



Dossier Professionnel : Expert / Experte en informatique & Système d'Information (EISI)

Entreprise : Quartz Insight, École : EPSI campus d'Auxerre
Responsable de formation : Alexandre Lemaire

Brunet Geoffrey

Année scolaire 2022/2023

Table des matières

1 Remerciements	5
2 Introduction	6
3 Environnement professionnel	7
3.1 Organigramme de Quartz-Insight	7
3.2 Présentation de l'étudiant	8
3.3 Activités de la structure	9
3.4 Présentation des activités confiées	9
3.5 Présentation du Système d'Information	10
4 Valorisation des compétences	11
4.1 Présentation globale du projet	11
4.2 Activité 1 - cartographie complète du système d'information existant	11
4.2.1 Compétence et son fondement	11
4.2.2 Présentation et réalisation de l'activité	12
4.2.3 Contexte de réalisation	16
4.2.4 Conclusion sur l'activité	17
4.3 Activité 2 - cartographie des informations sensibles, évaluation des risques et définition d'une politique de sécurité	17
4.3.1 Compétence et son fondement	17
4.3.2 Présentation et réalisation de l'activité	18
4.3.3 Contexte de réalisation	23
4.3.4 Conclusion sur l'activité	23
4.4 Activité 3 - analyse des données de références pour la création d'un référentiel de données	23
4.4.1 Compétence et son fondement	24
4.4.2 Présentation et réalisation de l'activité	24
4.4.3 Contexte de réalisation	27
4.4.4 Conclusion sur l'activité	27
4.5 Activité 4 - analyse des besoins métiers pour une solution applicative sur mesure	27
4.5.1 Compétence et son fondement	27
4.5.2 Présentation et réalisation de l'activité	28

4.5.3	Contexte de réalisation	30
4.5.4	Conclusion sur l'activité	30
4.6	Activité 5 - conception d'une architecture applicative évolutive et tolérante aux pannes	30
4.6.1	Compétence et son fondement	30
4.6.2	Présentation et réalisation de l'activité	31
4.6.3	Contexte de réalisation	33
4.6.4	Conclusion sur l'activité	33
4.7	Activité 6 - développement d'une application pour répondre aux besoins utilisateurs/directions métiers	33
4.7.1	Compétence et réalisation et son fondement	33
4.7.2	Présentation et réalisation de l'activité	34
4.7.3	Contexte de réalisation	38
4.7.4	Conclusion sur l'activité	38
4.8	Activité 7 - analyse de la qualité de la solution applicative et mise en place d'un plan de correction / d'amélioration	38
4.8.1	Compétence et son fondement	38
4.8.2	Présentation et réalisation de l'activité	39
4.8.3	Contexte de réalisation	41
4.8.4	Conclusion sur l'activité	41
4.9	Activité 8 - mise en place d'une solution d'intégration continue	41
4.9.1	Compétence et son fondement	41
4.9.2	Présentation et réalisation de l'activité	42
4.9.3	Contexte de réalisation	46
4.9.4	Conclusion sur l'activité	46
4.10	Activité 9 - validation fonctionnelle et rédaction de la documentation pour les utilisateurs	47
4.10.1	Compétence et son fondement	47
4.10.2	Présentation et réalisation de l'activité	47
4.10.3	Contexte de réalisation	49
4.10.4	Conclusion sur l'activité	49
4.11	Activité 10 - formation des utilisateurs, mise en place d'une enquête de satisfaction ultérieure	49
4.11.1	Compétence et son fondement	49
4.11.2	Présentation et réalisation de l'activité	50
4.11.3	Contexte de réalisation	51
4.11.4	Conclusion sur l'activité	51
5	Informations complémentaires	52
5.1	Gestion de projet	52
5.2	Gestion des coûts	52
6	Conclusion	53
6.1	Résumé	53
6.2	Ouverture sur l'avenir	53

6.2.1	L'entreprise et ses perspectives	53
6.2.2	Le service et ses évolutions	54
6.2.3	Les apports professionnels et personnels	54
6.2.4	Avenir professionnel	55

1 Remerciements

Je tiens à exprimer mes plus sincères remerciements à l'entreprise Quartz Insight, notamment à M. Christian MONTILLAUD, gérant et fondateur de l'entreprise pour m'avoir accueilli au sein de sa société ainsi qu'à M. Gaël ROUSTAN, Chief Technical Officer et tuteur pour m'avoir encadré tout au long de ma formation. Je tiens à remercier également tous les collaborateurs du pôle "recherche et développement" qui comme mon tuteur ont pris le temps de m'apporter leur aide sur les projets qui me sont confiés, ce qui m'a permis de monter en compétences tout au long de l'année.

Mes remerciements vont aussi vers les membres du personnel de l'EPSI Auxerre pour leur confiance et les formations prodiguées. L'enseignement de qualité du titre professionnel « Expert en Informatique et système d'information » a parfaitement été en adéquation avec mes objectifs tout au long de l'année. Plus spécifiquement, je tiens à remercier monsieur Sébastien GUILBERT (responsable Business Unit Enseignement supérieur au pôle formation 58-89) et Alexandre LEMAIRE (Responsable pédagogique) pour la confiance pour mon intégration au sein de l'EPSI, et ayant apporté leurs aides autant sur des plans professionnels que personnels.

J'aimerais aussi exprimer ma gratitude envers tous les formateurs étant intervenus tout au long de cette année de bachelor, qui ont pris le temps de nous préparer et de fournir des formations de qualité, et écouter nos difficultés lorsque nous en avions.

Pour finir, je tiens à présenter ma gratitude et mon respect à toutes les personnes qui m'ont accompagné tout au long de mon parcours pour le temps qu'ils m'ont consacré dans le but de faire de moi un meilleure développeur.

2 Introduction

Quartz-Insight est une société créée en 2015 par monsieur Christian Montillaud suite à une volonté de fonder une entreprise basée sur l'expertise, la performance et le management des processus d'*EPM*¹ et de *BI*² des entreprises. L'origine du nom de l'entreprise proviens de la passion de monsieur Montillaud pour les minéraux, et le Quartz étant choisis car étant un minéral transparent, valeur qu'il souhaite apporter à son entreprise. Le terme «Insight» quand à lui reprend la volonté d'avoir une expertise profonde, une véritable introspection sur les domaines de compétences de l'entreprise.

Les trois valeurs de l'entreprise sont l'accompagnement (de l'écoute à la proposition d'une solution adaptée), l'esprit d'équipe (allant de paire avec la solidarité) et l'honnêteté (avec nos différents clients et partenaires).

Ses activités sont principalement basé sur les solutions logicielles EPM de chez Oracle, allant d'outils comme Essbase, une base de données multidimensionnelle à Planning, ne solution de planification et de budgétisation. Quart-Insight a deux pôles d'activités différents. Le premier est un pôle « consulting », les collaborateurs membres apportant leurs expertises aux clients pour la gestion et l'amélioration de leurs différents services EPM. Le deuxième pôle est un pôle Recherche et Développement, avec une équipe d'ingénierie travaillant sur une solution logicielle sous forme d'Add-In pour la manipulation de logiciels de BI dans des tableurs ou logiciels de présentation comme Microsoft Excel, Google Sheets et Google Slides.

3 Environnement professionnel

3.1 Organigramme de Quartz-Insight

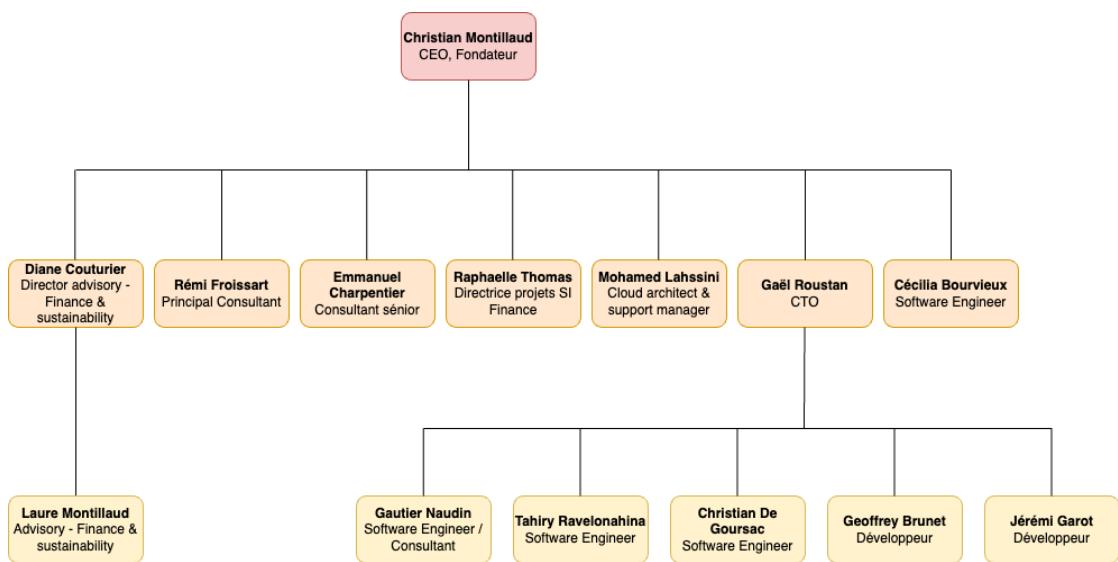


FIGURE 3.1 – Organigramme de Quartz-Insight

La société est gérée par monsieur Christian Montillaud. Monsieur Gaël Roustan, est directeur technique et supervise le pôle Recherche et Développement. Madame Cécilia Bourvieux gère un projet de test de charge pour les solutions de BI, du développement logiciel à la mise en place et l'exécution chez les clients. Les membres du pôle R&D (Recherche et développement) ont en charge le développement des Add-Ins pour les logiciels de chez Microsoft et Google, ainsi que la création, la mise en production et la gestion des microservices sous-jacents, avec l'aide de notre CTO. Le reste des collaborateurs sont des consultants apportant leurs expertises dans leurs domaines respectifs chez nos différents clients.

Je suis donc actuellement développeur au sein du pôle R&D, et travaille sur différents projets et microservices, dans le but d'améliorer ceux existants, résoudre des bugs et en créer de nouveaux lorsque cela est nécessaire.

3.2 Présentation de l'étudiant

BRUNET GEOFFREY

10 rue du Grand Caire, Auxerre, 89000
06.23.35.49.26 | geoffrey.brunet@icloud.com
Linkedin: <https://www.linkedin.com/in/geoffrey-brunet-558315ba/>
Github: <https://github.com/GeoffreyBrunet>

- Bac professionnel laboratoire contrôle qualité, en 2014, mention "assez bien".
- Bac professionnel laboratoire contrôle qualité, en 2014, mention "assez bien".
- Brevet d'études professionnelles travaux de laboratoire, en 2013.

OBJECTIF

En dernière année de bac +5, je souhaite acquérir le poste de software engineer, et me spécialiser dans l'amélioration des performances des applications.

EXPÉRIENCE

- Alternance depuis septembre 2020 chez Quartz Insight en tant que développeur
 - Création d'un outil de gestion de licences pour le logiciel de l'entreprise.
 - Création d'un outil d'acquisition et de gestion de journaux d'événements.
 - Création d'une API REST en java pour gérer et exécuter des tâches planifiées.
- Alternance de 2 ans dans chez Louis 21 d'aout 2016 à Juillet 2018.
- Entreprise Netquarks à Paris en mai/juin 2015 et décembre 2015/février 2016, en stage puis en emploi saisonnier.

DIPLÔMES

- Bachelor Concepteur / Développeur d'applications
- BTS SIO option Infrastructure Systèmes et Réseaux, en 2018 (major de promotion).
- Bac professionnel systèmes électroniques numériques, en 2016, mention "bien".

CONNAISSANCES TECHNIQUES

- Langages de programmation: Typescript, Rust, Java, Lua
- Éditeurs de texte & IDEs: Neovim (configuration maison en lua)
- Frameworks web: React, Sveltekit, Angular
- Librairie CSS: Tailwind CSS
- Bases de données: PostgreSQL, SQLite, Redis
- Management et gestion des versions: Git, Docker, Gitlab
- Web APIs: Axum, Spring
- Application multiplateforme: React Native, Tauri
- Outils de configurations et build: NPM, Cargo, Maven
- Autre: LaTeX, Markdown

CENTRES D'INTERÊT

- Informatique DIY et impression 3D
- Voyages
- Running / Trail (Tout terrains)
- Photographie argentique et numérique / cinéma

3.3 Activités de la structure

Quartz-Insight est une entreprise spécialisée dans la Business Intelligence (BI) et l’Enterprise Performance Management (EPM). Elle propose une gamme complète d’activités et de services destinés à aider les entreprises à optimiser leur gestion et à prendre des décisions éclairées. Voici un aperçu de nos principales activités :

1. **Consulting en BI et EPM** : Quartz-Insight offre des services de consulting personnalisés pour accompagner les entreprises dans leur utilisation efficace des outils de BI et d’EPM tels que les serveurs Essbase, les cubes OLAP et Hyperion Planning. Les consultants de Quartz-Insight travaillent en étroite collaboration avec les clients pour comprendre leurs besoins spécifiques et proposer des solutions adaptées à leurs objectifs.
2. **Réalisation de tests de charges sur des infrastructures de BI et d’EPM** : nous proposons une prestation de tests de charges pour les infrastructures BI et EPM des entreprises, avec divers scénarios possibles, pour tester les limites du matériel physique comme du logiciel. Le produit final est un fichier PDF contenant les résultats sous forme de graphes, ainsi qu’une proposition d’actions à mettre en place pour améliorer la résilience de l’infrastructure.
3. **Développement de Qibates** : développement d’un logiciel en mode SAAS (Software As A Service) pour explorer diverses sources de bases de données multidimensionnelles. Un abonnement par licence par utilisateur est proposé à la vente, pour une utilisation sur Microsoft Excel ou Google Sheets, avec des accès à des sources de données diverse grâce à des connecteurs développés par nos soins.
4. **Conseil sur la transition RSE des entreprises** : après une implementation d’actions RSE (Responsabilité sociétale des entreprises) et de formations pour notre entreprise, nous avons décidé de proposer à nos clients une formation comprenant tous les principaux axes des actions mises en place. Nos proposons aussi une animation pour la fresque du climat, pour démontrer l’implication écologique pour les personnes comme pour les entreprises.

3.4 Présentation des activités confiées

Voici une liste contenant les tâches qui m’ont été confiées durant mon alternance chez Quartz-Insight :

- Écriture de fichiers « Dockerfile » pour la conteneurisation de nos microservices
- Résolution de tickets d’incidents sur l’add-in QiBates pour Google Sheets
- Écriture d’un logiciel de récupération de journaux d’évènements, avec une page web pour les consulter

- Collaboration pour développer une API REST sur un projet pour un de nos client (BNP)
- Transformation d'un ancien service monolithique en microservices pour la gestion des licences utilisateurs

3.5 Présentation du Système d'Information

Comme vu précédemment, le pôle Recherche et Développement édite des logiciels sous forme d'Add-In pour Google Sheets, Google Slides et Microsoft Excel, pour la manipulation de bases de données multidimensionnelles et d'outils de Planning. Pour un soucis de performance, le traitement des données n'est pas fait sur le frontend de nos Add-Ins mais par des microservices hébergés sur nos machines virtuelles, dans un format de conteneur Docker.

Tous les microservices nécessitant d'être déployé sur des serveurs le seront sur des machines virtuelles hébergées chez notre partenaire OBS (pour Orange Business Services, filiale cloud du groupe Orange). Ces machines virtuelles utilisent Docker pour gérer les cycles de vie des conteneurs. Il en va de même pour les bases de données, que nous utilisons en version managée, donc dont toute la partie serveur est gérée par OBS directement.

