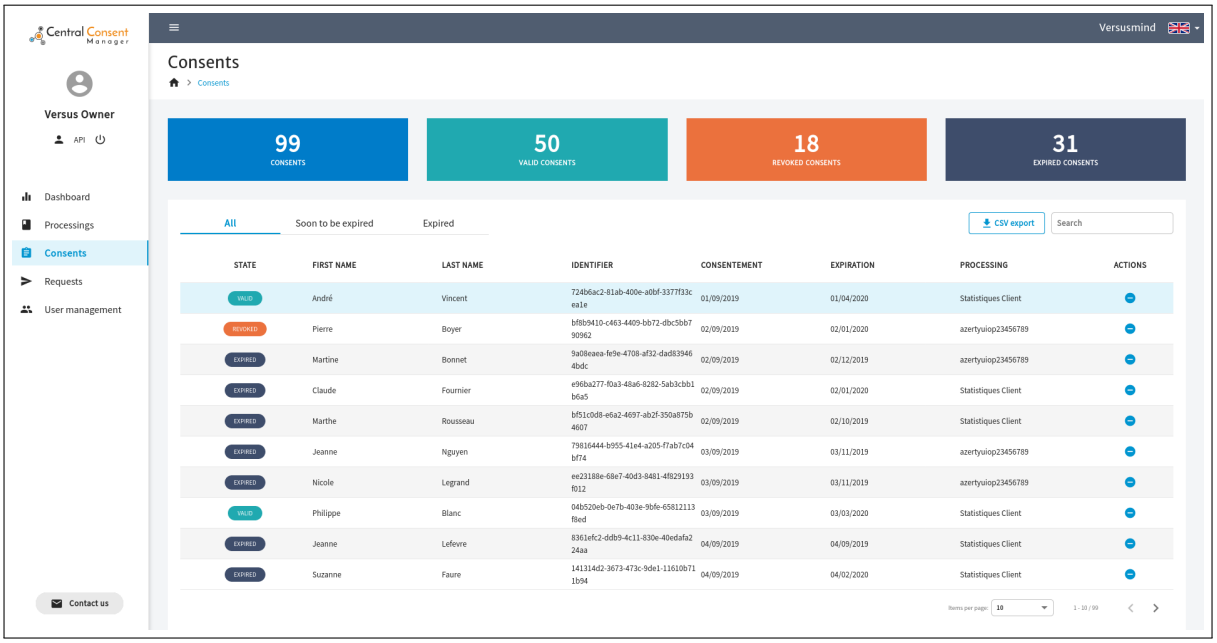

FIGURE 4.7 – Liste des consentements

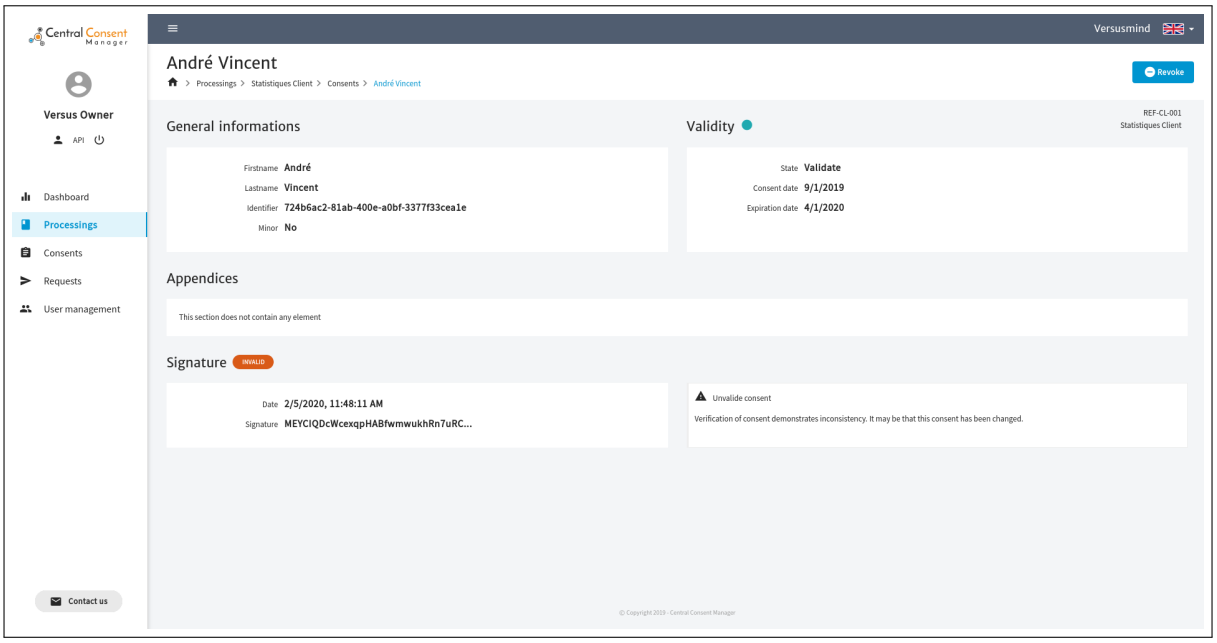

FIGURE 4.8 – Détail d'un consentement

4.3.3 Développement du Back-end

Le back-end est développé à l'aide de Spring, un framework open source permettant la création simple d'infrastructures Java simples et facilement testables.

L'application est divisée en trois niveaux,
Les contrôleurs qui vont recevoir les requêtes HTTP, que ce soit via une intégration de l'API sur le site d'un client, ou du backoffice de CCM. Leur rôle est de correctement mettre en forme et rediriger les données aux bons services.
C'est ensuite la responsabilité de la couche Business de vérifier les droits d'accès, l'intégrité des données et enfin de demander une mise à jour ou une récupération en base de données.
Cette dernière est gérée par la couche Repository qui va, au travers de requêtes générées par l'ORM ou de requêtes SQL forgées manuellement, communiquer avec la base de données. Les appels à la base sont gérés par Hibernate, qui s'occupe de la translation entre notre base de code et les appels à la base de données.

J'ai contribué au développement du back-end du projet, sous forme d'ajout de fonctionnalités, corrections de failles de sécurité, corrections de bugs, etc…

Gestion du cache

Je n'ai pas touché au développement de la gestion du cache côté serveur, mais j'ai participé aux discussions et aidé à la prise de décisions. Nous avons fait face à un problème de mise à jour des données présentes à la fois dans la base de données et dans le cache. La solution vers laquelle nous nous sommes orientés a été de simplifier les requêtes faites en base pour unifier la récupération des données (abstraire le cache et la base de données) et de filtrer les données dans une couche applicative.

