Table des matières

Remerciements **vii**

Table des matières **ix**

1 Introduction **1**

2 Présentation de l'entreprise **3**

3 Présentation du Sujet et problématique **4**

3.1 Vocabulaire 4

3.2 Etat de l'art 5

3.2.1 Analyse des technologies 6

3.3 Présentation du sujet 16

3.3.1 Problématique et intérêt pour l'entreprise 16

3.3.2 Objectifs et Livrables du projet 17

4 Développement du projet **19**

4.1 Introduction 19

4.2 Analyse et Conception 19

4.2.1 Analyse des besoins 19

4.2.2 Conception de la solution 22

4.2.3 Interface utilisateur et flux de l'application 33

4.3 Choix des technologies et outils 39

4.3.1 Choix de l'environnement de développement 39

4.3.2 Choix des frameworks et bibliothèques 41

4.3.3 Choix des outils pour l'intégration de l'IA (ChatGPT) 44

4.4 Développement de l'application 48

4.4.1 Intégration de Boond Manager 48

4.4.2 Intégration de l'api Google 48

4.4.3 Angular Material Design . . . 49
4.5 Sécurité . . . 50
4.5.1 Notion de développement sécurisé . . . 50
4.5.2 Surveillance . . . 53
4.5.3 Analyse statique des vulnérabilités avec Sonar Cube . . . 54
4.5.4 Analyse dynamique des vulnérabilités . . . 54
4.5.5 Authentification dans l'Application . . . 55
4.5.6 Authentification sur l'application . . . 56
4.5.7 Intégrité et Confidentialité des données . . . 58
4.6 Déployement et Intégration continu . . . 59
4.6.1 Ansible . . . 59
4.6.2 Pipeline GitLab CI/CD . . . 60
4.7 Tests et validation . . . 61
4.7.1 Tests unitaires . . . 61
4.7.2 Tests d'intégration . . . 63
4.7.3 Validation par les utilisateurs . . . 65
4.8 Gestion de projet . . . 65
4.8.1 Méthodologie de gestion de projet . . . 65
4.8.2 Outils de gestion de projet . . . 66
4.9 Conclusion et perspectives . . . 67
4.9.1 Bilan du projet . . . 67
4.9.2 Limitations et perspectives d'amélioration . . . 68

5 Conclusion **69**

Bibliographie / Webographie **71**

Liste des illustrations **73**

Listings **75**

Annexes **78**

A Conception **79**

B Choix des technologies et outils **81**

B.1 Environnement de Développement Optimal avec Visual Studio Code (VSCode) . . . 81

C Déploiement **82**

C.1 Docker 82

D GANTT **84**

Résumé **87**

Abstract **87**

1 Introduction

Dans le cadre de la dernière année du cursus ingénieur en France, le stage de dernière année revêt une importance capitale. Il constitue une véritable immersion professionnelle permettant aux étudiants de mettre en pratique les connaissances théoriques acquises tout au long de leur formation. Ce stage, d'une durée généralement étendue, représente une passerelle entre le monde académique et le monde du travail. Il offre aux étudiants l'opportunité de développer leurs compétences techniques, de découvrir le fonctionnement concret d'une entreprise et de saisir les enjeux spécifiques à leur domaine de spécialisation. En tant que tremplin vers l'insertion professionnelle, ce stage de dernière année permet aux étudiants de valoriser leurs compétences, d'élargir leur réseau de contacts et, éventuellement, de décrocher une offre d'emploi future.

Tout au long de cette expérience, les étudiants bénéficient d'un encadrement tant de la part d'un tuteur en entreprise que d'un tuteur académique, favorisant ainsi l'échange de connaissances et de compétences entre les deux milieux. Ainsi, le stage de dernière année joue un rôle clé dans la formation des ingénieurs en France, consolidant leur apprentissage et les préparant à leur carrière professionnelle.

La réponse aux appels d'offres est une tâche complexe pour toute entreprise de services informatiques, en particulier pour le responsable technique (CTO). Bien qu'analyser les exigences et proposer une solution et un devis soient les éléments clés de la proposition, la réponse à un appel d'offres peut contenir de nombreuses sections, telles que la présentation de l'entreprise, le résumé exécutif, les exigences, la solution proposée, le devis, les références, les CV, les conditions générales et les politiques de confidentialité et de protection des données. Le temps passé à fournir ces éléments pourrait être mieux utilisé pour fournir une excellente solution au client avec des informations et des diagrammes pertinents.

C'est dans ce contexte qu'une application va être développée, destinée à faciliter la création de réponses à des appels d'offres et de contrats de maintenance pour une entreprise de conseil en informatique. Cette application est destinée à répondre aux besoins du CTO, qui doit souvent répondre à des appels d'offres (RFP) dans lesquelles il doit organiser et remplir les différentes sections du document à l'aide de sources telles que Google Docs, Google Sheets ou l'API d'OpenAI, afin de générer un modèle de document. Les réponses doivent être produites, en français ou en anglais, en suivant les directives de la charte graphique de l'entreprise. D'autres types de contrats, tels que des contrats de maintenance, doivent également être produits.

Le développement de cette application représente un défi intéressant qui permettra la mise en pratique des compétences en matière de conception, développement et sécurisation d'applications Web. Le processus de développement et de sécurisation de cette application Web ainsi que les outils et les technologies utilisés tout au long du projet seront présentés dans ce mémoire d'ingénieur. Pour atteindre l'objectif de ce mémoire, la présentation du contexte et des enjeux liés au développement d'applications Web dans le monde professionnel actuel sera effectuée en premier lieu. Les spécifications de l'application seront décrites en détail, en présentant les différentes fonctionnalités et les

exigences de sécurité requises. Les méthodologies et les outils utilisés tout au long du projet, ainsi que les défis et les problèmes rencontrés seront également expliqués. Enfin, les résultats obtenus et les perspectives d'avenir pour cette application seront présentés dans la conclusion.

2 Présentation de l'entreprise

Caractéristiques	Définition / Description
Type d'organisation	Société privée à but lucratif créée en 2000 par Laurent Kratz et Pierre Gerard [3]
Finalité = raison d'être	— **économique** : vendre des services de conseil, de développement et de gestion de projets informatiques — **social** : favoriser l'innovation technologique et la création d'emplois
Nature de l'activité et but	Fournir des services de conseil, développement et gestion de projets informatiques aux entreprises, spécialisaton dans les applications Web et Mobiles
Statut juridique	Société anonyme (SA)
Ressources	— Humaines : entre 50 et 70 collaborateurs — Matériel : Locaux du siège, possédant un à deux serveurs en service et matériel informatique (écrans, PC portables, ...)
Champ d'action géographique	International (présence dans plusieurs pays)
Taille (en fonction du nombre de salariés)	PME
Nationalité	Luxembourgeoise (siège social à Luxembourg)

Table 2.1 – Fiche de présentation de l'entreprise NEOFACTO Luxembourg

NEOFACTO est une entreprise de conseil en technologies et en innovation, qui propose des solutions sur-mesure pour aider les entreprises à relever les défis numériques de demain. Avec une équipe de plus de soixante experts, NEOFACTO offre une gamme complète de services, allant de la conception et le développement de logiciels à la gestion de projets, en passant par la transformation digitale. Elle se distingue par son approche orientée client, son expertise technique et sa capacité à innover pour répondre aux besoins spécifiques de chaque entreprise. Elle se positionne comme un partenaire stratégique pour accompagner les entreprises dans leur transformation digitale et les aider à atteindre leurs objectifs de croissance et de performance.

3 Présentation du Sujet et problématique

3.1 Vocabulaire

Voici une compilation de termes employés au sein de ce mémoire, conçue pour familiariser le lecteur avec le contexte, et qui requièrent des définitions explicites tout au long de cet ouvrage.

- **Front-end** : La partie d'une application web ou logicielle avec laquelle les utilisateurs interagissent directement. Cela inclut l'interface utilisateur, les éléments visuels et les fonctionnalités accessibles côté client.
- **Back-end** : La partie d'une application qui gère les fonctionnalités en coulisses, tels que le stockage de données, le traitement des requêtes et la logique de l'application. Elle est gérée côté serveur.
- **Endpoint** : Un point d'accès spécifique dans une API (Interface de Programmation Applicative) qui expose une fonctionnalité ou une ressource particulière aux clients. Chaque endpoint est identifié par une URL unique et est associé à une méthode HTTP spécifique (comme GET, POST, PUT ou DELETE) pour indiquer le type d'action à effectuer sur cette ressource.
- **TypeScript** : Un sur-ensemble de JavaScript qui ajoute des fonctionnalités de typage statique. Cela permet de détecter et de prévenir les erreurs potentielles dès la phase de développement.
- **Callback** : Une fonction qui est passée comme argument à une autre fonction et est ensuite invoquée après l'exécution de cette fonction. C'était une méthode courante pour gérer l'asynchronicité avant l'introduction d'outils plus intuitifs tel que async/await.
- **Framework** : Un ensemble structuré d'outils, de bibliothèques et de conventions qui simplifient le développement d'applications en fournissant un cadre préétabli pour les tâches récurrentes.
- **12-factor app** : Une méthodologie de développement de logiciels qui fournit des meilleures pratiques pour la création d'applications modernes et évolutives. Les "12 factors" définis par cette méthodologie englobent des aspects tels que la configuration, le stockage, la gestion des dépendances et la scalabilité, visant à rendre les applications plus flexibles et faciles à gérer dans des environnements variés.
- **Pipeline** : Une pipeline est une séquence d'étapes automatisées qui permettent de construire, tester et déployer un logiciel de manière cohérente et efficace, garantissant une livraison continue et fiable. On peut facilement automatiser l'éxecution

3.2 Etat de l'art

La réponse à un appel d'offre est un processus complexe qui peut prendre beaucoup de temps et d'efforts. Actuellement, la plupart des entreprises utilisent une méthode manuelle pour générer des CVs et répondre aux appels d'offres. Cela implique souvent la recherche manuelle de compétences spécifiques dans les CVs des candidats, la sélection des CVs pertinents pour répondre à un appel d'offres, la personnalisation des CVs pour répondre aux exigences spécifiques de l'appel d'offres, et enfin la soumission de la candidature à l'appel d'offres.

Ce processus manuel peut entraîner une perte de temps et d'énergie pour les entreprises, car il nécessite un travail fastidieux et souvent répétitif. De plus, cela peut entraîner des erreurs humaines, telles que la sélection de CVs incorrects ou la soumission de candidatures inexactes.

Il peut donc être difficile de suivre les différentes demandes d'appels d'offres et de gérer les candidatures, car cela peut nécessiter des efforts supplémentaires pour surveiller les délais de soumission, répondre aux questions des clients, et effectuer des suivis auprès des candidats.

Ainsi, il est important pour NEOFACTO de trouver une solution automatisée pour répondre aux appels d'offres, qui permettra d'économiser du temps et de l'énergie tout en réduisant les erreurs humaines. Une telle solution pourrait inclure un système de gestion des CVs qui permettrait de trier et de classer automatiquement les CVs des candidats, ainsi qu'un système de réponse aux appels d'offres qui permettrait de générer automatiquement un modèle de document à remplir, ainsi que des candidatures en fonction des exigences de l'appel d'offres.

Dans le domaine des réponses aux appels d'offres, il existe déjà plusieurs solutions en ligne, comme rfp.plus[8] ou encore loopio.com[6], qui facilitent le processus de manière significative. Rfp.plus, sorti en 2021, propose une plateforme d'automatisation partielle permettant de gérer les réponses aux appels d'offres de manière efficace et précise. De son côté, Loopio, sorti en 2023, offre un logiciel de réponses aux appels d'offres qui permet de fournir les meilleures propositions dans les délais impartis.

Loopio met l'accent sur la qualité et la rapidité en offrant un logiciel de réponses aux appels d'offres pour les RFIs, les questionnaires de sécurité, les DDQs, et bien d'autres. Grâce à son système de gestion de bibliothèque de réponses aux appels d'offres, Loopio permet d'organiser, de rechercher et d'accéder facilement à tout le contenu pertinent. Avec des fonctionnalités de recherche puissantes et un système de saisie de contenu convivial, trouver les meilleures réponses devient rapide et facile.

Les deux solutions, rfp.plus et Loopio, offrent des fonctionnalités avancées pour gérer les réponses aux appels d'offres de manière efficace. Rfp.plus se distingue par sa flexibilité et sa capacité à s'adapter aux différentes structures d'entreprises, qu'il s'agisse d'entreprises proposant un seul produit ou de projets spécifiques. D'autre part, Loopio met l'accent sur l'organisation et la recherche rapides des réponses grâce à son système de gestion de bibliothèque bien structuré et à ses capacités de recherches avancées.

Les deux solutions offrent des fonctionnalités pour maintenir le contenu à jour, que ce soit par le biais de l'automatisation intelligente de rfp.plus ou des cycles de révision automatisés et des outils de détection de contenu neuf de Loopio. Elles offrent également des options de personnalisation, telles que l'organisation par catégories et tags personnalisés, afin de répondre aux besoins spécifiques de chaque entreprise.

Il existe donc des solutions modernes et puissantes pour répondre aux appels d'offres de manière efficace. Elles offrent des fonctionnalités avancées pour gérer, organiser et rechercher les ré-

ponses, permettant aux entreprises de gagner du temps, d'améliorer leur efficacité et d'accroître leur taux de réussite dans les appels d'offres. Le choix entre les deux dépendra des besoins spécifiques de chaque entreprise et de ses préférences en matière de fonctionnalités et d'interfaces utilisateur.

3.2.1 Analyse des technologies

Dans le cadre du développement de l'application, une étude comparative a été effectuée afin d'analyser les différentes technologies pouvant être utilisées pour le front-end, le back-end et la base de données. Cette étude vise à évaluer les options disponibles et à prendre des décisions éclairées pour le choix des langages de programmation, des frameworks, des bibliothèques et des bases de données les mieux adaptés à nos besoins. L'objectif est de garantir la performance, la sécurité et la stabilité de l'application. Au cours de cette étude comparative, nous examinerons les avantages et les inconvénients de chaque technologie, en tenant compte de nos exigences fonctionnelles et non fonctionnelles. Les résultats de cette analyse nous permettront de sélectionner les technologies les plus appropriées pour chaque composant de notre application, en maximisant ainsi ses chances de succès.

Pour chacune des composantes de l'application, nous allons d'abord évaluer l'importance de différents critères de sélections parmis une liste quasi-exhaustive, puis nous allons comparer les différents frameworks et langages parmi les plus utilisés actuellement à l'aide des critères retenus. Nous pourrons ainsi faire un choix réfléchit.

Dans le cadre du développement de l'application, une étude comparative approfondie a été entreprise pour évaluer les différentes composantes de celle-ci. Pour chaque composant, une analyse minutieuse des critères de sélection pertinents a été effectuée, par rapport aux besoins identifiés de l'application, parmi une liste de caractéristiques qui se veut quasi-exhaustive. Ensuite, une comparaison détaillée des frameworks et langages les plus couramment utilisés a été réalisée en se basant sur les critères préalablement identifiés. Cette approche rigoureuse nous permettra de prendre des décisions éclairées et réfléchies quant aux choix technologiques à adopter pour chaque composant de l'application. En mettant l'accent sur l'importance des critères de sélection et en évaluant attentivement les options disponibles, nous visons à sélectionner les solutions optimales qui répondront le mieux à nos besoins spécifiques.

On retrouve sur la figure 3.1 ci dessous les différents niveaux de priorité assignés à chaque critères d'évaluation.

FIGURE 3.1 – Visualisation des différents niveaux de priorité assignés à chaque critères d'évaluation

Pour chaque critère et pour chaque technologie, on utilisera les trois niveaux décrits sur la figure 3.2 pour définir les capacité des technologies à respecter les critères. Un justification résumée est présente en plus du code couleur pour expliciter l'évaluation.

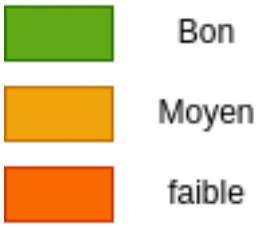

FIGURE 3.2 – Légende des couleurs associées à l'évaluation d'une technologie sur un critère

Front-end

On retrouve dans le tableau suivant les différents niveau de priorité pour différents critères d'évaluation en fonction des besoins de l'application, pour le front-end.

Critère d'évaluation	Détails	Priorité
Performance	Quelle est la performance offerte par le framework pour l'application ?	normal
Facilité d'utilisation	À quel point est-il facile d'apprendre et d'utiliser le framework ?	normal
Communauté de développeurs	Quelle est la taille et l'activité de la communauté de développeurs pour chaque framework ?	important
Flexibilité	Dans quelle mesure le framework est-il flexible pour s'adapter aux différents besoins de l'application ?	normal
Scalabilité	Dans quelle mesure le framework peut-il gérer des applications larges et complexes ?	moins important
Documentation	À quel point la documentation du framework est-elle complète et facile à comprendre ?	normal
Tests	À quel point est-il facile d'écrire et d'exécuter des tests pour le framework ?	Très important
Accessibilité	Dans quelle mesure le framework prend-il en charge les exigences en matière d'accessibilité ?	pas important
SEO (référencement)	Dans quelle mesure le framework prend-il en charge les exigences de référencement ? (facilité de recherche sur internet)	inutile
Rendu côté serveur	Dans quelle mesure le framework prend-il en charge le rendu côté serveur ?	important
Sécurité	Dans quelle mesure le framework prend-il en charge les meilleures pratiques de sécurité ?	Très important
Support mobile	Dans quelle mesure le framework prend-il en charge les appareils mobiles ?	important
Plugins et modules communautaires	Quelle est la disponibilité et la qualité des plugins et modules tiers ?	Très important
Mises à jour du framework	À quelle fréquence le framework est-il mis à jour et quel est le processus de mise à jour ?	important
Sûreté de type	Dans quelle mesure le framework prend-il en charge la sûreté de type ?	important
Gestion de l'état	Dans quelle mesure le framework gère-t-il la gestion de l'état ?	normal

Après avoir analysé la table des critères d'évaluation pour le développement du front-end, certains points ont été identifiés comme importants voire très importants. La performance et la vitesse ont été évaluées comme des critères de priorité normale, car il est essentiel que le front-end réponde rapidement et efficacement aux différents besoins des utilisateurs. La scalabilité et le support de la concurrence sont considérés comme moins importants, car le front-end ne traite généralement pas directement un grand nombre de requêtes concurrentes.

Le support de la communauté et la disponibilité des bibliothèques ont été jugés importants, car ils garantissent un écosystème solide de ressources, d'outils et de solutions pour le développement

du front-end. La fiabilité et la stabilité sont des critères très importants, car le front-end doit être fiable et stable dans des environnements de production afin de fournir une expérience utilisateur cohérente.

Les fonctionnalités de sécurité et la gestion des vulnérabilités sont également classées comme très importantes pour assurer la protection des données et la sécurité des utilisateurs. L'intégration avec d'autres technologies et services est considérée comme importante, car le front-end doit pouvoir se connecter de manière fluide avec d'autres systèmes et services.

En conclusion, les critères importants ou très importants pour le développement du front-end incluent la performance et la vitesse, la fiabilité et la stabilité, la sécurité, le support de la communauté et la disponibilité des bibliothèques, ainsi que l'intégration avec d'autres technologies et services. Ces points sont essentiels pour garantir un front-end efficace, sécurisé et évolutif, offrant une expérience utilisateur optimale.

On retrouve sur le tableau suivant l'évaluation de trois technologies de front-end, Vue.js, React.js et Angular, selon les critères retenus.

Technology	Community (March 2023)	Testing	Server-side rendering	Security (Built-ins)	Mobile support	Plugins and Modules	Updates	Type safety
Vue.js	200,000 stars on GitHub 1.5 million weekly downloads on npm subreddit with 140,000 members	built-in unit test, external integration testing	Nuxt.js static page rendering simple and intuitive API	XSS, CSP	NativeScript or Framework7 supports Progressive Web Applications	Vast ecosystem of third-party plugins (Vue CLI's built-in plugin system) some being highly polished and actively maintained some being outdated or no longer maintained.	major release = once year backward-compatible Vue CLI simplify process of upgrading	ensured by TypeScript support
React.js	190,000 stars on GitHub 15 million weekly downloads on npm subreddit with over 350,000 members	external unit test, external integration testing	Next.js static page rendering simple and intuitive API	XSS, CSP +React Developer Tools browser extension	React Native and React Native Web supports Progressive Web Applications	Vast ecosystem of third-party plugins (npm) some being highly polished and widely used some being less popular or experimental	new releases coming out every few months may require changes to the codebase create-react-app tool simplify process of upgrading	ensured by TypeScript support
Angular	76,000 stars on GitHub 1 million weekly downloads on npm subreddit with over 150,000 members	built-in unit test, built-in integration testing	Angular Universal Angular animations, lazy-loading modules,	XSS, CSP, CSRF + Angular Security Best Practices guide	NativeScript, Ionic Framework better in mobile-first design supports Progressive Web Applications	Vast ecosystem of third-party plugins (npm + Angular Package Manager) some being highly polished and widely used some being less popular or experimental	major releases < once a year Big changes = may require changes to the code robust CLI tool + comprehensive documentation	ensured by TypeScript support (native)

FIGURE 3.3 – Table de justification des choix de la technologie du front-end

Après avoir évalué les différentes technologies pour le back-end en fonction des critères définis, notre choix s'est porté sur Angular. Bien que toutes les technologies évaluées, à savoir Vue.js, React.js et Angular, présentent des avantages significatifs, Angular se démarque en répondant à nos besoins spécifiques.

En examinant le tableau d'évaluation, il est clair que React.js ne présente aucun avantage en termes de mises à jour et de tests, même si des solutions existent. Les nouvelles versions fréquentes de React.js peuvent entraîner des modifications dans le code existant, ce qui peut entraîner des complications lors des mises à jour. De plus, bien que React.js dispose d'un écosystème riche en

plugins, certains d'entre eux peuvent être moins populaires ou expérimentaux, ce qui peut limiter les options disponibles pour le développement de notre application. Comme pour une application Node.js classique, il faut choisir et importer une bibliothèque de testing pour revenir au niveau des autres frameworks.

D'autre part, bien que Vue.js ait une communauté active et une facilité d'utilisation, il présente moins de force en matière de sécurité, de prise en charge mobile et d'écosystème de plugins. Les préoccupations concernant la sécurité et la vulnérabilité peuvent être critiques pour notre application, et la compatibilité mobile est également un facteur essentiel pour répondre aux besoins de nos utilisateurs. De plus, un vaste écosystème de plugins est essentiel pour accéder à des fonctionnalités avancées et des intégrations avec d'autres technologies.

C'est pourquoi nous avons choisi Angular comme technologie pour le back-end de notre application. Angular présente des avantages solides en termes de performance, de stabilité et de sécurité. Avec Angular, nous bénéficions d'une communauté active, d'un écosystème de plugins vaste et d'une prise en charge mobile solide grâce à des frameworks tels que NativeScript et Ionic Framework. De plus, Angular facilite la gestion des mises à jour grâce à son outil CLI robuste et à sa documentation complète. La prise en charge du typage grâce à TypeScript est également un avantage supplémentaire pour garantir la qualité et la maintenabilité du code.

En résumé, notre choix d'Angular repose sur ses forces en matière de stabilité, de sécurité, de compatibilité mobile, d'écosystème de plugins et de facilité de mise à jour. Nous sommes convaincus que cette technologie répondra aux exigences de notre projet et nous permettra de développer une application performante, sécurisée et évolutive.

Back-end

On retrouve dans le tableau suivant les différents niveau de priorité pour différents critères d'évaluation en fonction des besoins de l'application, pour le back-end.

Critère d'évaluation	Question	Priorité
Performance et vitesse	À quelle vitesse et efficacité la technologie répond-elle à différents besoins ?	normal
Scalabilité et support de la concurrence	La technologie peut-elle gérer un grand nombre de requêtes concurrentes ? Dans quelle mesure est-elle évolutive ?	moins important
Support de la communauté et disponibilité des bibliothèques	Existe-t-il une large communauté et un écosystème de bibliothèques disponibles pour la technologie ?	important
Fiabilité et stabilité	Dans quelle mesure la technologie est-elle fiable et stable en environnements de production ?	très important
Fonctionnalités de sécurité et vulnérabilités	Quelles fonctionnalités de sécurité la technologie offre-t-elle ? Y a-t-il des vulnérabilités connues ?	très important
Intégration avec d'autres technologies et services	Dans quelle mesure la technologie s'intègre-t-elle avec d'autres technologies et services ?	important
Flexibilité et extensibilité	Dans quelle mesure la technologie est-elle flexible et extensible pour s'adapter aux besoins changeants ?	normal
Utilisation de la mémoire et efficacité des ressources	Comment la technologie gère-t-elle et utilise-t-elle les ressources système ?	moins important
Gestion des erreurs et capacités de débogage	À quel point est-il facile de gérer et de déboguer les erreurs dans la technologie ?	moins important
Documentation et facilité de déploiement	À quel point est-il facile de déployer et de maintenir des applications construites avec la technologie ? Quelle est l'exhaustivité de la documentation ?	normal
Outils de test et de débogage	Quels sont les outils de test et de débogage disponibles pour la technologie ?	très important
Intégration avec les bases de données et prise en charge de l'ORM	Dans quelle mesure la technologie s'intègre-t-elle avec les bases de données ? Y a-t-il une prise en charge de l'ORM (Mapping Objet-Relationnel) ?	important
Fonctionnalités de mise en cache et d'optimisation des performances	La technologie offre-t-elle des fonctionnalités de mise en cache et d'optimisation des performances ?	normal
Intégration avec les services tiers et prise en charge de l'architecture microservices	Dans quelle mesure la technologie prend-elle en charge l'intégration avec les services tiers et la construction d'architectures microservices ?	pas important

Après avoir examiné le tableau des critères d'évaluation pour le développement du back-end, nous avons identifié certains points comme importants voire très importants. La performance et la vitesse ont été évaluées comme des critères de priorité normale, car le back-end doit répondre efficacement aux différents besoins fonctionnels de l'application.

La scalabilité et le support de la concurrence sont considérés comme moins importants, car l'application est destinée à un faible nombre d'utilisateur. Cependant, il est tout de même important

de prendre en compte ces aspects pour s'assurer que le back-end peut gérer une charge de travail croissante.

Le support de la communauté et la disponibilité des bibliothèques sont classés comme importants, car ils permettent d'avoir un écosystème solide de ressources et de solutions pour le développement du back-end. La fiabilité et la stabilité sont des critères très importants, car le back-end doit être fiable et stable en environnement de production pour assurer le bon fonctionnement de l'application.

Les fonctionnalités de sécurité et la gestion des vulnérabilités sont également classées comme très importantes, car le back-end est responsable de la protection des données et doit être robuste contre les menaces potentielles. L'intégration avec d'autres technologies et services est considérée comme importante, car le back-end doit pouvoir se connecter de manière transparente avec d'autres systèmes et services.

En conclusion, les critères importants ou très importants pour le développement du back-end incluent la performance et la vitesse, la fiabilité et la stabilité, la sécurité, le support de la communauté et la disponibilité des bibliothèques, ainsi que l'intégration avec d'autres technologies et services. Ces points sont essentiels pour garantir un back-end performant, sécurisé et capable de s'intégrer efficacement à l'écosystème technologique global de l'application.

On retrouve sur le tableau suivant l'évaluation de trois technologies de back-end, Node.js, Golang et Python, selon les critères retenus.

