MÉMOIRE TECHNIQUE - DEVELOPPEUR

BANQUE DE FRANCE – DGSI DISCO

MESLIN THIBAUT
CESI - ROUEN
115 rue Réaumur – 75002 - PARIS

I. Remerciements

Je voudrais avant tout adresser tous mes remerciements à **M. Antoine MEHEUT**, responsable du pôle EXA & PO DevSecOps ECLAIR, pour m'avoir accueilli dans son pôle et de m'avoir permis de mettre en application mes compétences et d'en acquérir de nouvelles.

J'aimerais également remercier **M. Sébastien LE SAUSSE**, chef de projet de la ligne de développement JAVA, et son équipe pour leur professionnalisme, leurs conseils, leurs disponibilités et surtout le partage de leurs expériences.

Je saisis également cette occasion pour adresses mes remerciements à mon école et plus particulièrement ma tutrice **Mme. Sara ZACHARIE** pour sa réactivité pour compléter les documents du stage afin que je puisse débuter mon stage aux dates convenues.

Un immense merci à ma famille pour son aide et son soutien lors de la recherche de ce stage ainsi que pour son soutien moral et économique.

Table des matières

II. Entreprise

I. Présentation de l'entreprise

La Banque de France est une institution indépendante régie par le droit public français et européen, membre de l'Eurosystème (organe de l'UE chargé de définir et de mettre en œuvre la politique monétaire unique de tous les pays ayant adopté l'euro). L'entreprise a été créée en 1800, dès lors intimement liée à l'histoire de la nation et s'impose rapidement comme banque centrale et propose des services aux particuliers, mais aussi à l'État.

La Banque de France possède son réseau de succursales construit à partir du XIXe siècle, lui permettant d'assurer des missions dont la permanence est aujourd'hui avérée au sein du système européen de banques centrales.

L'entreprise possède le numéro de SIREN 572104891. Elle possède un effectif de 10 000 employés et plus selon la tranche INSEE et détient un capital social de 1 milliard d'euros.

Celle-ci possède trois missions principales étant la stratégie monétaire, la stabilité financière et le service économique à la collectivité.

- En matière de stratégie monétaire, son rôle est crucial puisque ses équipes préparent et mettent en œuvre les décisions prises par le Conseil des gouverneurs. De même, elle est gardienne de la monnaie et vise à la stabilité des prix et également la confiance dans l'Euro.
- En matière de stabilité financière, celle-ci doit assurer protection et surveillance des établissements financiers.
- Enfin, au niveau de services à l'économie, celle-ci s'adresse à différents types d'acteurs pour différents services.
    - Pour les ménages et entreprises, elle doit permettre d'établir des services concrets pour les ménages les plus fragiles et de couvrir les TPE/PME pour les cotations d'entreprise et médiation de crédits, tout en assurant l'accompagnement de celles-ci.
    - Pour l'État, elle doit gérer les comptes du Trésor public, l'adjudication de titres publics, la tenue de comptes courants de bons du Trésor, l'élaboration de la balance des paiements et d'autres activités sous conventions de l'État.

La Banque de France propose également différentes offres complémentaires et indépendantes de ses missions :

- Médiation de crédit
- Accompagnement des TPE/PME
- Éducation financière des entrepreneurs
- Accompagnement des start-ups
- Offre OPALE
- Accompagnement GEODE
- Offre ACSEL

Site de la Banque de France : Banque de France (banque-france.fr)

II. Présentation du service

Je suis intégré à la **Direction Générale des Services d'Informations** (**DGSI)** et plus précisément au pôle **EXpertise Applicative** (**EXA)**. Ce pôle à la responsabilité de définir les lignes de développement de la Banque de France, les normes, les bonnes pratiques et surtout de fournir un support à tous les utilisateurs de ces lignes de développement. Chacune de ces lignes est dirigée par un référent, lui-même responsable de sa ligne. On retrouve au total cinq lignes de développement, chacune autour d'un langage de programmation propre à la ligne :

- La ligne de dev BACK Java avec comme référent **Sébastien LE SAUSSE**
- La ligne de dev Python avec comme référent **Jean-Baptiste RENAULT**
- La ligne de dev Microsoft .NET avec comme référent **Sébastien LE SAUSSE**
- La ligne de dev Angular, IHM, UX/UI, RGPD, RGAA avec comme référente **Marie BITSCHENE**
- La ligne IAC (Infrastructure As Code) avec comme référent **Aymeric BLONDET**

Je suis personnellement membre de la ligne BACK Java. Il est à noter que cette équipe comporte du personnel basé à Paris et à Nantes.