4.3.4 Certification numérique

J'ai eu à déployer une instance EJBCA assignée à l'instance de production de la plateforme CCM.
EJBCA ou Enterprise JavaBeans Certificate Authority est une solution d'autorité de certification libre et open source proposée par PrimeKey.

Mon travail dans cette tâche s'est étendu de la génération de clés cryptographiques, à la manipulation de la plateforme Azure et du stockage de mots de passe sensibles car liés à la production. L'instance azure est déployée via Docker afin d'assurer un déploiement stable et reproductible, et les clés ont été générées avec OpenSSL.

Afin de sécuriser les mots de passe nous avons utilisé les KeyVault d'Azure. Il s'agit d'un coffre de mots de passe que l'on nomme afin de les récupérer automatiquement au lancement de l'application avec un mot de passe maître permettant d'ouvrir le coffre. Il

suffit dont de changer le lien du coffre afin de passer de l'environnement de développement à l'environnement de production.

4.3.5 Résolution des problèmes de montée en charge

Suite à la refonte graphique, nous sommes entrés dans une saison de mise en production et deux problématiques de performances sont apparues assez vite :

- Lorsqu'un nouveau client veux migrer vers CCM, une base de consentements potentiellement conséquente doit être importée.
- Un client doit pouvoir exporter cette base sous forme de fichier, ce qui implique une vérification de validité d'un grand nombre de consentements.