Technology	Community	Reliability / Stability	Security (Built-ins)	Integration	Testing	ORM Support	Performance
Node.js (Express.js)	Over 16,000 stars and more than 750 contributors on GitHub vast ecosystem of third-party plugins	PayPal, Uber, and IBM	TLS/SSL encryption CSRF protection HTTP response header configuration (XSS and clickjacking attacks) Middleware ecosystem (rate limiting and input validation) In the past: CSRF and JSON hijacking vulnerabilities (Patched)	Compatible with MongoDB, MySQL, and PostgreSQL Work with React, Vue, and Angular. Built-in support for JSON	• Mocha: Testing Express.js applications. • Chai: A BDD / TDD assertion library. • Supertest: Testing HTTP endpoints and APIs. • Debug: A debugging utility.	• Mongoose: Object Data Modeling library for MongoDB. • Sequelize: Object-Relational Mapping library for SQL databases. • Knex.js: SQL query builder.	High performance and low overhead (Node.js) Middleware overhead: can add some overhead to requests, which can affect performance.
Golang (Gin)	Over 22,000 stars and more than 380 contributors on GitHub Growing community (new) Lack of complex abstractions (requires more coding for simple features)	Google and Netflix	secure cookies CORS protection secure sessions Middleware ecosystem (rate limiting and input validation) In the past: path traversal and SQL injection attacks (Patched)	Compatible with MongoDB, MySQL, and PostgreSQL Work with React, Vue, and Angular.	• Go Test: Standard testing framework for Go applications. • Testify: Additional functionality and assertions. • Ginkgo: A BDD-style testing. • Gin Debug: Debugging information for requests and responses.	• GORM: Object-Relational Mapping library for SQL databases. • sqlx: Database toolkit for SQL databases. • go-pg: Object-Relational Mapping library for PostgreSQL.	Exceptional performance and low overhead (Go) Middleware overhead: can add some overhead to requests, which can affect performance.
Python (FastAPI)	Over 36,000 stars and more than 500 contributors on GitHub Vast ecosystem of third-party python plugins	NASA and Microsoft	TLS/SSL encryption CSRF protection JSON Web Tokens (JWT) authentication Middleware ecosystem (rate limiting and input validation) Pydantic library for data validation and serialization In the past: arbitrary code execution using the Pydantic library (Patched)	Compatible with MongoDB, MySQL, and PostgreSQL Work with React, Vue, and Angular. Built-in support for popular data formats such as JSON and OAuth2	• Pytest: Testing framework for Python. • Hypothesis: Property-based testing. • FastAPI TestClient: Built-in testing client for FastAPI applications, (endpoints and APIs). • logging: Built-in logging module for debugging.	• SQLAlchemy: Object-Relational Mapping library for SQL databases. • Tortoise-ORM: Asyncio Object-Relational Mapping library for SQL databases. • Pydantic ORM: ORM-like library for working with databases. • Motor : Coroutine-based API for non-blocking access to MongoDB.	Raw performance, slower than Go and Node.js. Async/await technique and type hinting to optimize performance and reduce overhead. ASGI server: (Asynchronous Server Gateway Interface) can improve performance and efficiency compared to traditional WSGI (Web Server Gateway Interface) servers.

FIGURE 3.4 – Table de justification des choix de la technologie du back-end

Après avoir évalué les différentes technologies pour le back-end en fonction des critères définis, notre choix s'est porté sur Node.js. Bien que Golang et Python aient également présenté des avantages significatifs, Node.js répond le mieux à nos besoins spécifiques.

En examinant le tableau d'évaluation, il est clair que Golang présente des avantages en termes de performances exceptionnelles et de faible surcharge. Cependant, sa petite communauté et ses lacunes en matière de sécurité et de support d'ORM ont été des facteurs décisifs pour ne pas retenir cette technologie.

D'autre part, Python, notamment avec FastAPI, a une communauté solide, une vaste gamme de plugins tiers et une excellente sécurité intégrée. Cependant, la performance brute de Python est inférieure à celle de Node.js, ce qui peut être un facteur critique pour notre application, qui nécessite une grande réactivité et une faible latence.

En plus des facteurs mentionnés précédemment, un autre élément déterminant dans notre choix d'opter pour Node.js en tant que technologie backend est l'infrastructure et l'expertise existantes au sein de notre entreprise.Neofacto dispose déjà d'une solide base et d'une expérience en matière de Node.js pour l'hébergement et la gestion d'applications back-end.

La normalisation sur une pile technologique spécifique, telle que Node.js, offre plusieurs avantages en termes d'intégration, de sécurité, de complexité, de maintenance et de support. En utilisant une technologie cohérente pour l'ensemble de nos applications backend, nous évitons les problèmes potentiels d'intégration liés à l'utilisation de différentes technologies. Cela nous permet également de mettre en place des mesures de sécurité robustes et de suivre systématiquement les meilleures pratiques.

De plus, la normalisation facilite la formation et le partage des connaissances au sein de l'organisation. Avec une pile technologique cohérente, les programmes de formation peuvent être rationalisés et les développeurs peuvent partager leurs connaissances et leurs expériences plus efficacement. Cela conduit à une productivité accrue et à une réduction des coûts associés à la formation et à l'intégration des nouveaux collaborateurs.

C'est pourquoi nous avons choisi Node.js comme technologie pour le back-end de notre application. Avec Node.js, nous bénéficions d'une communauté active, d'un écosystème riche en plugins tiers et d'une performance élevée avec une faible surcharge. Node.js est largement utilisé dans l'industrie et bénéficie d'un large soutien, avec des entreprises renommées telles que PayPal, Uber et IBM qui l'adoptent pour leurs besoins de développement.

En choisissant Node.js, nous pouvons tirer parti de son cadre populaire Express.js pour le développement rapide de l'API. De plus, Node.js offre une intégration transparente avec les bases de données courantes telles que MongoDB, MySQL et PostgreSQL, ainsi qu'avec les frameworks frontend populaires tels que React, Vue et Angular.

Il est également important de noter que Node.js dispose d'un solide ensemble d'outils de test, tels que Mocha, Chai, Supertest et Debug, qui facilitent la mise en œuvre de tests unitaires et d'intégration pour garantir la qualité du code.

Pour conclure, notre choix de Node.js repose sur ses forces en termes de performance, de communauté active, d'écosystème de plugins et d'intégration avec les bases de données et les frameworks frontend. Nous sommes convaincus que cette technologie répondra aux exigences de notre projet et nous permettra de développer un back-end robuste, réactif et évolutif pour notre application.

Base de données

Dans le cadre de notre étude comparative des bases de données pour notre projet, nous avons examiné les caractéristiques et fonctionnalités des bases de données relationnelles, NoSQL et basées sur les graphes. La base de données relationnelle, telle que MySQL, offre une structure tabulaire permettant des opérations de jointure complexes et une intégrité des données. Elle convient bien aux applications nécessitant des relations strictes entre les données.

D'autre part, les bases de données NoSQL, comme MongoDB, sont conçues pour gérer des données sans schéma et évolutives. Elles offrent une grande flexibilité et une évolutivité horizontale, ce qui en fait un choix adapté pour les applications nécessitant une grande volumétrie de données et des schémas flexibles.

Enfin, les bases de données basées sur les graphes, telles que Neo4j, se concentrent sur les

relations entre les données. Elles permettent de modéliser et de naviguer efficacement dans des structures de données complexes, ce qui les rend particulièrement adaptées pour les applications mettant l'accent sur les relations et les interconnexions entre les entités.

Dans le contexte de notre projet, qui vise à développer une application d'aide à la construction de réponses à des appels d'offres, nous avons identifié que la gestion des relations entre les différents éléments, tels que les sections, les modèles et les sources de données, est d'une importance primordiale. Les interdépendances entre ces éléments peuvent être modélisées en utilisant une base de données soit basée sur les graphes soit relationnelle.

Ainsi, en considérant les exigences spécifiques de notre projet et la nature des données à gérer, nous avons déterminé que Neo4j est l'option la plus appropriée. Son modèle de données orienté graphe et ses fonctionnalités avancées dans les requêtes et de navigation nous permettront de modéliser et de gérer les relations complexes entre les différentes entités de manière efficace. De plus, la flexibilité de Neo4j nous permettra d'adapter facilement notre modèle de données en fonction des évolutions futures de notre application.

En conclusion, l'utilisation de Neo4j comme base de données dans le projet est justifiée par sa capacité à gérer efficacement les relations entre les entités, en garantissant une modélisation précise et une navigation aisée. Cela nous permettra de développer une application répondant aux exigences spécifiques de notre domaine, en offrant une expérience utilisateur optimisée et une gestion efficace des données.

3.3 Présentation du sujet

Le sujet de ce mémoire porte sur le développement d'une application pour l'entreprise NEOFACTO, visant à aider les membres de l'entreprise dans la construction de documents de type réponse à appel d'offres et contrat de maintenance. Cette application permettra à la direction technique de répondre plus efficacement aux appels d'offres en proposant une structure standardisée pour les différentes sections du document. Elle permettra également d'organiser et de remplir automatiquement les différentes sections à l'aide de différentes sources, telles que Google Docs, Google Sheets ou encore l'API de ChatGPT. L'objectif principal de cette application est d'optimiser le temps passé à la production de ces documents tout en respectant les guidelines de branding de l'entreprise.

Cette application sera développée en utilisant les dernières technologies de développement web. Pour le front-end, nous utiliserons Angular, une plateforme de développement d'applications web en JavaScript maintenue par Google. Pour le back-end, nous utiliserons le framework Express.js de Node.js. Cette combinaison de technologies permettra de créer une application réactive, performante et facile à maintenir.

Le développement de l'application suivra une méthodologie agile, en suivant les principes de développement continu et de déploiement continu. Ainsi, les fonctionnalités pourront être itérativement développées, testées et déployées, permettant une rétroaction rapide et une adaptation en fonction des besoins.

En ce qui concerne la sécurité de l'application, elle sera abordée selon deux aspects : l'infrastructure et l'applicatif. Pour l'infrastructure, nous veillerons à maîtriser et documenter les accès aux serveurs et bases de données. Pour l'applicatif, une attention particulière sera portée à la gestion des identifiants permettant d'interagir avec d'autres API ou bases de données, afin qu'ils ne soient pas exposés dans le code source, conformément aux principes d'une application respectant les "12-factor app". De plus, la gestion de l'authentification et des autorisations sera un enjeu majeur pour assurer la sécurisation de l'accès à l'application et aux données.

Le choix des technologies est justifié dans la section de l'état de l'art, où un comparatif des technologies adaptées au développement de cette application est présenté. Ce choix est établi en fonction de plusieurs critères qui sont évalués en prenant en compte les fonctionnalités attendues dans cette application. En se concentrant sur le développement d'une seule application pour la construction de réponses à appels d'offre, nous pouvons nous concentrer sur l'optimisation de cette fonctionnalité spécifique et garantir une meilleure qualité et cohérence de l'application finale.

Avec cette nouvelle orientation, l'application répondra aux besoins spécifiques de NEOFACTO en matière de gestion des réponses à appels d'offre et de contrats de maintenance, en offrant une solution efficace et adaptée aux besoins de l'entreprise.

3.3.1 Problématique et intérêt pour l'entreprise

L'entreprise NEOFACTO est confrontée au défi de répondre efficacement aux appels d'offres en proposant des offres de qualité tout en respectant les délais impartis. Actuellement, la construction des réponses à ces appels d'offres demande beaucoup de temps et d'efforts, car les informations pertinentes sont dispersées et leur compilation est fastidieuse. De plus, il est essentiel de garantir la cohérence des informations et de respecter les guidelines de branding de l'entreprise dans chaque réponse.

La construction des réponses demandent également des compétences avancées afin de pouvoir répondre de manière cohérente à l'appel d'offre, ainsi que pour chiffrer les services fournis. C'est pourquoi le CTO de NEOFACTO s'occupe actuellement de la rédaction de ces documents, et le travail ne peut être délégué à un collaborateur dédié, qui n'a pas forcément les compétences techniques pour répondre efficacement. Le temps passé à construire manuellement la réponse est donc également très coûteux.

Dans ce contexte, il est nécessaire de développer un outil qui permettra à NEOFACTO de faciliter et d'optimiser le processus de construction des appels d'offres. Cet outil offrira une structure standardisée pour les différentes sections du document, facilitera l'organisation des informations et proposera des fonctionnalités d'autocomplétion à partir de différentes sources, telles que Google Docs, Google Sheets ou encore l'API de ChatGPT. De plus, il permettra de mettre à jour les réponses de manière centralisée, assurant ainsi la cohérence des informations.

La mise en place de cet outil d'aide à la construction d'appels d'offres permettra à NEOFACTO de gagner du temps et de réduire les efforts nécessaires pour répondre aux appels d'offres. L'entreprise pourra bénéficier d'une base de connaissances centralisée, garantissant l'accessibilité et la réutilisabilité des informations. De plus, en respectant les guidelines de branding de l'entreprise, cet outil permettra de maintenir une image cohérente et professionnelle dans toutes les réponses soumises.

L'aspect sécurité de l'application sera également un enjeu majeur, car les informations manipulées seront sensibles et confidentielles. Il sera nécessaire de mettre en place des mécanismes robustes d'authentification et d'autorisation pour garantir que seules les personnes autorisées peuvent accéder et modifier les données. De plus, la gestion des identifiants et des accès aux différentes sources externes sera réalisée de manière sécurisée, en suivant les principes d'une application respectant les normes de sécurité applicatives. Il sera également question d'agréger intelligemment les données pour éviter la redondance des données.

En développant cet outil d'aide à la construction d'appels d'offres, NEOFACTO pourra améliorer la qualité de ses réponses, optimiser ses processus de soumission et répondre aux exigences des clients de manière plus efficace. De plus, la centralisation des informations et la facilité de mise à jour contribueront à réduire les erreurs et à maintenir une base de connaissances précise et à jour. Enfin, l'ajout de fonctionnalités complémentaires telles que la possibilité de mettre à jour les CV des employés et de gérer les références vers les anciens projets, permettant de décrire les capacités actuelles de l'agence, renforcera encore davantage l'utilité et la pertinence de cet outil pour l'entreprise.

3.3.2 Objectifs et Livrables du projet

L'objectif de ce mémoire d'ingénieur est de présenter le développement et la sécurisation d'une application web permettant de consulter facilement les projets, les besoins et les informations liés à la création de réponses à appel d'offre chez NEOFACTO.

Pour cela, les livrables sont constitués des les éléments suivants :

- Une **conception détaillée** de l'application, comprenant des diagrammes et des spécifications fonctionnelles et techniques.
- Une **base de données** conçue pour supporter les différentes fonctionnalités de l'application.
- Une application **développée en fullstack**, en utilisant les technologies choisies lors de la phase de conception, avec une méthodologie Agile et une approche DevOps.

- Une amélioration de la **qualité des données** et de la cohérence de celles-ci.
- Une **gestion des utilisateurs et de leurs permissions** dans l'application, ainsi que l'intégration avec des outils existants tels que l'outil RH et/ou l'outil de formation via leur API respective.
- Une **documentation détaillée** de l'application, comprenant des instructions d'utilisation et de maintenance, ainsi que des descriptions des fonctionnalités et des choix techniques.
- Une **analyse et développement de la sécurité de l'application**, en prenant en compte les aspects infrastructurels et applicatifs, et en mettant en place des mesures préventives pour sécuriser l'application et les données manipulées.
- Une **gestion des références aux projets** permettant de mettre à jour et présenter les projets déjà réalisés par NEOFACTO, en choisissant un template parmi ceux renseignés sur l'application, afin de les insérer dans la réponse à Appel d'offre.
- Une **gestion des CVs des consultants** permettant de mettre à jour le CV des employés en choisissant un template parmi ceux renseignés sur l'application, afin de les insérer dans la réponse à Appel d'offre.

La réalisation de ces livrables permettra d'atteindre l'objectif de développer une application web sécurisée répondant aux besoins de l'entreprise en matière de gestion des profils et des informations de contact des employés, tout en respectant les régulations en vigueur et en garantissant la sécurité des données manipulées.

4 Développement du projet

4.1 Introduction

Cette section du mémoire se concentre sur le développement concret du projet "document-builder", qui vise à créer une application innovante pour faciliter la création de réponses à des appels d'offres et de contrats de maintenance dans le domaine des services informatiques. Dans cette étape cruciale du mémoire, nous plongerons dans les détails du processus de conception, de mise en œuvre et de sécurisation de l'application.

Le projet "document-builder" représente une réponse innovante à la complexité inhérente à la rédaction de propositions commerciales et de contrats dans le secteur des services informatiques. Cette application vise à simplifier et à rationaliser le processus de réponse aux appels d'offres en offrant une solution centralisée pour la création et la personnalisation de documents professionnels.

L'objectif principal de cette phase de développement est de concevoir et de mettre en œuvre une application robuste, conviviale et sécurisée qui répond aux besoins spécifiques du chef technique (CTO) et de l'équipe commerciale. L'application doit permettre la génération rapide de réponses pertinentes à des appels d'offres et de contrats de maintenance tout en respectant les normes de qualité et de présentation de l'entreprise.

La création de cette application revêt une grande importance pour l'entreprise NEOFACTO. En répondant aux défis complexes liés à la création de propositions commerciales et de contrats, cette application contribuera à optimiser le processus de soumission et à accroître l'efficacité opérationnelle. De plus, elle consolidera la position de NEOFACTO en tant qu'acteur innovant dans le domaine des services informatiques en proposant une solution moderne et adaptée aux besoins actuels du marché.

Au cours de cette section, nous explorerons en détail les différentes étapes de développement de l'application, en mettant l'accent sur l'analyse des besoins, la conception, le choix des technologies, l'implémentation des fonctionnalités clés et l'intégration d'éléments d'intelligence artificielle pour enrichir l'expérience utilisateur.

4.2 Analyse et Conception

4.2.1 Analyse des besoins

L'analyse des besoins constitue la pierre angulaire de tout développement réussi. Dans cette phase, nous avons entrepris une exploration approfondie des exigences spécifiques de l'application

de construction de documents d'appels d'offres. Cette exploration a commencé par l'identification minutieuse des besoins fondamentaux de l'entreprise NEOFACTO et des défis auxquels sont confrontés les responsables techniques et commerciaux lors de la réponse à des appels d'offres complexes.

Identification des besoins pour l'application de construction de documents d'appels d'offres

Le point de départ de notre démarche a été de cerner les lacunes et les inefficacités inhérentes au processus existant de réponse aux appels d'offres. Pour parvenir à cette compréhension, nous avons entamé des entretiens approfondis avec les membres clés de l'équipe technique et commerciale de NEOFACTO. Cette immersion nous a permis de dégager les obstacles majeurs auxquels ils font face dans ce processus. Parallèlement, nous avons entrepris une analyse approfondie de la manière dont l'entreprise aspire à présenter ses propositions et contrats, cherchant ainsi à refléter sa position avant-gardiste sur le marché.

Cette application proposera une structure standardisée pour les différentes sections du document, favorisant ainsi une organisation cohérente des informations. De plus, elle intégrera des fonctionnalités d'autocomplétion puisées à partir de diverses sources, telles que Google Docs, Google Sheets, l'API de ChatGPT et même l'outil d'aide à la gestion de l'entreprise (Boond Manager). Un avantage significatif de cette application résidera dans sa capacité à permettre des mises à jour centralisées des réponses, garantissant ainsi une uniformité et une pertinence continues des informations.

Collecte des exigences fonctionnelles et non fonctionnelles

En collaboration avec l'équipe NEOFACTO, nous avons recueilli des exigences fonctionnelles détaillées qui ont servi de base à la conception de l'application. Cela comprenait des fonctionnalités telles que la création de modèles de document et la personnalisation des réponses en fonction des exigences spécifiques de chaque appel d'offres. Parallèlement, nous avons identifié des exigences non fonctionnelles telles que la sécurité des données, la convivialité de l'interface et la compatibilité avec les normes graphiques de l'entreprise.

On retrouve ci-dessous la liste exhaustive des exigences de l'application.

Exigences Fonctionnelles

- **Gestion des Utilisateurs :**
  - Enregistrement des utilisateurs avec des rôles différents (administrateur, CTO, etc.).
  - Connexion sécurisée avec authentification Google OAuth2.
- **Construction de Documents :**
  - Création, modification et suppression de documents d'appels d'offres.
  - Structure modulaire pour organiser les différentes sections du document.
  - Possibilité d'ajouter, de supprimer et de réorganiser les sections du document.
- **Fonctionnalités de Texte :**
  - Éditeur de texte simple pour la saisie de contenu.
  - Mise en forme du texte (gras, italique, liste à puces, etc.).
  - Possibilité de copier-coller à partir de sources externes.
- **Fonctionnalités de Collaboration :**
  - Partage de documents avec d'autres utilisateurs.
  - Commentaires et annotations sur le contenu du document.

  - Suivi des modifications pour voir les versions précédentes du document.
- **Intégration avec des Sources Externes :**
  - Intégration des données de l'entreprise avec Boond Manager.
  - Intégration avec Google Docs et Google Sheets pour l'autocomplétion.
  - Intégration de l'API de ChatGPT pour la suggestion de contenu.
- **Gestion des Langues :**
  - Possibilité de rédiger les documents en français et en anglais.
  - Sélection de la langue pour l'autocomplétion et les suggestions.

Exigences Non Fonctionnelles

- **Performance :**
  - Temps de chargement rapide des documents.
  - Réactivité de l'interface lors de l'édition et de la manipulation des documents.
- **Sécurité :**
  - Sécurisation des données utilisateur et des documents.
  - Gestion des droits d'accès pour les différents rôles.
- **Convivialité :**
  - Interface utilisateur intuitive et conviviale.
  - Prise en charge de différentes résolutions d'écran.
- **Fiabilité :**
  - Sauvegarde automatique des documents en cours d'édition.
  - Prévention de la perte de données en cas de déconnexion ou de panne.
- **Interopérabilité :**
  - Compatibilité avec différents navigateurs web modernes.
  - Intégration fluide avec les outils tiers (Google Docs, ChatGPT, etc.).
- **Évolutivité :**
  - Capacité à gérer un grand nombre d'utilisateurs et de documents.
  - Possibilité d'ajouter de nouvelles fonctionnalités à l'avenir.
- **Accessibilité :**
  - Conformité aux normes d'accessibilité pour les personnes handicapées.
- **Performances :**
  - Optimisation des performances pour garantir une expérience fluide.
- **Soutien :**
  - Support technique et documentation pour les utilisateurs.

Modélisation des cas d'utilisation

Pour traduire ces besoins en actions concrètes, nous avons élaboré des cas d'utilisation détaillés qui décrivent comment les utilisateurs interagiront avec l'application. Une réponse à appel d'offre a servi d'exemple pour illustrer les différentes fonctionnalités qui pouvaient être utilisées pour la rédaction. Cela a permis de mieux comprendre les flux de travail, les interactions entre les différentes fonctionnalités et les exigences spécifiques de chaque tâche. La modélisation des cas d'utilisation a servi de base solide pour la phase de conception, garantissant que l'application serait alignée sur les besoins réels des utilisateurs.

L'analyse des besoins a jeté les bases de la conception de l'application, en veillant à ce qu'elle réponde de manière précise et complète aux défis rencontrés par NEOFACTO lors de la réponse aux appels d'offres. Cette phase a établi les paramètres pour le développement ultérieur, en s'assurant que

chaque fonctionnalité et chaque interaction sont soigneusement conçues pour offrir une expérience utilisateur fluide et efficace.

4.2.2 Conception de la solution

Lors de la conception de la solution, deux aspects majeurs ont été pris en compte : l'architecture globale de l'application et la conception de l'interface utilisateur. Pour garantir la modularité, la maintenabilité et l'évolutivité de l'application, une architecture propre a été mise en œuvre, tandis que la conception de l'interface utilisateur visait à offrir une expérience conviviale et intuitive.

Architecture Globale

La conception de l'architecture globale de l'application a été guidée par les principes de la Clean Architecture[5]. Cette approche permet de découpler les différentes couches de l'application en les hiérarchisant selon leur niveau d'abstraction. Au niveau le plus interne, le cœur de l'application abrite la logique métier et les règles de gestion. La couche externe, quant à elle, gère les détails techniques tels que les interfaces utilisateur et les API externes.

Pour le front-end, la Clean Architecture a été appliquée en utilisant Angular. Les composants d'interface utilisateur, basés sur Angular Material, sont soigneusement séparés des couches de gestion de l'application, ce qui permet une évolutivité aisée et une réutilisation efficace des composants.

Pour le back-end, une architecture REST a été choisie pour favoriser l'interopérabilité et la scalabilité. Les endpoints API sont structurés de manière à refléter les différentes fonctionnalités de l'application, assurant ainsi une organisation logique et cohérente de l'ensemble.

Quelques règles de l'architecture REST permettent de s'assurer du comportement de l'application en imposant la gestion des données en fonction du type de requête à l'API. Par exemple, les requêtes GET ne doivent pas altérer les données de l'application. Si c'est le cas, alors plutôt utiliser une requête POST sans corps de requête. Cela permet de clarifier le comportement de la route, sans aller voir profondément dans le code.

Conception de l'Interface Utilisateur

La conception de l'interface utilisateur a été réalisée en utilisant Angular Material, une bibliothèque de composants prêts à l'emploi qui favorise la création d'interfaces cohérentes et esthétiques. L'objectif était de proposer une expérience utilisateur agréable et ergonomique, tout en garantissant une navigation intuitive à travers les différentes fonctionnalités de l'application.

Les principes de la conception centrée utilisateur ont été suivis pour créer des interfaces intuitives et réactives. Des prototypes et des maquettes ont été élaborés pour valider les choix de conception avec les utilisateurs potentiels, afin de s'assurer que l'interface réponde aux besoins réels. On retrouve ces informations dans la section 4.2.3 de ce mémoire.

Conception de la Base de Données

La base de données joue un rôle crucial dans l'application, stockant les informations nécessaires au bon fonctionnement des fonctionnalités. La conception de la base de données a été effectuée en tenant compte des entités principales de l'application, de leurs relations et de leurs attributs.

Le modèle de données a été structuré de manière à optimiser l'accès aux informations et à garantir l'intégrité des données. Des considérations de sécurité ont également été prises en compte lors de la conception de la base de données, en utilisant des mécanismes de chiffrement et de gestion des accès pour protéger les données sensibles.

On retrouve section **??** l'implémentation de la base de données, et l'ensemble des procédés qui ont permis de mettre en application ces directives.

Interface avec les Autres APIs

Au cœur de son fonctionnement, l'application interagit de manière fluide avec différentes APIs externes pour enrichir sa gamme de fonctionnalités. Pour réaliser cette intégration, l'application se connecte à d'autres systèmes via des interfaces dédiées. Un exemple concret est la communication avec Boond Manager, une étape cruciale pour l'efficacité de NEOFACTO. Cette interaction s'effectue au moyen d'une API REST, en exploitant les points d'accès exposés par Boond.

Notablement, NEOFACTO se fonde sur l'écosystème Google Workspace, garantissant que tous les employés disposent d'une adresse e-mail associée à l'entreprise, suivant le format "@neofacto.com". Cette approche unifiée présente des avantages indéniables, tels que la simplification de la gestion des comptes utilisateur et la réduction de la charge administrative. C'est dans ce contexte qu'intervient le choix stratégique d'utiliser OAuth2 pour l'authentification. En dépit des alternatives, ce mécanisme offre une expérience utilisateur sans faille, en éliminant le besoin de mémoriser et de gérer plusieurs mots de passe. Cette solution assure une connexion sécurisée aux comptes des utilisateurs NEOFACTO, contribuant à renforcer la protection des données et à garantir un environnement d'utilisation convivial.

L'élaboration des interfaces pour ces interactions a été soumise à une approche rigoureuse. Elle repose sur l'adhésion aux spécifications détaillées fournies par les APIs externes. De plus, les meilleures pratiques de sécurité et d'intégration ont été scrupuleusement mises en œuvre. Cette démarche vise à garantir une communication sans faille entre l'application et les services externes, tout en préservant la confidentialité des données et en évitant les risques potentiels associés aux intégrations logicielles.

En résumé, la conception de la solution a été orientée vers une architecture propre, une interface utilisateur conviviale, une base de données bien structurée et des interfaces externes sécurisées. Cette approche vise à fournir une application solide, modulaire et répondant aux exigences de qualité et de convivialité.

Diagrammes de classes

Afin de définir la structure globale de l'application, il est nécessaire de modéliser les deux parties de l'application grâce à une sorte de diagramme de classes qui présente les différents composants utiles.

Front-end

Pour le front-end, on retrouve figure 4.1 le diagramme partiel qui présente les différents composants et services, ainsi que les relations entre ceux-ci. Ce diagramme a ensuite été complété avec les méthodes des classes afin de lister de manière exhaustive les différentes fonctionnalités existantes dans chaque classe. On retrouve ces informations plus détaillées dans les annexes, sur la figure **??**.