L'équipe fonctionne en méthode AGILE avec des réunions tous les mardis et jeudis vers 11 h afin de faire le point sur les avancées de chacun et les éventuels problèmes ou difficultés rencontrés. C'était également pour moi le moment de découvrir l'équipe et d'échanger avec les différentes personnes. Cette équipe ne s'occupait pas uniquement de développer, mais également de fournir du support aux utilisateurs via un système de tickets.

La procédure de développement sera détaillée plus en détails et inscrit dans le contexte de ma mission dans la partie Mission (Logigramme).

III. Sujet & Objectifs

I. Contexte

Avant de présenter le sujet et les objectifs, il est essentiel d'établir le contexte dans lequel j'ai effectué mon stage. J'ai été amené à travailler sur une application métier web intitulée AUDITOR. Cette application a pour but de recenser tous les projets, releases, dépendances, etc. dans le but de permettre de connaître l'avancement de ces projets. Cette application possède une partie front, une partie back et une base de données. Cette application est utilisée notamment par les PO (Product Owner) et les référents des lignes de développements.

Les différentes parties de l'application utilisent des technologies différentes :

- Front : Angular
- Back : Java
- Base de données : PostgreSQL

J'ai personnellement été amené à intervenir à tous les niveaux de cette application (Normalement, je n'aurai dû intervenir que pour la partie frontend). J'ai donc été amené à manipuler un grand nombre de technologies avec lesquelles je n'avais en aucun cas eu d'interactions avant et donc ne connaissait pas celles-ci ou seulement de nom ou dans la globalité.

II. Sujet

Le sujet de ce mémoire technique est le suivant : Développement d'une fonctionnalité d'export CSV pour l'application AUDITOR. Ce sujet inclus, comme dit précédemment, du développement en langage Angular pour la partie frontend et Java pour la partie backend.

Ce sujet correspond à un ticket adressé à l'équipe EXA via JIRA, un outil de gestion de bugs, de gestion des incidents et de gestion de projets développé par Atlassian. Dans le ticket se trouve une User story correspondant à l'expression du besoin utilisateur et donc de la tâche à réaliser pour résoudre cette story.

La mission se résume à créer et à mettre en place un moyen de générer un fichier CSV contenant toutes les informations demandées par le créateur de la story et à autoriser le téléchargement de celui-ci via un bouton ou autre. Le tout doit bien sûr être sécurisé, maintenable et compréhensible par toute autre personne.

III. Objectifs

L'objectif principal de cette mission est de permettre rapidement et aisément aux personnes consultant l'application de pouvoir exporter les données d'un ou tous les projets d'un simple clic, sans avoir à recopier manuellement et/ou en changeant d'onglet. Cela peut ainsi permettre de faire passer ce fichier par mail afin de transmettre ces informations à d'autres personnes.

Un autre objectif est de pouvoir conserver des traces des versions et autres de ces projets afin de pouvoir concevoir une sorte d'historique. Le fait de garder ces fichiers et les données contenues pourront notamment être utilisé à des fins statistiques pour par exemple évaluer le coût de maintenabilité d'un projet ou alors son coût médian. De multiples usages statistiques peuvent être réalisé grâce à ces données et à cet « historique de version ».