Suite à des tests de charge, nous avons observé qu'il faudrait environ trois jours pour importer une base d'un million de consentements et plusieurs heures pour générer un export des consentements enregistrés, ce qui n'était pas acceptable.
Nos tests de charges sont effectués pour la plupart avec l'outil open source Gatling, qui simule une multitude de connexion simultanées et indépendantes afin de représenter une population d'utilisateurs.

Afin de résoudre ce problème, nous avons mis en place des fonctions Azure ainsi qu'optimisé la vérification des consentements.

Développement des fonctions Azure

J'ai eu la responsabilité d'implémenter les fonctions Azure afin de paralléliser chaque création de signature numérique.
Voici une représentation simple de la signature d'un consentement :

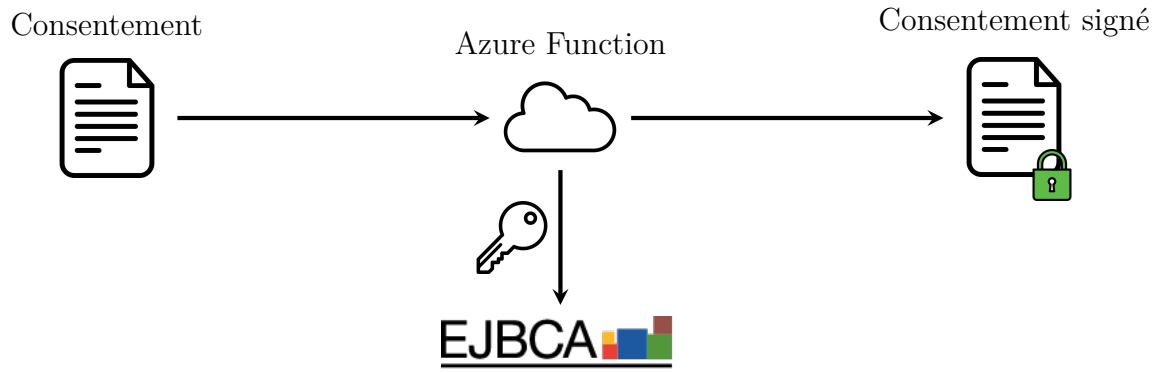

FIGURE 4.9 – Parcours de la signature d'un consentement

Dans la même tâche, j'ai pu implémenter la révocation de consentement en utilisant aussi des fonctions Azure.

Optimisation de la vérification

J'ai aussi eu à travailler sur la vérification des consentements et j'ai pu diviser le temps de signature par deux tout en augmentant drastiquement le niveau de sécurité.
Voici une représentation simple de la vérification de validité d'un consentement :