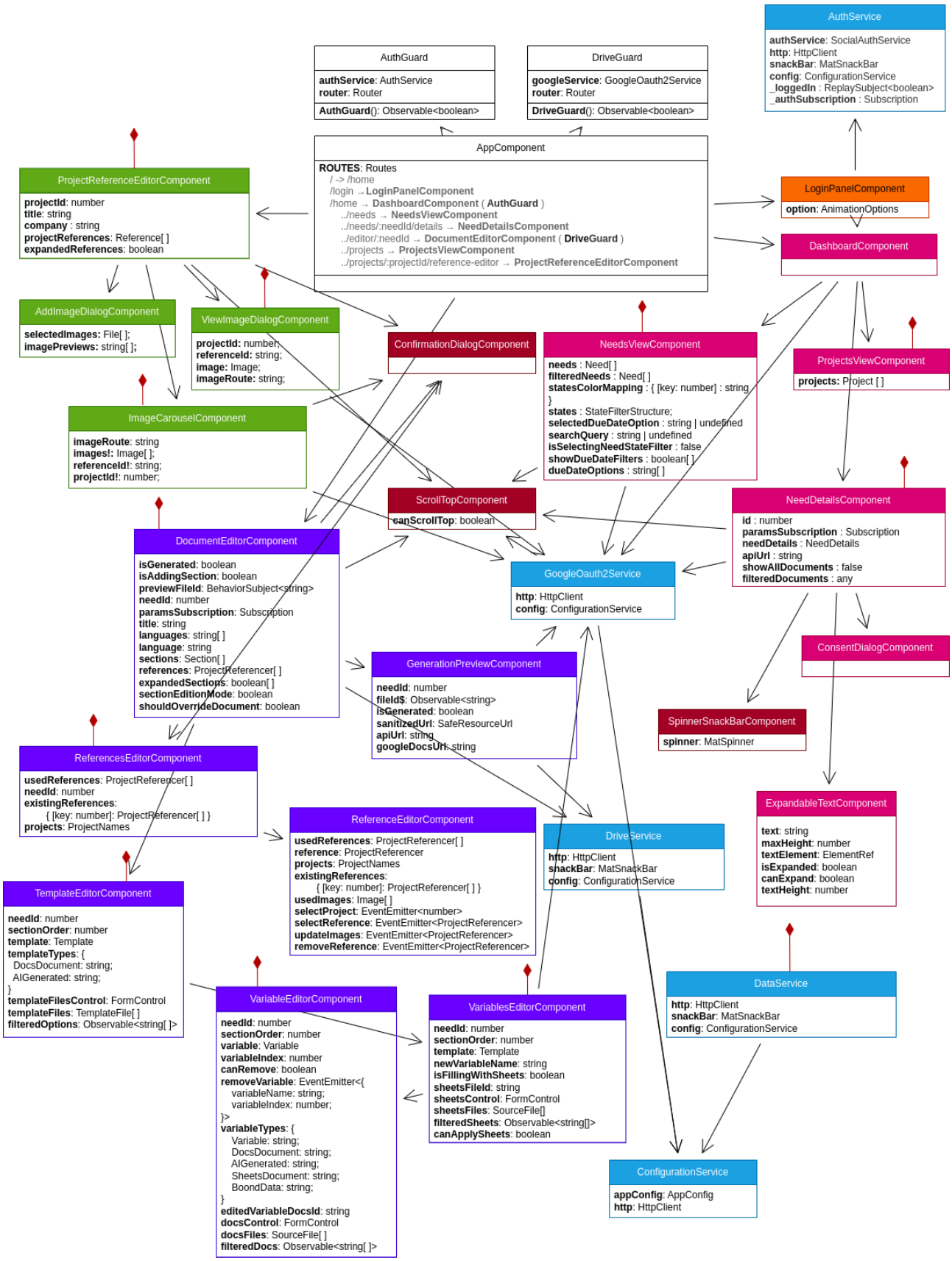

FIGURE 4.1 – Diagramme de classes partiel du front-end de l'application

La représentation visuelle illustrée dans la Figure 4.1 intègre un schéma de codage couleur qui se révèle essentiel pour la distinction des diverses fonctionnalités et services au sein de l'application.

On retrouve la définition du code couleur dans la liste suivante :

- la classe centrale de l'application ainsi que les 'Guards', lesquels englobent des fonctions de validation des autorisations préalables à l'accès à une route de l'application, sont signalés en teinte blanche.
- les composants en lien avec l'authentification, plus précisément la page d'identification, sont identifiés par la couleur orange.
- les éléments en rapport avec les données relatives aux exigences des clients se parent d'une teinte rose.
- les éléments associés à la manipulation de documents, destinée à la création de réponses aux appels d'offres, s'affichent en violet.
- les éléments concernant la gestion des références aux projets des clients s'illustrent en vert.
- les services sont discernables grâce à une teinte bleue.
- enfin, les composants utilitaires, réutilisés à plusieurs reprises dans le code, adoptent une nuance bordeaux.

Comme le service 'DataService' est utilisé par la plupart des composants, leur lien est représenté par une antenne rouge finissant par un losange. Cela permet simplement d'accroître la clarté du diagramme.

La conception globale repose sur une logique bien définie. Les données sont gérées temporairement au sein des composants, ces derniers assurant l'aspect purement utilisateur de l'interface (par exemple, la fonction de défilement vers le haut, qui est essentiellement destinée à l'utilisateur et n'a pas d'incidence sur le back-end). Lorsque les données entrées dans les composants sont validées, des actions sont initiées dans les services. Cependant, les actions liées au back-end sont principalement gérées au niveau des services, avec une configuration centralisée dans un service spécifique.

Le diagramme englobe également deux 'Guards' : l'un pour gérer l'authentification dans l'application, et l'autre pour actualiser le token OAuth2 pour l'accès au Drive. Ces éléments concourent à la robustesse et à la sécurité globale de l'application en contrôlant les autorisations d'accès et en gérant les interactions avec les services externes.

Back-end

Une API REST définit un ensemble de règles et de conventions pour créer des interfaces web permettant aux systèmes de communiquer entre eux de manière efficace et structurée. Cette approche repose sur des principes architecturaux fondamentaux qui favorisent la scalabilité, la flexibilité et l'interopérabilité des systèmes.

Lors de la conception d'une API REST, il est essentiel de suivre les normes établies pour assurer une expérience cohérente et intuitive aux développeurs qui l'utiliseront, notemment dans le cas où elle est utilisée par plusieurs applications. L'API doit être conçue de manière à refléter les ressources et les actions spécifiques que l'application expose. Les verbes HTTP tels que GET, POST, PUT et DELETE sont utilisés pour définir les opérations effectuées sur ces ressources.

La structure des URLs de l'API doit être réfléchie et organisée de manière logique pour faciliter la navigation et la compréhension. Les réponses de l'API sont généralement formatées en JSON (JavaScript Object Notation), un format léger et lisible. Les codes de statut HTTP tels que 200 (OK), 404 (Not Found) et 500 (Internal Server Error) sont utilisés pour indiquer le résultat d'une requête. L'ensemble des conventions pour les codes de retour peut être trouvé sur internet, comme sur la page wikipédia suivante dédiée[9].

Le diagramme de classes présenté dans la figure 4.2 illustre la conception de l'API REST pour

l'application. Les différentes classes représentent les ressources exposées par l'API, chacune avec ses propres attributs et méthodes. Les associations entre les classes indiquent les relations entre les différentes ressources, telles que les relations parent-enfant ou les dépendances.

Une conception soignée de l'API REST contribue à la simplicité, à la clarté et à la robustesse de l'interface, ce qui facilite son utilisation et son intégration dans diverses applications clientes.

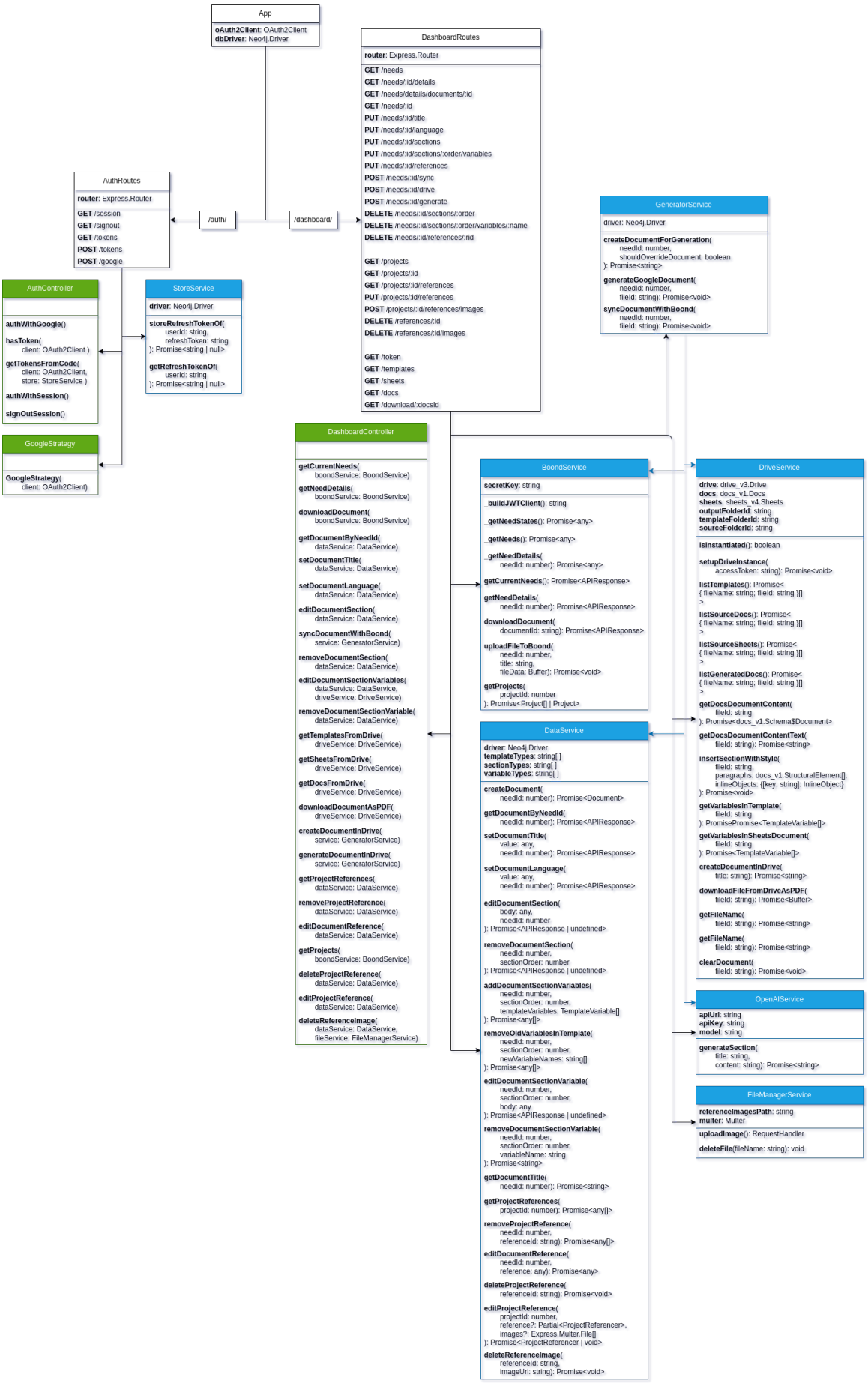

FIGURE 4.2 – Diagramme de classes du back-end de l'application

La représentation visuelle illustrée dans la Figure 4.2 intègre également un schéma de codage couleur qui se révèle essentiel pour la distinction des diverses fonctionnalités et services au sein de l'application. On retrouve la définition du code couleur dans la liste suivante :

— la classe centrale de l'application ainsi que les routes de l'API sont signalés en teinte blanche.
— les contrôleurs, concernant la gestion des actions relatives aux routes, s'illustrent en vert.
— les services sont discernables grâce à une teinte bleue.

Au cœur de ce schéma se trouvent deux contrôleurs principaux qui jouent des rôles cruciaux dans la gestion des fonctionnalités : l'AuthController et le DashboardController.

L'AuthController joue un rôle central dans le processus d'authentification et d'autorisation des utilisateurs. Il gère les interactions avec l'API d'authentification de Google pour vérifier les informations d'identification et générer des jetons d'accès. De plus, ce contrôleur assure également la gestion des jetons de rafraîchissement, qui sont stockés de manière sécurisée à l'aide du StoreService. Le StoreService, quant à lui, constitue une interface avec la base de données pour stocker et récupérer les informations sensibles, garantissant ainsi leur protection.

En parallèle, le DashboardController orchestre la logique derrière les fonctionnalités liées au tableau de bord de l'application. Il se connecte au DataService, qui agit comme une interface avec la base de données pour récupérer, mettre à jour et manipuler les données relatives aux projets et aux besoins des clients. Cette séparation entre le DataService et le StoreService permet de maintenir une clarté dans la gestion des données et assure une isolation des rôles et des responsabilités au sein du code.

Modèles de données

L'efficacité du stockage des informations au sein de l'application nécessite la mise en place de modèles de données bien structurés. Ces modèles servent de représentations abstraites des entités et des relations clés au sein de l'application. Ils assurent la cohérence et la normalisation des données échangées entre le front-end et le back-end.

Le diagramme présenté dans la figure 4.3 illustre les différents modèles de données utilisés dans l'application. Ces modèles sont catégorisés en fonction de leur implication dans le front-end, le back-end, ou leur partage entre les deux parties.

FIGURE 4.3 – Diagramme présentant les différents modèles de l'application

La section bleue du diagramme représente les modèles spécifiquement conçus pour le front-end. Ces modèles facilitent la présentation visuelle des données à l'utilisateur. Ils sont optimisés pour répondre aux besoins de l'interface utilisateur en fournissant des informations structurées et conviviales.

La section jaune du diagramme englobe les modèles exclusivement liés au back-end. Ces modèles servent de structures pour la logique métier et la gestion des données côté serveur. Ils sont conçus pour gérer efficacement les opérations de traitement, de stockage et de récupération des données.

La zone blanche au centre du diagramme représente les modèles partagés entre le front-end et le back-end. Ces modèles ont pour objectif de normaliser les échanges d'informations entre les deux parties de l'application. Ils assurent la synchronisation des données et réduisent le risque d'incohérence entre les vues présentées à l'utilisateur et les données réellement stockées.

L'architecture de modèles de données garantit une séparation claire des préoccupations entre les différentes composantes de l'application. Cela facilite la maintenance, l'évolutivité et la cohérence des données tout au long du cycle de vie de l'application.

Diagrammes de séquences

Pour illustrer de manière plus concrète le flux des données au sein de l'application, j'ai élaboré des diagrammes de séquence. Ces diagrammes permettent de suivre pas à pas les interactions entre les différents composants lors de scénarios clés, du moment où l'utilisateur initie une action jusqu'à ce que les données soient sauvegardées dans la base de données et que le résultat soit affiché en retour.

Le premier diagramme de séquence met en lumière le processus d'authentification de l'utilisateur. Il commence par l'initiation de l'authentification depuis l'interface utilisateur. L'AuthController prend le relais en communiquant avec l'API OAuth2 de Google. Une fois l'authentification réussie,

les jetons d'accès sont générés et stockés en toute sécurité via le StoreService. Ces jetons d'accès sont ensuite utilisés pour les requêtes aux API de l'application, garantissant ainsi l'accès sécurisé aux fonctionnalités. On retrouve ce diagramme sur la figure 4.4 suivante.

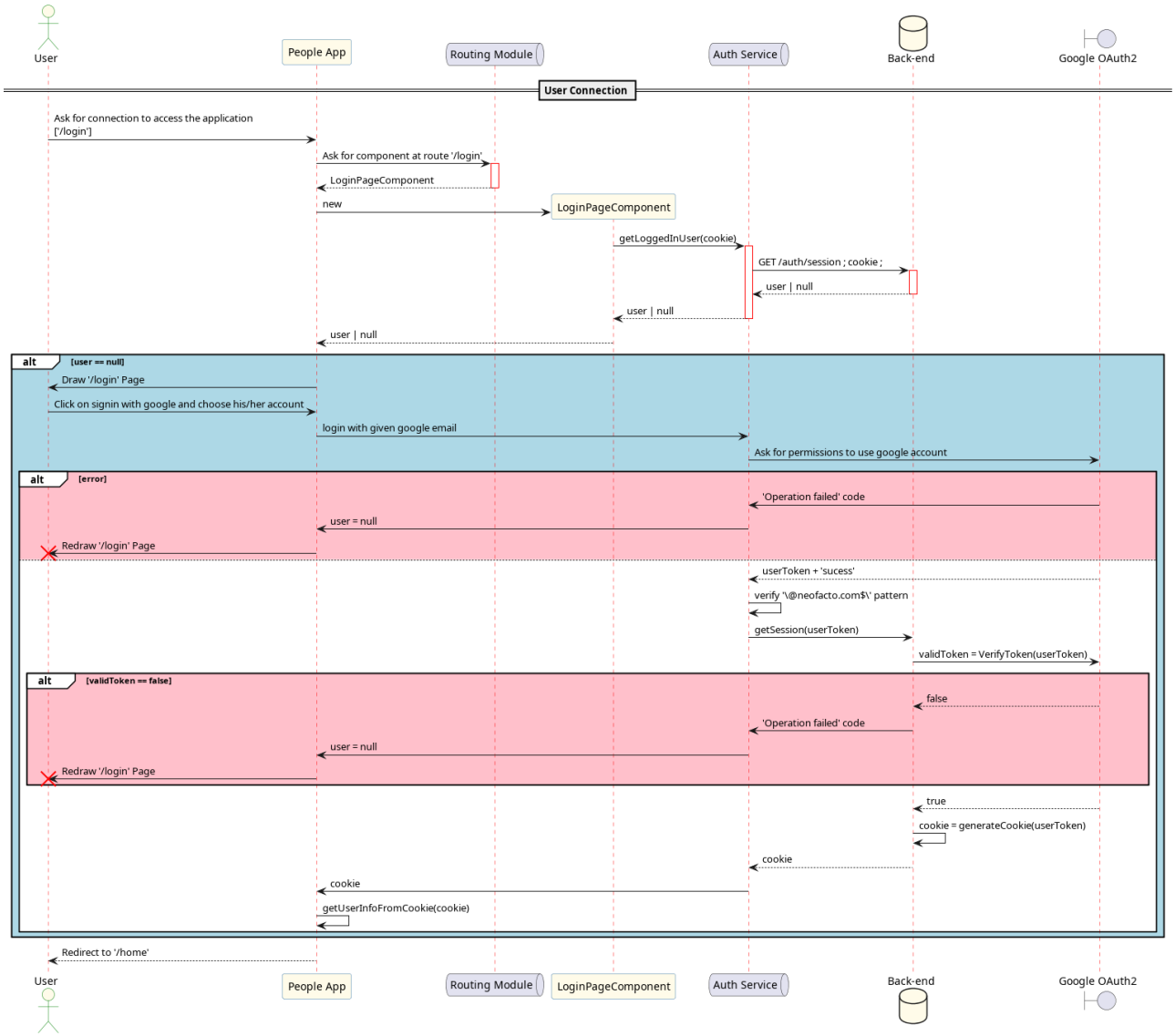

FIGURE 4.4 – Diagramme de séquence de l'authentification sur l'application

Le deuxième diagramme de séquence illustre le flux de modification de données dans l'application. L'utilisateur, depuis l'interface, initie une action pour mettre à jour des informations, telles que les détails d'un projet. Le DashboardController intervient pour traiter cette demande. Il communique avec le DataService pour récupérer les données pertinentes depuis la base de données. Une fois que l'utilisateur effectue ses modifications, le DashboardController coordonne le processus de mise à jour dans la base de données via le DataService. Une fois la mise à jour effectuée avec succès, le résultat est affiché à l'utilisateur pour confirmer que les données ont été sauvegardées. On retrouve ce diagramme sur la figure 4.5 suivante.

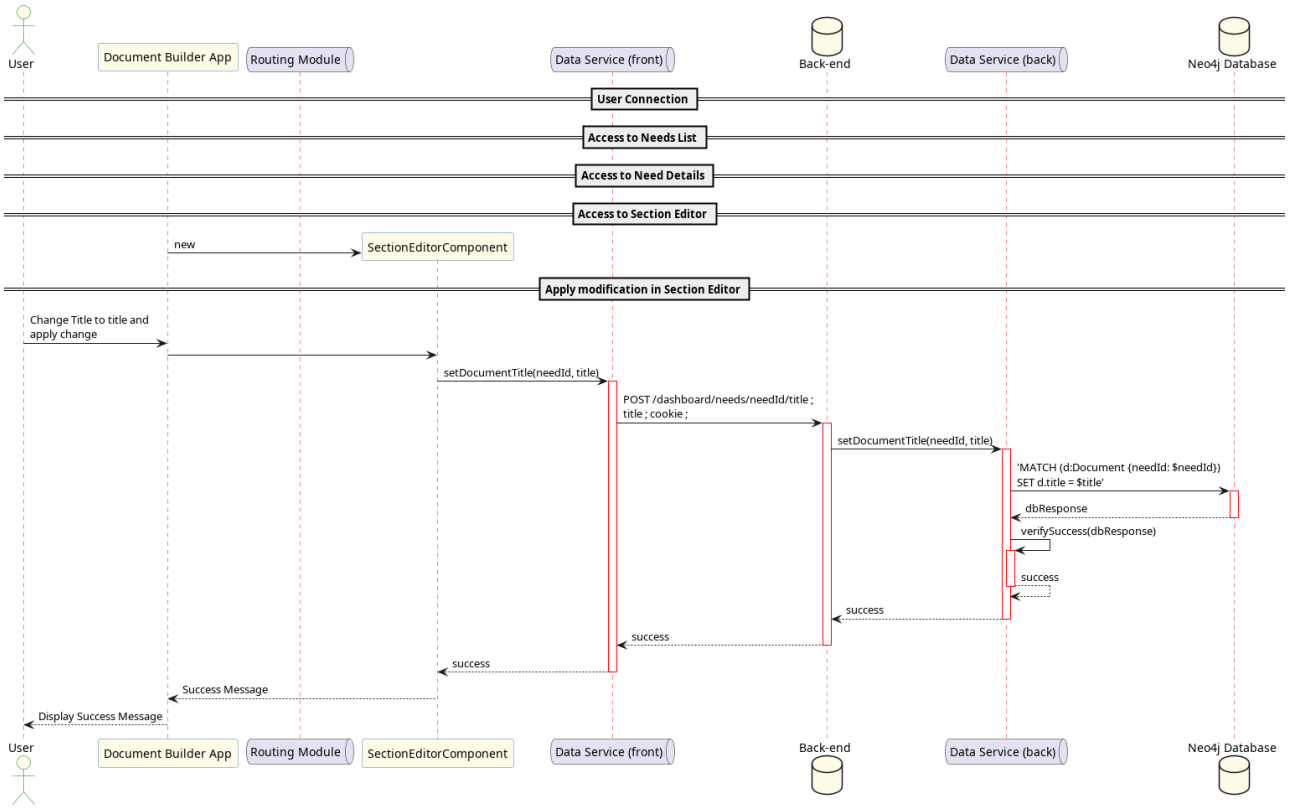

FIGURE 4.5 – Diagramme de séquence de modification et sauvegarde de données sur l'application

Ces diagrammes de séquence fournissent une vue détaillée et visuelle de la dynamique de l'application pendant des interactions clés, complémentaire aux diagrammes de classes. Ils permettent de mieux appréhender le cheminement des données à travers les composants, et d'apprécier comment l'architecture sous-jacente assure un flux fluide, sécurisé et cohérent des informations. Pour une visualisation complète de ces diagrammes, je vous invite à vous référer aux annexes.

On retrouve dans les annexes les diagrammes de séquence créés pour la fonctionnalité des références aux projets, dont le comportement est semblable au deuxième diagramme, et qui ont permis de concevoir l'application.

Modèle de base de données

Comme indiqué dans la phase d'état de l'art, une base de données relationnelle était un choix approprié pour les fonctionnalités attendues dans l'application. Nous avons cependant opté pour une solution orientée graphe, tel que Neo4j afin de profiter de la flexibilité de cette option dans le cadre du développement d'une application évolutive, et dont les relations sont abondantes parmi les entrées.

On retrouve alors un schéma relationnel sur la figure 4.6 suivante qui présente comment les données sont stockées et mis en relation dans la base de données.

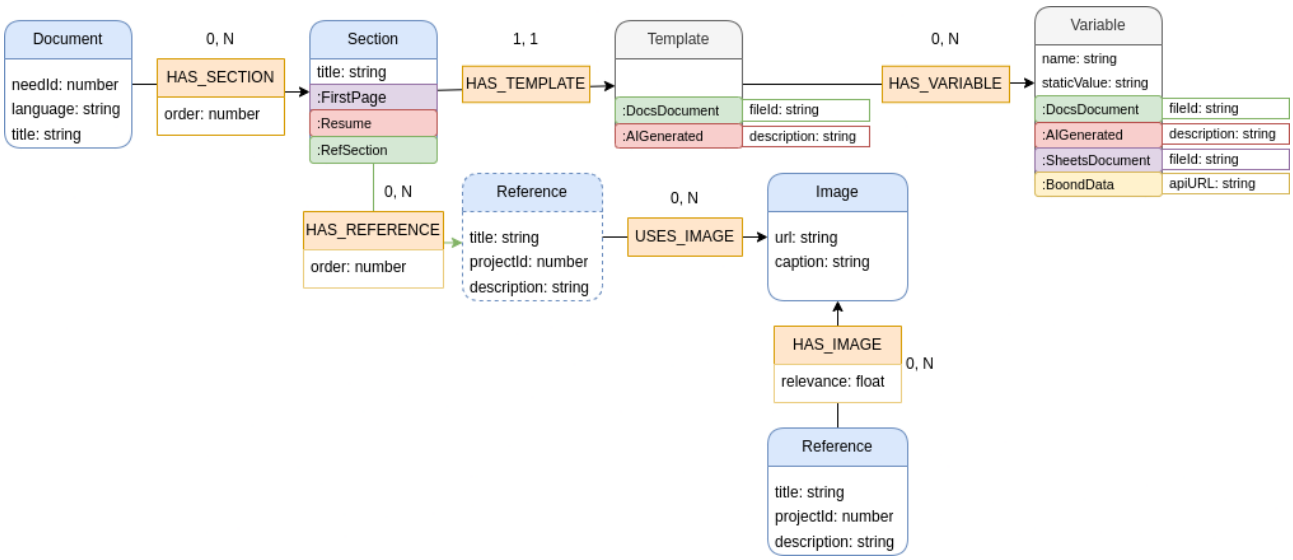

FIGURE 4.6 – Schéma relationnel modélisant la base de données de l'application

On remarque sur cette figure 4.6 que les relations entre les noeuds sont colorées en orange, alors que les noeuds sont colorés en bleu ou en gris. Afin de mieux comprendre les relations, des indices ont été ajoutés afin de définir les contraintes de cardinalité.

Les relations peuvent contenir des informations supplémentaires qui ne sont pertinentes, qu'en l'existence de lien entre les noeuds. Par exemple, la notion d'ordre pour une section n'a de sens qui si elle est reliée à un document (afin de les organiser entre-elles). Il est donc intéressant de stocker cette information dans la relation plutôt que dans l'un des noeuds.

De manière pratique, les bases de données orientées graphes facilite le développement en permettant de modifier de manière dynamique les attributs des objets ainsi que leurs relations. Le modèle de données peut donc évoluer avec l'application en appliquant des changement partiels sans nécessiter de modifier les autres données existantes.

4.2.3 Interface utilisateur et flux de l'application

Pour créer une interface aussi conviviale et intuitive que possible, l'outil de conception Figma a été utilisé, un choix qui s'est avéré particulièrement judicieux pour plusieurs raisons. Cette partie a été développée en partie par un autre membre de l'équipe sur le projet, qui est particulièrement à l'aise avec cet outil.