IV. Mission

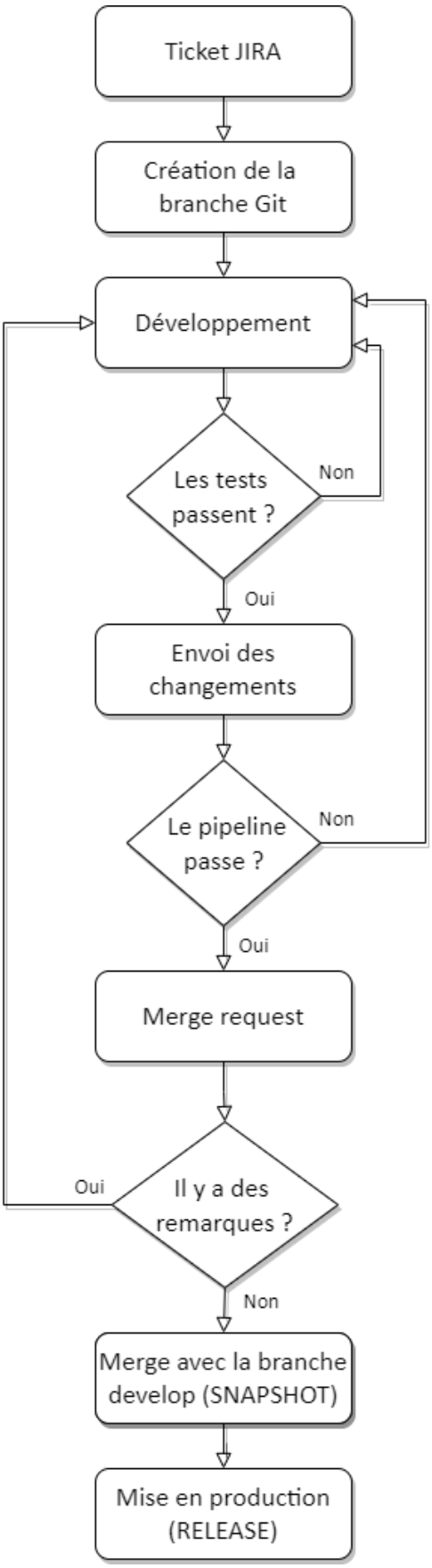
I. Logigramme

Le logigramme ci-contre représente toutes les étapes par lesquelles passent les développeurs de l'entreprise pour résoudre un ticket. C'est une procédure standard qui permet de répondre à un ticket JIRA et qui suffis dans la grande majorité des cas.

Il est néanmoins important de préciser qu'il y a également en place des normes de nommages pour tout ce qui concerne Git. Les normes sont les suivantes :

- Branche : feature/*nomDuTicket*
- Merge request : *nomDuTicket*

Dans le cas d'une merge request est créée entre le développement et les tests ou qu'il reste des modifications à faire on rajoute « WIP » pour *Work In Progress*. On signifie au propriétaire que la branche n'est pas prête à être fusionnée.



II. Réalisation de la mission

I. Généralités

Comme dit auparavant, la partie backend du projet est développée en langage Java. De nombreuses technologies ont été utilisées comme Spring boot ou encore Maven. Le projet inclus de nombreuses dépendances pouvant en conséquence chacune provoquer une faille de sécurité, il faut donc être prudent et essayer d'utiliser des dépendances sûres ou de ne pas avoir de dépendances inutiles. C'est donc là que l'utilisation du référentiel OWASP (Open Web Application Security Project) devient crucial au niveau de la sécurité applicative. On retrouve dans ce référentiel les dernières vulnérabilités qui ont été découvertes par la fondation et qui permettent donc de mettre à jour nos applications afin d'assurer un niveau de sécurité élevé et de se prémunir contre d'éventuelles attaques.

On retrouve également SonarQube qui est un outil permettant également d'assurer un niveau de sécurité élevé via son analyse, mais également d'assurer un code de qualité. Cet outil permet de s'assurer que le code analysé répond à certaines exigences comme en assurant que celui-ci ne comporte pas de répétitions ou autres problèmes qui pourraient dégrader la qualité globale du code.

Un autre outil que j'ai été amené à utiliser fréquemment est Jenkins. Jenkins est une plateforme proposant un service d'automatisation open-source pour automatiser des tâches redondantes comme faire les builds, déployer les changements, etc. Jenkins est utilisé pour les pipelines de notre service afin de s'assurer que le code est sécurisé et répond à toutes les exigences.

Le dernier outil que j'ai utilisé régulièrement était Sonarlint. C'est un outil permettant également d'assurer la sécurité et la qualité du code, mais qui permet également de s'assurer que le code se conformer à certaines exigences de formatage ou d'écriture. Il a été paramétré pour s'assurer que toutes les modifications apportées aux projets respectent des conventions spécifiques à la Banque de France.