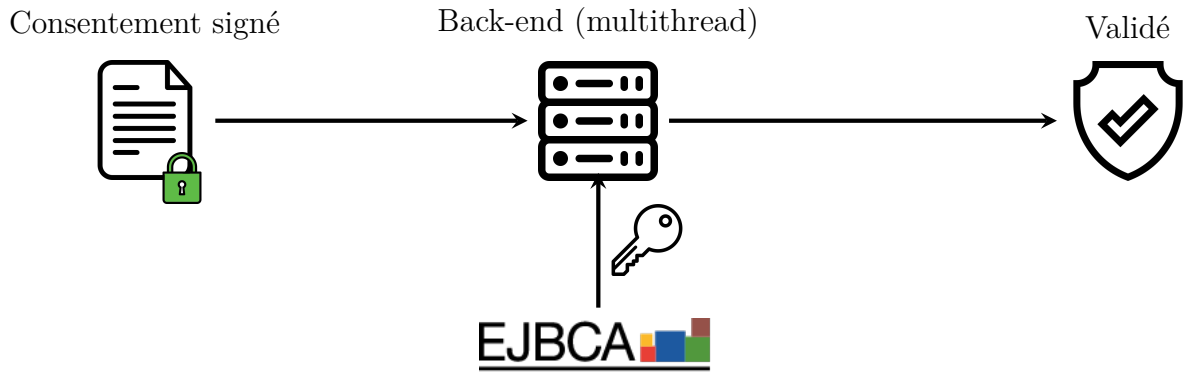

FIGURE 4.10 – Parcours de vérification d'un consentement non altéré

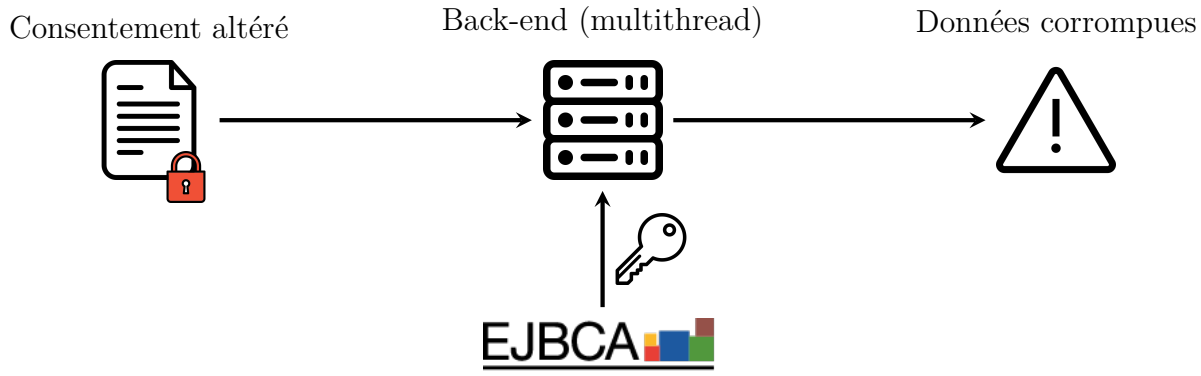

FIGURE 4.11 – Parcours de vérification d'un consentement altéré

Resultats des optimisations

Suite à ces optimisations, et d'autres qui ont suivies mon stage, nous avons réduit le temps d'insertion d'un million de consentement de trois jours à une quarantaine de minutes.

4.4 Développement dirigé par les tests (TDD)

Durant mon stage, j'ai appris à implémenter des tests afin de subvenir à plusieurs besoins :

1. Fournir une preuve de fonctionnement du code testé
2. Prévenir d'éventuels bugs futurs
3. S'assurer du bon fonctionnement de l'application avant son déploiement

Le développement suit un cycle circulaire où nous développons des nouvelles fonctionnalités, ce qui change le comportement de l'application, les tests ne passent donc plus. Nous réparons les tests, et le cycle recommence.

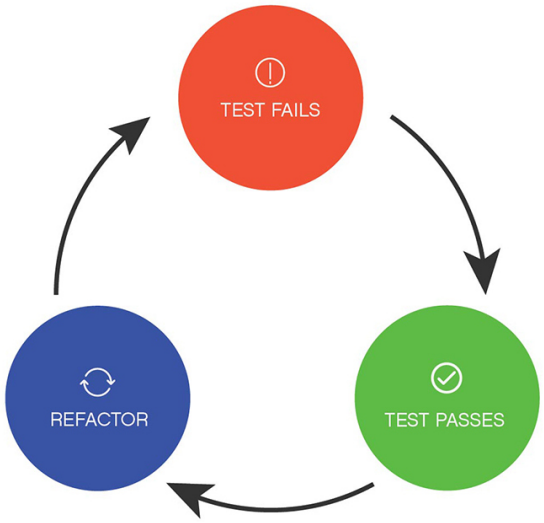

FIGURE 4.12 – Cycle de tests et de développement

Pour cela, il faut définir le comportement unitaire de chaque composant, ainsi qu'un ou plusieurs parcours d'utilisation critiques utilisant les éléments clés de l'application.
J'ai eu l'occasion de créer différents types de tests :

Les tests unitaires s'assurent que chaque composant de l'application rempli son rôle.

Les tests d'intégrations vérifient un fonctionnement comportant plusieurs composants.