L'interface de l'application se compose de plusieurs pages clés, chacune dédiée à une fonctionnalité spécifique. Pour les concevoir, j'ai adopté une approche axée sur la conception centrée sur l'utilisateur. J'ai commencé par établir un flux de navigation logique et intuitif, garantissant une expérience utilisateur fluide. Ensuite, j'ai exploité les fonctionnalités de Figma pour créer des maquettes visuelles de chaque page, définissant les éléments de l'interface tels que les boutons, les champs de saisie, les barres de navigation, et bien d'autres.

Un avantage considérable de Figma est sa compatibilité avec le module Angular Material Design, qui offre une gamme étendue de composants pré-construits et stylisés. Cela m'a permis d'implémenter les éléments de l'interface directement dans Figma, ce qui a facilité l'alignement visuel et

conceptuel avec les directives de conception spécifiques de l'entreprise.

L'une des grandes forces de Figma réside dans sa capacité à faciliter la collaboration. les maquettes ont pu être partagées avec les membres de l'équipe pour obtenir des retours et des commentaires, ce qui a permis d'apporter des améliorations continues. De plus, Figma autorise des modifications à la souris et en temps réel, permettant une exploration visuelle approfondie des changements avant leur implémentation.

L'utilisation de Figma a grandement enrichi la phase de conception de l'interface utilisateur. Sa flexibilité, ses outils de collaboration et son intégration transparente avec les composants Angular Material Design ont contribué à créer une interface cohérente, attrayante et hautement fonctionnelle.

Passons maintenant à l'examen des différentes maquettes élaborées au moyen de Figma. Commençons par explorer la page d'authentification de l'application, qui occupe une position de premier plan. En effet, cette page constitue le point d'entrée initial pour tout utilisateur lors de sa visite sur le site, particulièrement pour ceux qui n'ont jamais établi de connexion précédemment. Lorsqu'un utilisateur accède au site pour la première fois ou après s'être déconnecté, il est immédiatement redirigé vers cette page d'authentification. Ici, il peut fournir ses identifiants et s'authentifier pour accéder aux fonctionnalités de l'application. C'est un élément vital de l'expérience utilisateur, car il détermine le début de leur interaction avec l'application. On retrouve figure 4.7 la maquette Figma associée.

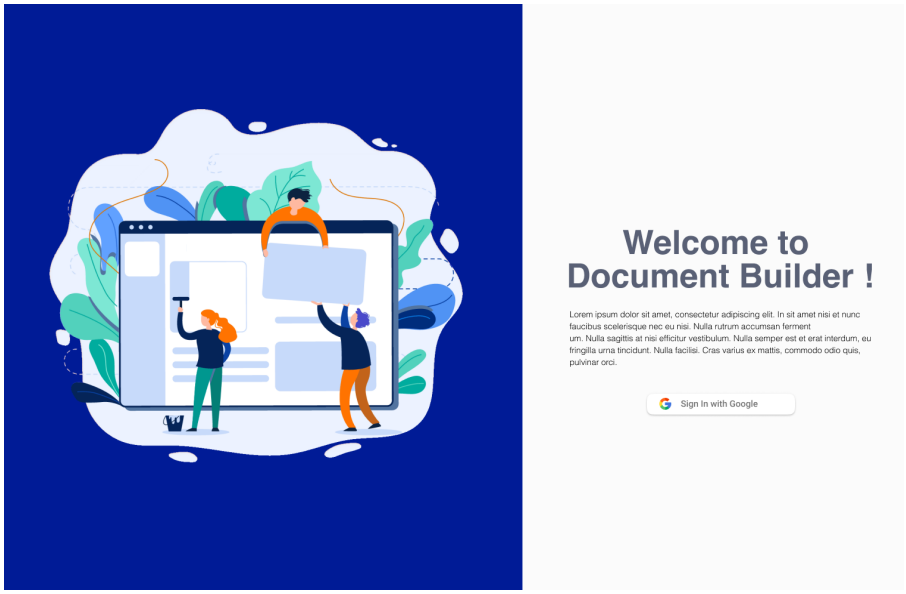

FIGURE 4.7 – Page d'authentification de l'application

Au sein de cette page, une animation captivante est intégrée en utilisant le service Lottie. Ce service offre un éventail varié d'animations stockées au format JSON, et il est mis à disposition grâce à une librairie dédiée. Cette animation dynamique constitue un élément visuel saisissant qui ajoute une dimension interactive à l'expérience de l'utilisateur. Grâce à l'utilisation de Lottie, l'animation peut être aisément chargée et affichée sur la page d'authentification. Le format JSON permet de décrire avec précision les mouvements, les transitions et les effets visuels, garantissant une représentation fidèle et fluide des animations souhaitées. L'intégration de cette animation contribue non seulement à l'attrait visuel de la page, mais elle renforce également l'engagement de l'utilisateur dès les premiers instants de son interaction avec l'application. Cela semble créer un effet positif, incitant davantage les utilisateurs à explorer davantage les fonctionnalités de l'application.

Immédiatement après l'animation, une brève description de l'application est présentée, offrant aux utilisateurs un aperçu des fonctionnalités clés qu'ils peuvent attendre. Cette description succincte sert à orienter les utilisateurs et à susciter leur intérêt pour la suite de l'expérience. Juste en dessous de cette description, un bouton stratégiquement placé offre aux utilisateurs la possibilité de se connecter à l'application en utilisant leur compte Google. Ce bouton "Se connecter avec Google" est une passerelle d'accès simple et sécurisée pour les utilisateurs. En cliquant sur ce bouton, les utilisateurs peuvent initier le processus d'authentification via OAuth2, en se connectant à leur compte Google et en autorisant l'application à accéder à certaines informations et permissions nécessaires.

On ressent immédiatement que l'intégration de cette option de connexion avec Google contribue à simplifier l'expérience d'authentification pour les utilisateurs. Plutôt que de créer et de mémoriser de nouveaux identifiants pour l'application, les utilisateurs peuvent utiliser leurs informations d'identification existantes sur leur compte Google pour accéder rapidement et en toute sécurité à l'application. Cela réduit les frictions lors du processus d'inscription et encourage potentiellement plus d'utilisateurs à adopter l'application.

Une fois que l'utilisateur a franchi l'étape de connexion, il est redirigé vers la page principale de l'application, où il découvre la liste complète des besoins clients. Cette liste est alimentée directement à partir de Boond Manager et est présentée sous la forme de cartes distinctes. Chaque carte représente un besoin spécifique et contient des informations pertinentes pour permettre à l'utilisateur de mieux comprendre la nature et l'état de ce besoin. Ces informations incluent le titre du besoin, le nom de l'entreprise cliente, la date de clôture prévue et l'état actuel du besoin (indiqué par des labels tels que "En cours", "En attente", "annulé", etc.). On retrouve la maquette Figma de cette page sur la figure 4.8 suivante.

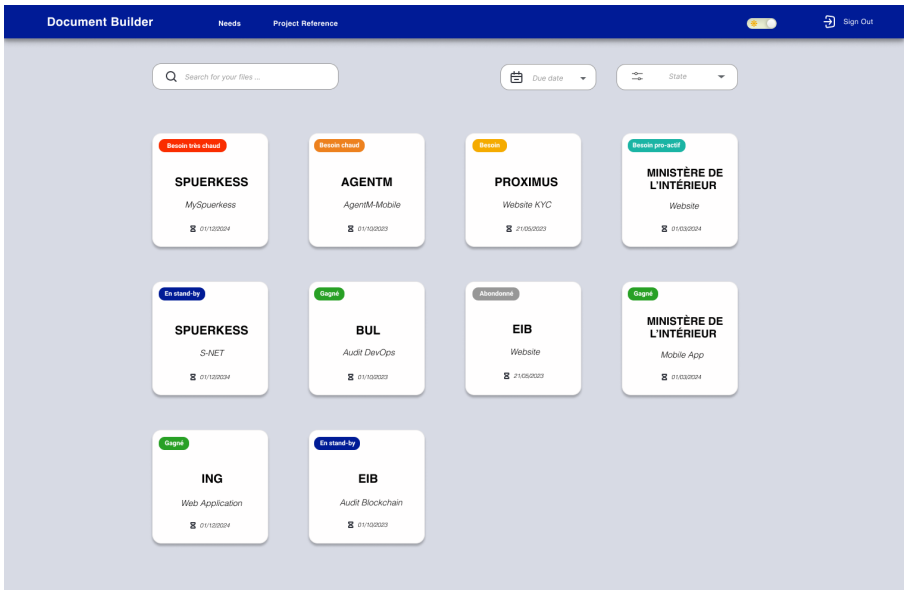

FIGURE 4.8 – Page d'accueil comportant la liste des besoins des clients

Pour faciliter la navigation et la recherche parmi cette liste, l'application offre à l'utilisateur la possibilité de filtrer les besoins en fonction de différents critères. Ces critères de filtrage comprennent le nom du besoin, la date de clôture et l'état du besoin. En fournissant ces options de filtrage, l'application améliore l'expérience de l'utilisateur en lui permettant de cibler rapidement les besoins qui l'intéressent le plus ou qui répondent à ses besoins spécifiques à un moment donné.

Lorsque l'utilisateur clique sur l'une des cartes représentant un besoin, il est dirigé vers une page qui présente en détail les informations spécifiques de ce besoin sélectionné. Cette page agglo-

mère les données extraites de Boond Manager et offre ainsi une vue exhaustive de toutes les données essentielles liées au projet choisi. On retrouve figure 4.9 la maquette Figma de cette page.

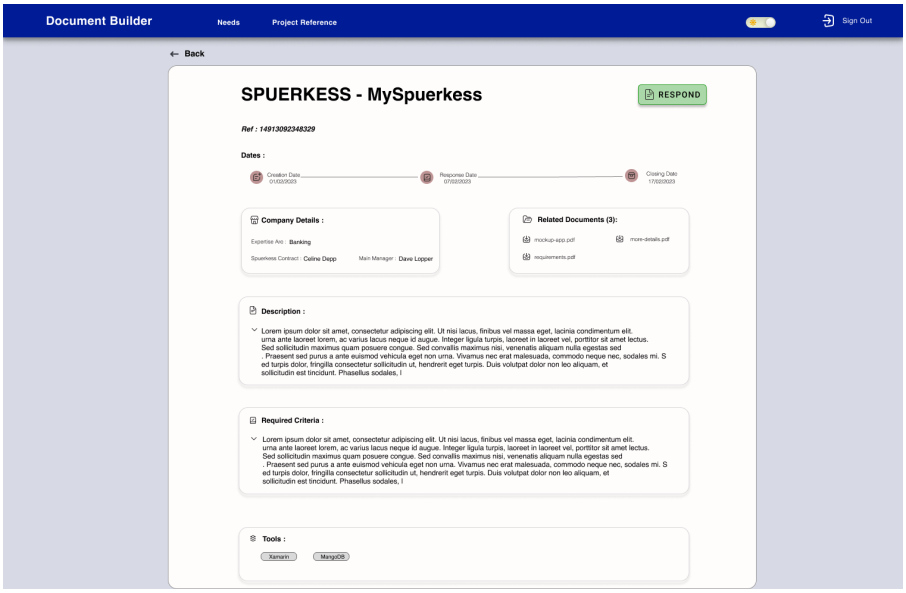

FIGURE 4.9 – Page de visualisation des détails d'un besoins client

Parmi les informations affichées figurent les différentes dates clés associées au projet, qui permettent de suivre son évolution temporelle de manière précise. Ces dates peuvent inclure des étapes importantes comme la date de début du projet, la date limite de soumission des offres, et d'autres jalons pertinents.

La description du projet est également mise en évidence sur cette page. Elle offre un aperçu textuel complet du projet, permettant à l'utilisateur de mieux comprendre ses spécificités, ses objectifs et ses contraintes éventuelles.

Une des caractéristiques majeures de cette page est la possibilité d'accéder au document PDF original de l'appel d'offres qui a donné naissance à ce besoin spécifique sur Boond Manager. En fournissant un accès direct à ce document, l'application facilite la consultation du cahier des charges complet et aide ainsi l'utilisateur à avoir une vision détaillée de l'appel d'offres à l'origine du projet.

Lorsque l'utilisateur choisit l'option "Répondre" dans la page de détails du besoin, l'application sollicite l'autorisation d'accéder au service Google Drive par le biais d'une intégration OAuth2. Cette démarche garantit un accès sécurisé et contrôlé au stockage en ligne de Google. Suite à cette demande, une page de redirection vers les serveurs OAuth2 de Google s'ouvre, où l'utilisateur est invité à consentir aux divers droits d'accès nécessaires pour l'application, notamment l'accès à Drive, Docs, Sheets, et d'autres services connexes. Une fois les autorisations accordées, l'utilisateur est dirigé vers l'environnement d'édition du document de réponse à l'appel d'offres qui est lié au besoin spécifique sélectionné. On retrouve la maquette Figma de la page sur la figure 4.10 suivante.

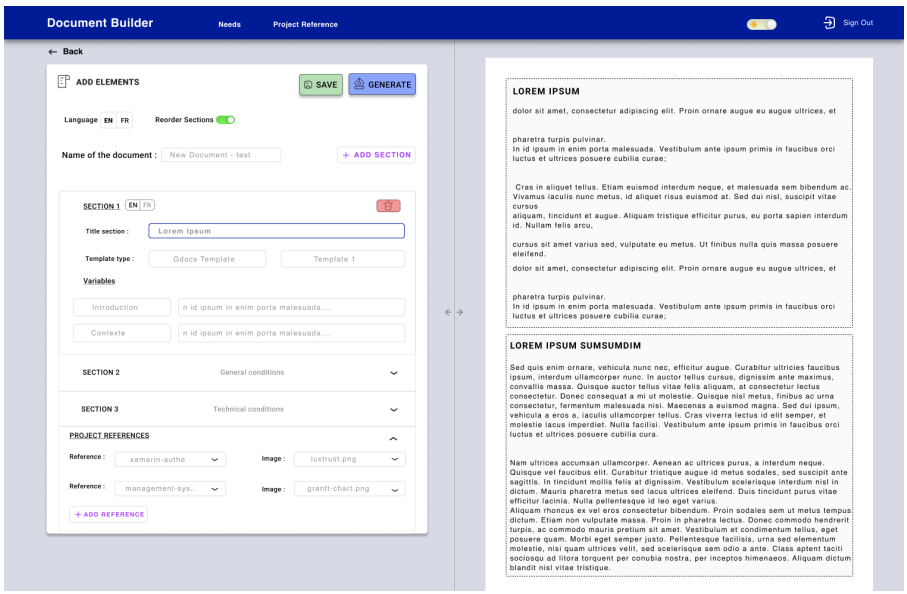

FIGURE 4.10 – Page d'édition du document de réponse au client avec pré-visualisation

La page d'édition est scindée en deux par un curseur mobile appelé "slider", qui offre une expérience flexible en permettant d'ajuster la taille des deux vues en fonction des besoins. À gauche se trouve l'éditeur, tandis qu'à droite se trouve la prévisualisation du document.

L'éditeur détient toute la logique associée à la base de données. Il autorise la modification des modèles à utiliser et l'ajout de références. L'interface est composée d'une série de champs (fields) issus du module Angular Material. Ces champs offrent une sélection intuitive des éléments. Par exemple, pour choisir les documents stockés sur le Drive, l'utilisateur bénéficie d'un formulaire équipé d'une auto-complétion intelligente. Celle-ci explore les documents existants sur le Drive, facilitant ainsi la sélection de documents en permettant à l'utilisateur de taper une sous-chaîne de caractères pour filtrer la liste.

La section de prévisualisation, quant à elle, permet à l'utilisateur de suivre en direct le processus de génération du document sur Google Drive. Cette visualisation est offerte à travers une iframe (page qui redirige vers le document sur Google Docs. Pour optimiser l'expérience, un paramètre est ajouté à l'URL pour basculer en mode prévisualisation, masquant ainsi les outils de l'interface habituelle de Google Docs et se concentrant uniquement sur le contenu du document en cours de création.

Une fois de retour à la page d'accueil, il est à noter que l'utilisateur a la possibilité de naviguer vers la vue des projets via la barre de navigation située en haut de l'application. Dans cette section, une interface similaire est présentée, mettant en avant des cartes qui représentent les projets issus des besoins clients ayant été engagés par l'entreprise. On retrouve la maquette Figma de cette page sur la figure 4.11 suivante.

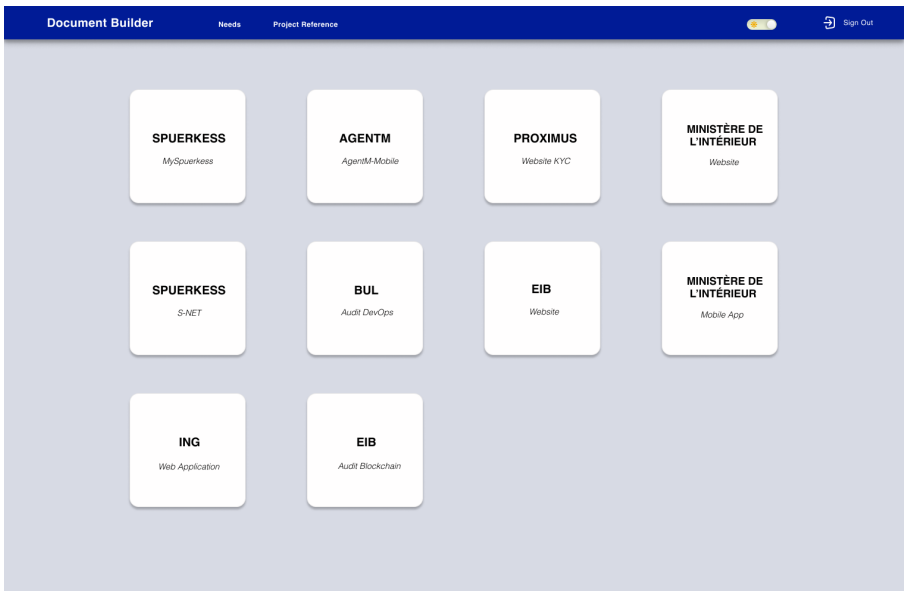

FIGURE 4.11 – Page de visualisation des projets de l'entreprise en cours

Parallèlement, la fonction de filtrage par nom de projet ou par nom de client est également disponible, permettant à l'utilisateur de retrouver rapidement les projets qui l'intéressent.

Si l'utilisateur clique sur l'une de ces cartes, il est redirigé vers la page d'édition des références. La maquette de cette page est disponnible sur la figure 4.12 suivante.

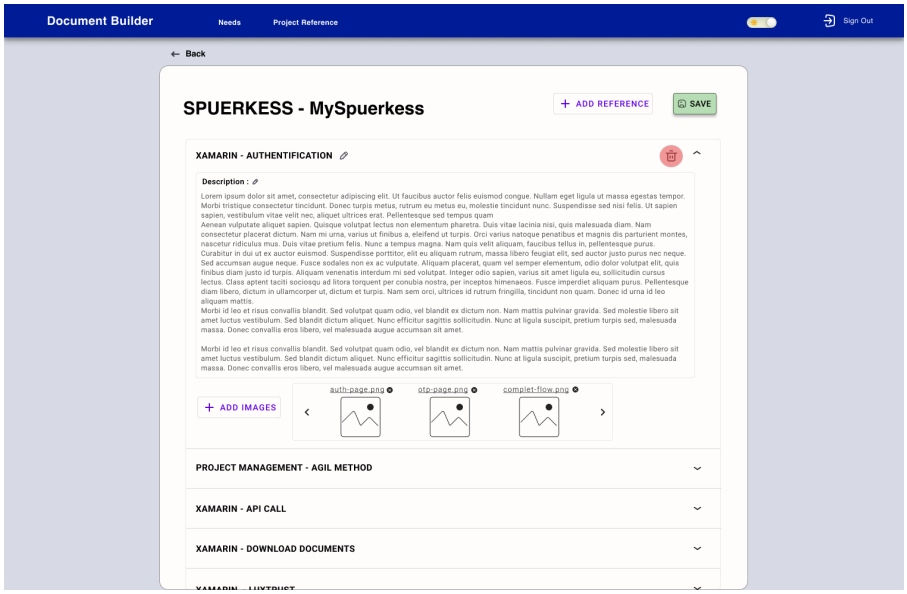

FIGURE 4.12 – Page de modification des références à un projet

Cette page offre la possibilité de créer des paragraphes pour décrire de manière unique un projet spécifique. Ces paragraphes seront ultérieurement utilisés comme contenu pour la section "Projets de références" dans une réponse typique à un appel d'offres. Pour chaque projet, il est possible d'associer plusieurs références, chacune contenant une description, un titre et des images illustratives. Cette fonctionnalité permet de présenter de manière détaillée les projets antérieurs, ce qui renforce la crédibilité et l'attractivité des réponses aux appels d'offres en montrant l'expertise et l'expérience de l'entreprise.

4.3 Choix des technologies et outils

4.3.1 Choix de l'environnement de développement

Le processus de développement d'une application efficace et stable nécessite des choix soignés en termes d'environnement de développement, d'outils et de technologies. Chaque élément choisi contribue à la productivité, à la qualité du code et à la facilité de collaboration au sein de l'équipe. Voici les choix d'environnement de développement pour le projet "document-builder", justifiés par leurs avantages et leur pertinence dans le contexte de l'entreprise NEOFACTO.

Gestionnaire de Paquets Yarn/npm

L'une des pierres angulaires du développement moderne d'applications est la gestion des dépendances. Dans le projet "document-builder", le choix s'est porté sur Yarn en tant que gestionnaire de paquets, avec une connaissance transposable à npm, pour plusieurs raisons fondamentales.

Installation et Gestion des Dépendances : Yarn et npm facilitent grandement l'installation des dépendances requises par le projet. En spécifiant ces dépendances dans un fichier de configuration, comme le fichier package.json, l'ensemble du processus d'installation devient automatisé. Cela garantit que tous les membres de l'équipe ont accès aux mêmes versions des dépendances, assurant ainsi la cohérence du code.

Sécurité des Dépendances : La sécurité des dépendances est une préoccupation majeure dans le développement logiciel. Les gestionnaires de paquets jouent un rôle crucial dans cette dimension. Yarn et npm permettent de détecter automatiquement les vulnérabilités connues au sein des dépendances. En utilisant la commande yarn audit, par exemple, les développeurs peuvent obtenir une liste de vulnérabilités potentielles dans les dépendances du projet. De plus, ces outils offrent des mécanismes pour mettre à jour intelligemment les dépendances vulnérables, ce qui est crucial pour maintenir la sécurité de l'application au fil du temps. Ce point sera développé dans la partie sécurité de ce mémoire.

Mise à Jour Intelligente : Outre la sécurité, Yarn et npm fournissent également des fonctionnalités avancées pour gérer les mises à jour des dépendances. Ils permettent aux développeurs de mettre à jour les dépendances en tenant compte des dépendances d'autres paquets. Cela garantit que les mises à jour sont compatibles et n'introduisent pas de conflits indésirables.

Gestion des Versions : Les gestionnaires de paquets permettent de spécifier des gammes de versions pour les dépendances. Cela offre une flexibilité dans le choix des versions tout en maintenant un contrôle sur les mises à jour automatiques. Les développeurs peuvent ainsi décider quand et comment mettre à jour les dépendances.

En somme, Yarn et npm simplifient considérablement la gestion des dépendances, renforcent la sécurité du projet en identifiant et en corrigeant les vulnérabilités, et offrent un cadre pour des mises à jour gérées et harmonieuses. Dans le contexte du projet "document-builder", ces gestionnaires de paquets sont devenus des alliés essentiels pour garantir un développement fluide et sécurisé.

Nodemon et Typescript pour un Développement Efficace

Pour optimiser le flux de développement dans le cadre du projet "document-builder", deux choix tactiques ont été faits : l'utilisation de Nodemon et la transcompilation du typescript.

Nodemon : Dans le cycle de développement, la répétition de modifications de code et de tests est la norme. Pour faciliter ce processus, l'outil Nodemon a été incorporé. Nodemon surveille en permanence les modifications apportées au code source. Dès qu'une modification est détectée, il redémarre automatiquement le serveur. Cette automatisation élimine le besoin de redémarrer manuellement le serveur à chaque changement, ce qui accélère considérablement le rythme d'itération. Les développeurs peuvent se concentrer sur la création de nouvelles fonctionnalités ou la correction de bugs sans être ralentis par les interruptions dues aux redémarrages manuels.

Typescript et Transcompilation : TypeScript a été préféré à JavaScript en raison de ses avantages en matière de développement. Avec sa vérification statique des types, TypeScript permet de détecter les erreurs dès la phase de développement, réduisant ainsi les bugs et améliorant la qualité du code. Le processus de développement implique la rédaction de code TypeScript, qui doit finalement être exécuté en tant que code JavaScript sur le serveur. Pour cette raison, le processus de transcompilation est introduit. Dans un premier temps, le code TypeScript est soumis à une vérification minutieuse des types grâce à TypeScript. Cela permet de détecter les erreurs potentielles dès les premières étapes du développement, réduisant ainsi les erreurs coûteuses à des étapes ultérieures. Ensuite, le code est transcompilé en JavaScript. Cette étape est rapide, prenant généralement environ une seconde. En production, ce code JavaScript transcompilé garantit des performances optimales et une exécution fluide. En combinant la vérification des types et la transcompilation, le processus de développement devient plus fiable et efficace.

Ainsi, avec Nodemon pour automatiser le redémarrage du serveur et la transcompilation pour garantir une conversion optimale du code TypeScript en JavaScript, le développement dans le projet "document-builder" est orchestré de manière à favoriser la productivité, la fiabilité et les performances.

Collaboration et Gestion de Projet avec Git et GitLab

Le choix de Git en tant que système de contrôle de version et de GitLab en tant que plateforme de gestion du code source pour le projet "document-builder" a apporté une dimension collaborative essentielle au développement. Ces outils sont fondamentaux pour la gestion de projets en équipe et s'intègrent particulièrement bien dans une méthodologie de développement agile.

Suivi Précis des Modifications : Git, en tant que système de contrôle de version décentralisé, permet à chaque membre de l'équipe de suivre et de comprendre précisément les modifications apportées au code source. Les commits, branches et merges offrent une vue claire de l'évolution du code au fil du temps.

Collaboration Harmonieuse : Pour le développement en équipe, Git est inestimable. Il facilite la création de branches spécifiques pour chaque nouvelle fonctionnalité ou tâche, permettant aux développeurs de travailler simultanément sur différentes parties du code sans interférer les uns avec les autres. Les merges ultérieurs facilitent l'intégration fluide de ces travaux.

Gestion Agile des Projets : La méthodologie agile, basée sur des itérations et une collaboration continue, est grandement simplifiée grâce à Git et GitLab. Chaque nouvelle fonctionnalité ou amélioration peut être traitée dans une branche distincte, puis fusionnée dans la branche princi-

pale (master) lorsque prête. Cela assure une gestion flexible des changements et une amélioration continue du projet.

Vérification du Code et Déploiement Automatisés : GitLab propose des fonctionnalités puissantes de CI/CD (Intégration Continue / Déploiement Continu). Les pipelines automatisent la construction et les tests du code à chaque modification. Cela garantit que le code répond aux normes et fonctionne correctement. De plus, le déploiement automatisé réduit les risques d'erreurs humaines lors du déploiement en production.

Facilitation des Revues de Code : Les merge requests dans GitLab facilitent les revues de code. Ils fournissent une vue d'ensemble claire des modifications apportées et permettent aux membres de l'équipe de commenter spécifiquement sur chaque partie du code. Cela accélère la collaboration et améliore la qualité du code.

En conclusion, l'utilisation de Git et GitLab pour le projet "document-builder" a grandement amélioré la collaboration et la gestion du code source. Cette combinaison a permis une approche de développement agile fluide, des revues de code plus efficaces et une automatisation des processus de construction, de test et de déploiement.

En somme, ces choix d'environnement de développement ont été faits en gardant à l'esprit l'efficacité, la qualité du code et la collaboration au sein de l'équipe. Ils ont contribué à créer un flux de travail fluide et efficace pour le projet "document-builder" au sein de l'entreprise NEOFACTO. Une section dédiée au choix de VSCode comme éditeur de code est disponnible dans les annexes.