II. Organisation du Git

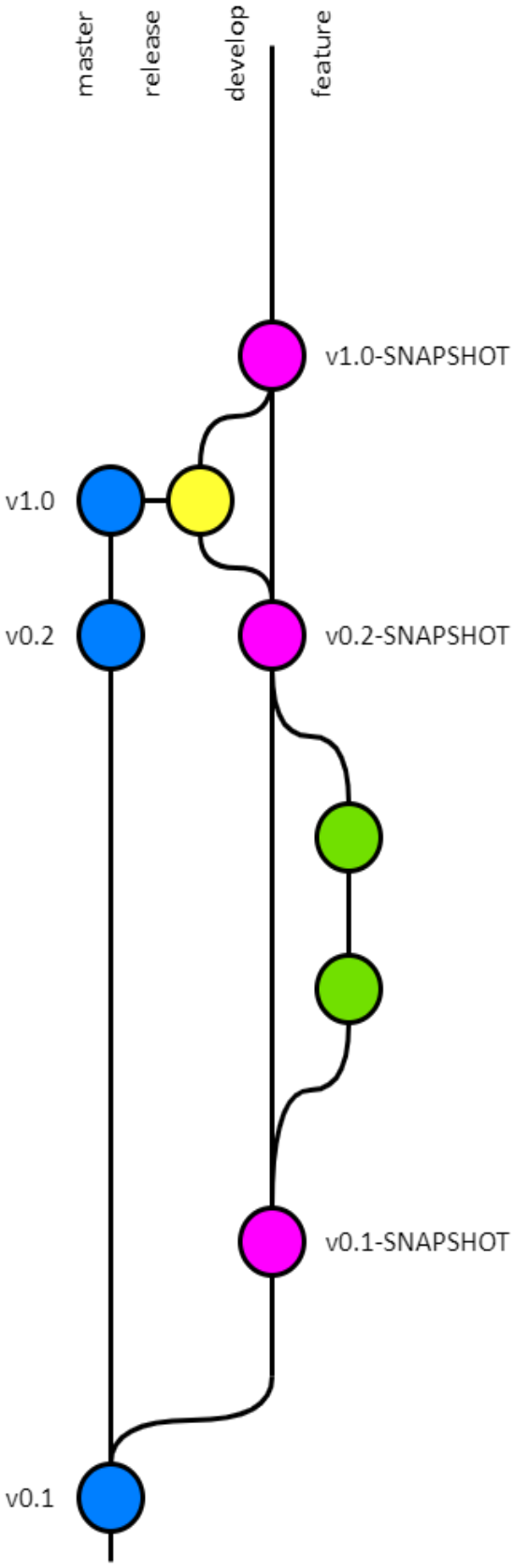
Pour ce projet, l'hébergement de celui-ci était fait sur GitLab. Il y a deux dépôts différents reprenant le même principe d'organisation des branches. Un dépôt pour le projet back et un projet pour le front.

On retrouve la branche d'origine qui est la branche master. Celle-ci est dupliquée dès le début pour obtenir la branche develop. C'est sur celle-ci que va se passer tous le développement, que ce soit les fonctionnalités, les corrections de bugs, failles de sécurité, etc. Sur la branche master on retrouvera les versions qui ont eu une release, c'est-à-dire les versions terminées. C'est également sur la branche release qu'on va retrouver ces informations.

Lorsqu'une fonctionnalité doit être développée et donc a eu son ticket sur JIRA, on crée une nouvelle branche basée sur la branche develop afin de récupérer les dernières modifications. On crée également la merge request qu'on définit en WIP (Work In Progress) le temps du développement de la fonctionnalité.

En général, lors des commits on appose en entête ses initiales en majuscules et entre crochets ([TM] par exemple) suivi du nom du ticket. On saute ensuite une ligne et on décrit le contenu de son commit. Par la suite, on regarde si le pipeline passe. Si oui, on peut enlever le WIP du titre de la merge request, sinon on modifie ce qui est à modifier et on renvoie les modifications sur le dêpot.

Pour avoir plus d'information sur le processus de livraison : Livraison.



III. Backend

a. Développement

Réaliser la partie backend de l'application a été ma première tâche. C'est là que j'ai dû mettre en place la récupération des données nécessaires et leur formatage afin de produire un fichier CSV. Tous ces nouveaux ajouts devaient bien sûr s'intégrer au projet déjà existant que je devais cloner depuis le dépôt Git.

Le projet utilisant Maven, la hiérarchie du projet reprenait celle d'un projet Maven classique. De nombreux ajouts avaient auparavant été faits et le projet est intégralement en langage Java.

Il existe de nombreux contrôleurs dans le projet, un contrôleur correspond à une option dans la barre de navigation d'AUDITOR. Comme ma fonctionnalité se trouve sur la partie projet de l'application, j'ai eu à modifier le contrôleur projet. Il existe également des services qui correspondent à des fonctionnalités plus ou moins récurrentes et utilisées dans un ou plusieurs contrôleurs. Comme la fonctionnalité d'export n'existait alors pas, j'ai dû concevoir un nouveau service que j'ai intitulé « CsvExportService ». Le nom peut être sujet à changement si jamais une nouvelle demande arrive et que le format de sortie change (JSON par exemple).

Dans ce fichier, j'ai conçu deux méthodes permettant d'exporter les informations soit d'un projet unique en fournissant son ID dans la base de données, soit en exportant toutes les informations de tous les projets dans un fichier unique si aucun numéro n'est fourni. Les informations exportées demeurent les mêmes. Le changement majeur représente le fait qu'il faut soit obtenir un seul projet ou alors tous.