Les test de bout en bout implémentent des User Stories complètes, utilisant toutes les couches de l'application.

Dans une démarche d'intégration continue, j'ai également participé à la mise en place de systèmes de tests et de déploiement automatisés sur la plateforme Azure DevOps.
J'ai pu utiliser les solutions SonarQube pour le Back-end et Eslint pour le Front-end qui s'occupent d'analyser le dépôt de code source afin d'y trouver des problèmes de sécurité, des bugs, des mauvaises pratiques ainsi que de la duplication de code.
Cela contribue à rendre le code plus stable et plus maintenable.
Voici une capture de l'interface de SonarQube :

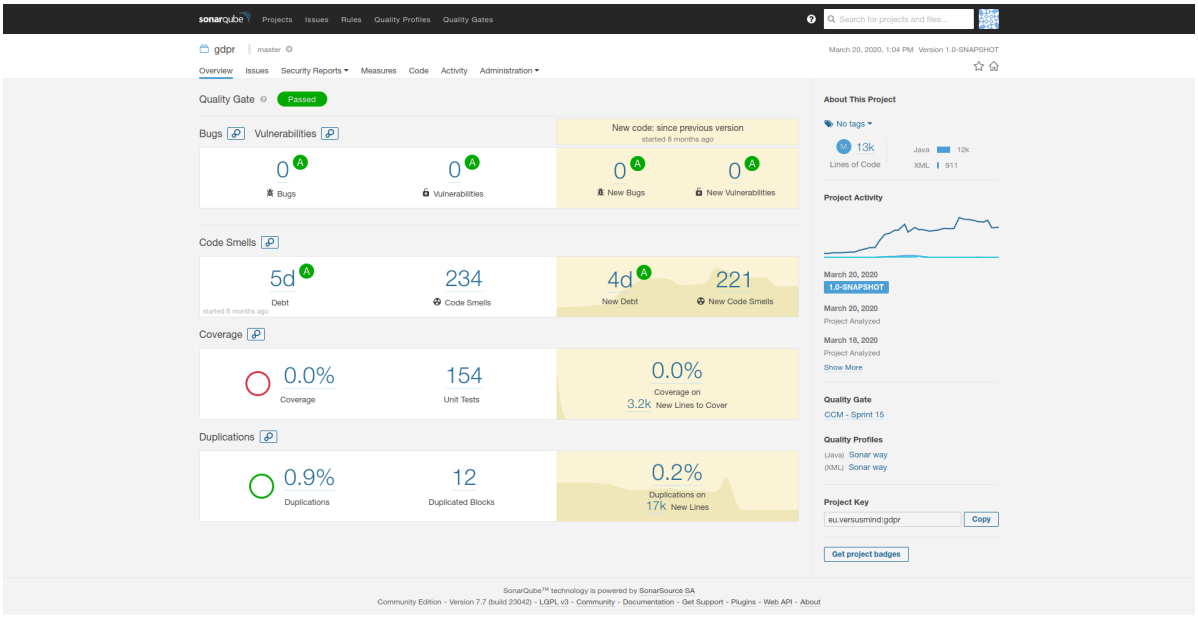

FIGURE 4.13 – Interface de SonarQube

Chapitre 5

Conclusion

Pour conclure, j'ai effectué mon stage de fin d'études en ingénierie informatique en tant que développeur pour une solution de gestion de consentements au sein de l'agence de Strasbourg de l'entreprise Versusmind.
Lors de ce stage de 6 mois, j'ai pu mettre en pratique mes connaissances théoriques acquises durant ma formation et me suis confronté aux difficultés du monde du travail dans le secteur du développement informatique. Ce stage m'a aussi permis de mieux cerner le métier de consultant.