4 Valorisation des compétences

4.1 Présentation globale du projet

En tant qu’alternant, le projet qui m’a été confié consistait à effectuer une transformation majeure d’un service existant, passant d’une architecture *monolithique*⁴ à une architecture de *microservices*⁵. Le service initial comprenait une web UI, une *API REST*³ et une base de données destinées à stocker les informations des utilisateurs, telles que leurs noms, domaines d’entreprise et types de profil. L’architecture monolithique, qui reposait sur *ASP.NET*⁸ avec SQL Server comme *SGBD*⁷, présentait des limitations en termes de flexibilité et d’évolutivité. L’objectif principal était donc de repenser cette infrastructure afin de permettre une évolutivité améliorée et des performances optimisées. Pour atteindre cet objectif, nous avons adopté une approche basée sur les microservices. Cela impliquait de diviser le service en plusieurs composants indépendants, chacun étant responsable d’une fonction spécifique. Ainsi, nous avons pu obtenir une architecture plus flexible, où chaque microservice peut être géré et déployé de manière autonome. Dans le cadre de cette transformation, nous avons choisi d’utiliser *Angular*⁹ pour la web UI, offrant ainsi une expérience utilisateur réactive et enrichissante. Pour l’API REST, nous avons opté pour *Spring Boot*¹⁰, un *framework*⁶ Java réputé pour sa simplicité et sa robustesse. De plus, nous avons migré la base de données vers *PostgreSQL*¹¹, un système de gestion de base de données open source réputé pour ses performances et sa fiabilité. Toutes ces technologies sont identiques pour le reste de nos microservices, nous permettant d’avoir une architecture uniformisée.

4.2 Activité 1 - cartographie complète du système d’information existant

4.2.1 Compétence et son fondement

Bloc de compétences : A1 – Analyse et définition de la stratégie des systèmes d’information

Compétence choisie : A1C4 – Cartographier un système d’information existant selon les 4 niveaux (métier, fonctionnel, applicatif et infrastructure) afin d’avoir une bonne connaissance de l’ensemble de ses composants

Détails : Cette compétence consiste à valider mon aptitude à réaliser une

cartographie du S.I. précisant pour chaque couche les éléments suivants :

- préciser dans le schéma les entités et les systèmes (en lien avec les Process Métier)
- préciser l'architecture applicative globale (les logiciels, les services et l'analyse des flux de données)
- présenter dans son schéma l'Infrastructure logique (VLAN, adresse.IP, filtrage et routage) infrastructure physique (équipements).

Elle consiste aussi à démontrer une utilisation appropriée d'un logiciel de cartographie :

- démarrage du logiciel adéquat
- temps d'utilisation approprié (min 10 minutes)
- le résultat de la cartographie informatique correspond aux attendus d'une cartographie d'un SI demandé par un comité de direction

4.2.2 Présentation et réalisation de l'activité

Une étape pour cartographier le système d'information existant selon les 4 niveaux (métier, fonctionnel, applicatif et infrastructure) serait la réalisation d'une analyse approfondie de l'environnement existant. Voici la suite d'étapes réalisées pour atteindre cet objectif :

Collecte des informations

Ce monolithe n'a pas de documentation proprement associée, la principale source de connaissance est son code source, présent dans son repository sur Gitlab. Pour analyser le service monolithique et étant programmé en .net, j'ai utilisé l'IDE fournis par Microsoft, Visual Studio. Les informations importantes à regarder pendant l'analyse du code sont :

1. Les endpoints de l'API REST
2. Les classes qui servent de modèles
3. Les requêtes effectuées par l'ORM vers le SGBD
4. Les autres microservices qui communiquent avec le monolithe.

La base de donnée utilisée par le monolithe a été utile pour observer les données sauvegardées dans les tables, et les jointures entre elles. Étant donné qu'il s'agit de Microsoft SQL Server comme service de gestion de base de données, j'ai utilisé SSMS (SQL Server Management Studio) pour effectuer mes recherches. Pour ce qui est des diagrammes et schémas, je les ai réalisé après analyse et sont présent dans ce dossier dans la sous-section « Analyse applicative ».

Identification des composants métier

Après consultation de l'équipe de développement, ce service monolithique n'est utilisé que par les dit-membres de cette équipe. La console n'est accessible et utilisée que par eux. L'objectif de ce service est de sauvegarder l'état des comptes et domaines clients qui utilisent la solution QiBates, ainsi que le

type de profil pour les comptes (qui défini quelles options sont accessibles ou non). Côté client, si il n'a jamais souscrit à une ou plusieurs licences QiBates, un domaine avec le nom de l'entreprise est créé. Le client envoie une liste des utilisateurs, avec entre autre leurs adresses mails professionnelles, et le type de profile demandé pour chaque compte. Toutes ces informations sont enregistrées dans un fichier CSV qui est envoyé au monolithe, que celui-ci utilisera pour créer et écrire les comptes avec les bonnes informations dans la base de données.

Analyse fonctionnelle

Pour l'analyse fonctionnelle, j'ai divisé celle-ci en 6 sous-analyses distinctes :

1. Identification des fonctionnalités, qui sont :

- Création de profils utilisateur avec des informations telles que leur nom, domaine d'entreprise et type de profil, ainsi que sa durée de validité.
- Recherche d'utilisateurs par domaine d'entreprise.
- Mise à jour des informations du profil utilisateur.
- Gestion de la durée de vie des profils avec possibilité de renouvellement ultérieur.
- Suppression d'un utilisateur, voir d'un domaine.

2. Définition des cas d'utilisation :

- Demande de création d'un nouvel utilisateur ou de plusieurs utilisateurs par un membre du pôle RD, avec validation du directeur technique ou du CEO, demande effectuée par un directeur technique ou un contrôleur de gestion, en fonction de nos clients.
- Modification d'un profil utilisateur par un membre du pôle RD, avec validation du directeur technique ou du CEO.
- Suppression d'un profil utilisateur par un membre du pôle RD, avec validation du directeur technique ou du CEO.
- Renouvellement de la durée de vie d'un profil utilisateur.

3. Modélisation des flux de travail :

Imaginons un premier cas d'utilisation qui serait "Demande de création, modification ou suppression d'un profil utilisateur" :

- Un de nos clients nous envoie une demande de création, modification ou suppression d'un profil utilisateur avec les informations nécessaires.
 - Le CEO ou le CTO valide la dite demande.
 - Les modifications sont transmises au pôle RD, et un membre de celui-ci apporte les modifications sur l'interface web de l'application.
- Dans le cadre d'un renouvellement de la durée de vie d'un profil utilisateur qui servira en second cas d'utilisation :
- Un de nos clients nous envoie une demande de création, modification ou suppression d'un profil utilisateur avec les informations nécessaires.

- Le CEO ou le CTO valide la dite demande.
- Les modifications sont transmises au pôle RD, et un membre de celui-ci apporte les modifications sur l'interface web de l'application.

4. Définition des entrées et sorties :

- Pour les demandes de création, modification ou suppression d'un profil utilisateur, les entrées comprennent les informations nécessaires pour le profil utilisateur concerné.
- Pour la demande de renouvellement de la durée de vie d'un profil utilisateur, l'entrée est l'identification du profil utilisateur concerné.
- La sortie attendue est la validation ou le refus de la demande par le directeur technique ou le CEO. Si la demande est validée, la sortie finale est la création, modification ou suppression d'un ou plusieurs comptes.

Analyse applicative

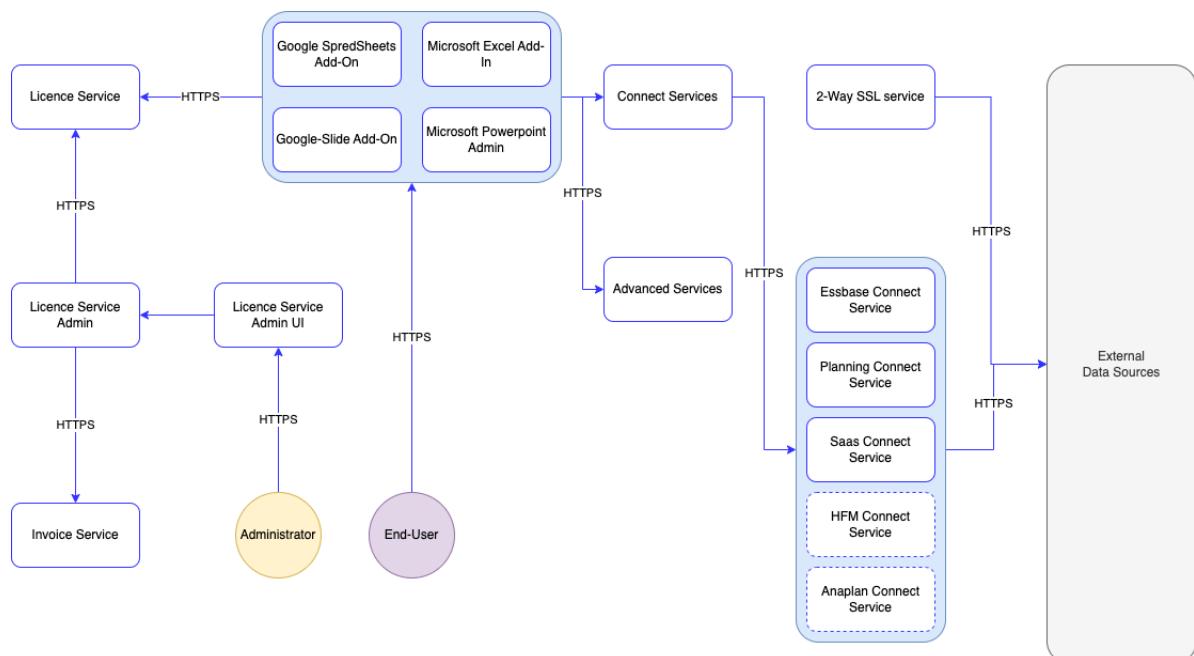


FIGURE 4.1 – Schéma de l'infrastructure actuelle

Pour plus de clareté et de compréhension des rôles du service monolithique dans le schéma, celui-ci a été fragmenté, mais le monolithe contient tous les services du schéma suivants :

- **Licence service** : service qui permet de connaître l'existence d'un compte, son état et son profile.
- **Licence service Admin** : Partie backend permettant la création, modification et suppression des comptes, mais uniquement utilisable par

- un administrateur depuis l'interface web.
- **service Admin UI** : Interface web permettant la création, modification et suppression des comptes.

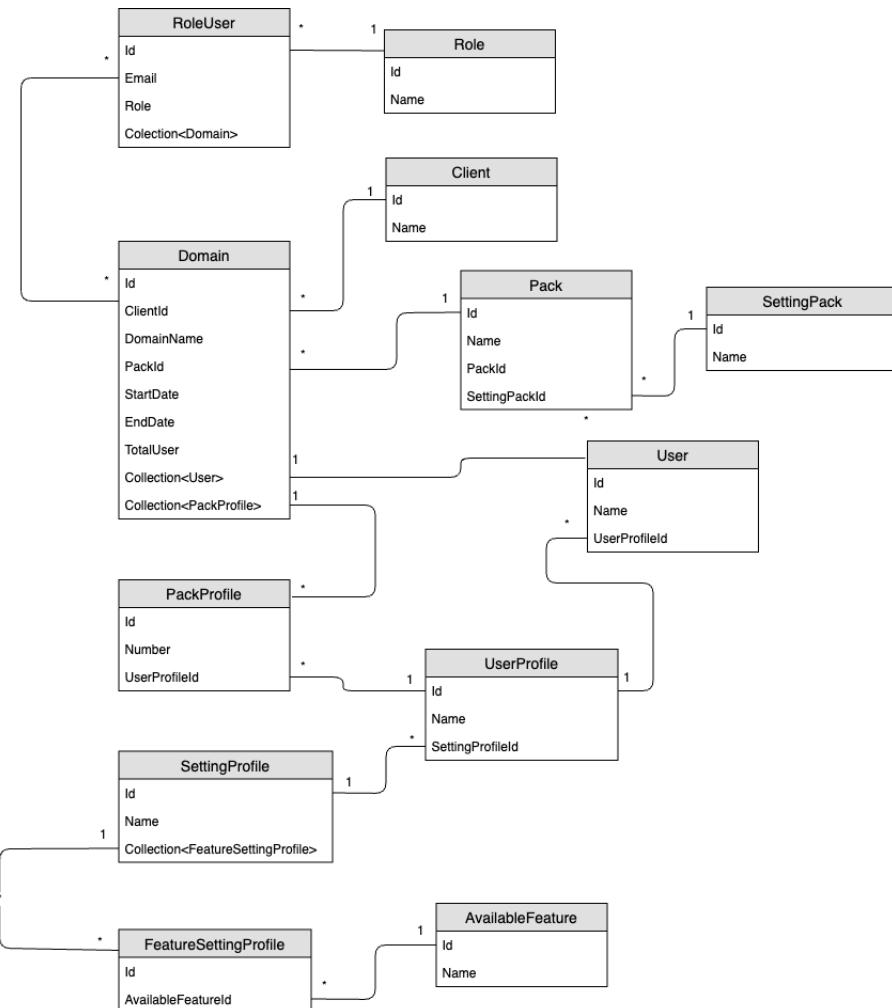


FIGURE 4.2 – Modèle de conception de données de la base de donnée du monolithique

Voici le MCD réalisé à partir des informations obtenues depuis le SGBD, et avec l'aide d'un collaborateur qui avais œuvré à son intégration. Une description de l'utilité de chaque table présente dans le schéma est effectuée ici :

- **Domain** : contient le domaine du client, ainsi que la durée de validité de ses comptes sous-jacents.
- **RoleUser** : utilisateur avec un rôle particulier pour l'administration de l'application.
- **Role** : rôle d'un utilisateur pour accéder et utiliser la console (droits de lecture ou rôle d'administrateur).

- **Client** : identifiant client, car un client peut avoir plusieurs domaines.
- **Pack** : permet de lier une liste prédéfinie de paramètres à un domaine (pour pouvoir les appliquer à tous les utilisateurs du domaine).
- **SettingPack** : liste prédéfinie de paramètres pour un utilisateur.
- **User** : utilisateur pour utiliser QiBates.
- **PackProfile** : permet de lier un domaine à un utilisateur.
- **UserProfile** : profile utilisateur permettant à un utilisateur d'avoir une liste de paramètres.
- **SettingProfile** : liste de paramètres de fonctionnalités pour un utilisateurs.
- **FeatureSettingProfile** : permet de changer les paramètres des fonctionnalités.
- **AvalibleFeature** : fonctionnalités spécifiques à QiBates (comme la coloration des cellules, le fait de se connecter à certains cubes, etc.).

Analyse de l'infrastructure

Les exigences de l'infrascture sont définis sur 3 paramètres :

- Performance : Le système doit être capable de traiter rapidement les demandes de création, modification ou suppression de profils, ainsi que les demandes de renouvellement.
- Sécurité : Les accès et les modifications aux profils utilisateurs doivent être contrôlés et sécurisés.
- Scalabilité : Le système doit être évolutif pour gérer une augmentation du nombre de demandes de gestion de profils sans compromettre les performances.

Chaque microservice est un conteneur Docker. Nos serveurs Docker sont des machines virtuelles hébergées chez Orange Business Service, avec Ubuntu comme système d'exploitation. Les caractéristiques physiques des machines virtuelles changent en fonction de la dite machine virtuelle, en fonction de l'OS (car nous avons aussi des machines virtuelles sur Windows pour les labs) et des zones de celle-ci (développement, pré-production et production).

Documentation

Toute l'analyse et la documentation produite a été mise à disposition des collaborateurs dans un dossier, sur le OneDrive (service de stockage en ligne collaboratif de chez Microsoft).

4.2.3 Contexte de réalisation

Cette activité a été réalisée en amon du projet, juste après la demande de refonte du service monolithique en microservices. Elle a donc été la première action réalisée après la dite demande.

Voici mes éléments personnels quand à la réalisation de cette activité :

J'ai utilisé mes compétences acquises en formation en architecture logicielle pour mieux analyser les différents microservices et leurs interactions.

J'ai utilisé mes compétences en modélisation de données acquises au cours de ma formation pour la réalisation du MCD précédemment présenté.

Ainsi que les éléments organisationnels et structurels :

J'ai analysé l'ancien SGBD pour comprendre comment les données étaient sauvegardées.

J'ai demandé aux autres développeurs du pôle RD quels microservices communiquent entre eux et comment pour réaliser le schéma précédent (numéro 4.1).