4.3.2 Choix des frameworks et bibliothèques

Dans le processus de développement de l'application, le choix des frameworks et bibliothèques appropriés joue un rôle crucial pour garantir une mise en œuvre efficace, maintenable et évolutive. Deux frameworks majeurs ont été sélectionnés pour différents aspects de l'application : Angular et Express.js avec Jest.

Angular

Gestion Asynchrone/Mutex Automatiques

Angular, en tant que framework basé sur TypeScript, intègre des fonctionnalités puissantes pour gérer de manière asynchrone les opérations. TypeScript offre une syntaxe de programmation qui facilite la manipulation des opérations asynchrones, ce qui est essentiel pour le développement d'applications modernes réactives. En utilisant des méthodes telles que les fonctions asynchrones et les promesses, Angular permet aux développeurs de gérer des tâches longues sans bloquer l'interface utilisateur. Cette approche évite les problèmes de latence et d'expérience utilisateur médiocre.

Sécurité dans le Contexte d'Angular

Comparé à des langages comme le C, Angular offre un environnement plus sécurisé grâce à sa nature basée sur le web. Cependant, il existe encore des considérations en matière de sécurité, notamment la protection contre les attaques XSS (Cross-Site Scripting) et CSRF (Cross-Site Request Forgery). Les développeurs doivent être vigilants lorsqu'ils manipulent les données d'entrée et mettent en place des mécanismes de validation et d'échappement appropriés pour prévenir les attaques. Angular propose des fonctionnalités intégrées pour aider à sécuriser les applications,

mais la vigilance des développeurs reste cruciale.

Auto-injection des Services

Angular propose un système d'injection de dépendances intégré qui facilite la gestion des dépendances entre les composants. Le concept d'auto-injection permet aux développeurs d'indiquer les dépendances requises pour un composant, et Angular se charge de les fournir automatiquement lors de la création d'une instance de ce composant. Cela favorise la modularité, la réutilisabilité et la facilité de test. Les services peuvent être injectés dans d'autres services, modules ou composants, offrant ainsi une architecture flexible et maintenable.

Gestion des Observables et Promises

Angular exploite le modèle de programmation réactive avec la gestion des observables, qui suit le design pattern Observer. Les observables permettent de traiter facilement les flux de données asynchrones, tels que les événements utilisateur, les appels API et les mises à jour de l'interface utilisateur. Ce modèle favorise la réactivité et la fluidité de l'application. D'un autre côté, les promesses sont également utilisées pour gérer les opérations asynchrones, mais elles suivent un modèle de traitement séquentiel. Les observables offrent une plus grande flexibilité dans la gestion des flux de données complexes, tandis que les promesses sont plus adaptées aux opérations asynchrones simples.

Tests Intégrés avec Jasmine dans Angular

Angular propose un cadre solide pour les tests intégrés, mettant en avant le framework de test Jasmine. Les tests intégrés sont essentiels pour garantir le bon fonctionnement de chaque composant, module et service dans une application. Jasmine offre une syntaxe claire et expressive pour écrire des suites de tests et des assertions, ce qui simplifie la création de cas de test complets et compréhensibles.

Le framework Jasmine permet aux développeurs de décrire les comportements attendus d'une manière proche de la langue naturelle. Chaque suite de tests commence par une description textuelle du comportement que le test va évaluer. À l'intérieur de chaque suite, les développeurs peuvent définir des spécifications qui vérifient des aspects spécifiques du comportement. Ces spécifications incluent des assertions qui valident les résultats attendus par rapport aux données fournies.

En conclusion, Angular apporte une multitude de fonctionnalités pour simplifier le développement d'applications web modernes, en automatisant la gestion asynchrone, en favorisant la sécurité, en facilitant l'injection des services et en offrant des mécanismes puissants pour gérer les observables et les promesses. Ces caractéristiques contribuent à la création d'applications performantes, réactives et sécurisées, tout en réduisant la complexité du développement.

Express.js

Gestion Asynchrone/Mutex Automatiques

Tout comme Angular, Express.js reconnaît l'importance de gérer les opérations asynchrones dans les applications web modernes. En utilisant des mécanismes tels que les fonctions de rappel (callbacks), les promesses et les middlewares asynchrones, Express.js permet aux développeurs de créer des applications web réactives sans sacrifier la performance. Les fonctions de rappel, bien qu'étant un modèle plus ancien, sont toujours utilisées dans Express.js pour gérer les opérations asynchrones. Cependant, les développeurs peuvent également tirer parti des avantages des promesses et des middlewares asynchrones pour des flux de contrôle plus propres et plus lisibles.

Avantages en Termes de Sécurité dans Express.js

Express.js offre une abstration de haut niveau pour la gestion des requêtes et des réponses HTTP, ce qui contribue à minimiser les risques de vulnérabilités courantes telles que les injections SQL et les attaques XSS. Les développeurs peuvent se concentrer sur la logique métier sans avoir à gérer directement les détails de la communication HTTP. Cependant, comme avec tout framework, la sécurité reste une préoccupation majeure. Les développeurs doivent mettre en œuvre des pratiques telles que la validation des données d'entrée, la protection contre les injections et la mise en œuvre de stratégies de sécurité pour se protéger contre les attaques potentielles.

Automatisation de l'Injection des Dépendances dans Express.js

Contrairement à Angular, Express.js ne possède pas de système d'injection de dépendances intégré. Cependant, cette approche plus légère offre également des avantages. Les développeurs ont plus de liberté pour choisir comment gérer les dépendances dans leurs applications. Ils peuvent opter pour des solutions tierces telles que Awilix ou gérer manuellement l'injection des dépendances. Cette flexibilité permet d'adapter l'architecture aux besoins spécifiques du projet.

Gestion de l'Asynchronicité dans les Middlewares

Express.js prend en charge les dernières fonctionnalités d'ECMAScript, notamment les fonctions asynchrones async/await. Cela permet aux développeurs de gérer les opérations asynchrones de manière plus concise et lisible, en évitant les structures de code complexes liées aux callbacks. De plus, les middlewares asynchrones simplifient la gestion des opérations telles que l'authentification, l'autorisation et la gestion des erreurs. Cela facilite la création d'applications modulaires et hautement personnalisables.

En somme, Express.js offre des mécanismes d'asynchronicité similaires à Angular pour gérer les opérations non bloquantes, tout en fournissant des avantages spécifiques aux développeurs, tels que la flexibilité de l'injection des dépendances, la gestion intuitive des opérations asynchrones avec async/await et la simplicité des middlewares pour la création d'applications hautement personnalisées et sécurisées.

Manipulation de fichiers binaires

Pour manipuler efficacement des fichiers binaires, tels que les images destinées à illustrer les références aux projets dans les réponses aux appels d'offres, il est crucial de déterminer comment l'application stocke ces données en arrière-plan.

Il existe plusieurs approches pour stocker des fichiers binaires comme des images. Une option consiste à les enregistrer dans une base de données dédiée qui gère automatiquement le stockage et la récupération des images. Cependant, cette approche nécessite la mise en place d'un service supplémentaire au sein de l'application. Étant donné que la base de données Neo4j déjà en place n'est pas conçue pour ce type de données, cette solution a été écartée.

Le choix a été fait de stocker les fichiers directement sur le système de stockage de fichiers du serveur, pour des raisons de simplicité. Pour réaliser cela, des bibliothèques comme Multer sont utilisées pour faciliter la communication entre des frameworks tel que Express.js et le système de fichiers du serveur. Cependant, il est également nécessaire d'attribuer des identifiants uniques à ces fichiers pour les distinguer.

L'une des méthodes courantes pour générer des identifiants uniques est l'utilisation des UUID (Universally Unique Identifier). Il existe deux principales versions d'UUID :

Version 1 : Cette version génère un ID unique basé sur l'adresse MAC d'une carte réseau et le temps actuel. Cependant, si ces informations sont sensibles d'une manière quelconque, il est préférable de ne pas utiliser cette méthode. Cette version permet de reconnaître facilement si plusieurs UUID ont été générés par la même machine et offre des indications temporelles.

Version 4 : Ces UUID sont générés à partir de nombres aléatoires (ou pseudo-aléatoires). Si l'objectif est simplement de générer un UUID, c'est probablement la meilleure option. Cette version facilite également la recherche de correspondances lorsque l'on examine une longue liste d'informations associées à des UUID.

Les versions 2 et 3 proposent des algorithmes qui permettent de créer des UUIDs reproductibles, ce qui n'est pas nécessaire dans le cadre de notre application.

Dans le contexte de cette application, l'utilisation d'UUID de la version 4 pour l'identification des fichiers semble être la plus appropriée, compte tenu de sa simplicité et de sa pertinence pour le stockage de fichiers.

Tests avec Jest dans Express.js

Dans le processus de développement d'applications, l'assurance de la qualité du code est primordiale. C'est là que Jest entre en jeu en tant que framework de test très populaire pour les applications Express.js. Jest simplifie grandement la création et l'exécution de tests unitaires et d'intégration. Grâce à sa configuration minimale et à sa syntaxe conviviale, les développeurs peuvent se concentrer sur la création de tests plutôt que sur la mise en place de l'infrastructure de test.

L'un des avantages majeurs de Jest est sa fonction de détection automatique des modifications du code source, qui déclenche automatiquement l'exécution des tests pertinents. Cela accélère le processus de développement et assure la stabilité continue du code. De plus, Jest fournit des rapports de test clairs et informatifs, facilitant l'identification rapide des erreurs et des problèmes potentiels.

4.3.3 Choix des outils pour l'intégration de l'IA (ChatGPT)

L'intégration d'une intelligence artificielle comme ChatGPT au sein de l'application "document-builder" nécessitait une approche réfléchie pour garantir une expérience utilisateur fluide et une utilisation efficace des fonctionnalités d'IA. Deux fonctionnalités sont attendues de l'IA : comprendre des documents PDF des appels d'offre afin d'en extraire toute information qui serait nécessaire à la réponse, et la rédaction de sections ou de parties de section en réutilisant les informations du document. Parmi les différentes options disponibles, la bibliothèque Langchain s'est avérée être l'outil le plus approprié pour répondre à nos besoins spécifiques.

Introduction à LangChain

Langchain[4] est une bibliothèque open-source spécialement conçue pour simplifier l'intégration d'IA basée sur le langage dans des applications JavaScript. Cette bibliothèque vise à faciliter l'utilisation de services d'IA tels que ChatGPT en gérant de manière transparente des opérations asynchrones complexes, tout en offrant des fonctionnalités de composition et de traitement simplifié des réponses générées par l'IA.

Simplification de l'Utilisation de l'IA

L'une des principales raisons pour lesquelles Langchain a été choisi réside dans sa capacité à simplifier l'interaction avec ChatGPT. Cette bibliothèque permet la création de chaînes d'opérations asynchrones, ce qui est crucial pour l'intégration de l'IA dans une application qui nécessite des interactions fluides avec l'utilisateur. La mise en place de chaînes d'opérations permet de séquencer les appels à l'IA et aux autres services, évitant ainsi les problèmes de synchronisation et améliorant la réactivité de l'application.

Utilisation de Fonctions de Composition

Langchain offre également des fonctions de composition, telles que le patron map/reduce, qui permettent de manipuler les réponses générées par ChatGPT de manière efficace. Cela est particulièrement utile lorsque les réponses nécessitent un formatage spécifique ou doivent être combinées avec d'autres données.

Dans notre cas, chacune des sources de données (document PDF de l'appel d'offre, données de l'entreprise sur Boond Manager, CV à jour des employés, templates de documents sur le drive ...) seraient d'abord traitées pour être compressées (résumé de document ou extraction d'informations importantes dans le contexte de la réponse à l'appel d'offre) correspondant à l'opération 'map', puis seraient agrégées pour générer le contenu d'une section ou partie de section de la réponse. On retrouve un schéma qui présente la chaîne associée à cette suite d'opération sur la figure 4.13 suivante.

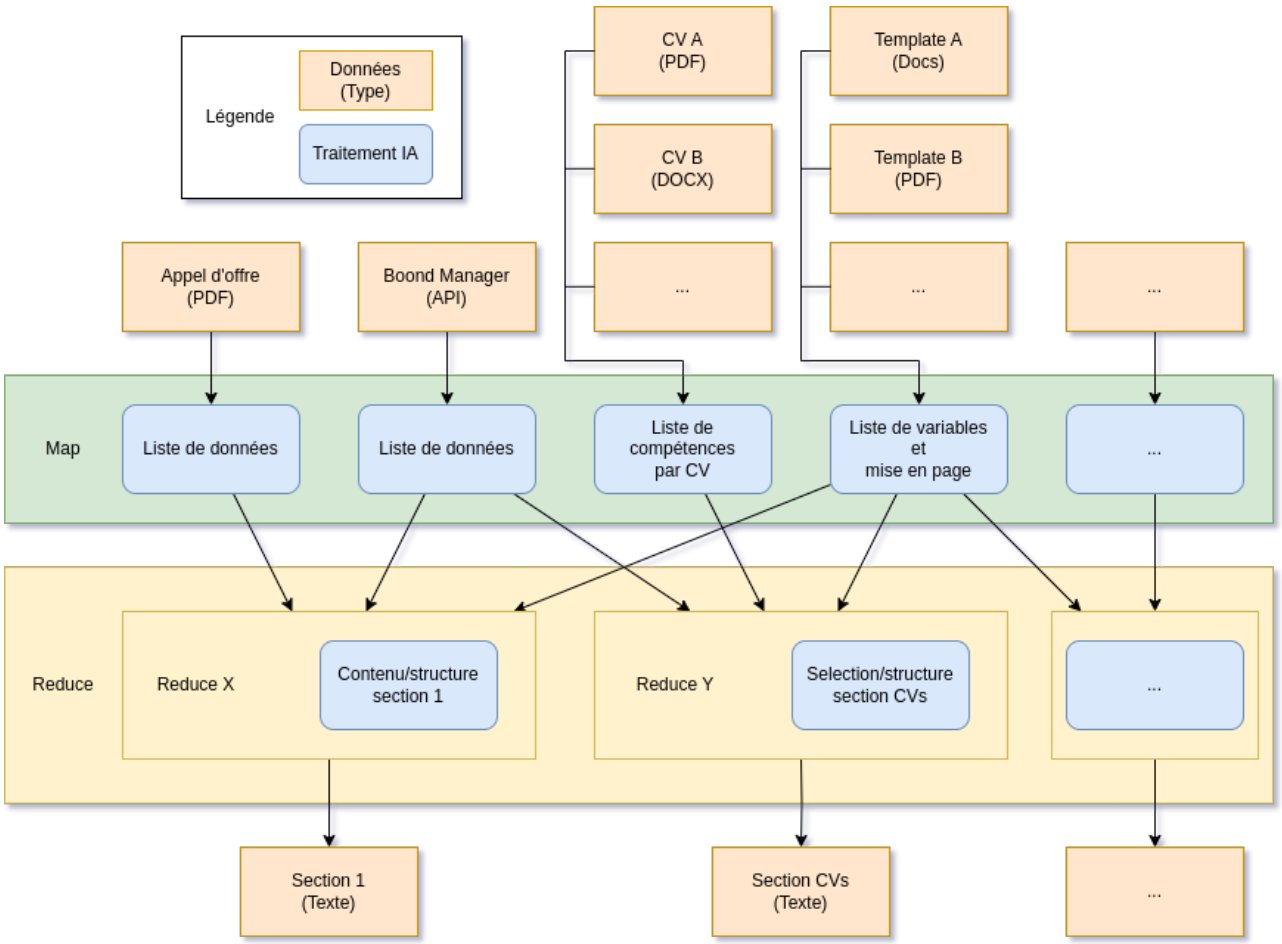

FIGURE 4.13 – Exemple de traitement des données à l'aide d'une chaîne Map/Reduce d'appel à GPT3

Sur cette figure on observe deux types d'entités, les données et les traitement effectués par l'intelligence artificielle. Les différentes données sont envoyées au début de la chaîne, qui demande le premier traitement, puis l'utilisation de ces données agrégées afin d'obtenir le contenu souhaité pour la section ou sous-section.

À chaque traitement est associé une entrée spécifique, qui permet d'orienter l'IA sur la démarche à suivre. Il suffit ensuite d'insérer dans l'entrée les données à traiter pour permettre à l'IA de générer un contenu convenable pour le contexte de l'appel d'offre.

On remarque cependant que les traitements en entrées peuvent recevoir des informations sensibles provenant du service de gestion de l'entreprise. Comme il est coûteux d'héberger sois-même le service d'IA, il est pertinent d'imaginer lors de l'implémentation, un système d'anonymisation et de désanonymisation des données à travers des services situés avant l'entrée et après la sortie de la chaîne. Cela permettrait de limiter les données sortantes de l'entreprise, qui pourraient être accessibles aux fournisseur du service d'IA (OpenAI dans notre cas). On retrouve sur la figure 4.14 le schéma complété par la mise en évidence des points de traitements liés à l'anonymisation des données envoyées à l'IA.

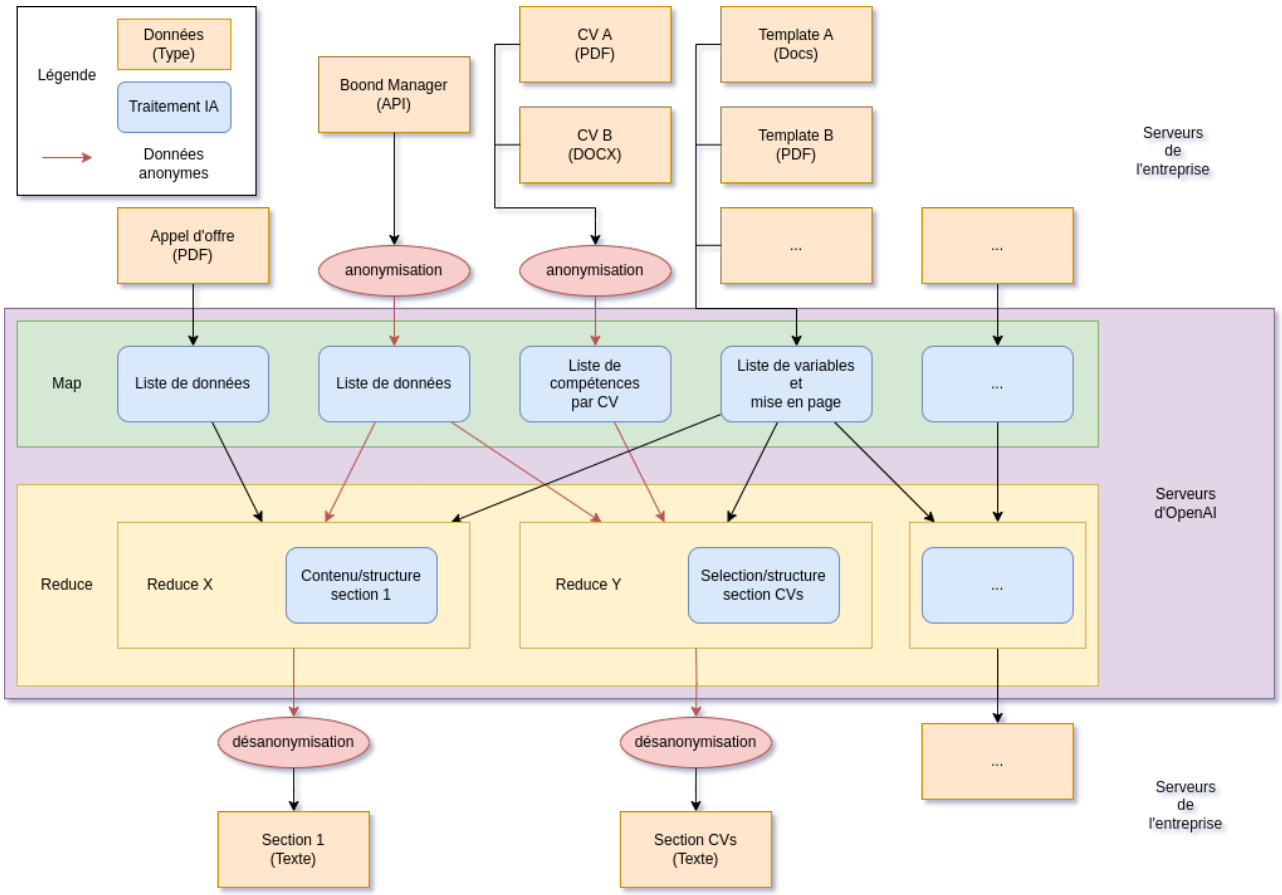

FIGURE 4.14 – Exemple de traitement des données avec mise en évidence de l'anonymisation lors d'appel à GPT3

Avec cette conception de l'utilisation de l'IA de manière anonymisée, comme on peut le voir sur la figure précédente, les données sensibles ne sortent pas des serveurs de l'entreprise. Seules les données non sensibles (comme les templates) sont envoyées directement aux serveurs hébergeant le service d'IA.

Finallement, le choix de la bibliothèque Langchain pour l'intégration de l'IA dans l'application "document-builder" a permis de simplifier considérablement l'interaction avec ChatGPT. Les fonctionnalités d'asynchronicité, la gestion transparente des chaînes d'opérations et les fonctions de composition ont grandement amélioré l'expérience utilisateur et ont facilité le développement de fonctionnalités basées sur l'IA.

La composante d'intelligence artificielle de ce projet n'a pas pu être mise en œuvre avant la clôture de la période de stage. Néanmoins, cette fonctionnalité conserve sa pertinence et devrait être maintenue en tant que projet pour l'après-stage.

4.4 Développement de l'application

4.4.1 Intégration de Boond Manager

Présentation et Contexte

Boond Manager représente un service essentiel pour les Entreprises de Services Numériques (ESN) telles que NEOFACTO, offrant une solution de gestion simplifiée et centralisée. Principalement géré par le service des ressources humaines de NEOFACTO, ce logiciel facilite la gestion des plannings horaires des employés sur différents projets. De plus, il sert de plateforme pour suivre les besoins des clients et les projets qui en découlent.

L'équipe enregistre une variété d'informations dans le système, en faisant une source fiable et centralisée d'informations. Ainsi, l'intégration de cet outil dans l'application de réponse à appel d'offres est devenue cruciale pour garantir la cohérence et la centralisation des données liées aux clients.

L'API Boond Manager

Boond Manager fournit une API REST accompagnée d'une documentation en RAML[7]. La Figure X présente une capture d'écran de cette documentation, extraite depuis un navigateur web.

La documentation propose une liste exhaustive d'endpoints avec des exemples de réponses, facilitant ainsi la compréhension et l'utilisation de l'API.

Pour interagir avec l'API Boond Manager, nous utilisons la bibliothèque Axios en TypeScript. Voici un exemple de code qui illustre comment effectuer une requête via Axios :

```
import axios from 'axios';

const API_URL = 'https://api.boondmanager.com'; // URL de l'API Boond Manager

// Exemple de êrequte GET
axios.get(`${API_URL}/projects`)
    .then(response => {
        console.log(response.data);
    })
    .catch(error => {
        console.error(error);
    });
```

Listing 4.1 – Exemple d'utilisation d'Axios pour les requêtes API

Cette approche permet une intégration transparente de l'API Boond Manager dans notre application, permettant ainsi un accès fluide aux données cruciales pour les réponses aux appels d'offres.

4.4.2 Intégration de l'api Google

Cette section se concentre sur l'interaction de l'application avec l'API Google pour enrichir ses fonctionnalités, notamment la gestion des documents sur Google Drive. Cette intégration a été réalisée en utilisant la librairie JavaScript "googleapis" mise à disposition par Google.

La documentation de la librairie "googleapis" fournie par Google offre un guide complet sur l'utilisation des différentes fonctionnalités de l'API Google[2]. Cette documentation, constitue une ressource essentielle pour comprendre les bases de l'intégration et de l'interaction avec les services Google.

Pour interagir avec Google Drive, l'application utilise le client OAuth2 avec un refresh token. Cela permet de garantir que l'application peut accéder aux ressources Google de manière sécurisée et avec des autorisations adéquates.

L'intégration de l'API Google implique la construction de requêtes appropriées pour interagir avec les services Google. Cela comprend la structuration correcte du corps de la requête, la définition des paramètres nécessaires et la gestion des réponses reçues. Cette approche garantit que l'application communique efficacement avec Google Drive et peut exécuter des opérations telles que la création de documents d'appels d'offres.

Parmi les fonctionnalités implémentées, la création de documents d'appels d'offres est au cœur de l'intégration de l'API Google. Cela implique l'utilisation de requêtes spécifiques pour créer de nouveaux documents sur Google Drive. De plus, l'application a été dotée de fonctionnalités de personnalisation et de génération automatique de texte. Ces fonctionnalités sont mises en œuvre en utilisant des méthodes telles qu'"insertText" et "updateStyle" pour manipuler le contenu et le formatage des documents.

En résumé, l'intégration de l'API Google dans l'application a été effectuée en utilisant la librairie "googleapis", avec une attention particulière portée à la documentation fournie par Google. Cette intégration a permis d'étendre les fonctionnalités de l'application en exploitant les capacités de Google Drive, notamment la création de documents et la personnalisation du contenu.

4.4.3 Angular Material Design

L'intégration visuelle cohérente et esthétique des composants de l'application est essentielle pour offrir aux utilisateurs une expérience utilisateur homogène. C'est ici que le framework Angular Material Design, développé par Google, entre en jeu. Il s'agit d'une bibliothèque de composants d'interface utilisateur prêts à l'emploi, suivant les principes du Material Design, un langage visuel de conception moderne et élégant, utilisé sur les téléphones Android.

Angular Material Design offre une gamme complète de composants, de directives et de services qui permettent de créer des interfaces utilisateur interactives et engageantes. Parmi les composants disponibles, on retrouve des boutons, des cartes, des menus déroulants, des formulaires, des boîtes de dialogue, des barres de progression, etc. Chaque composant est soigneusement conçu pour répondre aux critères d'esthétisme, de convivialité et de cohérence visuelle.

La documentation d'Angular Material Design[1] est particulièrement riche et détaillée. Elle propose des guides, des exemples et des démonstrations pour chaque composant, ainsi que des instructions pour leur implémentation. Cette documentation facilite grandement la prise en main et l'utilisation de ces composants dans le développement de l'application.

En intégrant Angular Material Design dans le projet, j'ai pu profiter de plusieurs avantages significatifs. Tout d'abord, l'utilisation de composants prédéfinis a permis d'unifier l'apparence et le comportement des éléments d'interface, créant ainsi une expérience utilisateur fluide et cohérente. De plus, la bibliothèque inclut également des animations intégrées, ajoutant un niveau supplémentaire d'interactivité et d'attrait visuel aux interactions utilisateur.

En somme, l'intégration d'Angular Material Design a grandement simplifié et amélioré le processus de conception d'interface utilisateur. Cette approche a permis de créer un environnement visuel attrayant et convivial tout en maintenant une uniformité visuelle à travers l'ensemble de l'application.

4.5 Sécurité

À travers mon cursus spécialisé en sécurité informatique à TELECOM Nancy, j'ai eu l'opportunité d'explorer les dimensions critiques de la cybersécurité, un domaine en constante évolution. Mes professeurs ont souligné les besoins croissants en matière de développement sécurisé dans un monde numérique de plus en plus interconnecté. Cette convergence entre ma passion pour la cybersécurité et mon attrait pour le développement m'a conduit à me plonger dans le domaine de la programmation sécurisée.

La programmation sécurisée, une sous-discipline de la cybersécurité, se focalise sur la création de code de haute qualité afin de réduire la surface d'attaque pour les hackers. Son objectif est de concevoir du code résistant à diverses attaques connues et de le rendre robuste face aux scénarios imprévus et aux données d'entrée incertaines (effets de bords). Bien que de nombreux concepts appris au cours de ma formation ne soient utiles que de manière indirecte, ils fournissent une perspective précieuse qui me permet d'adopter la mentalité d'un hacker, en m'aidant à envisager les moindres détails qui pourraient devenir des sources de perturbations ou d'incohérences.

Cette section de mon mémoire se consacre à l'approfondissement du concept de programmation sécurisée à travers le développement d'un service pour NEOFACTO. Nous examinerons comment les principes de programmation sécurisée sont mis en œuvre dans la création de l'application, en veillant à ce que chaque aspect du développement contribue à la robustesse et à la sécurité globale du système.