Ce stage m'a beaucoup apporté, d'un point de vue technique mais aussi au niveau organisationnel.
C'est pour moi une première expérience professionnelle suivant une méthode agile et j'en suis satisfait. Les réunions régulières m'ont permises de prendre du recul sur le travail effectué, et le travail en équipe agile a créé une dynamique dans laquelle nous nous poussions mutuellement vers le haut.
J'y ai appris à suivre un processus de développement plus stable, grâce au développement dirigé par les tests ainsi qu'à l'intégration continue via Azure DevOps.
J'ai aussi appris à développer des tests de charge et à interpréter leurs résultats afin d'en tirer des directions à prendre.

En définitive, même si le développement web n'est pas mon domaine de prédilection, j'ai eu la chance d'utiliser de nombreux outils open source et cela m'a permis de confirmer ma volonté de participer au monde du logiciel libre. J'ai vécu une belle expérience durant ce stage et j'ai la chance de pouvoir rester au près de Versusmind et de continuer à approfondir des sujets qui me tiennent à cœur.

Annexe A

Personas

Les différents rôles dans l'application :

Versusmind

La société éditrice du logiciel

Propriétaire

La personne physique représentant le Responsable de traitement (Directeur général/-Président par exemple) et ayant la capacité d'engager l'organisme.

Admin client

Par exemple : Un RSSI et/ou un Délégué à la protection des données (DPO) qui doivent pouvoir avoir tous les droits de lecture/écriture ainsi qu'un droit de gestion des Utilisateurs / Consultant.
Le RSSI et le DPO gèrent conjointement le projet de mise en conformité au RGPD au sein d'un organisme.
Si aucun des deux acteurs n'est nommé au sein d'un organisme et n'a pas vocation à être nommé, il est recommandé de désigner un "Référent données personnelles" ayant en charge le projet de mise en conformité.

Utilisateurs

Par exemple : Le directeur/directrice du service communication de l'organisme qui souhaite créer une newsletter en lien avec les clients de l'organisme ou des prospects ayant consenti sur le site internet de l'organisme.
Cette newsletter nécessite le consentement des clients et doit apparaître au registre des traitements.
Le directeur/directrice du service communication complète donc la partie correspondante dans le registre après que le RSSI et/ou le DPO ou, à défaut, le "Référent données personnelles" lui a octroyé les droits.

Consultant

Un consultant est une personne qui n'a que des droits de lecture sur le registre ou le consentement.
Par exemple : Une personne du service communication qui doit suivre les consentement sans avoir besoin de les modifier.
Ou encore : Une stagiaire qui a besoin d'y accéder pour travailler mais n'a pas soit la responsabilité nécessaire, soit le besoin de bénéficier d'un droit d'écriture.

Annexe B

Guide CSS

B.0.1 Et si ca devenait agréable de faire du CSS ?

Pour beaucoup de développeurs, le CSS a une image d'un langage qu'il faut sans arrêt bricoler pour arriver à nos fins. Une des raisons principales est que le CSS est très permissif et ne propose pas d'architecture en lui-même, c'est un grand bac a sable. En conséquences, si un projet grossi, la structure CSS choisie (ou construite de manière empirique) finie par être un cauchemar à maintenir et à étendre tant il y a de collisions de règles. Un indicateur d'un projet ayant grossi sans avoir fait évoluer sa structure CSS est le nombre de **!important** dans le code. Cela montre des conflits n'ayant pas été résolus, mais masqués (Je reviendrais sur ce point).
Je vais donc ici vous proposer une architecture CSS qui va vous permettre d'organiser votre code, de le documenter, d'éviter les conflits mais aussi d'améliorer la maintenabilité et faciliter l'extension.
Rapidement pour que tout le monde soit d'accord sur les termes que j'utilise :

```
regle {
    propriété : valeur ;
    propriété : valeur !important ;
}
```

La précision

Afin de comprendre et d'éviter les conflits dans le code, il faut se préoccuper de la notion de 'précision' d'une règle CSS. Si il y a conflit sur une propriété, c'est la règle avec la plus grande précision qui l'emporte, sauf si il y a la directive **!important**. Si deux propriétés sont en conflit avec la même précision, c'est la règle déclarée en dernier qui l'emporte (l'ordre a son importance !).
La précision d'une règle augmente avec le nombre de membres qu'elle contient.

```
body { // précision : 1
[...]
}
```

```
body p { // précision : 2
[...]
}

body > p { // précision : 2
[...]
}

.container { // précision : 1
[...]
}

.container.disabled { // précision : 2
[...]
}
```