4.2.4 Conclusion sur l'activité

J'ai réalisé avec succès une cartographie complète du système d'information existant dans le cadre de la refonte du service monolithique en microservices. J'ai utilisé mes compétences en architecture logicielle et en modélisation de données pour analyser les microservices et leurs interactions, les flux de données et les interactions entre les services. Cette cartographie approfondie m'a permis d'identifier les fonctionnalités clés du projet, fournissant une base solide pour la conception des microservices. J'ai mis à disposition de l'équipe la documentation complète pour faciliter la collaboration. Mis à part un manque de documentation, aucun problème n'a eu lieu quand à la réalisation de cette activité, donc les documents produits ont été validé par mon directeur technique.

4.3 Activité 2 - cartographie des informations sensibles, évaluation des risques et définition d'une politique de sécurité

4.3.1 Compétence et son fondement

Bloc de compétences : A1 – Analyse et définition de la stratégie des systèmes d'information

Compétence choisie : A1C5 – Identifier les informations sensibles, les risques, les zones critiques et les chemins d'attaque possibles d'un système d'information existant à l'aide de la cartographie afin d'aider le/la RSSI à définir une politique de sécurité **Détails** : Cette compétence consiste à démontrer ma capacité à présenter une cartographie des risques du SI analysant les éléments suivants :

- la confidentialité : préciser les règles d'authentification et d'accès aux ressources informatiques du SI
- l'intégrité : préciser les règles garantissant l'exactitude, la complétude et la non altération des données du SI ;

- la disponibilité : préciser les procédures mises en place pour garantir la disponibilité des ressources informatiques du SI.
- formaliser à l'écrit une analyse du résultat de cette cartographie mettant en exergue les risques et zones critiques du SI

4.3.2 Présentation et réalisation de l'activité

Analyse de l'existant et des points chauds

Voici les informations principales relatives à la sécurité actuellement mise en place, en amont de la réalisation d'un PSSI (Politique de sécurité du système d'information) :

- la confidentialité : Les accès aux ressources se font avec des rôles définis par le directeur technique et en accord avec les membres du pôle RD. La double authentification est obligatoire, ainsi qu'un mot de passe répondant à une politique définie au préalable. Les données dans les bases de données sont chiffrées.
- l'intégrité : préciser les règles garantissant l'exactitude, la complétude et la non altération des données du SI; une sauvegarde incrémentale des bases de données a lieu chaque jour et les journaux d'événements sont aussi sauvegardés.
- la disponibilité : un démarrage d'un nouveau SGBD avec une sauvegarde de l'ancienne base de données peut être réalisé depuis OBS rapidement mais (mais aucun système de réplication comme avec PG-Pool II n'est actuellement en place).

Voici la liste des points chauds présents dans la réalisation de ce projet :

1. **La migration de la base de données** peut être un point chaud en raison du risque de perte de données ou de corruption pendant le processus de migration.
2. **La sécurité des microservices**, la communication entre les microservices, l'authentification des utilisateurs et la gestion des autorisations sont des points critiques qui doivent être soigneusement mis en place pour éviter les vulnérabilités potentielles.
3. **La gestion des accès aux API**, avec la gestion des identifiants et des jetons d'authentification doit être sécurisée pour protéger les ressources sensibles.
4. **La sécurité de l'interface web**, dont celle-ci doit être protégée contre les attaques de sécurité courantes, telles que les attaques XSS (Cross-Site Scripting) ou les attaques CSRF (Cross-Site Request Forgery).
5. **La gestion des priviléges d'accès** est cruciale pour éviter les fuites d'informations sensibles ou les accès non autorisés.
6. **La surveillance et détection des incidents** pour détecter les activités anormales ou les tentatives d'intrusion est essentielle pour identifier rapidement les incidents de sécurité et y répondre de manière appropriée.

Pour faciliter l'identification des informations sensibles, les risques, les zones critiques et les chemins d'attaque possibles, j'ai réalisé une première mouture d'un Plan de Sécurité des Systèmes d'Information (PSSI). Celui ci se base sur 3 principaux axes, les principes organisationnels, les principes de mise en œuvre et les principes techniques.

Voici les **principes organisationnels en œuvre** du projet de refonte du service :

1. Politique de sécurité :

- Élaboration d'une politique de sécurité qui définit les objectifs, les responsabilités et les exigences pour la sécurité de l'ensemble de l'organisation.

2. Organisation de la sécurité :

- Établissement d'une structure des organisations claire pour la gestion de la sécurité, avec une identification des responsabilités et des rôles spécifiques liés à la sécurité de l'information.
- Désignation d'un RSSI (Responsable de la sécurité des systèmes d'information) chargé de coordonner les activités de sécurité et de veiller à leur mise en œuvre.

3. Gestion des risques SSI :

- Mise en place un processus de gestion des risques pour identifier, évaluer et traiter les risques liés à la sécurité de l'information.
- Développement des plans d'action pour atténuer les risques identifiés.

4. Sécurité et cycle de vie :

- Intégration des principes de sécurité dès les phases de conception, de développement, de déploiement et de maintenance du système.
- Mise en place d'évaluations régulières de sécurité tout au long du cycle de vie du projet pour identifier les vulnérabilités et prendre les mesures correctives nécessaires.

5. Assurance et certification :

- Établissement des mécanismes d'assurance et de certification pour valider la conformité de votre système à des normes et réglementations en matière de sécurité.
- Recherche des certifications appropriées, telles que ISO 27001 (norme relative à la sécurité), pour attester de la maturité et de la conformité de votre système en matière de sécurité.

Voici les **principes de mise en œuvre** du projet de refonte du service :

1. Aspects humains :

- Implication les parties prenantes clés, telles que le directeur technique, le CEO, les membres du pôle RD, et confirmer leurs engagements envers les objectifs de sécurité du projet, au travers de réunions.
- Établissement clair des rôles et des responsabilités en matière de sécurité, en définissant les tâches et les autorisations appropriées pour chaque acteur du projet (en corrélation avec les différents droits d'administration du service gestionnaire de licences).

2. Planification de la continuité des activités :

- Identification des activités critiques du système et élaboration de plans de continuité des activités pour assurer leur disponibilité en cas d'incident ou de sinistre.
- Mise en place de mécanismes de sauvegarde réguliers et des procédures de récupération des données pour minimiser les temps d'arrêt et préserver l'intégrité des informations.

3. Gestion des incidents :

- Création et écriture de procédures claires pour la gestion des incidents de sécurité, y compris la notification, l'investigation, la résolution et le suivi.
- Mise en place d'une ou plusieurs personnes pour répondre rapidement aux incidents et atténuer les impacts.

4. Sensibilisation et formation :

- Mise en place de sessions de sensibilisation à la sécurité pour les employés (clients comme internes à Quartz Insight), afin de les informer des meilleures pratiques, des politiques et des procédures de sécurité. Cela peut faire référence à des points comme le SSO, la double authentification, la gestion des mots de passe (vérifier son poste lorsqu'un utilisateur utilise QiBates et s'absente de son poste).

5. Exploitation :

- Mise en place d'une solution de supervision en temps réel, avec une définition des métriques et journaux d'évènements, au préalable.
- Formation des membres du pôle RD à la lecture et l'analyse des dites métriques et des dits journaux d'évènements.
- Écriture d'une politique de mise à jour des micro-services pour réduire le temps d'arrêt à un strict minimum.

6. Aspects physiques et environnementaux :

- Mise à part les ordinateurs portables, Quartz-Insight ne possède aucune machine physique.
- Nos machines virtuelles sont toutes hébergées chez Orange Business Service (OBS), dans des régions.
- Un accès VPN est requis pour accéder aux serveurs et applications en cours de développement, en plus d'un couple identifiant et mot de passe propre à chacun.

PSSI

Voici une analyse des **principes techniques** de la PSSI :

1. Identification des composants métier :

- Le système comprendra les composants métier suivants : interface web, API REST et base de données.
- L'interface web sera développée en utilisant Angular.
- L'API REST sera développée avec Spring Boot.
- La base de données sera migrée vers PostgreSQL, un système de gestion de base de données open source réputé pour ses performances et sa fiabilité.
- Les autres microservices du projet utiliseront également ces technologies pour uniformiser l'architecture.

2. Gestion des accès et des privilèges :

- Les utilisateurs doivent s'authentifier avant d'accéder au système, en utilisant des identifiants et des mots de passe, ainsi qu'un compte Google ou Microsoft pour le SSO.
- Des politiques de contrôle des privilèges sont mises en place pour limiter l'accès aux fonctionnalités et aux données sensibles.
- Les membres du pôle RD auront des privilèges spécifiques pour créer, modifier et supprimer des profils utilisateurs, sous réserve de validation par le directeur technique ou le CEO.

3. Protection des données :

- Les données des utilisateurs, telles que leurs noms, domaines d'entreprise et types de profil, seront stockées dans la base de données PostgreSQL.
- Des mesures de sécurité, telles que le chiffrement des données sensibles et la sauvegarde régulière des données, seront mises en place pour assurer la confidentialité et la disponibilité des données.
- Un cluster de SGBD peut être mis en place, avec un outil logiciel comme PG-Pool II.

4. Surveillance et détection des incidents :

- Un système de surveillance des activités anormales et des incidents de sécurité est mis en place pour détecter les éventuelles violations de sécurité ou les comportements suspects. Il utilise notre système de supervision et de gestion de journaux d'événements.
- Des procédures de gestion des incidents seront définies, notamment pour la notification, l'investigation, la résolution et le suivi des incidents de sécurité.

5. Conformité et réglementations :

- Le projet doit se conformer aux réglementations applicables en matière de protection des données et de confidentialité, telles que le RGPD (Règlement général sur la protection des données).
- Des audits réguliers seront effectués pour évaluer la conformité et l'efficacité des mesures de sécurité mises en place.

Les 4 critères de sécurité pour le service d'information (SI) dans un PSSI sont :

1. **Confidentialité** : le SI de mettre en place une politique de gestion des droits pour avoir la fiabilité que seules les personnes autorisées ont accès aux données.
2. **Disponibilité** : le SI doit rendre accessible les données, rapidement et avec une constance dans le temps.
3. **Intégrité** : le SI doit être en mesure de vérifier et prouver qu'aucune modification n'a été apportée aux données.
4. **Tracabilité** : le SI doit mettre en place un historique des accès aux données, historique qui se doit d'être lisible et vérifiable.

Norme ISO 27000

Comme indiqué précédemment dans le PSSI, être conforme à la norme ISO 27000 serait un avantage considérable, car elle est un ensemble de normes internationales qui fournit des lignes directrices et des bonnes pratiques pour la gestion de la sécurité de l'information au sein d'une organisation. Voici les principales recommandations après une lecture des documents des normes ISO 27000, 27001, 27005, et une liste de d'actions à mettre en place pour y répondre :

1. **Politique de sécurité** : élaboration d'une politique de sécurité claire et documentée qui énonce les principes, les objectifs et les responsabilités en matière de sécurité de l'information.
2. **Gestion des risques** : effectuer une évaluation approfondie des risques liés à la totalité des microservices de QiBates, en identifiant les informations sensibles, les menaces potentielles et les vulnérabilités. Établissement de mesures de sécurité appropriées pour atténuer ces risques.
3. **Contrôles de sécurité** : mise en place des contrôles de sécurité techniques et managériales conformes aux recommandations de la norme ISO 27000. cela peut inclure la mise en œuvre de mécanismes d'authentification (OAuth a été utilisé pour ce projet) et d'autorisation, le chiffrement des données, la surveillance des activités, la gestion des journaux (avec Loki et Grafana) et la protection contre les attaques.
4. **Sensibilisation et formation** : sensibilisation et formation des membres de l'équipe du le projet aux principes de sécurité de l'information, aux bonnes pratiques de développement sécurisé et aux politiques de sécurité établies.
5. **Gestion des incidents** : mise en place un processus de gestion des incidents pour détecter, signaler, investiguer et répondre aux incidents de sécurité éventuels. Établissement de mécanismes de signalement des incidents et vérification de l'application des mesures correctives appropriées.

6. **Évaluation et amélioration continue** : effectuer des audits réguliers pour évaluer la conformité aux exigences de la norme ISO 27000 et pour identifier les domaines d'amélioration. Mise en place un processus d'amélioration continue pour renforcer la sécurité de l'information tout au long du projet.

4.3.3 Contexte de réalisation

La mise en place du PSSI et de l'analyse des normes ISO a été réalisée en cours de projet, après le prototypage de celui-ci, mais avant la mise en production des micro-services développés.

Voici mes éléments personnels quand à la réalisation de cette activité : Grâce à notre formation en RGPD et sécurité des S.I. suivie en entreprise, j'ai été informé des principes de protection des données et des bonnes pratiques de sécurité.

La formation sur les normes ISO 27000 et sur la réalisation d'un PSSI, suivie à l'EPSI, m'a doté d'une solide compréhension des normes de sécurité de l'information et des étapes nécessaires pour mettre en place un PSSI.

Ainsi que les éléments organisationnels et structurels :

En me référant à différents PSSI trouvés en ligne, j'ai lu et analysé les bonnes pratiques de sécurité de l'information mises en œuvre par d'autres organisations pour les adapter à mon projet (tel le PSSI de l'université de Poitiers). Le formateur de l'EPSI m'a fourni des ressources telles que les normes ISO 27000-27001-27002-27005, me permettant de me familiariser avec les exigences et les recommandations des normes citées précédemment.

4.3.4 Conclusion sur l'activité

J'ai donc réalisé une maquette de PSSI et une analyse des normes ISO 27000 dans le cadre de cet activité. La formation à l'EPSI ainsi que l'application pour ce projet m'a permis de capitaliser et d'appliquer mes connaissances dans une situation concrète.

Les analyses produites m'ont servis pendant la réalisaiton du projet, pour analyser les points sensibles des microservices, et d'actions à mettre en place pour améliorer la sécurité du système d'information.

Les évolutions possibles sont les mises à jour des normes ISO, et des bonnes pratiques pour la réalisation d'un PSSI, tout en effectuant une veille sur la sécurité informatique.

4.4 Activité 3 - analyse des données de références pour la création d'un référentiel de données

4.4.1 Compétence et son fondement

Bloc de compétences : A4 – Pilotage de l'informatique décisionnelle d'un système d'information (business intelligence big data)

Compétence choisie : A4C6 – Définir les données de référence de l'entreprise à partir des données utilisées pour créer un référentiel de données afin d'assurer la mise à disposition de données cohérentes aux directions métiers

Détails : Cette compétence consiste à démontrer ma capacité à identifier le référentiel de donnée de l'entreprise :

- identifier et formaliser par écrit des critères de sélection et de validation des données existantes.

4.4.2 Présentation et réalisation de l'activité

Dans le chapitre "4.2.2", avec la figure "4.2 - Modèle de conception de données de la base de donnée du monolithe" et la description présente en dessous, le MCD est présent avec la description de chaque table. L'architecture microservices m'a orienté quand à la division de la précédente base de données en 3 nouvelles bases de données distinctes.

Voici en dessous chaque MCD pour chacune des 3 bases de données, ainsi que la description de celui-ci. Voici la description de la base de données pour sauvegarder les fonctionnalités :

- **Domain** : enregistre le domaine du client.
- **User** : enregistre le nom de l'utilisateur.
- **FeatureGroup** : enregistre un groupe de fonctionnalités.
- **Feature** : enregistre une fonctionnalité.