4.5.1 Notion de développement sécurisé

Méthodologie durant le développement

Lors de la phase de développement, adopter une approche sécurisée est essentiel pour garantir la fiabilité du code. Cette approche implique de prendre en compte divers scénarios, y compris les effets de bords potentiels pour chaque fonction. En adoptant une perspective à la fois d'utilisateur et d'attaquant, il est possible d'anticiper les vulnérabilités potentielles et les vecteurs d'attaque. Cette approche proactive permet d'identifier les points faibles du code dès le début du développement, ce qui facilite la correction de problèmes avant qu'ils ne deviennent des failles de sécurité.

Vérification des Librairies

Les librairies externes jouent un rôle crucial dans le développement d'applications modernes. Cependant, elles peuvent également être une source potentielle de vulnérabilités. Il est primordial de surveiller régulièrement la sécurité des librairies utilisées. Cela inclut de s'assurer que les librairies sont régulièrement mises à jour avec les derniers correctifs de sécurité. De plus, la vérification de l'activité des développeurs et de la communauté autour d'une librairie peut aider à évaluer sa fiabilité. Opter pour des librairies populaires et bien entretenues réduit les risques de failles de sécurité potentielles.

Vérification des Entrées Utilisateur

La sécurité d'une application dépend en grande partie de la manière dont elle gère les données entrantes. Les attaques de type "injection" et autres vulnérabilités similaires peuvent être exploitées si les données utilisateur ne sont pas correctement validées et échappées. La mise en œuvre de mécanismes de validation et de filtrage pour les entrées utilisateur est donc fondamentale. Les champs de saisie, les formulaires et autres points d'interaction avec les utilisateurs doivent être rigoureusement contrôlés pour empêcher les attaques telles que les injections SQL ou les scripts cross-site.

Programmation Concurrente et Comportements Indésirables

La programmation concurrente peut introduire des défis particuliers en matière de sécurité. Des problèmes tels que les conditions de concurrence et les deadlocks peuvent compromettre la stabilité et la sécurité de l'application. L'utilisation de mécanismes de verrouillage et de gestion de mutex (verrous mutualisés) est essentielle pour éviter les conflits entre threads ou processus. En identifiant et en résolvant les comportements indésirables potentiellement liés à la programmation concurrente, on peut garantir que l'application fonctionne de manière fiable et sécurisée même dans des environnements multitâches.

Tests et Intégration Continue

Les tests jouent un rôle crucial dans le développement sécurisé. L'utilisation de tests unitaires et d'intégration permet de vérifier la résistance du code face aux différents scénarios. Les mocks, qui simulent des composants du système, sont utiles pour isoler des parties spécifiques du code et tester leur fonctionnement. L'intégration continue, qui inclut l'automatisation des tests lors de chaque modification du code, permet de détecter rapidement les erreurs et les vulnérabilités potentielles. En adoptant une approche de tests rigoureuse, on peut créer un code plus sûr et résistant aux attaques.

La partie Test est développée en détails dans la section 4.7 de ce mémoire. L'apport de la sécurité dans les tests assure que ces derniers couvrent la quasi-totalité (selon l'imagination du développeur) des cas sensibles.

Développement sécurisé sous Angular

Angular est un framework JavaScript moderne qui intègre de nombreuses fonctionnalités de sécurité par défaut, réduisant ainsi la surface d'attaque potentielle. Par exemple, Angular protège naturellement contre les attaques de type XSS (Cross-Site Scripting) en appliquant automatiquement des mécanismes de échappement aux données dynamiques insérées dans les templates. De plus, Angular offre des protections CSP (Content Security Policy) pour limiter les sources de contenu exécutable sur une page web et prévenir les attaques XSS.

Pour garantir un développement sécurisé sous Angular, il est essentiel de suivre les bonnes pratiques recommandées par la documentation officielle et la communauté. Le guide de sécurité d'Angular fournit des conseils détaillés sur la manière de concevoir et de développer des applications résistantes aux attaques. Cela comprend la validation et la échappement des données d'entrée, la protection contre les attaques CSRF (Cross-Site Request Forgery), la gestion des permissions d'accès et la mise en œuvre de mécanismes de contrôle d'accès.

Cependant, certaines vulnérabilités spécifiques peuvent persister malgré les protections intégrées d'Angular. Par exemple, l'authentification peut présenter des sensibilités, notamment lors de l'intégration de services tiers pour l'authentification, tels que l'utilisation d'Angularx-social-login pour l'authentification via Google OAuth2. Il est crucial de comprendre en détail comment ces ser-

vices fonctionnent et comment ils gèrent les jetons d'accès et les données utilisateur pour éviter tout risque de fuite d'informations sensibles.

Un autre aspect à considérer concerne la prévisualisation cross-site, qui implique la nécessité de contourner les règles de la Content Security Policy (CSP), laquelle est utilisée pour visualiser les documents construits sur le Drive.

Des cybercriminels peuvent exploiter les failles pour lancer des scripts malveillants via cette prévisualisation, lesquels seraient exécutés dans le navigateur de l'utilisateur. Une méthode pour éviter cela consiste à rendre statique les pages de prévisualisation et à restreindre l'exécution de scripts externes. En appliquant des mesures de développement sécurisé spécifiques à ces scénarios, la résilience de l'application face à d'éventuelles attaques est renforcée.

Le développement sécurisé sous Angular nécessite donc une compréhension approfondie des fonctionnalités de sécurité intégrées, ainsi qu'une vigilance continue en ce qui concerne les points sensibles de l'application. En suivant les meilleures pratiques et en restant au fait des dernières vulnérabilités et solutions, vous pouvez construire une application robuste et sécurisée.

Programmation défensive

La programmation défensive est une approche de développement qui vise à anticiper et à gérer les erreurs et les situations inattendues pour garantir la robustesse et la fiabilité d'une application. Elle se concentre sur la validation des données d'entrée, la gestion des erreurs et la mise en place de mécanismes de récupération appropriés.

L'utilisation de déclarations throw et de blocs try/catch est une pratique courante en programmation défensive. Lorsqu'une condition inattendue ou une erreur survient, une exception est levée à l'aide de l'instruction throw, ce qui interrompt le flux normal d'exécution. Les blocs try/catch permettent de capturer ces exceptions et de gérer les erreurs de manière contrôlée, en prenant des mesures appropriées pour garantir que l'application continue de fonctionner correctement.

Voici un exemple en TypeScript qui illustre l'utilisation de la programmation défensive avec des blocs try/catch :

```
try {
    // Code susceptible de provoquer une exception
    const result = getInfoFromDistantServer();
    console.log("éRsultat:", result);
} catch (error) {
    // Gestion de l'erreur
    console.error("Une erreur s'est produite:", error.message);
}
```

Listing 4.2 – Exemple de programmation défensive avec le bloc try/catch en TypeScript

Dans cet exemple, si une exception est levée lors de l'exécution de la fonction getInfoFromDistantServer, par exemple si le serveur distant n'est pas disponnible, le bloc catch capture l'exception et affiche un message d'erreur approprié. Le code peut continuer sans être perturbé par ce dysfonctionnement.

La programmation défensive est particulièrement importante lors du développement d'applications sécurisées, car elle permet de minimiser les risques liés aux erreurs et aux comportements inattendus. Elle nécessite une planification minutieuse dès la conception de l'application, en identifiant les points sensibles où des erreurs pourraient se produire et en mettant en place des mécanismes de gestion appropriés.

En conclusion, c'est une stratégie essentielle pour développer des applications résilientes et robustes. En anticipant les erreurs et en mettant en œuvre des mécanismes de récupération adéquats, il est possible d'améliorer la qualité et la sécurité du code.

4.5.2 Surveillance

La sécurité d'une application ne se limite pas à sa phase de développement, mais nécessite également une surveillance continue pour détecter et réagir aux vulnérabilités potentielles. L'une des pratiques essentielles pour maintenir la sécurité de votre application est la surveillance régulière des dépendances et des vulnérabilités associées. Pour cela, des outils tels que l'audit de dépendances peuvent être utilisés.

L'outil `yarn audit` (équivalent de `npm audit`) est un moyen puissant pour évaluer la sécurité de vos dépendances. En exécutant cette commande, l'outil analyse les packages utilisés dans votre projet et vérifie s'il y a des vulnérabilités connues. La Figure 4.15 montre un exemple de la sortie de la commande `npm audit` pour un projet Node.js.

```
(nodejs-12) 9888 /tmp/myapp  » npm audit

                    === npm audit security report ===

                              Manual Review
           Some vulnerabilities require your attention to resolve

         Visit https://go.npm.me/audit-guide for additional guidance

┌──────────────┬──────────────────────────────────────────────────────────────┐
│ Moderate     │ Regular Expression Denial of Service                         │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Package      │ postcss                                                      │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Patched in   │ >=8.2.10                                                     │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Dependency of│ @vue/cli-service [dev]                                       │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ Path         │ @vue/cli-service > @intervolga/optimize-cssnano-plugin >     │
│              │ cssnano > cssnano-preset-default > css-declaration-sorter >  │
│              │ postcss                                                      │
├──────────────┼──────────────────────────────────────────────────────────────┤
│ More info    │ https://npmjs.com/advisories/1693                            │
└──────────────┴──────────────────────────────────────────────────────────────┘
```

FIGURE 4.15 – Capture d'écran de la sortie de la commande npm audit dans un example de projet Node.js

Lorsque vous exécutez `yarn audit`, vous obtiendrez une liste de vulnérabilités identifiées, accompagnée d'informations sur l'impact potentiel de chaque vulnérabilité. Ces impacts sont généralement classés en niveaux tels que *Low*, *Moderate*, *High*, etc. Il est recommandé de traiter chaque vulnérabilité détectée en fonction de son niveau d'impact et de sa pertinence pour votre application.

Une pratique courante pour résoudre les vulnérabilités consiste à utiliser la commande `yarn upgrade-interactive`. Cette commande permet d'afficher les mises à jour disponibles pour les packages vulnérables, ce qui facilite la mise à jour sélective de ces packages vers des versions sécurisées. Cela peut être particulièrement utile pour les vulnérabilités critiques qui nécessitent une mise à jour rapide.

Un exemple concret pourrait être une vulnérabilité détectée dans une dépendance utilisée pour gérer les fichiers téléchargés. Si cette vulnérabilité est identifiée comme critique et est patchée par les développeurs du package, une nouvelle version sécurisée sera publiée. Utiliser `yarn upgrade-interactive` permettra alors de mettre à jour cette dépendance spécifique vers la version corrigée, renforçant ainsi la sécurité de l'application.

Il existe des moyens automatisés d'effectuer cette tâche, comme dependaBot qui effectuent les mises à jour des dépendances automatiquement, avec une intégration facile dans la Pipeline, sous GitHub. Il existe un sservice similaire sous GitLab, qui est beaucoup plus complexe à mettre en place. Le programme est intégré à Github ou Gitlab et peut créer des branches ainsi que des merge request que les développeurs ont juste à valider pour mettre à jour leur dépendance.

Il est également possible d'anticiper la mise à jour, dans le cas ou la vulnérabilité est critique, mais le service ne peut être arrêté le temps du développement du patch, et de créer une version alternative d'un librairie pour corriger temporairement la faille manuellement. Cette solution est coûteuse et n'assure pas toujours la protection face à de nouvelles failles causées par les corrections. Ce procédé existe néanmoins dans le cas de situations critiques qui nécessite une certaine immédiateté.

En surveillant régulièrement les vulnérabilités et en appliquant des correctifs rapidement, vous pouvez maintenir un niveau élevé de sécurité pour votre application, réduisant ainsi le risque d'exploitation de failles potentielles.

4.5.3 Analyse statique des vulnérabilités avec Sonar Cube

L'analyse statique des vulnérabilités est un processus crucial pour identifier les vulnérabilités potentielles dans le code source d'une application. Dans le cadre de ce projet, l'outil SonarQube a été choisi pour mener cette analyse approfondie du code back-end de l'application. SonarQube est un outil d'analyse statique du code qui examine le code source pour identifier les problèmes de sécurité, les erreurs de programmation, les violations de règles de codage et d'autres éléments susceptibles de compromettre la qualité et la sécurité du code. Cette analyse peut aider à prévenir les vulnérabilités et à améliorer la robustesse du code. Bien que SonarQube n'ait pas encore été utilisé dans ce projet, il est prévu de l'intégrer dans les étapes ultérieures du développement. Son utilisation permettra d'effectuer une évaluation approfondie du code back-end, garantissant ainsi que les meilleures pratiques de sécurité sont respectées tout au long du processus de développement.

4.5.4 Analyse dynamique des vulnérabilités

L'analyse dynamique des vulnérabilités joue un rôle essentiel dans l'identification des vulnérabilités au niveau du réseau et des systèmes. OpenVAS est l'outil sélectionné pour réaliser cette analyse dynamique dans le contexte de ce projet. OpenVAS est un scanner de vulnérabilités open-source qui inspecte les systèmes, les applications et les réseaux pour détecter les failles de sécurité et les vulnérabilités. Bien que l'analyse dynamique des vulnérabilités avec OpenVAS n'ait pas encore

été mise en œuvre dans le projet, elle est planifiée pour les étapes ultérieures. L'utilisation d'OpenVAS aidera à identifier les vulnérabilités potentielles dans les systèmes et les réseaux, en fournissant des rapports détaillés sur les faiblesses détectées. Cette approche proactive permettra de prendre des mesures pour atténuer les risques de sécurité et de renforcer la résistance de l'application aux attaques potentielles.

Dans le but de garantir un environnement de déploiement sécurisé pour notre application, nous avons pris des mesures substantielles pour configurer NGINX en tant que Web Application Firewall (WAF) et proxy pour l'application. Notre objectif était de mettre en place un déploiement qui adhère aux meilleures pratiques en matière de sécurité des communications et de gestion des certificats SSL/TLS.

Pour assurer la robustesse de notre mise en œuvre sécurisée, nous avons suivi le guide "Nginx SSL/TLS configuration for "A+" Qualys SSL Labs rating". Cette configuration NGINX a été soumise à un test d'évaluation de sécurité SSL à l'aide de l'outil SSL Labs. Les résultats obtenus ont confirmé l'efficacité de notre configuration, obtenant ainsi un score A+.

Voici un aperçu de l'architecture que nous avons utilisée, basée sur trois conteneurs Docker interconnectés :

- Le conteneur "nginx" agit comme notre serveur web principal. Il écoute sur les ports 80 (HTTP) et 443 (HTTPS) pour gérer le trafic entrant. Ce conteneur est configuré pour gérer les connexions HTTPS sécurisées et assure également le rôle de reverse proxy pour acheminer les requêtes vers les services appropriés de notre application.
- Le conteneur "dockergen" est responsable de générer de manière dynamique les configurations nécessaires pour NGINX en fonction des conteneurs et services en cours d'exécution. Cette étape automatisée garantit que la configuration NGINX est toujours à jour et cohérente avec les services actifs.
- Le conteneur "acme" s'occupe de la gestion des certificats SSL/TLS à l'aide de Lets Encrypt, un service de certification automatisé. Il garantit que les certificats sont générés et renouvelés automatiquement, éliminant ainsi le besoin d'une intervention manuelle et garantissant des communications sécurisées.

Cette approche modulaire et automatisée nous a permis de déployer un environnement sécurisé avec des connexions HTTPS chiffrées. Les résultats de notre mise en œuvre sécurisée sont tangibles, améliorant la confidentialité des données et la confiance des utilisateurs envers notre application.

Le résultat de cette mise en place est un déploiement sécurisé qui garantit des communications chiffrées entre les utilisateurs et l'application. La configuration minutieuse de NGINX, couplée à l'utilisation de Lets Encrypt pour les certificats SSL/TLS, a permis de créer un environnement où la sécurité des données et la confidentialité des utilisateurs sont des priorités essentielles.

4.5.5 Authentification dans l'Application

L'authentification dans l'application est gérée à l'aide d'une combinaison de protocoles OAuth2 et de JSON Web Tokens (JWT). Ces mécanismes garantissent un accès sécurisé aux utilisateurs et aux services autorisés.

OAuth2 est utilisé en conjonction avec l'API OAuth2 de Google pour assurer une connexion sécurisée aux comptes d'utilisateurs NEOFACTO. Cette stratégie permet une authentification centralisée tout en bénéficiant de la robustesse de l'infrastructure Google. Une attention particulière a été

portée à l'utilisation de Google Workspaces, anciennement G Suite, qui permet à tous les employés d'avoir une adresse e-mail avec le domaine "@neofacto.com". Cette uniformité simplifie l'authentification en restreignant l'accès aux comptes autorisés.

Pour que l'application puisse interagir avec les comptes NEOFACTO sur Google, des credentials spécifiques ont été créés et configurés via la Google Cloud Console. Ces credentials sont utilisés pour obtenir des tokens d'accès, notamment les tokens JWT, qui sont ensuite inclus dans les requêtes envoyées aux API Google. Cela garantit que seules les applications authentifiées et autorisées peuvent accéder aux données et aux services de Google.

La gestion des droits de l'application sur la Google Cloud Console est un élément crucial de cette intégration. Elle permet de contrôler les autorisations accordées à l'application et les actions qu'elle est autorisée à entreprendre. L'application a été configurée de manière à restreindre l'accès uniquement au domaine "@neofacto.com", garantissant que seuls les utilisateurs appartenant à ce domaine peuvent se connecter.

En somme, l'authentification dans l'application repose sur l'utilisation judicieuse d'OAuth2 et de JWT en collaboration avec l'API OAuth2 de Google. Cette approche garantit une connexion sécurisée, tout en prenant en compte les besoins spécifiques de NEOFACTO en termes de gestion des comptes et des droits d'accès.

4.5.6 Authentification sur l'application

L'authentification constitue un élément critique dans tout système informatique, et l'application en question ne fait pas exception. Pour garantir un accès sécurisé et bien géré aux utilisateurs, un protocole d'authentification robuste a été mis en œuvre, basé sur OAuth 2.0 avec l'utilisation de JSON Web Tokens (JWT).

OAuth 2.0 est un protocole d'autorisation standard qui permet aux applications tierces d'accéder aux ressources d'un utilisateur sans divulguer les informations d'identification. Dans le contexte de cette application, Google OAuth 2.0 a été utilisé pour gérer l'authentification. Cela signifie que seules les personnes disposant d'un compte Google lié à un domaine "@neofacto.com" ont la possibilité de se connecter.

L'authentification se déroule selon le flux d'autorisation implicite illustré dans la figure 4.16. L'utilisateur est redirigé vers la page de connexion Google où il doit fournir ses informations d'identification. Une fois authentifié, Google génère un JWT qui est ensuite renvoyé à l'application. Ce JWT contient des informations sur l'utilisateur ainsi que des autorisations spécifiques. L'application utilise ce JWT pour identifier l'utilisateur et autoriser les actions en fonction des autorisations accordées.

L'utilisation d'OAuth 2.0 exige une conformité stricte aux meilleures pratiques afin d'assurer un niveau de sécurité adéquat. Les éléments clés de cette conformité incluent :

Utilisation d'HTTPS : L'utilisation d'HTTPS est nécessaire pour toutes les opérations d'échange de jetons et d'appels d'API. Cela garantit que les jetons et les données sensibles sont chiffrés pendant la transmission, réduisant ainsi le risque d'écoute indue et d'attaques de type « homme du milieu ».

Durée limitée des jetons d'accès : Les jetons d'accès doivent avoir une durée de vie limitée. Cela réduit la fenêtre d'opportunité pour les attaquants d'utiliser des jetons volés. Google émet généralement des jetons d'accès avec une courte durée d'expiration.

Utilisation de rafraîchissement des jetons : Les jetons de rafraîchissement sont utilisés pour obtenir de nouveaux jetons d'accès sans nécessiter l'interaction de l'utilisateur. Ils sont à longue durée de vie et doivent être stockés en toute sécurité sur le serveur. Ils ne doivent jamais être exposés côté client.

Stockage sécurisé des jetons : Les jetons doivent être stockés côté client de manière sécurisée, soit dans des cookies sécurisés et en lecture seule, soit dans un stockage local avec des mécanismes de sécurité appropriés. Il est crucial d'éviter l'exposition des jetons dans les URL, les journaux ou les endroits accessibles au public.

Gestion de l'expiration des jetons : Gérez élégamment l'expiration des jetons en utilisant les jetons de rafraîchissement. Mettez en place une logique pour obtenir automatiquement de nouveaux jetons lorsqu'ils expirent.

Transmission sécurisée des jetons : Lors de l'utilisation de jetons dans les appels d'API, transmettez-les de manière sécurisée dans l'en-tête d'autorisation en utilisant le schéma de jeton Bearer.

Cette approche d'authentification solide garantit que seuls les utilisateurs autorisés peuvent accéder aux ressources de l'application, tout en fournissant une couche supplémentaire de protection contre les attaques potentielles. Le diagramme de flux d'authentification OAuth 2.0 provenant de la documentation et la séparation des front et back-ends sont illustrés respectivement dans les figures 4.16 et 4.17.

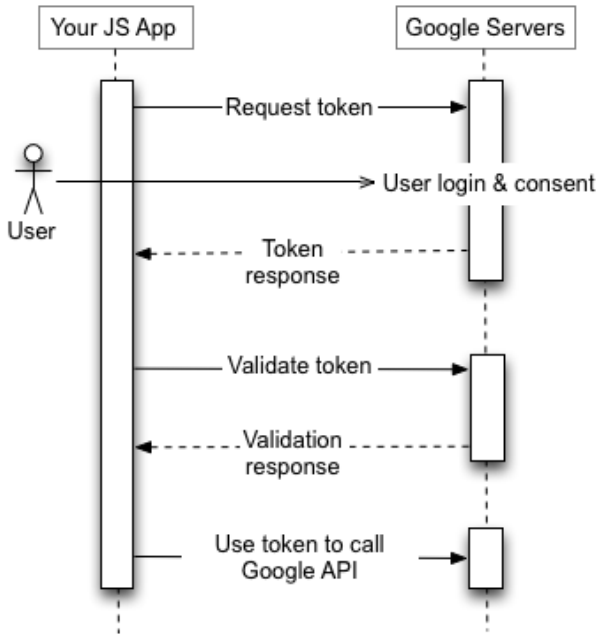

Figure 4.16 – Flux d'authentification avec le service OAuth2 de Google

Ce flux est particulièrement adapté aux applications monopages et mobiles, car il évite l'échange d'un secret client avec le navigateur de l'utilisateur. Dans ce flux, l'application front-end initie le processus d'authentification en redirigeant l'utilisateur vers la page de connexion Google. Une fois que l'utilisateur fournit ses informations d'identification, Google génère un JSON Web Token (JWT) contenant des informations d'identification et d'autres autorisations spécifiques. Ce JWT est ensuite renvoyé à l'application front-end. Le JWT peut être utilisé par l'application front-end pour identifier l'utilisateur et lui accorder des autorisations spécifiques.

Cependant, pour des raisons de sécurité et de séparation des responsabilités, certaines actions nécessitent des appels aux API depuis le back-end. Par exemple, récupérer des données sensibles ou effectuer des opérations nécessitant des privilèges plus élevés. L'application front-end doit donc récupérer un code d'autorisation après l'authentification, qui sera ensuite transmis au back-end. Le back-end échangera ce code contre un jeton d'accès en utilisant son propre secret d'application pour vérification. Ce jeton d'accès pourra ensuite être utilisé par le back-end pour accéder aux ressources de l'utilisateur, conformément aux autorisations accordées lors de l'authentification. Cette séparation des responsabilités entre le front-end et le back-end permet de garantir une sécurité accrue tout en facilitant l'expérience utilisateur. On retrouve le schéma disponnible sur la documentation Google de leur sservice OAuth 2, séparé entre front-end et back-end pour clarifier le flux sur la figure 4.17 suivante.

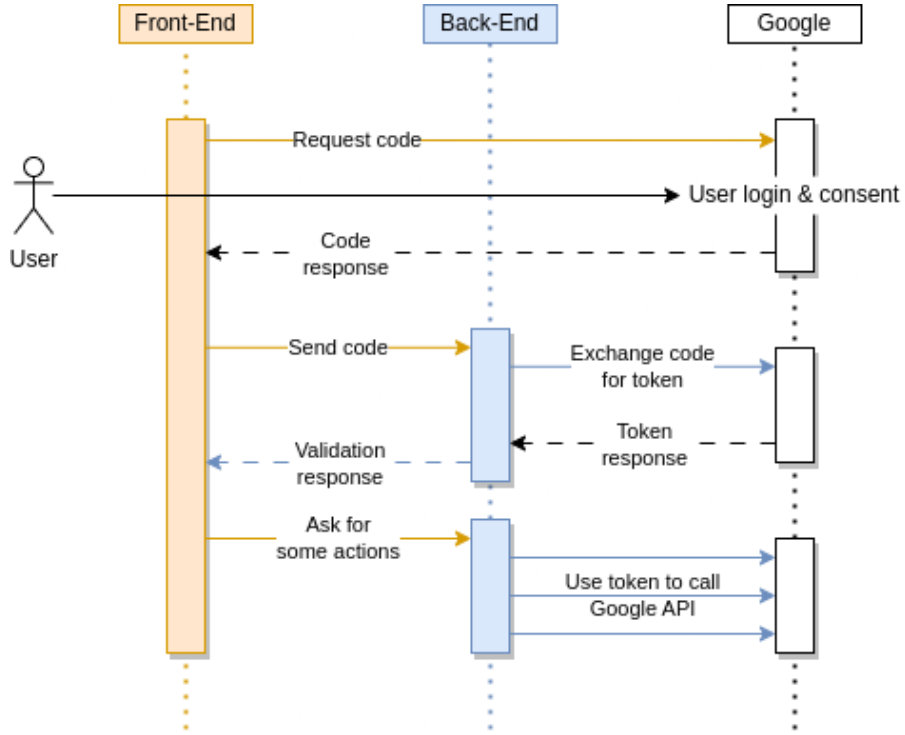

FIGURE 4.17 – Flux d'authentification OAuth2 avec spéaration du Front et Back

4.5.7 Intégrité et Confidentialité des données

La sécurité des données est d'une importance capitale dans une application. En particulier, la protection de l'intégrité et de la confidentialité des données stockées en base de données est essentielle pour garantir la confiance des utilisateurs (dans notre cas, un accès restreint à des fichiers dans le Drive pourraient être récupérés par les attaquants). Pour cela, plusieurs mesures de sécurité doivent être mises en place.

Tout d'abord, lors du stockage de données sensibles comme les jetons de rafraîchissement dans une base de données, il est crucial de veiller à ce que les données soient correctement chiffrées. En cas de compromission de la base de données, le chiffrement ajoute une couche de sécurité supplémentaire pour protéger ces informations sensibles.

Le contrôle d'accès à la base de données doit également être rigoureusement géré. L'accès à la base de données doit être limité uniquement au personnel ou aux services autorisés. Cela contribue à prévenir les accès non autorisés aux jetons stockés.

Il est recommandé de stocker les données sensibles, telles que les clés de chiffrement ou les identifiants de base de données, en tant que variables d'environnement plutôt que de les intégrer directement dans le code. Cette pratique évite toute exposition accidentelle si le code est partagé ou versionné. La base de données est chiffré et les identifiants sont sécurisés à l'aide du service BOLT, qui permet la configuration rapide de chiffrement de données sensibles.

Le principe du moindre privilège doit être appliqué pour les utilisateurs ou les comptes de service qui accèdent aux jetons en base de données. Accordez uniquement les privilèges nécessaires, par exemple, en octroyant uniquement les permissions de lecture et d'écriture aux tables pertinentes.

L'audit régulier est crucial pour maintenir un niveau de sécurité élevé. Il est recommandé de vérifier régulièrement qui a accès aux jetons stockés et quand ces accès ont eu lieu. Ces vérifications régulières permettent d'identifier rapidement toute tentative d'accès non autorisé.

La rotation des jetons peut être une stratégie judicieuse. Par exemple, il est possible de périodiquement renouveler les jetons de rafraîchissement et de mettre à jour leur version en base de données. Cela réduit la fenêtre d'opportunité pour un attaquant en cas de compromission d'un jeton.