Attention à l'utilisation de préprocesseurs CSS, toujours penser à la précision des règles qui seront génerées :

```
body {
    [...]

    p {
        [...]

        &.selected {
            [...]
        }
    }
}
```

Compilera en :

```
body { // précision : 1
[...]
}

body p { // précision : 2
[...]
}

body p.selected { // précision : 3
[...]
}
```

La précision d'une règle contenant un identifiant est infinie, elle écrasera n'importe quelle propriété lors d'un conflit. Cela rend l'extension du code impossible, c'est donc à proscrire dans vos bases de code CSS.

```
#page { // précision infinie
 [...]
}
```

De manière générale, il faut essayer de toujours garder la précision la plus basse possible.

Structure HTML

Il arrive régulierement que lors de corrections ou d'extensions des fonctionnalités, qu'il faille changer la structure HTML d'une page. Si les règles CSS se basent exclusivement sur cette dernière, les propriétés ne cibleront peut-être plus les bons blocs HTML. Cela implique plus de maintenance et de la résilience à l'extension. Il faut donc éviter au maximum de se baser sur la structure HTML afin de définir du style. De manière générale il ne faut pas définir un élément en fonction de où il est, mais ce qu'il est. Il faut donc préférer définir des classes spécifiques qui concernent le cas d'application précis de l'élément.

```
nav button {
    [...]
}
```

Pourrait être :

```
.button--nav {
    [...]
}
```

Cela réduit la dépendence à la structure HTML et cela a aussi réduit la précision, parfait.

Découpage en composants + convention de nommage

Voici un point qui va grandement (et je pèse mes mots) améliorer la facilité de lecture et donc de maintenance de votre code. C'est la suite logique du point précédent où il va falloir définir des composants et les isoler de leur environnement. Je vous propose la convention de nommage BEM pour (Bloc, Element, Modifier) Cela va permettre de standardiser le nommage de vos classes et faciliter la lecture du code.
Prenons l'exemple d'un simple article :

- définition de la racine du bloc : **article**
- des éléments qui le composent : **titre**, **contenu**
- et enfin des spécialisations du bloc : **informatique**, **general**
- dans son propre fichier du nom du composant : **_article.scss**

```
.article {
    [...]
```

```
}

.article--menuiserie {
    [...]
}

.article--informatique {
    [...]
}

.article__titre {
    [...]
}

.article__contenu {
    [...]
}
```

La même chose avec Sass :

```
.article {
    [...]

    &--menuiserie {
        [...]
    }

    &--informatique {
        [...]
    }

    &__titre {
        [...]
    }

    &__contenu {
        [...]
    }
}
```

Et appliqué à du HTML :

```
<body>
<div class="article article--informatique">
<div class="article__titre">
Et si ça devenait agréable de faire du CSS ?
```

```
</div>
<div class="article__contenu">
[...]
</div>
</div>
<div class="article article—Menuiserie">
<div class="article__titre">
L'utilisation de la scie sauteuse en milieu urbain
</div>
<div class="article__contenu">
[...]
</div>
</div>
</body>
```

C'est en effet plus verbeux qu'une méthode moins structurée, mais à grande échelle, cela compartimente correctement les comportements attendus. Attention toutefois au nommage de vos spécialisations ! Il ne faut pas nommer une spécialisation par une valeur qu'elle comporte, car si cette valeur change, le nom de la classe n'aura plus de sens. Par exemple, ne pas utiliser **.article–rouge** mais plutôt spécifier la sémantique, le sens qu'il porte : **.article–important**.