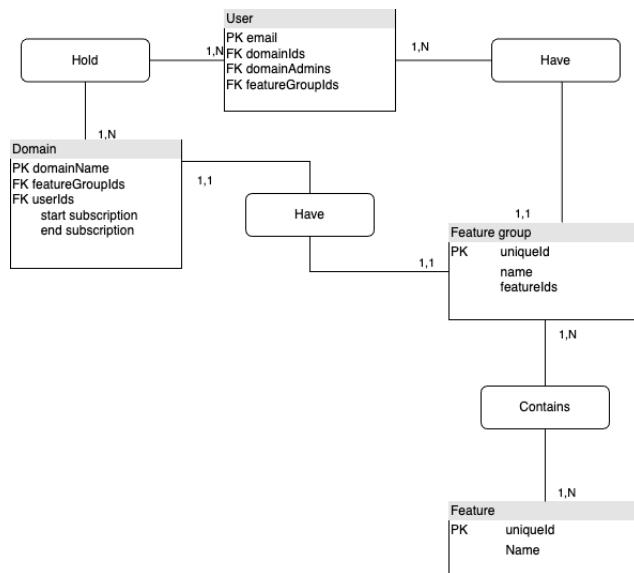


FIGURE 4.3 – Modèle de conception de données des domaines clients

Voici la description de la base de données pour sauvegarder les paramètres des utilisateurs :

- **Domain** : enregistre le domaine du client.
- **User** : enregistre le nom de l'utilisateur.
- **DomainUserSetting** : enregistre un groupe de paramètres pour l'utilisateur.
- **DefaultSetting** : enregistre un paramètre par défaut.

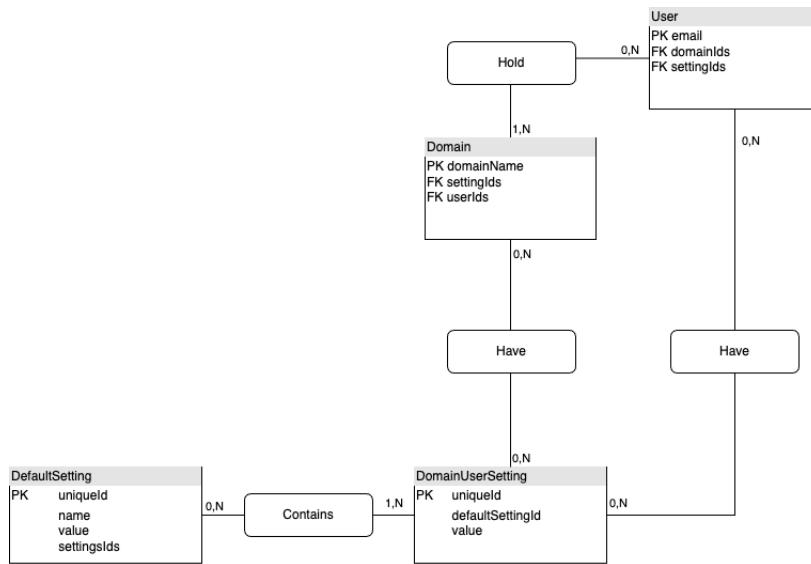


FIGURE 4.4 – Modèle de conception de données pour la sauvegarde des paramètres clients

Voici la description de la base de données pour sauvegarder les notifications à envoyer :

- **Domain** : enregistre le domaine du client.
- **User** : enregistre le nom de l'utilisateur.
- **Notification** : enregistre les notifications à envoyer à l'utilisateur.

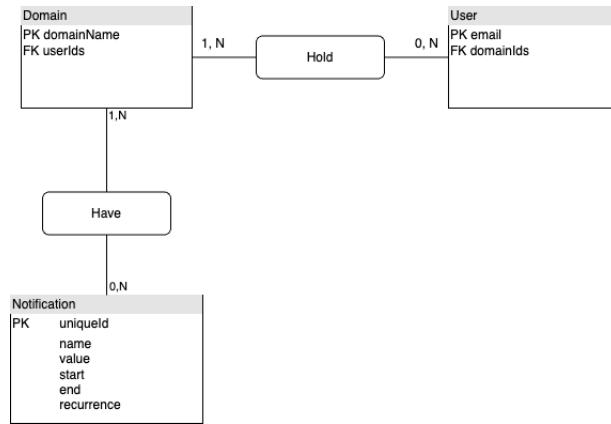


FIGURE 4.5 – Modèle de conception de données de sauvegarde des notifications

Pour le choix du gestionnaire de base de données, j'ai utilisé le théorème CAP. Il permet selon 3 critères, de sélectionner un SGBD en fonction des problématiques auxquelles il doit répondre, comme la disponibilité, la cohérence et la distributivité. La volonté de garder un SGBD avec un modèle relationnel, et en prenant en compte les autres SGBD présents dans l'infrastructure de QiBates, le choix c'est porté sur PostgreSQL, étant Open Source, et ayant une bonne documentation ainsi qu'une grande communauté.

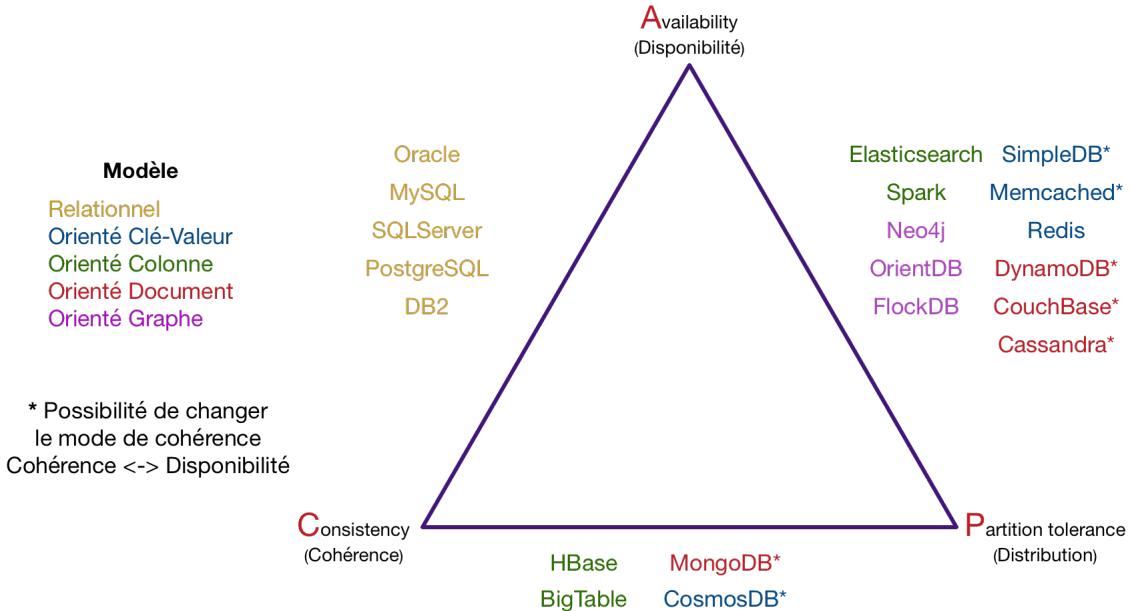


FIGURE 4.6 – Théorème CAP

Chacune des 3 bases de données sera présente sur le même SGBD pour des

raisons de coûts et de maintenance, car permettant d'éviter de payer 3 SGBD chez OBS, et n'ayant qu'un seul SGBD à répliquer, avec un outil comme PG-Pool II.

4.4.3 Contexte de réalisation

J'ai collaboré avec un développeur pour valider les MCD, et obtenu l'approbation de mon directeur technique après lui avoir présenté la refonte totale du SGDB. Trois bases de données ont ainsi été créées pour stocker les fonctionnalités, les paramètres des utilisateurs et les notifications, avec PostgreSQL comme choix de gestionnaire de base de données, validé en considération de l'infrastructure existante. Le SGBD est toujours managé par OBS, j'ai donc créé une nouvelle instance pour héberger les 3 nouvelles bases de données.

4.4.4 Conclusion sur l'activité

Au cours de cette activité, j'ai exploré les tenants et aboutissants de l'analyse des données de référence pour la création d'un référentiel cohérent. En adaptant le modèle conceptuel de données du monolithe à l'architecture microservices, j'ai orchestré la création de trois bases distinctes pour sauvegarder les fonctionnalités, les paramètres des utilisateurs et les notifications. En optant pour PostgreSQL comme gestionnaire de base de données, en accord avec les besoins et la structure existante, j'ai mis en oeuvre mes capacités à réaliser cette activité, avec des choix justifiés et justifiables.

Pour les évolutions dans 10 ans, les bases de données relationnelles ainsi que les schémas sous-jacents comme les MCD seront toujours d'actualité pour la sauvegarde de données, malgré l'essor des SGBD NoSQL (Not Only SQL, sauvegardant des données en format clef-valeur, JSON ou autre).

4.5 Activité 4 - analyse des besoins métiers pour une solution applicative sur mesure

4.5.1 Compétence et son fondement

Bloc de compétences : A5 – Développement d'une solution applicative spécifique et métier selon le projet de développement S.I.

Compétence choisie : A5C1 – Collecter les besoins métiers des utilisateurs en menant des interviews auprès d'eux pour comprendre leurs activités et leurs contraintes métier afin d'étudier les opportunités et la faisabilité technologique d'une solution applicative spécifique ou métier

Détails : Cette compétence consiste à démontrer ma capacité à présenter mes éléments de préparation à la collecte des besoins en détaillant :

- l'interview-type à mener avec le questionnaire à utiliser en précisant la méthodologie de transcription (tableau, graphique,...)

- besoins métiers attendus par le client en termes de fonctionnalités
- contraintes métiers à prendre en compte

4.5.2 Présentation et réalisation de l'activité

Étapes de réalisation de l'interview

1. Préparation de l'Interview :
 - J'ai identifié les modules du monolithe à transformer en microservices.
 - J'ai identifié les personnes clefs à interroger (trois développeurs du pôle R&D seront interrogés).
 - J'ai créé un modèle d'interview incluant des questions sur les fonctionnalités actuelles, les défis rencontrés et les besoins non satisfaits.
2. Conduite des Interviews :
 - J'ai réalisé une enquête avec Google forms
3. Identification des Besoins Métiers :
 - J'ai analysé les réponses du formulaire pour identifier les besoins spécifiques.
 - J'ai identifié les fonctionnalités clés à découper en microservices et les interactions nécessaires.
4. Formalisation des Contraintes Métiers :
 - J'ai inclus les exigences de sécurité, de performances, d'évolutivité, etc.
5. Création des Questionnaires de Retranscription :
 - J'ai créé des questionnaires spécifiques pour retranscrire les réponses des entretiens.
 - J'ai utilisé des tableaux ou des graphiques pour organiser les besoins et contraintes.
6. Présentation des Résultats :
 - J'ai présenté les besoins et contraintes identifiés pour chaque module aux parties prenantes.
 - J'ai expliqué comment la transformation en microservices répondra aux besoins et contraintes métiers.

Outil pour la réalisation de l'interview

J'ai utilisé Google Forms, un outil de création de formulaire gratuit et en ligne, pour réaliser mon enquête. Celui-ci m'a permis de présenter les questions et d'obtenir les réponses des personnes concernées directement sur la page associée au formulaire.

Refonte du service de g

Toutes les modifications ont été enregistrées dans Drive

Envoyer

Questions Réponses Paramètres

Refonte du service de gestion de licences utilisateurs

Cette enquête a pour but de confirmer l'utilisation précédente de l'ancien service monolithique, les problèmes rencontrés, et les améliorations à apporter.

Quelles sont les fonctionnalités les plus critiques de chaque module du monolithique ?

Réponse longue

Comment interagissent ces fonctionnalités entre elles et avec d'autres modules ?

Réponse longue

Quels sont les principaux problèmes ou limitations que vous rencontrez actuellement avec le monolithique ?

Réponse longue

Pouvez-vous identifier des domaines où la performance du monolithique est insatisfaisante ?

FIGURE 4.7 – Google Forms pour la réalisation de l'enquête utilisateurs

Les résultats sont présents sur l'onglet "Réponses", et ne sont visibles que par le créateur du Google Forms par défaut.

Questions de l'enquête

Voici une liste de 7 questions proposées aux trois développeurs ciblés, ces questions ayant pour but de confirmer l'utilisation précédente de l'ancien service monolithique, les problèmes rencontrés, et les améliorations à apporter.

1. Quelles sont les fonctionnalités les plus critiques de chaque module du monolithique ?
2. Comment interagissent ces fonctionnalités entre elles et avec d'autres modules ?

3. Quels sont les principaux problèmes ou limitations que vous rencontrez actuellement avec le monolithique ?
4. Pouvez-vous identifier des domaines où la performance du monolithique est insatisfaisante ?
5. Quels sont les éléments de sécurité importants à prendre en compte pour chaque module ?
6. Y a-t-il des fonctionnalités que vous souhaiteriez ajouter à chaque module à l'avenir ?
7. Quels sont les améliorations en terme d'UX/UI design que vous souhaitez rencontrer ?

4.5.3 Contexte de réalisation

J'ai élaboré l'enquête en utilisant la plateforme Google Forms pour recueillir les informations nécessaires à la transformation du monolithique en microservices. Les questions ont été formulées de manière à comprendre comment les utilisateurs interagissent avec le système et à identifier leurs préférences en matière de fonctionnalités. Toute cette compétence a été réalisée à distance avec des outils de travail collaboratif en ligne, comme Microsoft Teams, pour communiquer avec les différents collaborateurs. Une réunion finale a eu lieu pour partager les résultats de l'enquête aux différents membres de l'équipe R&D, pour valider les réponses et propositions de chacun.

4.5.4 Conclusion sur l'activité

Cette compétence n'a pas à mon avis d'évolution majeure à venir dans les 10 ans à venir, la plus grande dernière étant le passage du papier au numérique pour la réaliser.

Je n'ai pas eu de difficulté particulière quand à la réalisation de cette compétence, et je ne vois non plus de piste de progression.

4.6 Activité 5 - conception d'une architecture applicative évolutive et tolérante aux pannes

4.6.1 Compétence et son fondement

Bloc de compétences : A5 – Développement d'une solution applicative spécifique et métier selon le projet de développement S.I.

Compétence choisie : A5C2 – Concevoir une architecture applicative selon la complexité du système d'information existant de type architecture distribuée, ou micro service évolutive et tolérante aux pannes

Détails : Cette compétence consiste à démontrer ma capacité à proposer une architecture applicative :

- argumenter mon choix en formalisant par écrit les critères de stabilité, d'efficacité et de pérennité.
- lister l'environnement technique adéquat à son architecture applicative (architecture distribuée, clusters, architecture micro services, REST, appel des services....)
- présenter sous forme de schéma la logique du fonctionnement de notre architecture applicative

4.6.2 Présentation et réalisation de l'activité

Analyse en amont de la réalisation du schéma d'infrastructure

Étant déjà dans une architecture en microservices avec Qibates, j'ai choisis de continuer sur cette architecture, en rajoutant les nouveaux services créés depuis l'ancien service monolithique. J'ai décomposé les principaux points à prendre en compte, que voici :

1. **Analyse des Besoins** : J'ai commencé par examiner les besoins spécifiques de l'application, en mettant l'accent sur la séparation des fonctionnalités en modules indépendants et en identifiant les services critiques.
2. **Décomposition des Fonctionnalités** : J'ai découpé l'application monolithe en modules autonomes, chacun représentant une fonctionnalité ou un domaine spécifique. Chaque module serait développé et déployé en tant que service indépendant.
3. **Choix Technologiques** : J'ai sélectionné les technologies appropriées pour la mise en œuvre de l'architecture microservices. J'ai opté pour Spring Boot pour le développement des services REST et Angular pour l'interface utilisateur.
4. **Communication entre Services** : J'ai mis en place des API REST pour permettre la communication entre les différents services. J'ai également utilisé un système de messagerie pour les notifications asynchrones entre les services lorsque nécessaire.
5. **Gestion des Erreurs et Tolérance aux Pannes** : J'ai intégré des mécanismes de gestion des erreurs tels que les messages d'erreur standardisés et les codes d'état HTTP. De plus, j'ai conçu des stratégies de récupération pour garantir la continuité du service en cas de défaillance d'un module.
6. **Équilibrage de Charge** : J'ai mis en place un équilibrage de charge pour répartir les requêtes entre les différents services, assurant ainsi une utilisation optimale des ressources et une meilleure réactivité de l'application.
7. **Sécurité** : J'ai intégré des mécanismes de sécurité tels que l'authentification et l'autorisation au niveau des services, ainsi qu'une API Gateway (en utilisant notre serveur Nginx existant) pour centraliser la gestion des accès.