Enfin, la durée de validité des jetons doit être prise en compte. Les jetons de rafraîchissement peuvent avoir une durée de vie prolongée, mais les risques associés doivent être évalués. En cas de vol d'un jeton de rafraîchissement, il peut être utilisé pour accéder potentiellement indéfiniment aux ressources. Des mécanismes de révocation ou d'expiration des jetons doivent être mis en place pour prévenir toute utilisation abusive. Google met à disposition une méthode universelle pour révoquer ses jetons, à travers les paramètres de sécurité du compte Google.

4.6 Déployement et Intégration continu

4.6.1 Ansible

L'intégration continue et le déploiement automatisé sont des pratiques essentielles pour garantir une livraison rapide et stable des nouvelles versions d'une application. Dans ce contexte, Ansible se révèle être un outil puissant pour automatiser et orchestrer les tâches liées au déploiement, à la configuration et à la gestion de l'infrastructure.

Ansible est une plateforme d'automatisation open source qui permet de déployer, configurer et gérer des applications et des systèmes à grande échelle. Une des principales forces d'Ansible réside dans son approche basée sur la déclaration d'état souhaité. Au lieu de spécifier comment effectuer une tâche, vous indiquez simplement l'état que vous souhaitez atteindre, et Ansible se charge de mettre en œuvre les actions nécessaires pour y parvenir.

L'un des avantages majeurs d'Ansible est sa capacité à réduire la complexité du déploiement en automatisant les tâches répétitives et en utilisant des templates pour créer des fichiers de configuration dynamiques. Par exemple, lors du déploiement d'applications avec Docker Compose, Ansible peut utiliser des modèles Jinja pour générer les fichiers docker-compose.yml en remplaçant les valeurs spécifiques en fonction de l'environnement cible.

Dans le contexte de ce projet, Ansible a grandement simplifié l'intégration du reverse-proxy nginx. Plutôt que de configurer manuellement les fichiers de configuration nginx pour chaque déploiement, nous avons pu réutiliser des templates provenant d'autres projets de NEOFACTO. Ces templates ont été adaptés à nos besoins spécifiques en utilisant des variables dynamiques, ce qui a

permis de déployer le reverse-proxy de manière cohérente et sans erreurs.

En somme, Ansible a considérablement amélioré l'efficacité du processus de déploiement en automatisant les tâches complexes et en réduisant les erreurs humaines. L'utilisation de templates pour la génération de fichiers de configuration a favorisé la cohérence et la reproductibilité des déploiements.

4.6.2 Pipeline GitLab CI/CD

La mise en place d'une intégration continue et d'un déploiement continu (CI/CD) est essentielle pour garantir la qualité du code et permettre des déploiements fréquents et automatisés. Pour ce faire, une pipeline GitLab CI/CD a été configurée, automatisant les étapes clés du processus de développement.

La configuration de la pipeline est définie dans le fichier .gitlab-ci.yml. Ce fichier spécifie les étapes de la pipeline, les images Docker utilisées pour l'exécution des étapes, ainsi que les règles de déclenchement pour chaque étape en fonction de certaines conditions, telles que la branche de développement. On retrouve ci-dessous un exemple de Pipeline pour le font-end de notre application.

```
image: docker:latest
services:
  - docker:dind

stages:
  - build
  - test
  - package
  - deploy

app-build:
  stage: build
  image: node:18
  script:
    - yarn
    - yarn build
  artifacts:
    paths:
      - dist

unit-tests:
  stage: test
  image: node:18
  before_script:
    - apt-get update && curl https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb --output google-chrome.deb
    - apt install -y ./google-chrome*.deb;
    - export CHROME_BIN=/usr/bin/google-chrome
  script:
    - yarn
    - yarn test:ci

docker-build:
  rules:
    - if: $CI_COMMIT_BRANCH == 'dev'
  stage: package
  script:
    - docker login -u ${CI_REGISTRY_USER} -p ${CI_REGISTRY_PASSWORD} ${CI_REGISTRY}
    - docker build -t ${TAG_LATEST} -f Dockerfile .
    - docker push ${TAG_LATEST}
```

Listing 4.3 – Configuration de la pipeline GitLab CI/CD

La pipeline est divisée en plusieurs étapes, chacune ayant un objectif spécifique pour assurer la qualité et la stabilité du code. Les principales étapes sont les suivantes :

Build (Construction de l'application) : Cette étape utilise une image Node.js pour installer les dépendances à l'aide de Yarn et générer les fichiers de construction. Assurer que le code compile correctement est essentiel avant de procéder aux tests.

Test (Tests unitaires et d'intégration) : Dans cette étape, les tests unitaires et d'intégration sont exécutés à l'aide d'une image Node.js. Avant l'exécution des tests, Google Chrome est installé pour les besoins de certains tests. Les tests garantissent que les fonctionnalités existantes fonctionnent comme prévu et détectent les éventuelles régressions.

Package (Création de l'image Docker) : Cette étape est conditionnée à la branche de développement. Elle construit l'image Docker à partir du fichier Dockerfile du projet. L'image Docker est ensuite taguée avec le nom et la version de la branche pour suivre les versions. L'image est ensuite poussée vers le registre Docker pour le déploiement ultérieur.

Chacune de ces étapes est exécutée automatiquement à chaque fois que des modifications sont poussées sur le référentiel GitLab. Si une étape échoue, le pipeline est interrompu et l'équipe de développement est notifiée. Cette approche garantit une détection rapide des problèmes, ce qui permet de maintenir un haut niveau de qualité et de stabilité. On retrouve sur la figure 4.18 suivante un schéma qui représente l'enchaînement de ces étapes.

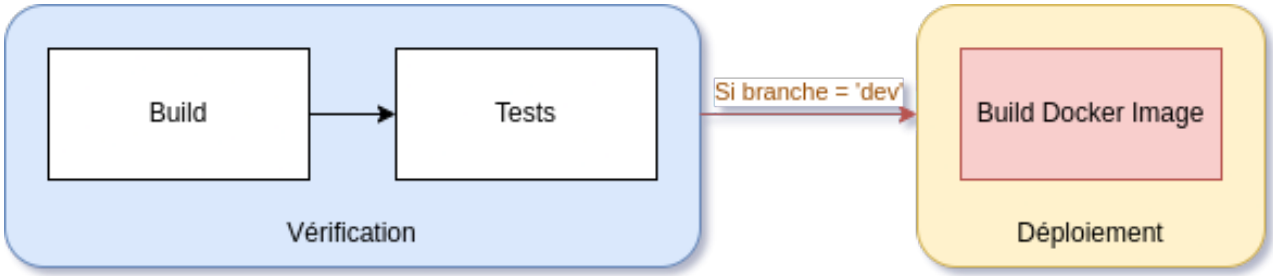

FIGURE 4.18 – Schéma qui représente l'enchaînement des étapes de la Pipeline GitLab

En somme, la pipeline GitLab CI/CD joue un rôle crucial dans le processus de développement en garantissant que le code est continuellement testé, construit et déployé de manière cohérente et automatisée. Cela facilite la gestion des versions, réduit les erreurs humaines et accélère le processus de développement et de déploiement.

On retrouve une section qui explicite le flux des images docker pour le déploiement, avec une évolution du schéma représentant les tâches de la Pipeline, dans les annexes.

4.7 Tests et validation

4.7.1 Tests unitaires

Dans le but de garantir la qualité, la fiabilité et le bon fonctionnement de chaque fonctionnalité développée, des tests unitaires ont été mis en place. Les tests unitaires sont une partie essentielle de notre processus de développement, permettant de détecter et de corriger rapidement d'éventuelles erreurs, de vérifier les comportements attendus et d'assurer la robustesse de notre application, notamment lors des phases d'évolution de fonctionnalités déjà présentes.

Pour chaque nouvelle fonctionnalité ou module développé, des tests unitaires ont été écrits pour couvrir différents scénarios d'utilisation, y compris les cas de réussite, les erreurs et les cas limites. Notre approche s'est concentrée sur deux aspects cruciaux :

- **Scénarios de base :** Dans un premier temps, nous avons testé les scénarios les plus courants pour chaque fonctionnalité afin de nous assurer que les fonctionnalités principales fonctionnent correctement. Cela implique de vérifier si l'application renvoie les résultats attendus pour les entrées typiques.
- **Cas limites et bords :** En plus des cas de base, nous avons accordé une attention particulière aux cas limites et aux scénarios extrêmes. Tester les valeurs aux limites et les situations exceptionnelles permet d'identifier des problèmes potentiels et d'améliorer la robustesse de l'application. Nous avons vérifié comment l'application réagit face à des valeurs inattendues, à des données manquantes ou à des situations inhabituelles.

Pour les tests côté front-end développé avec Angular, nous avons utilisé le framework de test Jasmine en conjonction avec Jest, qui est devenu le moteur de test par défaut pour les applications Angular. Ces outils nous ont permis d'effectuer des tests unitaires pour chaque composant et service que nous avons développés.

Concernant le côté back-end, nous avons utilisé principalement le framework Jest pour écrire des tests unitaires pour nos API. Les tests ont vérifié la logique métier, les contrôleurs, les modèles de données et l'interaction avec les bases de données.

La figure suivante présente un exemple de code de test unitaire pour le service de communication avec le back-end du côté du front-end, illustrant la manière dont nous avons appliqué cette approche de test.

```
it('should return a Promise of all needs when calling getAllNeeds and server is up', waitForAsync(()
    => {
    const needsList: Need[] = [
      {
        id: 1,
        title: 'Client 1',
        companyName: 'Project X',
        dueDate: new Date(1111, 11, 11),
        state: { id: 0, value: 'test State' },
      },
    ];

    const snackBarOpenSpy = spyOn(snackBar, 'open');

    service
      .getAllNeeds()
      .then((needs) => {
        expect(needs).toEqual(needsList);
        expect(snackBarOpenSpy).not.toHaveBeenCalled();
      })
      .catch((error) => {
        expect(error).toBeFalsy();
      });

    const req = httpTestingController.expectOne(API_URL + '/dashboard/needs');
    expect(req.request.method).toEqual('GET');
    req.flush({
      data: needsList,
    });
  }));

  it('should show error message in snackbar when calling getAllNeeds and server is not available',
    waitForAsync(() => {
    const snackBarOpenSpy = spyOn(snackBar, 'open');

    service
      .getAllNeeds()
      .then((needs) => {
```

```
        expect(needs).toBeFalsy();
      })
      .catch((error) => {
        expect(error).toBeTruthy();
        expect(snackBarOpenSpy).toHaveBeenCalledOnceWith(
          'Unable to retrieve clients needs, try again later !',
          'Close',
          { duration: 5000, panelClass: 'error-snackbar' }
        );
      });

    const req = httpTestingController.expectOne(API_URL + '/dashboard/needs');
    expect(req.request.method).toEqual('GET');
    req.flush('simulate server down', {
      status: 404,
      statusText: 'Server down',
    });
  }));
```

Listing 4.4 – Exemple de code de test unitaire pour un service du front-end

Dans le cadre de cette fonctionnalité (récupération de données liées aux besoins) les cas limites se résument à une réponse inattendue du serveur ou son indisponnibilité. Cet exemple a été volontairement choisi pour sa simplicité. Cependant, il met en avant le fait que même les fonctionnalité les plus basiques nécessitent reflexion.

D'autres fonctionnalités plus complexes, comme la modification des paramètres d'un document ont nécessite plus de tests afin d'assurer qu'aucune des entrées possibles ne soit source potentielle d'erreurs.

Chaque test a été conçu pour valider un aspect spécifique de la fonctionnalité et a été exécuté de manière automatique à chaque nouvelle modification du code. Cette approche a permis d'identifier rapidement les problèmes, de garantir la stabilité de l'application et de maintenir une qualité élevée tout au long du développement.

4.7.2 Tests d'intégration

En complément des tests unitaires, nous avons réalisé des tests d'intégration pour s'assurer du bon fonctionnement de l'application dans son ensemble. Ces tests ont pour objectif de vérifier que les différentes parties de l'application interagissent correctement les unes avec les autres et que les différentes fonctionnalités collaboratives fonctionnent harmonieusement.

Tout comme pour les tests unitaires, les tests d'intégration se sont concentrés sur les scénarios de base ainsi que sur les cas limites et extrêmes. L'objectif était de détecter d'éventuels problèmes qui pourraient survenir lorsque les différentes parties de l'application sont combinées.

Les tests d'intégration ont été conçus en respectant les principes de la méthodologie Agile et en se basant sur les exigences fonctionnelles de chaque fonctionnalité. Les étapes typiques des tests d'intégration incluent :

- **Test non authentifié :** Nous avons vérifié comment l'application réagit lorsqu'un utilisateur non authentifié tente d'accéder à certaines fonctionnalités ou routes sécurisées. Cela a permis de s'assurer que les mécanismes d'authentification fonctionnent correctement.
- **Tests avec mauvais paramètres :** Nous avons effectué des tests en fournissant intentionnellement des paramètres incorrects ou manquants pour évaluer comment l'application gère ces cas. Cela nous a aidé à identifier les points de vulnérabilité potentiels.
- **Tests avec données valides :** Nous avons testé les fonctionnalités avec des données valides

pour vérifier que l'application produit les résultats attendus. Nous avons inclus différents scénarios pour couvrir différentes combinaisons de données d'entrée.

L'exemple de code suivant illustre un test d'intégration pour une route du côté du back-end :

```
describe('PUT /dashboard/needs/:needId/language', () => {
  const needId = 7357;
  const language = 'New Language Test';
  test('should return 401 status when cookie is invalid', async () => {
    const response = await request(app)
      .put('/dashboard/needs/' + needId + '/language')
      .send({ value: language });
    expect(response.statusCode).toBe(401);
    expect(response.body).toEqual({
      message: 'You are no longer connected',
    });
  });

  describe('with authenticated user', () => {
    let firstCookieValue: string;
    let secondCookieValue: string;

    beforeAll(async () => {
      const cookies = await authWithTestUser();
      firstCookieValue = cookies[0].split(';')[0];
      secondCookieValue = cookies[1].split(';')[0];
    });
    afterAll(async () => {
      await request(app)
        .get('/auth/signout')
        .set('Cookie', firstCookieValue + '; ' + secondCookieValue);
    });

    test('should return 200 status and the new language when user is authenticated', async () => {
      const response = await request(app)
        .put('/dashboard/needs/' + needId + '/language')
        .send({ value: language })
        .set('Cookie', firstCookieValue + '; ' + secondCookieValue);

      expect(response.status).toBe(200);
      expect(typeof response.body.data === 'object').toBe(true);
      expect(response.body.data.value === language).toBe(true);
    });

    test('should return 404 status when user is authenticated but language is undefined in request
    body and error was thrown', async () => {
      const response = await request(app)
        .put('/dashboard/needs/' + needId + '/language')
        .set('Cookie', firstCookieValue + '; ' + secondCookieValue);

      expect(response.status).toBe(404);
      expect(response.body).toEqual({ message: 'Data not found' });
    });
  });
});
```

Listing 4.5 – Exemple de code de test d'intégration pour une route du back-end

Chaque test d'intégration a été conçu pour tester la cohérence des interactions entre les différents composants de l'application et a été exécuté automatiquement dans le processus de déploiement continu. Ces tests ont contribué à garantir un niveau élevé de qualité et de fiabilité de l'application dans son ensemble.

À présent, on peut exécuter aisément les tests aussi bien pour le front-end que pour le back-end en utilisant la commande yarn test, laquelle est configurée dans les fichiers package.json de chaque répertoire de projet. Dans le contexte du front-end, cette commande enclenchera l'exécution de *ngtest*, qui initie les tests grâce à Angular CLI. Pour ce qui est du back-end, l'exécution de yarn test déclenchera la commande *jest* – –*coverage*.

L'invocation de *ngtest* dans le contexte du front-end déclenchera l'exécution des tests unitaires pour le code Angular. Angular CLI orchestre le processus en lançant les tests définis dans les fichiers de spécification, vérifiant ainsi le bon fonctionnement de chaque composant, service et module. Cela permet de s'assurer que chaque partie de l'application front-end fonctionne conformément aux attentes.

D'autre part, l'utilisation de la commande *jest – –coverage* dans le cadre des tests du back-end a un double rôle. Jest, en tant que framework de test, gère l'exécution de vos tests unitaires et d'intégration. En incluant l'option –coverage, Jest calcule également la couverture de code, c'est-à-dire la proportion du code source qui est couverte par les tests. Cela permet de visualiser les parties du code qui sont testées de manière exhaustive et celles qui pourraient nécessiter une attention supplémentaire en matière de tests.

4.7.3 Validation par les utilisateurs

La phase de validation par les utilisateurs a joué un rôle essentiel dans l'assurance de la qualité et de l'efficacité de l'application. Dans cette étape, l'application a été présentée aux utilisateurs finaux afin de recueillir leurs commentaires et leurs retours d'expérience. Cette rétroaction précieuse a permis d'identifier les points forts de l'application, ainsi que les domaines qui nécessitaient des améliorations.

Pour faciliter cette validation, des réunions hebdomadaires ont été organisées. Au cours de ces réunions, l'équipe de développement a présenté les nouvelles fonctionnalités et les mises à jour de l'application aux utilisateurs finaux. Cette présentation devant l'équipe a offert l'occasion de valider les comportements des fonctionnalités directement sur l'interface utilisateur.

Ces sessions de validation ont permis de mettre en évidence des problèmes qui n'avaient pas été anticipés lors de la phase de développement. Par exemple, grâce aux retours des utilisateurs, il a été découvert qu'il y avait une limitation de taille de fichiers dans le proxy NGINX, fixée à 100 Mo. Cela a posé des problèmes pour le chargement d'images associées aux références de projets. En réponse à ce retour, l'équipe de développement a réévalué cette limitation et l'a finalement ajustée à 1 Go, ce qui a résolu ce problème et amélioré l'expérience utilisateur.

La collaboration étroite entre l'équipe de développement et les utilisateurs finaux a donc permis d'identifier des aspects cruciaux pour l'optimisation et l'ajustement de l'application en fonction des besoins réels des utilisateurs. Cette interaction continue a renforcé la qualité et la pertinence de l'application, garantissant ainsi sa conformité aux exigences et son utilité dans un contexte opérationnel.

4.8 Gestion de projet

4.8.1 Méthodologie de gestion de projet

La gestion de projet a joué un rôle essentiel dans le développement fructueux de l'application "document-builder". Pour ce faire, une méthodologie de gestion de projet précise a été adoptée, favorisant une approche itérative et collaborative pour répondre aux besoins du projet. Tout au long du processus de développement, deux périodes distinctes ont été observées, chacune avec sa propre orientation et dynamique.

La première période a été caractérisée par une approche exploratoire et de recherche. L'objectif initial était de concevoir et d'implémenter l'application de manière progressive, permettant ainsi l'émergence et la clarification des idées. Durant cette phase, j'ai bénéficié d'une grande autonomie, accompagnée par des micro-cours fournis par mon encadrant pour approfondir certains concepts pertinents. Cela m'a permis d'orienter mes recherches de manière efficace et intuitive pour la conception initiale de l'application.

La deuxième période a marqué la première version concrète du prototype, englobant le flux complet de l'application, de l'affichage des besoins à la génération des réponses aux appels d'offres. Pendant cette étape, le projet a évolué vers une approche plus structurée et collaborative, reflétant des principes similaires à ceux de la méthodologie Scrum. J'ai été rejoint par un autre stagiaire sur le projet, ce qui a permis un partage des responsabilités et une collaboration efficace.

Le processus de développement durant cette phase a suivi un modèle de revue par les pairs, où nous avons mutuellement révisé le code de chacun. Ces revues étaient renforcées par des séances quotidiennes de revue du code menées par un membre sénior de l'équipe. En outre, des réunions hebdomadaires ont été organisées pour consolider ces pratiques dans un cadre de type sprint. Lors de ces réunions, nous avons effectué des démonstrations de nos progrès réalisés pendant le sprint précédent, défini les tâches pour le sprint suivant et procédé à une répartition efficace des tâches entre nous deux.

Cette approche a permis de fusionner les avantages d'une gestion de projet rigoureuse avec l'aspect créatif et exploratoire nécessaire dans les premières phases de développement. En conséquence, le projet a pu suivre une trajectoire claire, passer par des phases de développement intense tout en maintenant une communication transparente et une collaboration étroite avec les parties prenantes, contribuant ainsi au succès global de l'application "document-builder".

4.8.2 Outils de gestion de projet

Pour garantir une planification efficace, un suivi précis et une coordination sans faille des tâches tout au long du projet, nous avons opté pour l'utilisation d'outils de gestion de projet. Ces outils ont joué un rôle essentiel dans l'organisation et la réalisation de notre projet en suivant les meilleures pratiques de gestion.

Jira

Jira, un outil de gestion de projet développé par Atlassian, a été au cœur de notre méthodologie de travail. Il nous a permis de planifier les tâches, de suivre les avancements et de coordonner les activités au sein de l'équipe. La fonction de timeline de Jira nous a aidés à visualiser la répartition temporelle des tâches, tandis que les user stories ont été utilisées pour décomposer les fonctionnalités en éléments plus petits et gérables. Le tableau d'avancement nous a fourni une vue d'ensemble de l'état de chaque tâche et de son attribution. On retrouve dans les annexes, le diagramme de GANTT généré à partir de la timeline de Jira.

GitLab

GitLab a été notre plateforme de choix pour la gestion de notre code source et de notre processus de développement. Les fonctionnalités de gestion des branches et des merge requests nous ont

permis de travailler efficacement en équipe. Les reviews de code et les discussions intégrées dans les merge requests ont amélioré la qualité du code et facilité la collaboration entre les membres de l'équipe, notemment pour le transfert de connaissance.

Google Chat

Google Chat a été notre principal moyen de communication en équipe. Nous avons créé un groupe de discussion pour faciliter la communication en temps réel et échanger rapidement des informations. De plus, les conversations en tête-à-tête ont été essentielles pour le travail en binôme, permettant une collaboration étroite même à distance.

Confluence

Pour la documentation de notre projet, nous avons utilisé Confluence, un outil collaboratif également développé par Atlassian. Confluence nous a permis de créer et de partager des documents, des rapports et des spécifications de manière organisée et accessible à tous les membres de l'équipe. Cela a grandement facilité la gestion des informations et des connaissances tout au long du projet, ce qui a notamment permis à un collaborateur de s'intégrer au projet pendant durant son développement.

En combinant ces outils de gestion de projet, nous avons pu maintenir une communication fluide, une coordination efficace et une documentation complète, contribuant ainsi au succès global de notre projet.

4.9 Conclusion et perspectives

4.9.1 Bilan du projet

La réalisation du projet "document-builder" a abouti à la création d'une application fonctionnelle répondant aux objectifs fixés. L'application permet la génération automatisée de documents à partir de modèles stockés sur Google Drive, simplifiant ainsi le processus de création de réponses aux appels d'offres. En récapitulant les fonctionnalités mises en œuvre, l'application offre une interface d'authentification sécurisée, la gestion des besoins clients récupérés depuis Boond Manager, la personnalisation des documents via l'éditeur intégré et la prévisualisation en temps réel de la génération.

Dans le choix de prioriser une première version fonctionnelle, l'accent a été mis sur la création d'une application opérationnelle. Bien que l'intégration d'intelligence artificielle n'ait pas été abordée dans cette version, une attention particulière a été portée à la qualité de la génération de documents. Cette étape a nécessité une compréhension approfondie de l'API Google pour manipuler avec précision les aspects stylistiques du document généré, tels que la copie de style, la mise en page et les couleurs des paragraphes.

La démonstration finale a mis en avant l'ensemble des fonctionnalités, détaillant leur utilisation efficace. Un exemple concret a été présenté en générant un document à partir d'une ancienne réponse à un appel d'offres de NEOFACTO. Cette démonstration a également permis d'illustrer les

limites actuelles de la génération en termes de duplication de contenu. En pratique, le processus de génération d'un document s'effectue en quelques secondes, avec un maximum de 20 secondes lors des tests impliquant des documents de taille conséquente (une trentaine de pages).

Cependant, l'application atteint un stade où sa véritable valeur se révélera lorsqu'elle sera utilisée en production pour répondre à un besoin réel au sein de NEOFACTO. Les résultats plus approfondis et les avantages tangibles seront observés lors de cette phase, permettant de mesurer pleinement l'impact de l'application dans le contexte opérationnel de l'entreprise. L'espérance en terme de gain de temps pour le directeur technique pour rédiger les réponses à appel d'offres est estimé à quelques heures, sur le long terme (le temps de mettre en place les différents templates pour les sections courantes).

4.9.2 Limitations et perspectives d'amélioration

Actuellement, l'application possède quelques limitations qu'il convient de relever. Malgré les efforts mis en place pour garantir une copie précise des éléments des documents Google Docs, certaines fonctionnalités de formatage et de mise en page nécessitent encore des ajustements pour assurer une cohérence totale lors de la génération. Une liste exhaustive de ces éléments a été communiquée à NEOFACTO à travers Confluence, afin que l'entreprise puisse poursuivre le développement en les prenant en compte.

De plus, l'API Google a évolué tout au long du projet, offrant de nouvelles fonctionnalités pour la manipulation de documents. Cependant, il est essentiel de suivre attentivement les futures évolutions de l'API pour garantir que l'application demeure à jour et que les opportunités d'amélioration soient exploitées.

En termes d'améliorations futures, une piste prometteuse serait l'intégration de l'intelligence artificielle à l'aide de l'API ChatGPT. Cette approche pourrait permettre une génération encore plus automatisée et intelligente des documents, en proposant des suggestions de contenu, des corrections grammaticales et même en apprenant des habitudes d'écriture de l'utilisateur pour optimiser la qualité des documents générés.

D'une manière plus générale, l'architecture modulaire de l'application offre des possibilités d'extension considérables. En envisageant une intégration avec d'autres systèmes de gestion d'entreprise et des solutions de stockage de documents, l'application pourrait être proposée comme une solution globale aux entreprises. Cette perspective ouvre la voie à de nouvelles opportunités commerciales, bien que cela nécessite un investissement significatif en termes de développement et de personnalisation pour chaque entreprise cliente.

En somme, bien que l'application "document-builder" ait déjà atteint un niveau fonctionnel et utile, son potentiel d'amélioration et d'expansion offre des perspectives intéressantes pour l'avenir. Les futures versions pourront capitaliser sur ces améliorations pour offrir une expérience utilisateur encore plus fluide et efficace, tout en élargissant les possibilités d'utilisation pour répondre aux besoins variés des entreprises.

5 Conclusion

En conclusion de ce mémoire, nous avons abordé l'importance du stage de dernière année dans le cursus des étudiants en ingénierie en France. Ce stage représente une étape cruciale dans la transition entre l'apprentissage académique et la pratique professionnelle. Il offre aux étudiants l'occasion de mettre en pratique leurs compétences techniques, de découvrir le fonctionnement réel d'une entreprise et de se familiariser avec les enjeux spécifiques de leur domaine de spécialisation. Le stage de dernière année est un véritable tremplin vers l'insertion professionnelle, permettant aux étudiants de valoriser leurs compétences, de développer leur réseau de contacts et de saisir de futures opportunités d'emploi.

Dans le cadre de ce stage, j'ai travaillé sur le développement d'une solution visant à faciliter la création de réponses à des appels d'offres et de contrats de maintenance pour une entreprise de conseil en informatique, NEOFACTO. Cette application a été conçue pour répondre aux besoins spécifiques du CTO, qui doit fréquemment répondre à des demandes de propositions en organisant et en remplissant les différentes sections du document, en suivant les directives de la charte graphique de l'entreprise, tout en s'assurant de proposer un service de qualité à un prix cohérent.

Le développement de cette application a été un défi passionnant qui m'a permis de mettre en pratique mes compétences en matière de conception, de développement et de sécurisation d'applications Web. Tout au long de ce projet, j'ai utilisé diverses méthodologies et technologies pour atteindre les objectifs et répondre aux exigences de sécurité imposées.

Les principales fonctionnalités de l'application ont été implémentées avec succès, permettant de générer rapidement et efficacement des modèles de documents en réponse aux appels d'offres et aux contrats de maintenance. L'application a été conçue pour répondre aux besoins spécifiques de NEOFACTO, mais elle offre également un potentiel d'extension pour s'intégrer à d'autres entreprises du secteur.