!important

Maintenant que nous avons aborder comment éviter les conflits, pourquoi avoir besoin de la directive **!important** ? Pourquoi existe-t-elle si nous pouvons nous en passer ? Simplement car cette directive doit-être utilisée de manière proactive et non réactive. L'utiliser en réponse à un conflit que difficile à résoudre soulève un problème de précision dans l'architecture. Par contre, il est possible de l'utiliser dans un cas où nous savons déjà en amont qu'une propriété DOIT surpasser toute autre règle, par exemple dans le cas d'un plein écran :

```
.article {
    [...]

    &—fullscreen {
        position: absolute !important;
        width: 100vw !important;
        height: 100vh !important;
    }
}
```

Si ces propriétés ne sont plus désirées, c'est qu'il faut retirer la classe du HTML.

Les dépendances externes

Les dépendances externes peuvent parfois avoir des précisions plus grandes que les règles de notre code. Au lieu de placer la directive **!important** afin de résoudre le problème et passer à la suite, ou encore de se baser sur la structure du HTML pour augmenter la précision, voici une petite astuce :

```
.classe-externe.classe-externe { // Précision : 2
[...]
}
```

Cela augmente la précision de la règle, ce qui est obligatoire si nous voulons changer le comportement du code, mais sans aucune dépendance de classe et sans détruire l'arbre de précision avec un **!important**.

Documentation

Et enfin le meilleur pour la fin, la documentation. Après avoir refondu toute la structure et appliqué tous ces superbes conseils, il nous manque encore de la documentation afin de se rappeler du comportement d'un bloc, d'un élément ou d'une spécialisation. Je vous propose une documentation simple en entête de chaque fichier de composant. Grouper chaque élément et ses spécialisations et séparer les éléments entre eux afin de bien visualiser les différents groupes de règles :

```
// .article
//     Composant représentant un article de blog
// .article--menuiserie
//     Spécialisation d'un article concernant la menuiserie
// .article--informatique
//     Spécialisation d'un article concernant l'informatique
//
// .article__titre
//     Élement titre d'un article
//
// .article__contenu
//     Élement de contenu d'un article
//
// Exemple :
// <div class="article article--informatique">
//     <div class="article__titre">
//         [...]
//     </div>
//     <div class="article__contenu">
//         [...]
//     </div>
// </div>
```

```
.article {
    [...]

    &—menuiserie {
        [...]
    }

    &—informatique {
        [...]
    }

    &__titre {
        [...]
    }

    &__contenu {
        [...]
    }
}
```

Conclusion

Nous y voilà, nous avons vu qu'il faut décrire des composants pour ce qu'ils sont et non pour leur emplacement dans la page. Nous savons maintenant qu'il ne faut pas utiliser d'identifiant dans le CSS mais aussi qu'il faut utiliser **!important** avec parcimonie. Nous avons aussi vu qu'utiliser une convention de nommage en CSS permet d'ajouter du sens dans notre code sans augmenter la précision. Et enfin nous avons vu comment documenter nos classes CSS pour s'y retrouver et que chaque composant soit indépendant.
Si vous étiez fachés avec le CSS et que vous êtes arrivés jusqu'ici, déjà bravo, et j'espère vous avoir au moins un peu réconcilié avec ce language. Sinon j'espère vous avoir quand même apporté quelque chose au travers de cet article.

Table des figures

Mots clefs

Versusmind - RGPD - Consentement - Signature numérique - Cloud Microsoft Azure
Méthodologie Scrum - Azure DevOps - Angular - Spring Boot - Azure Functions

Nicolas BALLET

Rapport de stage ST50 - A2019

Résumé

J'ai pu, durant mon stage de fin d'études à Versusmind (Strasbourg), participer au développement d'un plateforme de recueil de consentement hébergée dans le cloud Microsoft Azure.
L'équipe dont j'ai fait partie, utilise la methode agile Scrum. J'ai aidé à terminer une refonte graphique, mais aussi, à faire face à des problèmatiques de montée en charge et d'optimisation. Le tout sur une base de tests d'intégrations à l'aide d'Azure DevOps.

Entreprise Versusmind
30 avenue du Rhin
67000 Strasbourg
versusmind.eu

utbm
université de technologie
Belfort-Montbéliard