8. **Monitoring et Journalisation** : J'ai inclus des outils de supervision pour surveiller les performances et la disponibilité des services. J'ai également configuré la journalisation pour faciliter le débogage en cas de problème. Telegraf est utilisé pour récupérer les métriques, InfluxDB pour les stocker, et Grafana pour les afficher. Ces trois solutions sont open-sources.
9. **Documenter l'Architecture** : J'ai préparé une documentation détaillée de l'architecture, y compris les interactions entre les services, les dépendances, les flux de données et les points d'entrée.
10. **Validation et Tests** : J'ai testé l'architecture en simulant différentes conditions, notamment des pannes de services et des pics de charge, pour m'assurer de sa stabilité et de sa capacité à répondre aux besoins. L'utilisant en interne pour d'autres projets, j'ai utilisé Jmeter, une solution open-source, pour effectuer ces tests de charge.

Réalisation du schéma d'infrastructure

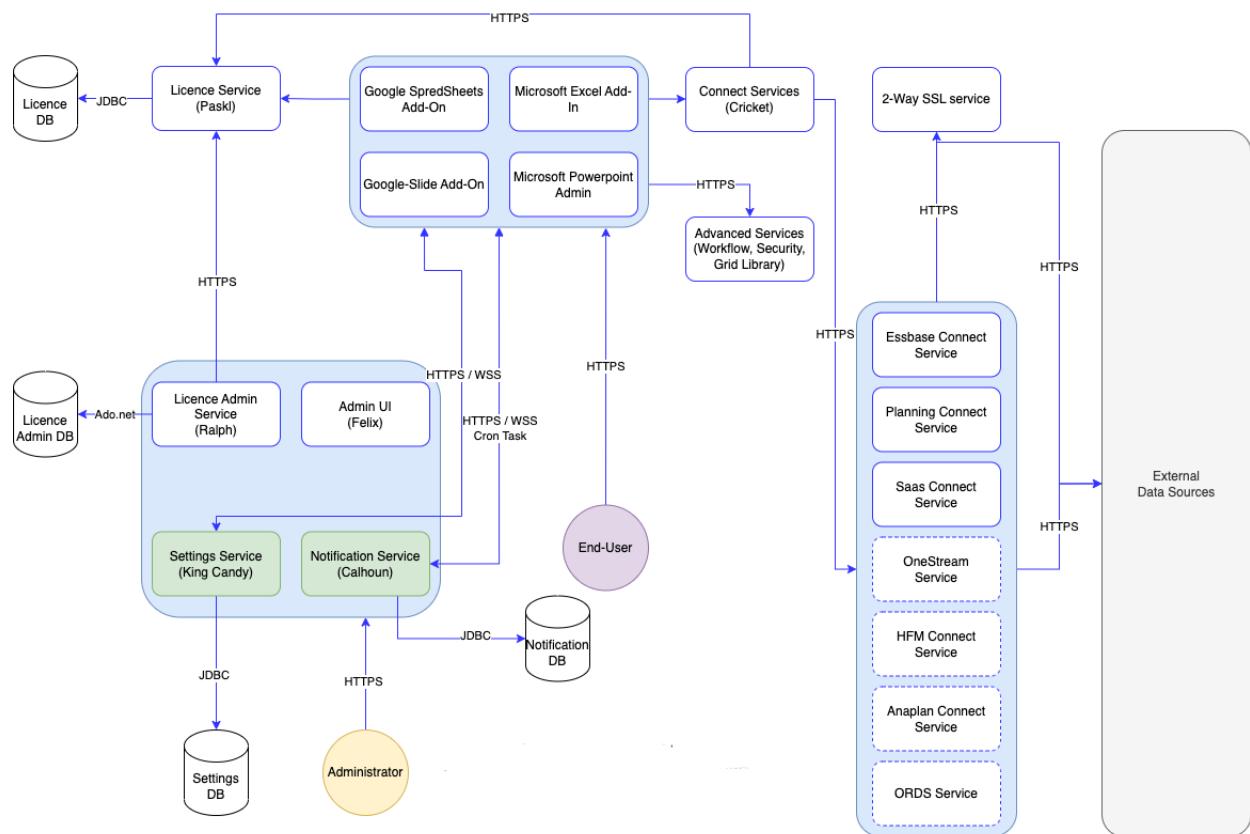


FIGURE 4.8 – Schéma de la refonte en microservices

Voici donc le nouveau schéma de l'architecture de QiBates. La zone bleu en bas à gauche, ainsi que les bases de données liées aux microservices dans cette zone, sont les nouveaux microservices remplaçant l'ancien service monolithique.

Le service “**Admin UI**” est l’interface web permettant de gérer les 3 autres microservices liés.

“**Licence Admin**” permet comme son nom l’indique de gérer les licences utilisateurs des clients, pour utiliser QiBates.

“**Settings Service**” permet à un utilisateur de sauvegarder des paramètres pour QiBates comme bon lui semble.

“**Notification Service**” permet l’envoie de notifications à un utilisateur, comme par exemple une notification de fin de tâche de calcul sur QiBates.

L’interface d’administration est une application web, les 3 autres services sont des API REST, et PostgreSQL est utilisé comme SGBD pour les bases de données. Toutes les communications entre chaque service utilisent le protocole HTTP.

4.6.3 Contexte de réalisation

J’ai utilisé Draw.io, un logiciel gratuit et open-source pour réaliser les schémas d’architecture. J’ai aussi utilisé Microsoft Teams pour valider mes schémas auprès de mon directeur technique, lors de réunions en visioconférences.

4.6.4 Conclusion sur l’activité

Dans 10 ans, les architectures micro-services seront toujours présentes, même si celle-ci a des problèmes liés aux coûts qui peuvent se montrer excessif, malgré un avantage certain quand à la réduction de complexité et une augmentation de la fiabilité par rapport à une architecture monolithique.

Ayant travaillé précédemment sur d’autres microservices de QiBates, je n’ai pas rencontré de difficulté particulière quand à la réalisation de cette activité. Le choix des technologies a été effectué en prenant en compte l’architecture actuelle, et les moyens de communications entre microservices sont basés sur les standards du web (émis par la W3C).

4.7 Activité 6 - développement d’une application pour répondre aux besoins utilisateurs/directions métiers

4.7.1 Compétence et réalisation et son fondement

Bloc de compétences : A5 – Développement d’une solution applicative spécifique et métier selon le projet de développement S.I.

Compétence choisie : A5C3 Développer une application adéquate selon la stratégie applicative de l'environnement en utilisant un langage de programmation approprié dans le respect du cahier des charges établi afin de répondre aux besoins utilisateurs/directions métiers

Détails : Cette compétence consiste à démontrer ma capacité à développer une application en adéquation avec l'environnement technique de l'entreprise (développement web, développement mobile, développement embarqué, développement IOT) :

- utiliser le langage de programmation approprié à l'environnement technique
- concevoir et détailler une logique de développement dans le respect du cahier des charges
- en résultat attendu, faire une démonstration technique de l'application (interprétation de certaines lignes de codes)

4.7.2 Présentation et réalisation de l'activité

Introduction

Pour développer les microservices, j'ai utilisé Angular pour réaliser l'interface web, Spring Boot pour les API REST, et PostgreSQL comme SGBD. Ces technologies sont déjà utilisées pour nos microservices précédents, j'ai choisis de les réutiliser, car ayant déjà des connaissance avec, je peux aussi demander de l'aide aux autres collaborateurs en cas de problème, et notre stack technologique est uniforme dans notre infrastructure.

Angular est un framework web utilisant TypeScript, conçu pour créer des applications interactives et dynamiques côté client. Il facilite la gestion des composants, de l'état de l'application et des interactions avec les données.

Spring Boot est un framework Java qui simplifie la création d'applications robustes et évolutives en gérant automatiquement la configuration et en fournissant des fonctionnalités prêtes à l'emploi, comme la création de routes pour une API REST, un *ORM*¹², et des outils pour tester notre code.

PostgreSQL est un système de gestion de base de données relationnelle open source, permettant de stocker et d'organiser des données de manière structurée, et sa fiabilité n'est plus à démontrer.

Structure des projets

The screenshot shows two terminal windows side-by-side. The left window is for the project 'felix' and the right window is for 'ralph-2.0'. Both windows use the 'tree -d .' command to display the directory structure.

felix Project Structure:

- app
 - core
 - components
 - add-user
 - domain-table
 - footer
 - header
 - profile-table
 - role-table
 - user-table
 - interfaces
 - views
 - domain-view
 - profile-view
 - role-view
 - user-view- assets
 - images
- environments

20 directories

ralph-2.0 Project Structure:

- main
 - java
 - com
 - quartzinsight
 - ralph2
 - controller
 - dto
 - model
 - repository
 - service
 - resources
 - test
 - java
 - com
 - quartzinsight
 - ralph2
 - Initializers
 - controller
 - dto
 - model
 - repository
 - service
 - resources

24 directories

Both terminals show the command being run: `tree -d .` and the output count of directories (20 for felix, 24 for ralph-2.0). The bottom of the terminal shows the user's name: geoffreybrunet and Mac-de-Geoffrey.

FIGURE 4.9 – Structure des dossiers pour deux microservices

J'ai utilisé la structure commune aux projets Spring Boot, pour une meilleure compréhension, une meilleure collaboration et une meilleure maintenance :

1. main

- **java** : C'est ici que se trouve le code Java principal de l'application.
- **com.quartzinsight.ralph2.TITI** : C'est le package de base de l'application.
- **controller** : Contient les classes qui gèrent les requêtes HTTP.
- **model** : Contient les classes représentant les entités de données de l'application.
- **repository** : Contient les interfaces pour interagir avec la base de données.
- **service** : Contient la logique métier de l'application.

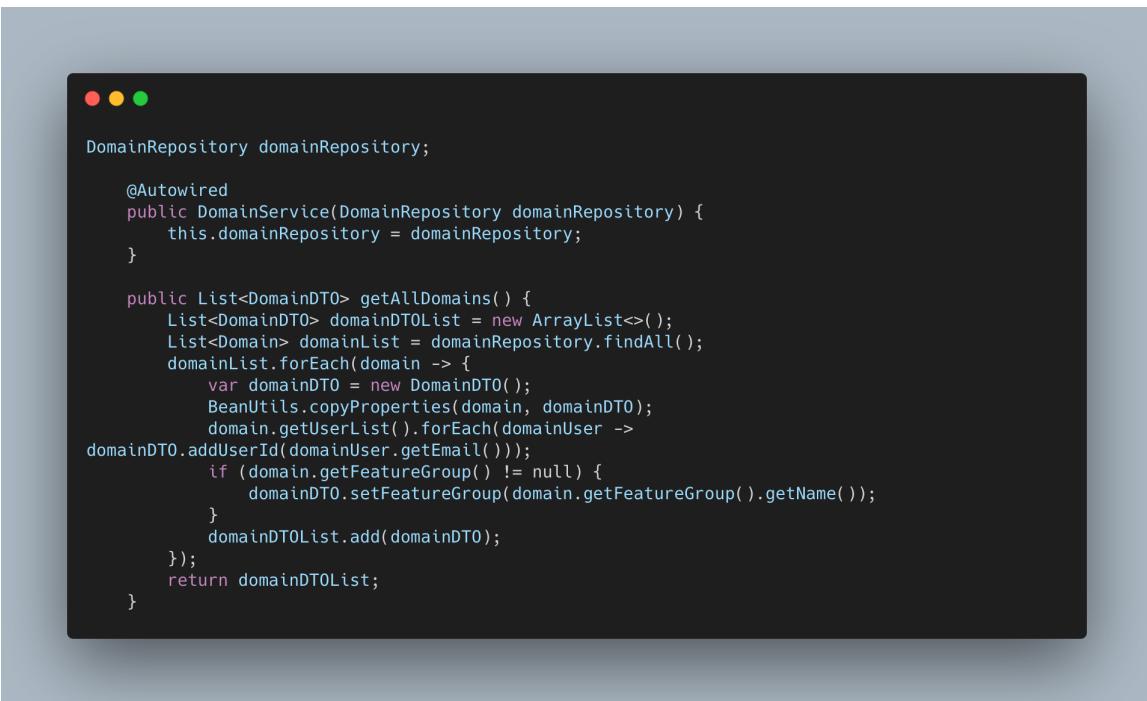
2. resources : Contient les fichiers de configuration et les ressources statiques de l'application.

J'ai aussi utilisé la même structure pour l'application Angular que dans la documentation officielle :

- **app** : C'est le répertoire principal de l'application.
- **core** : Contient les éléments centraux de l'application tels que les services partagés et les intercepteurs HTTP.
- **components** : Contient les composants réutilisables de l'application.

- **add-user** : Un composant pour ajouter un nouvel utilisateur.
- **domain-table** : Un composant pour afficher un tableau de domaines.
- **footer** : Un composant pour le pied de page.
- **header** : Un composant pour l'en-tête.
- **profile-table** : Un composant pour afficher un tableau de profils.
- **role-table** : Un composant pour afficher un tableau de rôles.
- **user-table** : Un composant pour afficher un tableau d'utilisateurs.
- **interfaces** : Contient les interfaces TypeScript utilisées dans l'application.
- **views** : Contient les vues principales de l'application.
 - **domain-view** : Une vue pour afficher les détails d'un domaine.
 - **profile-view** : Une vue pour afficher les détails d'un profil.
 - **role-view** : Une vue pour afficher les détails d'un rôle.
 - **user-view** : Une vue pour afficher les détails d'un utilisateur.
- **assets** : Contient les ressources statiques de l'application, comme les images.
- **environments** : Contient les fichiers de configuration pour différents environnements (par exemple, développement et production).

Exemple de code



```
DomainRepository domainRepository;

@Autowired
public DomainService(DomainRepository domainRepository) {
    this.domainRepository = domainRepository;
}

public List<DomainDTO> getAllDomains() {
    List<DomainDTO> domainDTOList = new ArrayList<>();
    List<Domain> domainList = domainRepository.findAll();
    domainList.forEach(domain -> {
        var domainDTO = new DomainDTO();
        BeanUtils.copyProperties(domain, domainDTO);
        domain.getUserList().forEach(domainUser ->
            domainDTO.addUserId(domainUser.getEmail());
            if (domain.getFeatureGroup() != null) {
                domainDTO.setFeatureGroup(domain.getFeatureGroup().getName());
            }
            domainDTOList.add(domainDTO);
        });
    return domainDTOList;
}
```

FIGURE 4.10 – Exemple de code d'un service Spring Boot renvoyant une liste de domaines

Ce code utilise l'annotation Spring `@Autowired` pour injecter une instance de `DomainRepository` dans le service `DomainService`. Ce service récupère tous les domaines à partir de la base de données en utilisant la méthode `findAll()` du repository. Ensuite, pour chaque domaine, il crée un objet `DomainDTO` (Data Transfer Object), qui est une structure pour transporter les données. Les propriétés du domaine sont copiées vers le DTO à l'aide de `BeanUtils.copyProperties()`. En outre, les ID des utilisateurs associés sont ajoutés au DTO, et si le domaine a un groupe de fonctionnalités, le nom du groupe est défini dans le DTO. En fin de compte, une liste de DTOs de domaine est renvoyée.



```
constructor(private domainTableService: DomainTableService) {}

ngOnInit() {
  this.fillDomainsTable();
}

private fillDomainsTable() {
  this.domainTableService.getDomains().subscribe((response) => {
    this.domainList = response;
    this.dataSource = new MatTableDataSource(this.domainList);
    this.dataSource.sort = this.sort;
    this.dataSource.paginator = this.paginator;
  });
}

addRow() {
  const newRow = {
    client: '',
    domain: '',
    selected_pack: '',
    user_profile: '',
    nb_used: 0,
    total: 0,
    creation_date: new Date(),
    end_date: new Date(),
    status: ''
  } as EditableDomain;
  newRow.isEditing = true;
  this.dataSource.data.push(newRow);
  this.dataSource._updateChangeSubscription();
  this.isLastRowEditing = true;
}
}
```

FIGURE 4.11 – Exemple de code d'un service Spring Boot renvoyant une liste de domaines

Ce code Angular exécute les actions suivantes :

1. Dans la méthode `ngOnInit()`, au démarrage du composant, `fillDomainsTable()` est appelée pour remplir une table de données de domaines.
2. La fonction `fillDomainsTable()` utilise `domainTableService` pour obtenir des domaines via une requête asynchrone. Une fois la réponse reçue, elle configure une source de données de table (`MatTableDataSource`) avec les domaines pour permettre le tri et la pagination.
3. `addRow()` ajoute une nouvelle ligne à la table. Elle crée un nouvel objet avec des valeurs par défaut, le marque comme en cours d'édition, et l'ajoute à la liste des données de la table. Ensuite, la table est mise à jour pour refléter cette modification. Cette fonction est utilisée pour ajouter de nouvelles entrées à une table éditable.