Pour les perspectives futures, deux orientations sont envisageables. D'une part, l'application pourrait être retravaillée et adaptée pour répondre aux besoins spécifiques d'autres entreprises de conseil en informatique, leur offrant ainsi un outil puissant pour gérer et optimiser leurs réponses aux appels d'offres. D'autre part, l'application pourrait rester en interne, réservée à NEOFACTO, afin de renforcer sa compétitivité sur le marché en lui permettant d'obtenir plus facilement des contrats grâce à des réponses plus rapides et mieux structurées.

Enfin, il est important de souligner que ce projet a été une expérience enrichissante qui m'a permis de développer mes compétences professionnelles et techniques. Je suis absolument convaincu que les connaissances acquises tout au long de ce projet me seront utiles dans ma future carrière d'ingénieur en informatique.

En conclusion, ce mémoire d'ingénieur reflète mon engagement envers l'excellence et mon désir de contribuer de manière significative au monde professionnel de l'informatique. J'espère que ce

travail ouvrira la voie à de nouvelles opportunités pour NEOFACTO et qu'il inspirera d'autres projets novateurs dans le domaine du développement d'applications Web, notemment avec le support d'intelligences artificielles.

Bibliographie / Webographie

[1] Google. Documentation officielle d'angular matérial. https://material.angular.io/. 49

[2] Google inc. Documentation officielle de google api. https://developers.google.com/docs/api/reference/rest. 49

[3] Laurent Kratz. Entreprise esn neofacto luxembourg. https://www.neofacto.com. 3

[4] LangChain. Site officiel de la documentation de la librairie langchain en javascript. https://js.langchain.com/docs/get_started/introduction. 44

[5] Lixtec. Présentation complète avec exemples de l'architecture en onion. https://lixtec.fr/architecture-oignon-onion-architecture/. 22

[6] Loopio. Site officiel de présentation du service loopio. https://loopio.com/. 5

[7] Boond Manager. Documentation officielle de l'api de boond manager. https://doc.boondmanager.com/api-externe/. 48

[8] RFP plus. Site officiel de présentation du service rfp plus. https://rfp.plus/. 5

[9] Wikipedia. List of http status codes. https://en.wikipedia.org/wiki/List_of_HTTP_status_codes. 26

Liste des illustrations

3.1 Visualisation des différents niveaux de priorité assignés à chaque critères d'évaluation 6

3.2 Légende des couleurs associées à l'évaluation d'une technologie sur un critère . . . 7

3.3 Table de justification des choix de la technologie du front-end 9

3.4 Table de justification des choix de la technologie du back-end 13

4.1 Diagramme de classes partiel du front-end de l'application 25

4.2 Diagramme de classes du back-end de l'application 28

4.3 Diagramme présentant les différents modèles de l'application 30

4.4 Diagramme de séquence de l'authentification sur l'application 31

4.5 Diagramme de séquence de modification et sauvegarde de données sur l'application 32

4.6 Schéma relationnel modélisant la base de données de l'application 33

4.7 Page d'authentification de l'application 34

4.8 Page d'accueil comportant la liste des besoins des clients 35

4.9 Page de visualisation des détails d'un besoins client 36

4.10 Page d'édition du document de réponse au client avec pré-visualisation 37

4.11 Page de visualisation des projets de l'entreprise en cours 38

4.12 Page de modification des références à un projet 38

4.13 Exemple de traitement des données à l'aide d'une chaîne Map/Reduce d'appel à GPT3 46

4.14 Exemple de traitement des données avec mise en évidence de l'anonymisation lors d'appel à GPT3 47

4.15 Capture d'écran de la sortie de la commande npm audit dans un example de projet Node.js 53

4.16 Flux d'authentification avec le service OAuth2 de Google 57

4.17 Flux d'authentification OAuth2 avec spéaration du Front et Back 58

4.18 Schéma qui représente l'enchaînement des étapes de la Pipeline GitLab 61

A.1 Diagramme de classes complet du front-end de l'application 79
A.2 Diagramme de séquence de modification d'un attribut d'une références de projet . . 80
A.3 Diagramme de séquence d'ajout d'une image à une référence de projet 80

C.1 Schéma qui représente l'enchaînement des étapes de la Pipeline GitLab avec redéploiement automatique 83

D.1 GANTT regroupé par fonctionnalités 84
D.2 GANTT développé avec les User Stories 85

Listings

4.1 Exemple d'utilisation d'Axios pour les requêtes API 48
4.2 Exemple de programmation défensive avec le bloc try/catch en TypeScript 52
4.3 Configuration de la pipeline GitLab CI/CD 60
4.4 Exemple de code de test unitaire pour un service du front-end 62
4.5 Exemple de code de test d'intégration pour une route du back-end 64

Annexes

A Conception

On retrouve sur la figure A.3 suivante le diagramme de classes complet du front-end de l'application. C'est une version du diagramme disponible dans ce mémoire complétée des méthodes de chaque classes, et une organisation plus claire, mais plus volumineuse.

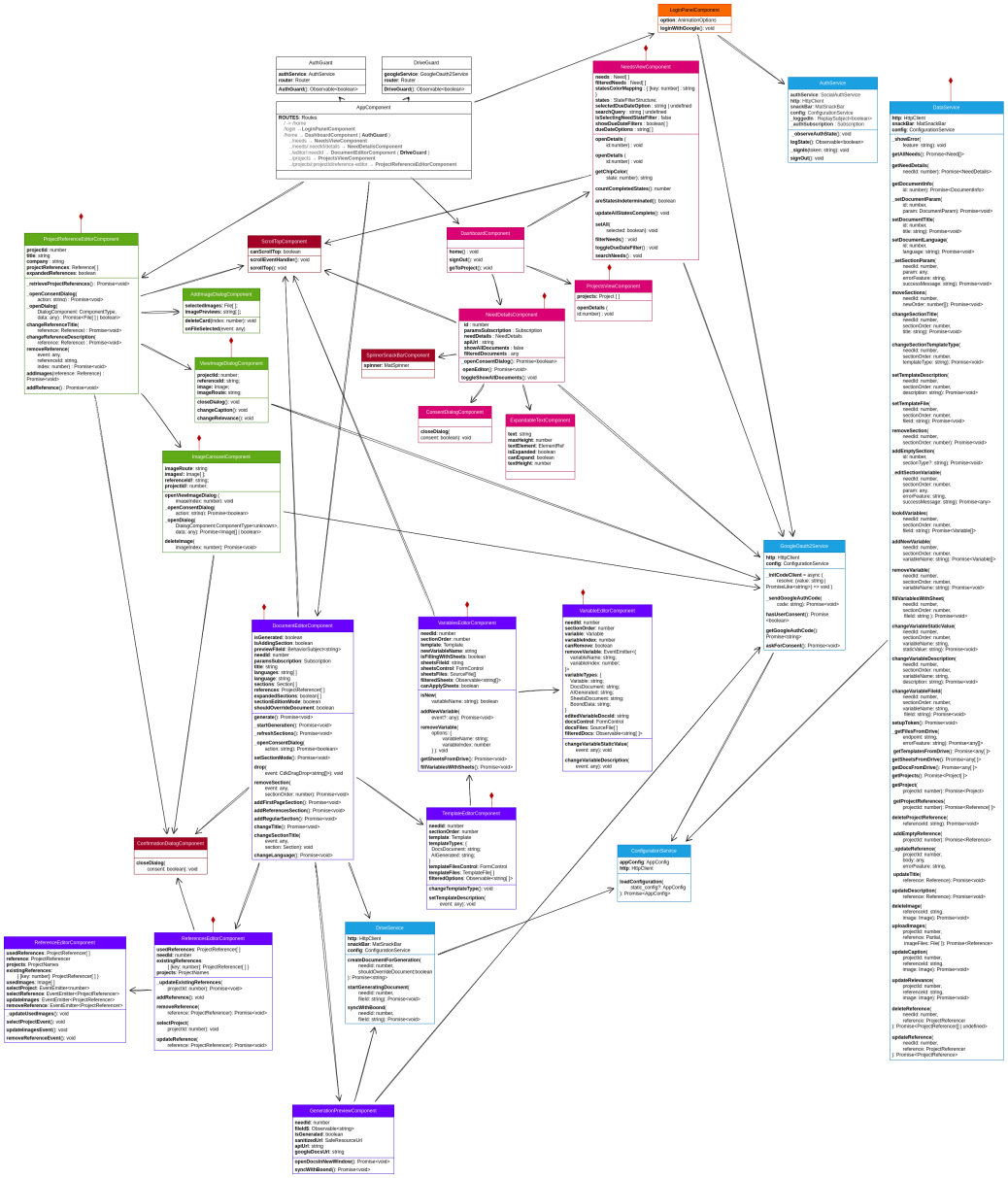

FIGURE A.1 – Diagramme de classes complet du front-end de l'application

On retrouve sur les figures ?? et ?? respectivement les diagrammes de séquences de change-

ment de propriété et de l'ajout d'une image dans les références aux projets.

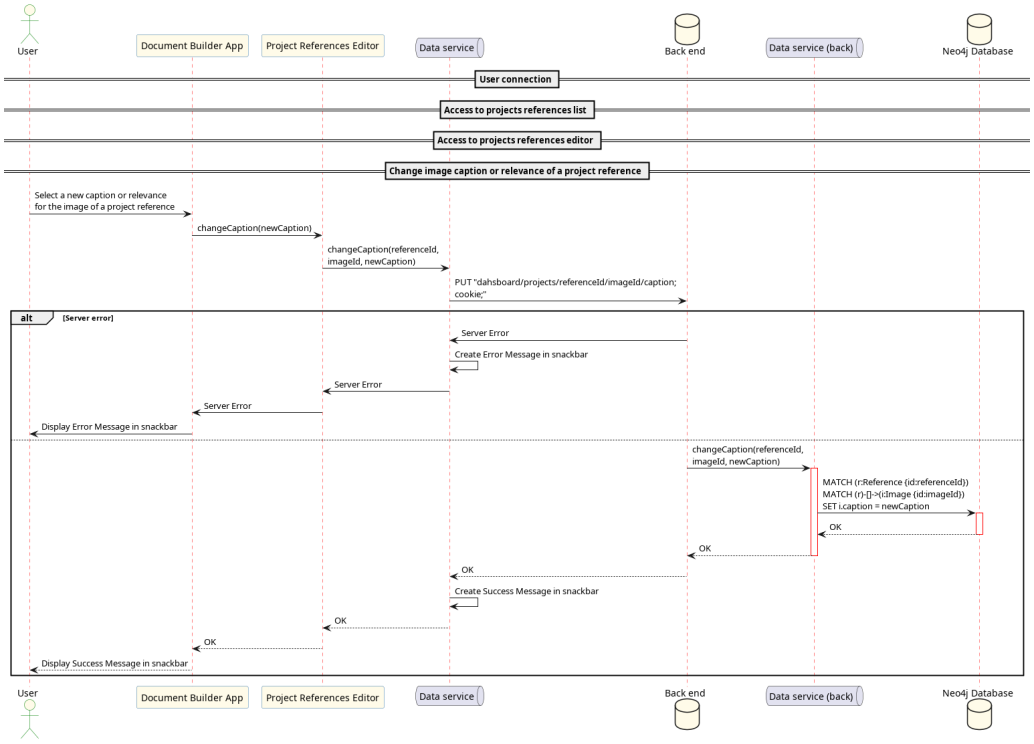

FIGURE A.2 – Diagramme de séquence de modification d'un attribut d'une références de projet

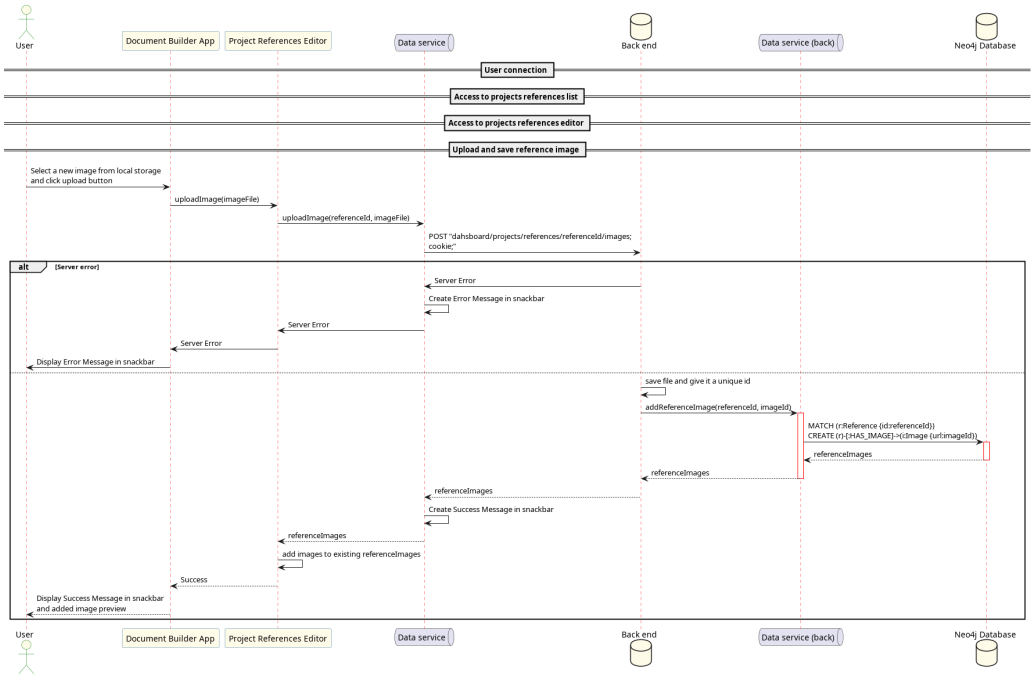

FIGURE A.3 – Diagramme de séquence d'ajout d'une image à une référence de projet

B Choix des technologies et outils

B.1 Environnement de Développement Optimal avec Visual Studio Code (VSCode)

Le choix de Visual Studio Code (VSCode) comme environnement de développement principal pour le projet "document-builder" a été délibéré et judicieux. VSCode offre une combinaison impressionnante de puissance, de flexibilité et de support étendu pour une variété de langages et d'extensions.

Puissance et Polyvalence : VSCode se distingue par sa capacité à gérer des projets de toute envergure, qu'il s'agisse d'applications Web complexes ou de scripts plus légers. Son interface intuitive et sa configuration personnalisable font de lui un outil adapté à divers besoins de développement.

Prise en Charge Étendue : VSCode prend en charge une gamme étendue de langages de programmation, ce qui en fait un choix optimal pour un projet qui implique TypeScript et Node.js. Cette prise en charge native facilite l'écriture, le débogage et la maintenance du code, augmentant ainsi l'efficacité du développement.

Automatisation Simple : L'un des avantages majeurs de VSCode réside dans sa capacité à automatiser les tâches courantes. Les tâches d'exécution de l'application et de tests peuvent être configurées avec des scripts personnalisés, ce qui simplifie le processus de développement. Les développeurs peuvent ainsi se concentrer sur la création plutôt que sur les opérations manuelles.

Extensions Adéquates : Les extensions spécifiques à TypeScript et à Node.js offrent un soutien supplémentaire pour le développement et le débogage dans ces langages. Cela permet une programmation plus fluide, avec des fonctionnalités avancées telles que l'analyse statique (mise en évidence des erreur dans l'éditeur de texte), la navigation aisée dans le code et des fonctionnalités de débogage étendues.

En somme, Visual Studio Code se révèle comme un choix judicieux pour le développement du projet "document-builder". Son potentiel à la fois puissant et flexible, associé à sa prise en charge complète des langages TypeScript et Node.js ainsi qu'à la facilité d'automatisation, crée un environnement idéal pour les développeurs du projet, favorisant la productivité et la qualité du code.

C Déploiement

C.1 Docker

L'intégration de Docker dans le processus de déploiement est un élément clé pour assurer la cohérence entre les environnements de développement et de production. Grâce à la pipeline GitLab CI/CD précédemment évoquée, l'image Docker de l'application est générée automatiquement à chaque fois que des modifications sont apportées au code. Cette image contient tous les composants nécessaires à l'exécution de l'application, y compris les dépendances, le serveur web et les configurations spécifiques.

Une fois l'image Docker créée dans la pipeline, le processus de déploiement sur le serveur de production entre en jeu. Pour gérer efficacement cette étape, Ansible est utilisé. Ansible est un outil d'automatisation qui permet de déployer, de configurer et de gérer des applications sur des serveurs distants.

L'une des étapes importantes dans le processus de déploiement est la configuration des variables d'environnement. Ces variables permettent de paramétrer l'application en fonction de l'environnement dans lequel elle est déployée. Ansible facilite cette tâche en permettant la modification des variables d'environnement lors du déploiement. Cela permet de personnaliser le comportement de l'application sans nécessiter de modifications directes dans le code.

Une fois les variables d'environnement configurées, Docker Compose est utilisé pour déployer l'application sur le serveur de production. Docker Compose est un outil qui permet de définir et de gérer des applications multi-conteneurs dans Docker. En se basant sur le fichier docker-compose.yml, Docker Compose récupère l'image Docker de l'application depuis le registre Docker et la déploie sur le serveur de production.

Ce processus de déploiement automatisé garantit que l'application est déployée de manière cohérente et reproductible sur le serveur de production. Il réduit les risques d'erreurs humaines et assure que l'application fonctionne de la même manière dans tous les environnements, ce qui facilite la détection des problèmes et la résolution des bugs. En outre, cette approche permet de mettre à jour l'application en toute confiance en utilisant la dernière version de l'image Docker générée par la pipeline.

En optant pour une approche automatisée, l'intégration de GitLab avec une clé privée SSH offre une solution efficiente pour orchestrer le processus de déploiement directement au sein de la Pipeline. Cette méthode implique la configuration préalable d'une clé privée SSH dans les variables système de GitLab, qui peut ensuite être assignée à chaque projet ou groupe de projets spécifique. Cette clé privée SSH autorise l'automatisation des tâches liées au déploiement sur le serveur de production.

Cette automatisation se concrétise par l'ajout de la clé publique associée à la clé privée utilisée dans GitLab au sein du serveur de production. Cette démarche assure l'accès en toute sécurité via SSH à un utilisateur disposant des privilèges requis pour gérer Docker sur le serveur. Grâce à cette configuration, la Pipeline acquiert la capacité d'interagir directement avec le serveur, lui permettant ainsi d'effectuer automatiquement la mise à jour des images Docker après leur construction. On retrouve figure C.1 suivante le schéma de l'enchaînement des étapes de la pipeline avec l'ajout de la tâche de redéploiement automatique.

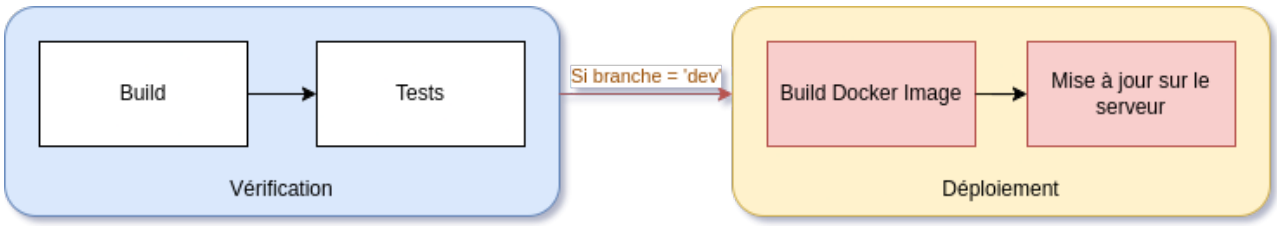

FIGURE C.1 – Schéma qui représente l'enchaînement des étapes de la Pipeline GitLab avec redéploiement automatique

L'adoption de cette approche présente des avantages significatifs. En effet, elle réduit les étapes manuelles et les risques d'erreur humaine tout en garantissant un déploiement cohérent et fiable de l'application. L'automatisation permet de simplifier le processus de mise à jour des images Docker sur le serveur de production, facilitant ainsi la maintenance et les évolutions de l'application.

D GANTT

On retrouve sur les figures D.1 et D.2 suivantes, respectivement le GANTT regroupé par Fonctionnalité, montrant la timeline prévisionnelle et les différents sprints et le GANTT étendu avec les différentes User Stories.

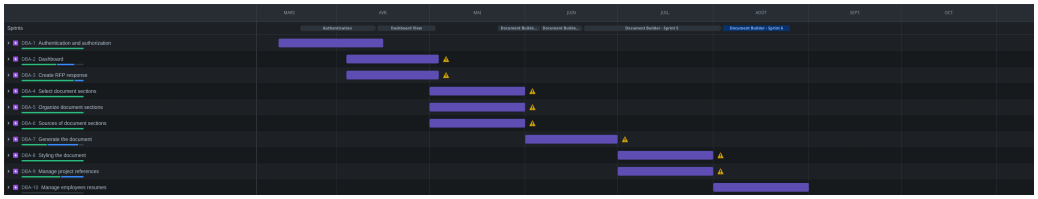

Figure D.1 – GANTT regroupé par fonctionnalités

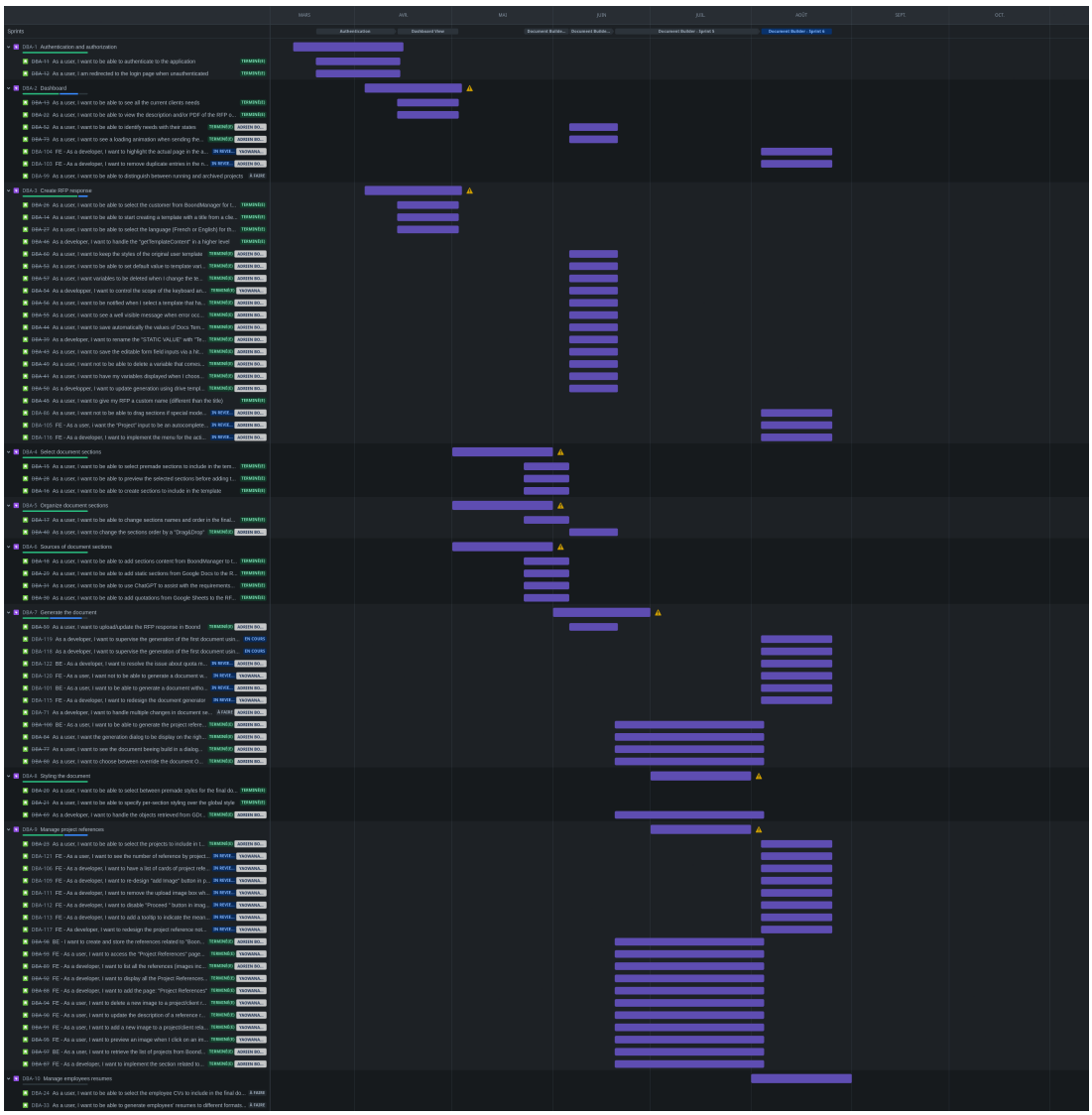

FIGURE D.2 – GANTT développé avec les User Stories

Résumé

Vers l'efficacité et l'innovation : Développement d'une application pour optimiser la rédaction de réponses aux appels d'offres dans le secteur des services informatiques

Le stage de dernière année en ingénierie marque une étape cruciale pour les étudiants, offrant l'opportunité de monter ses compétences et sa capacité à en acquérir. Dans ce contexte, ce mémoire présente le développement d'une application novatrice répondant aux défis du secteur des services informatiques.

Répondre aux appels d'offres représente une tâche complexe pour les entreprises de conseil en informatique, nécessitant des ressources considérables. Afin d'optimiser cette démarche, une application web puissante a été conçue pour simplifier la création de réponses pertinentes, tant pour les appels d'offres que les contrats de maintenance.

Tout au long du projet, des compétences en conception, développement et sécurité des applications web ont été mises à l'épreuve. L'application intègre une interface conviviale et efficace, permettant au responsable technique de générer rapidement des documents en utilisant l'intelligence artificielle, respectant les normes graphiques de l'entreprise.

L'application est intégrée dans le parc informatique de l'entreprise, et interagit étroitement avec les autres services du parc. Cette intégration en profondeur permet de limiter la duplication des données et de services qui sont déjà utilisés par l'entreprise.

Au-delà des résultats obtenus, ce mémoire met en exergue les méthodologies de gestion de projet et les technologies utilisées pour surmonter les défis rencontrés. En considérant les perspectives d'avenir, cette application offre un potentiel d'extension pour d'autres entreprises du secteur, mais également l'opportunité de renforcer la compétitivité de NEOFACTO sur le marché.

Ce travail représente ainsi une expérience riche et prometteuse pour un futur ingénieur en informatique, démontrant sa capacité à relever les défis du monde professionnel. Le mémoire se conclut avec enthousiasme sur l'ouverture de nouvelles opportunités pour NEOFACTO et sur l'inspiration que cette application peut susciter dans le domaine du développement d'applications web.

Mots-clés : Typescript, Full-Stack, Appel d'offre, IA

Abstract

Towards efficiency and innovation : Development of an application to optimize the drafting of responses to calls for tenders in the IT services sector

The final year internship in engineering marks a crucial step for students, offering the opportunity to develop their skills and their ability to acquire them. In this context, this thesis presents the development of an innovative application responding to the challenges of the IT services sector.

Responding to tenders is a complex task for IT consulting companies, requiring considerable resources. To optimize this process, a powerful web application has been designed to simplify the creation of relevant responses, both for calls for tenders and maintenance contracts.

Throughout the project, skills in the design, development and security of web applications

were put to the test. The application incorporates a user-friendly and efficient interface, allowing the technical manager to quickly generate document using artificial intelligence, respecting the company's graphic standards.

The application is integrated into the company's computer park, and interacts closely with the other park services. This in-depth integration limits the duplication of data and services that are already used by the company.

Beyond the results obtained, this thesis highlights the project management methodologies and the technologies used to overcome the challenges encountered. Considering the future prospects, this application offers a potential for extension for other companies in the sector, but also the opportunity to strengthen the competitiveness of NEOFACTO on the market.

This work thus represents a rich and promising experience for future computer engineers, demonstrating their ability to meet the challenges of the professional world. The thesis concludes with enthusiasm on the opening of new opportunities for NEOFACTO and on the inspiration that this application can arouse in the field of web application development.

Keywords : Typescript, Full-Stack, RFP, AI