4.7.3 Contexte de réalisation

J'ai utilisé **Neovim** comme éditeur de code, **Maven** comme gestionnaire de builds et dépendances pour les microservices en Java (Version 17), et **Node.JS / NPM** comme runtime et gestionnaire de dépendances pour l'application Angular.

4.7.4 Conclusion sur l'activité

Cette compétence m'a permis de m'améliorer sur le développement, et de collaborer en équipe pour que les microservices soient en adéquation avec notre architecture actuelle et répondent aux besoins des utilisateurs.

Le choix des technologies et langages est sujet à fort changement, en effet, depuis 10 ans, les technologies utilisées ont bien changées, et certaines comme Angular n'existaient pas.

4.8 Activité 7 - analyse de la qualité de la solution applicative et mise en place d'un plan de correction / d'amélioration

4.8.1 Compétence et son fondement

Bloc de compétences : A5 – Développement d'une solution applicative spécifique et métier selon le projet de développement S.I.

Compétence choisie : A5C5 – Effectuer les tests de la solution applicative paramétrée ou développée pour identifier les erreurs et dysfonctionnements et établir les plans de correction/d'amélioration avant sa mise en production

Détails : Cette compétence consiste à démontrer ma capacité à tester une solution applicative :

- rédiger un plan de test en précisant la typologie de test (unitaire, récursive, d'intégration), les données de test et les résultats attendus
- mettre en œuvre le plan de test à l'aide des jeux d'essai
- utiliser un outil de testing de mon choix
- Résultat attendu : présenter les facteurs de la bonne testabilité de son plan de test

4.8.2 Présentation et réalisation de l'activité

Pour vérifier la qualité de mon code, j'écris des tests unitaires et d'intégration. Nous utilisons en interne un outil appelé SonarQube pour vérifier la qualité de notre code. Voici un exemple d'un test unitaire pour l'application Angular, ainsi que la page SonarQube associé à cette même application.

```

1 {
2   "client": "client_1",
3   "domain": "domain_1",
4   "selected_pack": "selected_pack_1",
5   "user_profile": "user_profile_1",
6   "nb_used": 42,
7   "total": 42,
8   "creation_date": "2012-04-23T18:25:43.511Z",
9   "end_date": "2012-04-23T18:25:43.511Z",
10  "status": "ENABLED"
11 },
12 {
13   "client": "client_2",
14   "domain": "domain_2",
15   "selected_pack": "selected_pack_2",
16   "user_profile": "user_profile_2",
17   "nb_used": 24,
18   "total": 24,
19   "creation_date": "2012-04-23T18:25:43.511Z",
20   "end_date": "2012-04-23T18:25:43.511Z",
21   "status": "DISABLED"
22 }
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

```

FIGURE 4.12 – Exemple de fausses données et d'un test unitaire les utilisant

Ce code comprend des tests unitaires pour un service. Dans le premier test, il vérifie si le service a été créé avec succès en utilisant le fournisseur `DomainTableService`. Ensuite, il simule l'utilisation de MirageJs pour gérer les requêtes HTTP en simulant des données de domaine similaires à “client” : “client_1”, “domain” : “domain_1”, etc. Dans le deuxième test, il vérifie si le service `getDomains()` est appelé à l'aide de `jest.spyOn()`. Le code utilise également `httpTestingController` pour simuler les requêtes HTTP et comparer les données reçues avec les données de test définies. L'ensemble des tests est exécuté dans un environnement de test Angular grâce à `TestBed`.

MirageJS est une bibliothèque JavaScript qui permet de simuler des appels API HTTP côté client lors des tests. Elle est particulièrement utile pour les tests unitaires et d'intégration, car elle permet de créer un faux backend et de définir des endpoints avec des données simulées, évitant ainsi les appels réels aux serveurs et les dépendances externes.

Pour les tests sur les API REST, j'utilise des “mocks”, des simulations

de requêtes HTTP, grâce aux outils présents nativement dans Spring Starter Test.

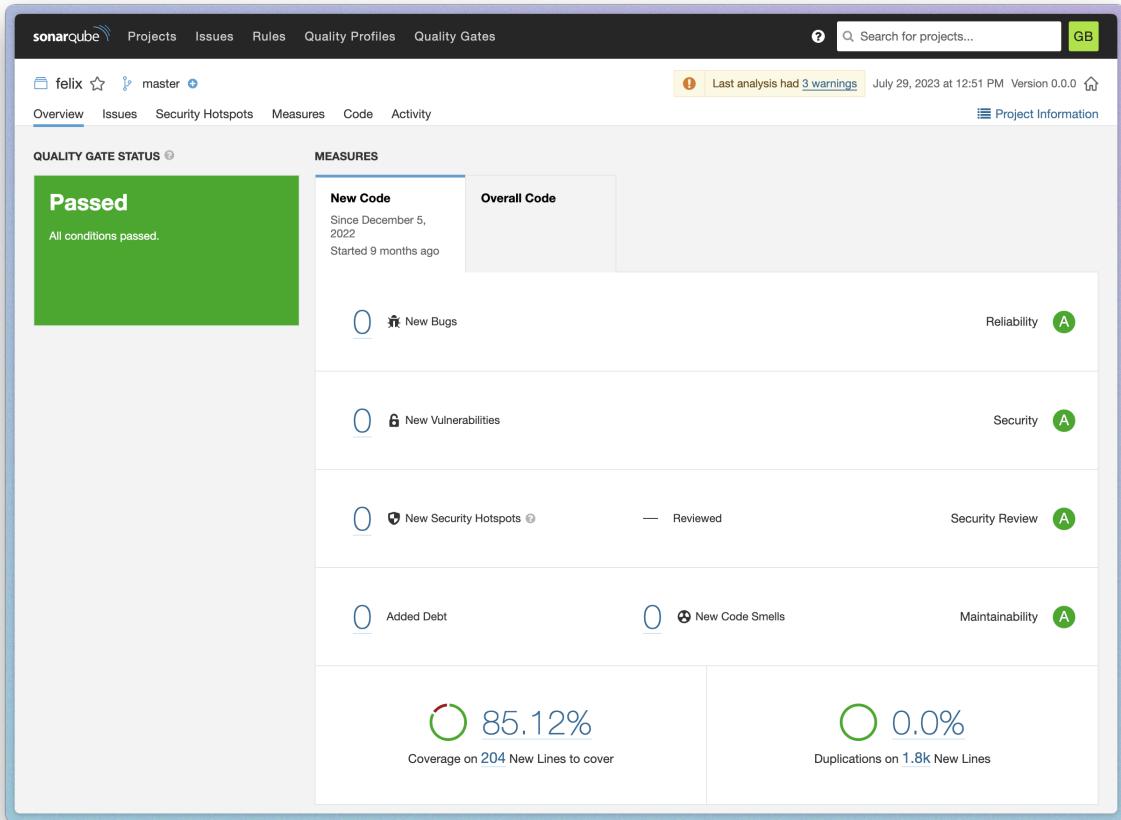


FIGURE 4.13 – Logiciel de couverture de tests et d’analyse de code

SonarQube est un outil que nous intégrons dans notre processus d’intégration continue (CI). Il nous aide à évaluer la qualité de notre code en identifiant les problèmes potentiels et les opportunités d’amélioration. Pendant notre CI, SonarQube analyse notre code pour repérer les bugs, les vulnérabilités et les duplications, tout en mesurant la couverture des tests.

Dans notre approche, nous exigeons qu’au moins **80%** du code soit testé pour que notre pipeline de déploiement puisse fonctionner. Cette exigence garantit une meilleure stabilité de notre application et réduit les risques d’erreurs. L’intégration de SonarQube dans notre CI assure que notre code reste de qualité, sécurisé et conforme aux meilleures pratiques de développement.

La capture d’écran présente les points suivants :

- 85.12% du code de l'application Angular est testé
- Aucun bug n'est présent (erreur dans le code qui provoque un comportement indésirable)
- Aucun code smell n'est présent (indicateur suggérant la présence d'un problème potentiel dans le code)
- Il n'y a pas de code dupliqué
- Aucun temps de déte technique c'est présent (temps à résoudre les divers problèmes cités dans cette liste).

4.8.3 Contexte de réalisation

J'ai utilisé les mêmes outils que pour l'activité 6 pour écrire les tests unitaires, avec en plus :

- Jest comme framework de tests pour mon application Angular
 - Spring Starter Tests pour réaliser les tests sur les microservices Angular.
- Comme montré précédemment j'ai aussi utilisé SonarQube comme logiciel pour vérifier ma couverture de tests et la qualité du code que j'ai produit.

4.8.4 Conclusion sur l'activité

Comme pour l'activité 6, peut d'évolutions peuvent avoir lieu, sauf sur le plan technologique, avec des outils toujours plus performants et simple d'utilisations.

Cette activité m'a permis d'améliorer ma compréhension des tests et de leur importance sur un projet architecturé en microservices. La prochaine étape est de réaliser des tests End-To-End, pour vérifier que les applications se comportent totalement comme désiré.

4.9 Activité 8 - mise en place d'une solution d'intégration continue

4.9.1 Compétence et son fondement

Bloc de compétences : A5 – Développement d'une solution applicative spécifique et métier selon le projet de développement S.I.

Compétence choisie : A5C6 –Appliquer l'intégration continue dans le cadre du développement d'une application en utilisant un outil d'intégration continue afin de vérifier la conformité de la solution et les besoins utilisateurs

Détails : Cette compétence consiste à démontrer ma capacité à utiliser un outil d'intégration continu :

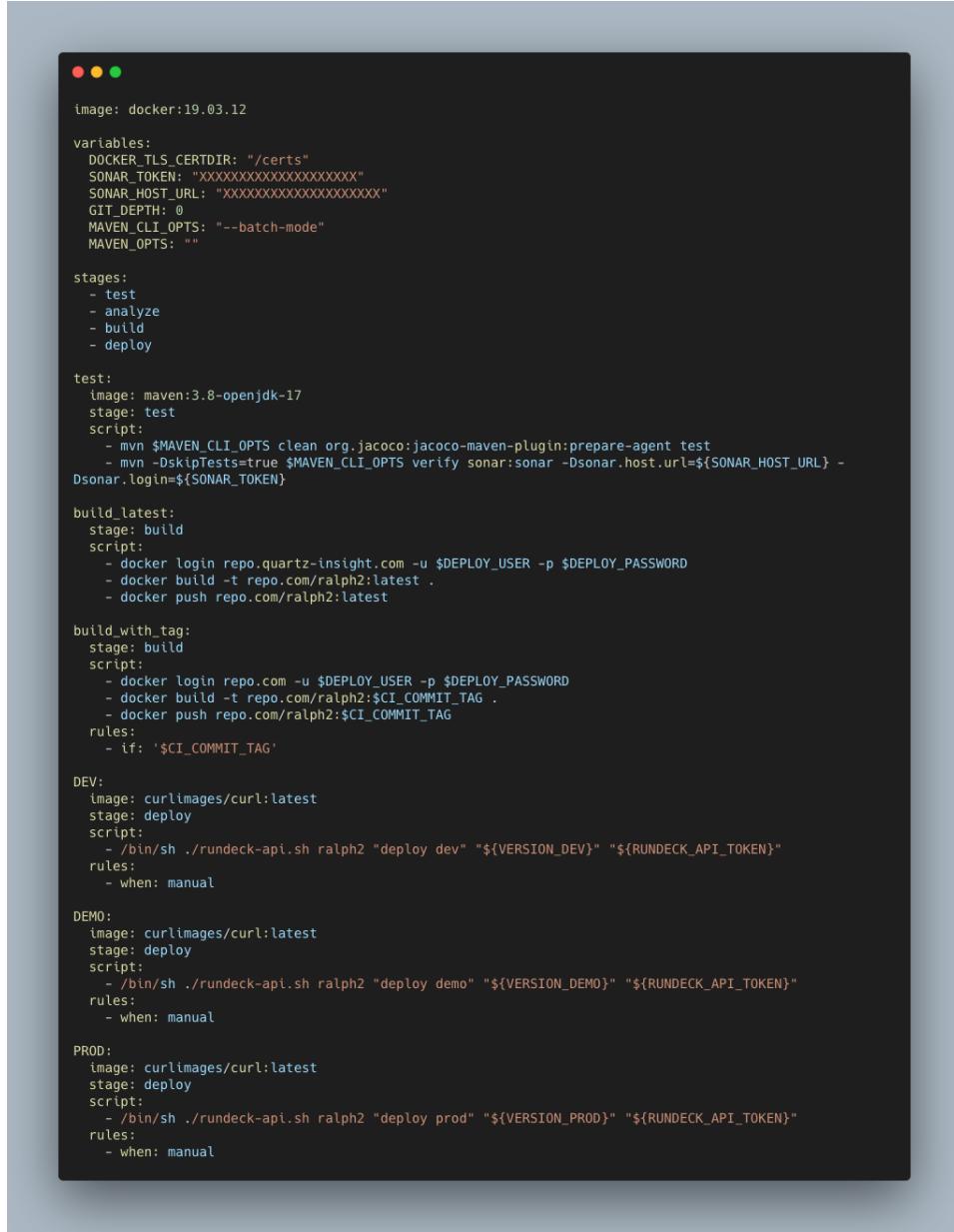
- installer et paramètre un outil d'intégration continu de mon choix
- utiliser les différentes fonctionnalités de l'outil d'intégration choisi

4.9.2 Présentation et réalisation de l'activité

Gitlab CI

GitLab est une plateforme tout-en-un pour la gestion des projets et du code. Sa fonctionnalité GitLab CI/CD facilite l'intégration continue et le déploiement automatique. J'ai opté pour GitLab CI pour l'intégration continue car nous l'utilisons en interne, et sa prise en main, même pour un novice, est relativement simple. En effet, il s'agit de fichier YAML à écrire, est un format de donnée relativement simple à écrire, à l'aide de tabulations et de données "clefs-valeurs".

Pipeline d'un microservice du serveur pour l'API REST



```
image: docker:19.03.12

variables:
  DOCKER_TLS_CERTDIR: "/certs"
  SONAR_TOKEN: "XXXXXXXXXXXXXXXXXXXX"
  SONAR_HOST_URL: "XXXXXXXXXXXXXXXXXXXX"
  GIT_DEPTH: 0
  MAVEN_CLI_OPTS: "--batch-mode"
  MAVEN_OPTS: ""

stages:
  - test
  - analyze
  - build
  - deploy

test:
  image: maven:3.8-openjdk-17
  stage: test
  script:
    - mvn $MAVEN_CLI_OPTS clean org.jacoco:jacoco-maven-plugin:prepare-agent test
    - mvn -DskipTests=true $MAVEN_CLI_OPTS verify sonar:sonar -Dsonar.host.url=${SONAR_HOST_URL} -Dsonar.login=${SONAR_TOKEN}

build_latest:
  stage: build
  script:
    - docker login repo.quartz-insight.com -u $DEPLOY_USER -p $DEPLOY_PASSWORD
    - docker build -t repo.com/ralph2:latest .
    - docker push repo.com/ralph2:latest

build_with_tag:
  stage: build
  script:
    - docker login repo.com -u $DEPLOY_USER -p $DEPLOY_PASSWORD
    - docker build -t repo.com/ralph2:$CI_COMMIT_TAG .
    - docker push repo.com/ralph2:$CI_COMMIT_TAG
  rules:
    - if: '$CI_COMMIT_TAG'

rules:
  - if: '$CI_COMMIT_TAG'

DEV:
  image: curlimages/curl:latest
  stage: deploy
  script:
    - /bin/sh ./rundeck-api.sh ralph2 "deploy dev" "${VERSION_DEV}" "${RUNDECK_API_TOKEN}"
  rules:
    - when: manual

DEMO:
  image: curlimages/curl:latest
  stage: deploy
  script:
    - /bin/sh ./rundeck-api.sh ralph2 "deploy demo" "${VERSION_DEMO}" "${RUNDECK_API_TOKEN}"
  rules:
    - when: manual

PROD:
  image: curlimages/curl:latest
  stage: deploy
  script:
    - /bin/sh ./rundeck-api.sh ralph2 "deploy prod" "${VERSION_PROD}" "${RUNDECK_API_TOKEN}"
  rules:
    - when: manual
```

FIGURE 4.14 – Pipeline d'intégration et de déploiement continu d'un micro-service d'API REST

Cette pipeline GitLab suit plusieurs étapes pour tester, construire et déployer une application :

1. Configuration globale :

- L'image Docker `docker:19.03.12` est utilisée comme image de base.

- Différentes variables sont définies, comme les informations d'authentification Docker, le token SonarQube, l'URL du serveur SonarQube, la profondeur du git, et des options Maven.

2. Stages :

- Plusieurs étapes sont définies pour la pipeline : `test`, `analyze`, `build latest`, `build_with_tag`, `DEV`, `DEMO`, et `PROD`.

3. Étape de test (`test`) :

- Utilisation de l'image Maven pour exécuter les tests.
- Nettoyage du projet Maven.
- Exécution des tests avec JaCoCo pour mesurer la couverture.
- Exécution de SonarQube pour l'analyse de code statique avec le token et l'URL fournis.

4. Étapes de construction d'images Docker (`build latest` et `build_with_tag`) :

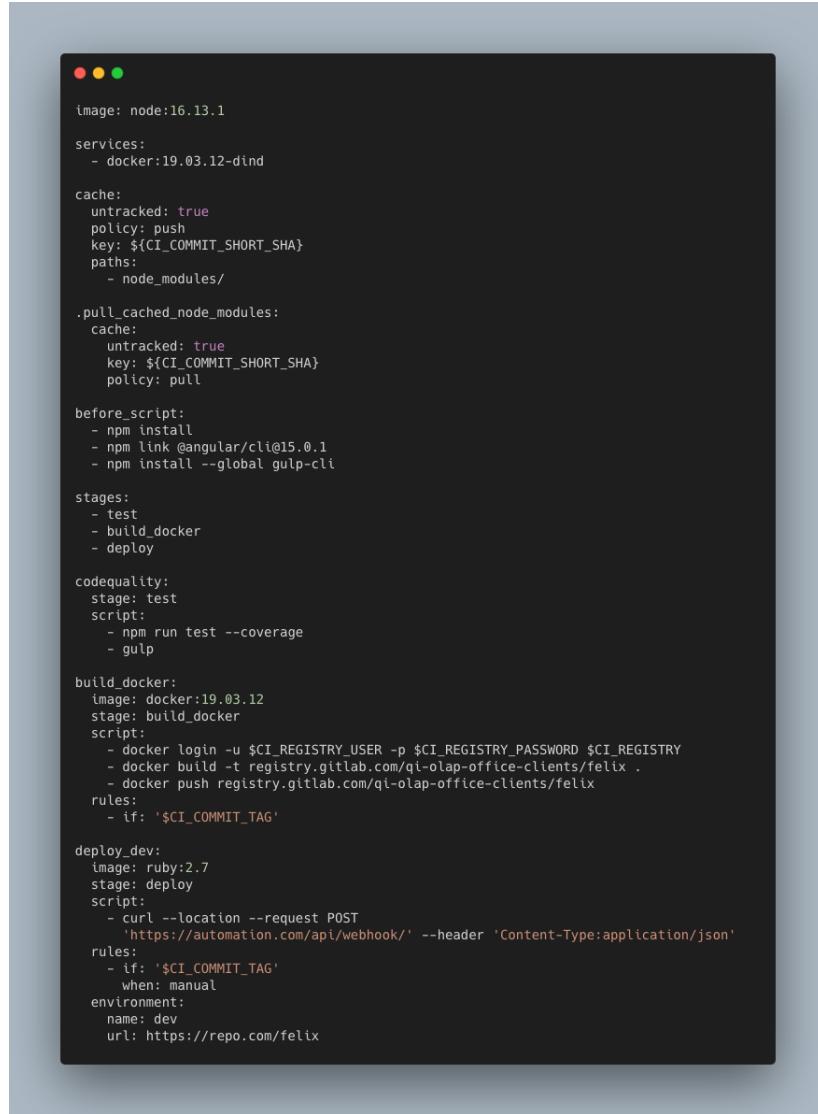
- Ces étapes construisent et poussent des images Docker vers un registre.
- L'étape `build latest` construit une image avec l'étiquette `latest`.
- L'étape `build_with_tag` construit une image avec l'étiquette égale au tag du commit.
- La règle conditionnelle permet à `build_with_tag` de s'exécuter uniquement si un tag de commit est présent (`if: '$CI_COMMIT_TAG'`).

5. **Étapes de déploiement (DEV, DEMO, PROD) :

- Utilisation d'une image Curl pour effectuer des appels HTTP vers Rundeck, notre outil d'automatisation de déploiement (pour mettre à jour les images Docker et les déployer sur notre infrastructure).
- Chaque étape correspond à un environnement de déploiement.
- Les scripts d'exécution de déploiement sont déclenchés manuellement (`when: manual`) et appellent l'API Rundeck pour déclencher le déploiement vers chaque environnement avec les versions appropriées.

Cette pipeline réalise des tests, construit des images Docker, et effectue des déploiements vers différents environnements en utilisant des scripts et des appels d'API spécifiques.

Pipeline d'un microservice du serveur pour le client web



```
image: node:16.13.1

services:
  - docker:19.03.12-dind

cache:
  untracked: true
  policy: push
  key: ${CI_COMMIT_SHORT_SHA}
  paths:
    - node_modules/

.pull_cached_node_modules:
  cache:
    untracked: true
    key: ${CI_COMMIT_SHORT_SHA}
    policy: pull

before_script:
  - npm install
  - npm link @angular/cli@15.0.1
  - npm install --global gulp-cli

stages:
  - test
  - build_docker
  - deploy

codequality:
  stage: test
  script:
    - npm run test --coverage
    - gulp

build_docker:
  image: docker:19.03.12
  stage: build_docker
  script:
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $CI_REGISTRY
    - docker build -t registry.gitlab.com/ql-olap-office-clients/felix .
    - docker push registry.gitlab.com/ql-olap-office-clients/felix
  rules:
    - if: '$CI_COMMIT_TAG'

deploy_dev:
  image: ruby:2.7
  stage: deploy
  script:
    - curl --location --request POST
      'https://automation.com/api/webhook/' --header 'Content-Type:application/json'
  rules:
    - if: '$CI_COMMIT_TAG'
      when: manual
  environment:
    name: dev
    url: https://repo.com/felix
```

FIGURE 4.15 – Pipeline d'intégration et de déploiement continu du microservice de l'interface web

Cette pipeline GitLab comporte les étapes suivantes pour la construction, les tests, la construction d'images Docker et le déploiement d'une application :

1. Configuration de l'image et des services :

- L'image de base utilisée est `node:16.13.1`.
- Les services comprennent Docker (`docker:19.03.12-dind`), pour construire des images Docker dans le pipeline.

2. Configuration du cache :

- Le cache est configuré pour des fichiers non suivis (`untracked: true`).
- La politique de cache est définie pour le push et le pull.
- La clé du cache est basée sur le hachage court du commit (`CI_COMMIT_SHORT_SHA`).
- Les dossiers `node_modules/` et d'autres dépendances sont inclus dans le cache.

3. Étapes du pipeline :

- `codequality` (étape de test) :
 - L'application est testée et sa couverture est mesurée.
- `build_docker` (étape de construction d'image Docker) :
 - Une image Docker est construite à partir du dossier de l'application.
 - Authentification Docker est effectuée via les informations du registre GitLab.
 - L'image est ensuite poussée vers le registre GitLab.

4. Règles de déploiement conditionnel :

- L'étape `build_docker` est exécutée seulement si la condition `$CI_COMMIT_TAG` est vraie (lorsqu'un tag est ajouté au commit).

5. Déploiement vers l'environnement de développement (dev) :

- `deploy_dev` (étape de déploiement) :
 - Une image Ruby est utilisée pour cette étape.
 - Une requête POST est envoyée à l'URL du webhook pour déclencher le déploiement automatisé.
 - Cette étape est déclenchée manuellement (`when: manual`).
 - Les détails de l'environnement de développement sont configurés, y compris l'URL de déploiement.

Cette pipeline accomplit des tâches allant de la construction à la distribution automatisée de l'application, en passant par les tests et la création d'images Docker, tout en étant flexible grâce à des conditions et des déclencheurs spécifiques.

4.9.3 Contexte de réalisation

Cette activité, hormis les outils et leurs utilisations, n'a que peu de changement à venir dans son fonctionnement. Les améliorations disponibles seront une meilleure interface, sûrement à travers une page web, sans avoir à écrire un fichier au format YAML.

4.9.4 Conclusion sur l'activité

En concluant sur cette activité, j'ai pu utiliser GitLab CI/CD dans un cadre d'intégration continue. Cette solution a grandement facilité la gestion et l'automatisation de des tâches lors du développement des microservices. La simplicité d'utilisation de GitLab CI/CD m'a permis de mettre en place les pipelines d'intégration continue, sans difficulté particulière.

4.10 Activité 9 - validation fonctionnelle et rédaction de la documentation pour les utilisateurs

4.10.1 Compétence et son fondement

Bloc de compétences : A5 – Développement d'une solution applicative spécifique et métier selon le projet de développement S.I.

Compétence choisie : A5C7 – Vérifier la conformité entre la solution développée ou paramétrée et les fonctionnalités attendues à partir des retours des directions métiers afin de rédiger la documentation et les référentiels orientés utilisateurs

Détails : Cette compétence consiste à démontrer ma capacité à rédiger une documentation utilisateur :

- comparer la réalisation technique et le cahier des charges
- détailler avec une bonne qualité rédactionnelle la documentation utilisateur

4.10.2 Présentation et réalisation de l'activité

J'ai utilisé GitLab pour comparer la réalisation technique avec le cahier des charges en suivant ces étapes. Tout d'abord, j'ai créé des "issues" pour chaque fonctionnalité décrite dans le cahier des charges. Ensuite, j'ai lié ces issues aux "merge requests" correspondants dans lesquels les modifications techniques ont été apportées. Pendant le processus de développement, j'ai régulièrement mis à jour les états des issues pour refléter l'avancement. Une fois les développements terminés, j'ai procédé à des revues de code et de tests avec les membres de l'équipe pour s'assurer de la conformité de la réalisation avec les exigences du cahier des charges. Enfin, j'ai utilisé les fonctionnalités de suivi de GitLab, comme les tableaux Kanban et les rapports d'avancement, pour visualiser les progrès et m'assurer que chaque issue était liée à la réalisation technique correspondante.

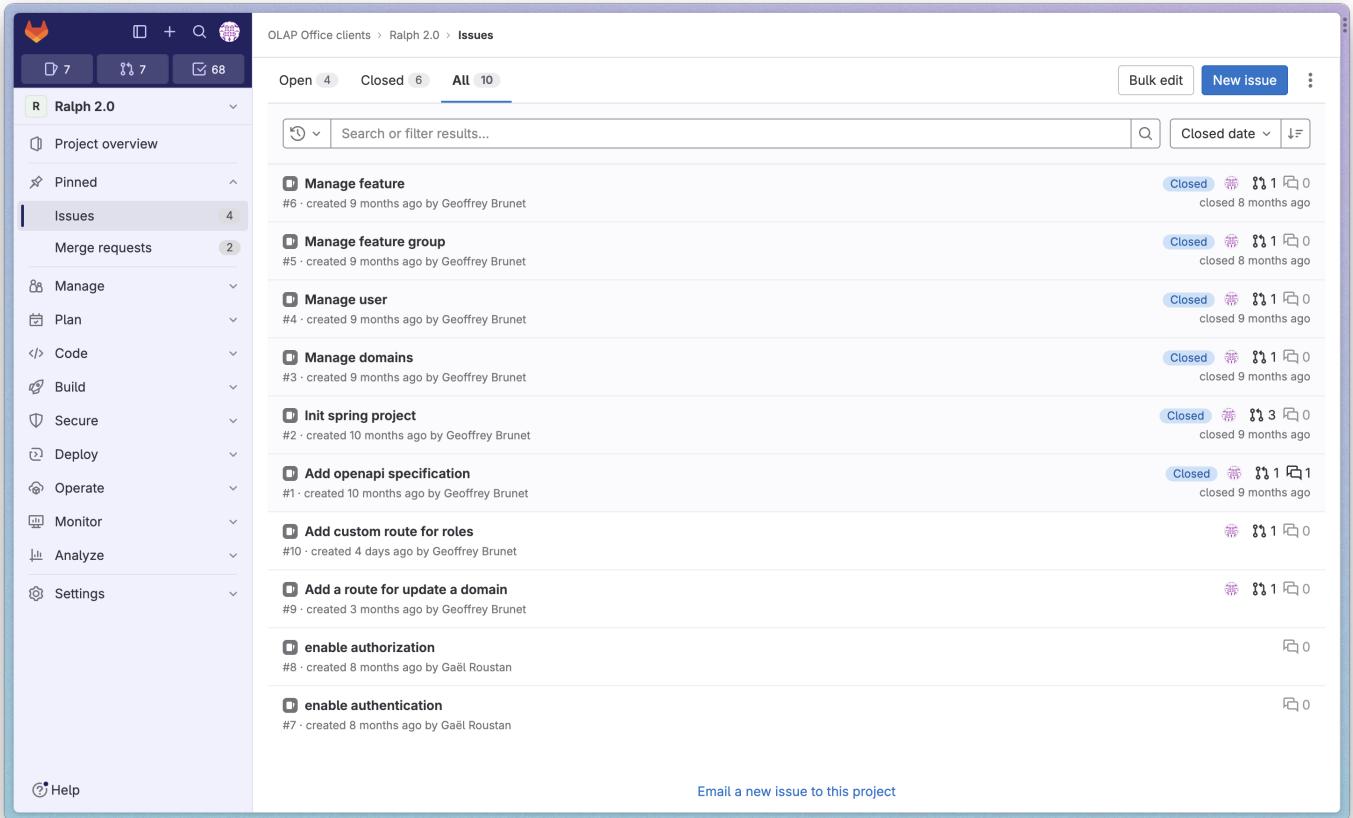


FIGURE 4.16 – Suivi des étapes clefs du développement sur Gitlab

J'ai utilisé Docusaurus pour créer la documentation utilisateur. J'ai suivi ces étapes :

- 1. Installation et Configuration** : J'ai installé Docusaurus et configuré le projet.
- 2. Organisation du Contenu** : J'ai structuré la documentation en sections basées sur les fonctionnalités, et les APIs.
- 3. Rédaction en Markdown** : J'ai rédigé le contenu dans les fichiers Markdown Extended, en incluant des images et des liens.
- 4. Validation Utilisateur** : J'ai sollicité les retours des membres du pôle RD pour m'assurer de la clarté et de l'utilité du contenu.
- 5. Publication en Ligne** : J'ai publié la documentation en ligne pour offrir un accès facile aux utilisateurs.

J'ai utilisé docusaurus qui permet la création d'une documentation simplement et accessible pour les utilisateurs.

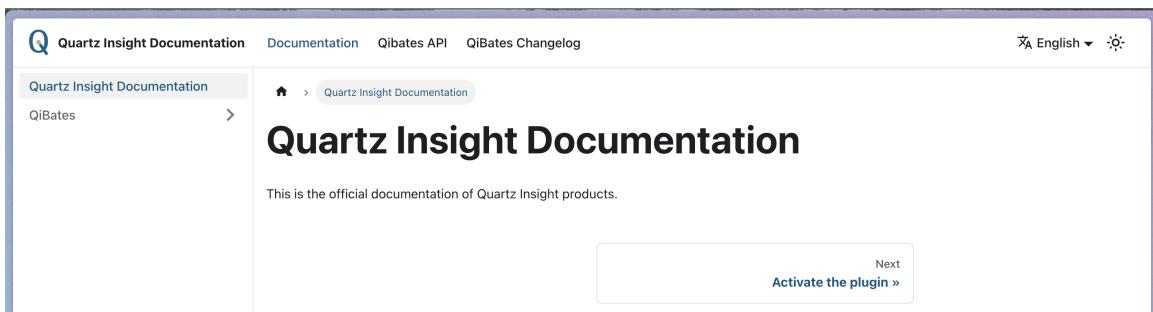


FIGURE 4.17 – Documentation utilisateur en ligne

4.10.3 Contexte de réalisation

J'ai donc utilisé Gitlab pour la gestion et le suivi de projet, et j'ai utilisé Docusaurus pour réaliser la documentation utilisateur et des APIs. Ces deux technologies sont open-source, et pour Gitlab nous utilisons la version hébergée mais gratuite.

4.10.4 Conclusion sur l'activité

La compétence liée à la validation fonctionnelle et à la rédaction de la documentation a évolué ces dix dernières années en raison de l'apparition de nouveaux outils de documentation et de collaboration en ligne.

J'ai acquis des compétences en comparant la réalisation technique avec le cahier des charges grâce à l'utilisation de GitLab pour suivre les changements et les commentaires, sans rencontrer de grande difficulté, et en ayant été force de proposition pour l'utilisation de Docusaurus.

4.11 Activité 10 - formation des utilisateurs, mise en place d'une enquête de satisfaction ultérieure

4.11.1 Compétence et son fondement

Bloc de compétences : A5 – Développement d'une solution applicative spécifique et métier selon le projet de développement S.I. **Compétence choisie :** A5C8 - Conduire le changement auprès des métiers lors du déploiement d'une solution applicative ou intégrée en mettant en place une démarche de participation, de communication et de formation pour accompagner les utilisateurs à l'intégration du nouvel outil dans leurs habitudes de travail

Détails : Cette compétence consiste à démontrer ma capacité à proposer un plan d'action de conduite de changement :

- identifier les actions à mener sur les 4 axes (Informer, communiquer, former, faire participer) :

- présenter différents outils de changement (FutureWheel, Modèle transactionnel de William Bridge,...)
- mettre en application son plan de conduite de changement via un outil

4.11.2 Présentation et réalisation de l'activité

Actions sur 4 axes

Pour réussir la conduite du changement dans mon projet, j'ai mis en place des actions spécifiques sur les quatre axes clés :

- 1. Informer** : J'ai élaboré des communications claires et concises pour informer les utilisateurs (actuellement les membres du pôle R&D) des raisons du changement, de ses avantages et des étapes à venir. J'ai utilisé des emails, des messages sur Microsoft Teams et des réunions pour diffuser ces informations.
- 2. Communiquer** : J'ai créé une communication continue en mettant en place des canaux de discussion tels que des groupes de discussion en ligne ou des sessions de questions-réponses, toujours avec Microsoft Teams.
- 3. Former** : J'ai organisé des sessions de formation adaptées aux besoins des utilisateurs. Ces sessions ont couvert les fonctionnalités clés de la nouvelle solution, avec des démonstrations pratiques et des exercices. Cela a renforcé leur compréhension et leur confiance dans l'utilisation du nouvel outil.
- 4. Faire Participer** : J'ai encouragé la participation active des utilisateurs tout au long du processus. J'ai sollicité leurs commentaires lors des phases de test, en tenant compte de leurs suggestions pour améliorer l'expérience utilisateur. Leur implication a contribué à créer un sentiment d'appartenance au projet.

Présentation des outils de changement

Lors de la conduite du changement, j'ai exploré divers outils pour faciliter la transition en douceur vers la nouvelle solution. Deux de ces outils sont :

- 1. FutureWheel** : Le FutureWheel est un modèle visuel qui permet aux individus et aux équipes de visualiser leurs objectifs futurs. J'ai utilisé cet outil pour aider les utilisateurs à envisager les avantages et les opportunités que la nouvelle solution apporterait à leurs activités quotidiennes. Cela a permis de créer un sentiment positif et d'anticiper les résultats positifs du changement.
- 2. Modèle transactionnel de William Bridge** : Ce modèle propose une approche pour gérer les émotions et les résistances liées au changement. J'ai appliqué ce modèle pour aider les utilisateurs à passer par les différentes étapes émotionnelles du changement, telles que la phase

de deuil pour ce qui est laissé derrière, la phase de transition où ils se familiarisent avec le nouveau, et enfin la phase d'acceptation.

Application d'un plan de conduite de changement

Dans mon projet, j'ai mis en pratique le Modèle transactionnel de William Bridge en organisant des sessions de sensibilisation et de discussions avec les membres du pôle R&D. J'ai identifié leurs préoccupations, leurs résistances et leurs émotions vis-à-vis du changement. Ensuite, j'ai adapté ma communication en fonction de ces étapes émotionnelles, en fournissant des informations rassurantes et en expliquant les bénéfices concrets de la nouvelle solution. J'ai également encouragé la participation active des utilisateurs en leur offrant des opportunités de poser des questions, de partager leurs idées et de contribuer au processus. En utilisant ce modèle, j'ai pu aider les membres de l'équipe à surmonter leurs appréhensions et à embrasser le changement de manière plus positive et constructive.

4.11.3 Contexte de réalisation

J'ai réalisé ce plan avec les collaborateurs avec des communications synchrones (réunions en visioconférence) et asynchrones (par messages et emails), étant en télétravail la totalité du temps. Les outils utilisés sont donc Microsoft Teams et Outlook, ainsi que le modèle transactionnel de William Bridge comme cité précédemment.

4.11.4 Conclusion sur l'activité

Au cours des dernières années, la conduite du changement a évolué en raison des avancées technologiques et des nouvelles réglementations. Les évolutions à venir pourraient inclure une plus grande utilisation des plateformes en ligne pour la communication et la formation, ainsi qu'une adaptation aux méthodes de travail à distance.

Pour mettre en place le plan de conduite de changement, je me suis aidé des compétences acquises en formation et des ressources fournies par les formateurs.

5 Informations complémentaires

5.1 Gestion de projet

J'ai choisi d'adopter une approche de gestion de projet agile pour réaliser mon projet. J'ai opté pour la méthodologie Scrum, avec des itérations de 2 semaines appelées "sprints".

Lors de la planification de chaque sprint, j'ai utilisé la technique du poker planning pour estimer la complexité des tâches et des problèmes à résoudre. Cette méthode collaborative a impliqué tous les membres de l'équipe dans la définition des difficultés et des délais, me permettant d'avoir un avis extérieur sur la difficulté possible des tâches du projet.

Les réunions matinales quotidiennes ont favorisé la communication et la résolution rapide des problèmes rencontrés. Ces réunions étaient essentielles pour permettre aux équipes de connaître l'avancé du projet.

En fin de sprint, j'ai effectué une rétrospective pour évaluer ce qui a bien fonctionné et ce qui pourrait être amélioré. Cette approche itérative m'a permis d'adapter le projet en cours de route et de répondre aux changements.

5.2 Gestion des coûts

En tant qu'alternant, j'ai pris en compte la gestion des coûts pour mon projet. Les principaux éléments de coûts comprenaient mon salaire en tant qu'alternant, ainsi que les ressources informatiques nécessaires pour le développement. J'ai utilisé des machines virtuelles et des systèmes de gestion de base de données (SGBD) hébergés sur le cloud d'Orange pour déployer et tester mon application. Ces services cloud ont engendré des coûts liés à leur utilisation. J'ai veillé à surveiller et à optimiser ces coûts tout au long du projet en utilisant des outils de suivi et en ajustant les ressources selon les besoins. Cette approche m'a permis de maintenir le projet dans les limites budgétaires définies tout en garantissant la continuité et la qualité du développement.

Les coûts ont donc été divisés en deux catégories :

- mon salaire en tant qu'alternant
- le coût de ma formation à l'EPSI et des formations en interne (comme sur Angular par exemple)
- le coût de l'infrastructure sur le cloud Orange.

6 Conclusion

6.1 Résumé

Ce projet m'a permis d'acquérir une expérience précieuse dans le développement de microservices, en utilisant une combinaison de technologies telles que PostgreSQL, Spring Boot et Angular. La réalisation de ce projet m'a permis de mettre en pratique mes connaissances théoriques acquises au cours de mes études. J'ai dû faire face à des défis techniques, tels que la modélisation de la base de données et la mise en œuvre des fonctionnalités clés de l'application. Ce projet m'a également sensibilisé à l'importance de la planification, de la gestion du temps et de la communication efficace dans un contexte professionnel. J'ai appris à hiérarchiser les tâches, à gérer les priorités et à respecter les délais, tout en maintenant une communication régulière avec mon équipe et en fournissant des mises à jour régulières sur l'avancement du projet.

En conclusion, ce projet de gestion des licences utilisateurs chez Quartz-Insight a été une expérience enrichissante qui m'a permis de consolider mes compétences en développement de microservices et en gestion de projet de bout-en-bout pour un logiciel. Je suis fier d'avoir réussi à réaliser un produit fonctionnel et utile pour les entreprises, tout en respectant les exigences et les contraintes du projet. Je suis reconnaissant envers l'équipe de Quartz-Insight pour leur soutien et leur expertise tout au long du projet.

Le code de mon dossier professionnel peut être trouvé à l'adresse suivante : <https://github.com/GeoffreyBrunet/memoire-i2>.

6.2 Ouverture sur l'avenir

6.2.1 L'entreprise et ses perspectives

Grâce à son expertise approfondie dans le domaine de la BI et de l'EPM, l'entreprise se distingue par sa capacité à proposer des stratégies de gestion efficaces, des modélisations de données précises et des analyses approfondies pour aider les entreprises à prendre des décisions éclairées. Son équipe de consultants hautement qualifiés travaille en étroite collaboration avec les clients pour comprendre leurs besoins spécifiques et résoudre leurs problèmes, voir être pro-

actifs par rapport à ceux-ci.

Les perspectives de Quartz-Insight en tant que PME sont prometteuses. Avec son approche personnalisée, son agilité et son engagement continu envers l'innovation, l'entreprise est bien positionnée pour étendre sa clientèle et se démarquer sur le marché de la BI et de l'EPM. La taille réduite de l'entreprise lui permet également de maintenir une culture d'entreprise forte et un haut niveau de service client, ce qui contribue à sa réputation positive.

6.2.2 Le service et ses évolutions

Grâce à la vente de licences en constante augmentation, auprès d'une clientèle de plus en plus variée, la pérennité du service est assurée. Mais de nouvelles contraintes et nouvelles améliorations peuvent être mises en place :

1. **Ajout de nouveaux connecteurs et intégration de nouvelles sources de données** : En permettant l'intégration de nouvelles sources de données, telles que des bases de données supplémentaires, des services cloud ou des API externes, les utilisateurs auront une plus grande flexibilité dans l'exploration de diverses sources.
2. **Amélioration de la supervision** : En intégrant des fonctionnalités de supervision avancées, telles que des tableaux de bord de surveillance en temps réel, des alertes automatisées en cas d'anomalies ou de dépassement de seuils, et des rapports de performance détaillés, les administrateurs et responsables pourront superviser efficacement les performances et l'utilisation de Qibates. Cela facilitera la détection précoce des problèmes, l'optimisation des ressources et la prise de décisions basées sur des données précises.
3. **Élaboration d'un Plan de Reprise d'Activité (PRA)** : Pour assurer la continuité des opérations en cas d'incident majeur ou de catastrophe, il est essentiel de développer un PRA solide pour Qibates. Ce plan devrait inclure des procédures détaillées pour la sauvegarde régulière des données, la restauration du système, la reprise des opérations critiques, la communication avec les parties prenantes, ainsi que des tests réguliers pour s'assurer de l'efficacité du plan. Un PRA bien élaboré garantira une reprise rapide et efficace des activités en cas d'urgence, minimisant ainsi l'impact sur les utilisateurs et l'entreprise dans son ensemble.
4. **Amélioration de la documentation** : En fournissant une documentation détaillée et des ressources de support, la prise en main de Qibates par les utilisateurs sera facilitée.

6.2.3 Les apports professionnels et personnels

Ce projet réalisé au sein de Quartz-Insight lors de mon alternance a été une expérience professionnelle et personnelle enrichissante. Sur le plan profession-

nel, j'ai pu mettre en pratique mes compétences techniques en développement de microservices, en utilisant des technologies telles que PostgreSQL, Spring Boot et Angular. J'ai renforcé ma compréhension des processus de développement logiciel et des bonnes pratiques. Travailler en équipe m'a permis d'améliorer ma communication et ma collaboration. Sur le plan personnel, j'ai gagné en confiance en moi grâce aux succès obtenus et j'ai développé mon autonomie en prenant des initiatives et en résolvant des problèmes de manière indépendante. En résumé, ce projet m'a apporté des compétences techniques solides, renforcé ma confiance en moi, développé mon autonomie et élargi mon réseau professionnel, ce qui constituera des atouts précieux pour ma future carrière en tant qu'ingénieur logiciel.

6.2.4 Avenir professionnel

Pour mon avenir professionnel, je suis passionné par l'optimisation des performances en tant qu'ingénieur en logiciel. Je suis attiré par le développement frontend avec des langages et libraries tels que TypeScript, React ou Svelte, ainsi que le développement backend avec Rust. Mon objectif est de créer des applications web rapides et performantes en réduisant les temps de réponse et en optimisant les requêtes. J'explore également l'utilisation de Rust compilé en WebAssembly pour exécuter du code hautement performant dans les navigateurs web. Cela me permet de développer des applications web puissantes et réactives, offrant une expérience utilisateur améliorée. En résumé, mon ambition est de contribuer à des projets stimulants axés sur l'amélioration des performances. Je souhaite devenir un ingénieur en logiciel polyvalent, capable de créer des solutions performantes en utilisant React ou Svelte, TypeScript, Rust et WebAssembly, afin de garantir des expériences utilisateur exceptionnelles et d'optimiser l'efficacité des systèmes.

Annexes

1. Définition d'EPM (Entreprise Performance Management)

Processus de gestion qui intègre la planification stratégique, la budgétisation, la prévision et la gestion des performances d'une entreprise pour atteindre ses objectifs.

2. Définition de BI (Business Intelligence)

Processus de collecte, d'analyse et de présentation d'informations pertinentes pour prendre des décisions éclairées dans une entreprise.

3. Définition d'API REST

Interface qui permet aux applications informatiques de communiquer entre elles via des requêtes HTTP en suivant les principes de l'architecture REST.

4. Définition de Service monolithique

Application logicielle regroupant toutes ses fonctionnalités en un seul bloc, sans division en composants distincts.

5. Définition de Microservices

Architecture logicielle où une application est décomposée en petits services indépendants, chacun exécutant une fonction spécifique.

6. Définition de Framework

Ensemble de composants et de conventions prédéfinis facilitant le développement d'applications en fournissant une structure et des fonctionnalités réutilisables.

7. Définition de SGBD

Logiciel permettant de stocker, organiser et manipuler des données de manière structurée.

8. Définition d'ASP.NET

Framework de développement web de Microsoft permettant de créer des applications web dynamiques et interactives.

9. Définition d'Angular

Plateforme de développement front-end open-source développée par Google, permettant de créer des applications web interactives et réactives.

10. Définition de Spring Boot

Framework open-source basé sur Java qui facilite le développement rapide et la création d'applications Java autonomes, en simplifiant la configuration et la gestion des composants.

11. Définition de PostgreSQL

Système de gestion de base de données open-source puissant et performant, basé sur le modèle relationnel, capable de gérer de grandes quantités de données et offrant une large gamme de fonctionnalités avancées.

12. Définition d'ORM (Object-Relational Mapping)

Technique de programmation qui permet de faire correspondre les objets d'une application avec les enregistrements d'une base de données relationnelle de manière transparente.