Pour avoir accès aux informations sur les projets, j'ai dû faire appel à deux autres services : « ProjectService » et « ReleaseService ». J'ai par la suite utilisé l'ID pour récupérer toutes les données nécessaires et les enregistrer dans des variables. Pendant le développement, j'ai évoqué la possibilité de permettre une possibilité d'évolutivité en récupérant immédiatement les objets. De cette manière, si jamais les informations à exporter évoluent, il y a seulement à faire appel à un autre getter pour récupérer une nouvelle valeur.

Pour créer l'entête du fichier, j'ai dû rajouter une énumération dans le fichier common du projet. Cette énumération contient toutes les colonnes de la base de données qui ont un rapport avec les projets comme l'ID dont on parlait juste au-dessus. L'appel de cette énumération et de la méthode « getColumn » permet de récupérer le nom de la colonne en minuscule.

Il restait à importer la dépendance « Apache commons-csv ». Cette dépendance permet de créer et de lire des fichiers CSV. Grâce à cette dépendance, j'ai pu stocker dans une variable le header avec les colonnes et les données à exporter. La dernière étape est alors de retourner cette variable.

La dernière étape était d'ajouter deux méthodes au contrôleur projet pour pouvoir les appeler depuis la partie frontend grâce à un URL spécifique qui est à définir. Il faut donc une méthode pour l'export d'un projet et une autre pour l'export de tous les projets. Chacune de ces méthodes inclut une requête http permettant de transmettre les informations d'une partie à l'autre. Une autre ligne de code importante était de définir que le contenu de cette requête était du type « text/csv » afin que le navigateur qui recevra les données sache que c'est un fichier CSV.

b. Tests

Afin de tester si la méthode retournait effectivement ce qu'on lui demandait, il fallait procéder à un programme de tests. Les tests d'appel http ne peuvent pas être réalisé sous éclipse alors, j'ai utilisé le logiciel SoapUI qui permet de tester ces appels et d'étudier les réponses, entêtes, etc.

Une des difficultés rencontrées a été le fait que dans l'environnement de développement comme dans l'environnement réel, il y a un système d'authentification nommée Keycloak. Ce logiciel fourni un token lorsque la connexion est effective et ce token possède un lifespan prédéfini. Pour réaliser le test d'une requête, il faut fournir en entête ce token pour accréditer l'accès.

Une fois le tout fait, on peut tester l'appel et vérifier que les informations contenues dans la réponse correspondent à nos attentes ou alors adopter les modifications qui s'imposent. Visualiser les données en « brut » permet de s'assurer que tout est correctement paramétré et fonctionne selon le fonctionnement voulu.

c. Code

```
public void writeAllProjectToCsv(Writer writer) {

    List<ProjectTableDto> projects = projectService.getProjects();

    try (CSVPrinter csvPrinter = new CSVPrinter(writer, CSVFormat.DEFAULT)) {
        csvPrinter.printRecord(ProjectColumn.MATUR_CODE.getColumn(), ProjectColumn.RUNTIME.getColumn(), ProjectColumn

        for (ProjectTableDto project : projects) {
            ReleaseDetailsDto releaseData = releaseService.getReleaseById(project.getId());
            csvPrinter.printRecord(project.getName(), releaseData.getRuntime(), releaseData.getVersion());
        }
    } catch (IOException e) {
        log.error("Unable to use CSVPrinter", e);
    }
}
```

Méthode d'export CSV de masse

IV. Frontend

a. Développement

Réaliser la partie frontend de l'application n'était pas prévu mais j'ai eu la possibilité de la faire après avoir terminé la partie backend. J'ai ainsi enchaîné sur la partie frontend qui est entièrement développé en utilisant le framework Angular et différentes dépendances.

Le projet est découpé selon la hiérarchie classique Angular et au même titre que le backend on retrouve un dossier par élément de la barre de navigation et des sous-dossiers pour les pages qui sont des pages enfants de ces éléments. Il est à noter que chaque sous-page se compose d'un dossier incluant un fichier de style CSS, un fichier HTML et deux fichiers TypeScript, dont un utilisé pour les tests.

Il existe un service par élément de la barre de navigation et celui que j'ai dû modifier correspondait encore une fois au projet donc le « project.service.ts ». C'est dans ce fichier que j'ai ajouté deux méthodes permettant d'appeler le backend et donc de disposer des données récupérées dans la partie frontend. On appelle ainsi via l'URL défini plus tôt afin d'avoir la méthode qui y est associé et de récupérer les données pour les proposer au téléchargement.

La prochaine étape est de modifier la partie component afin d'y inscrire une méthode permettant de télécharger les données récupérées sous forme de fichier CSV. Pour faire cela, on appelle donc la méthode créée dans le paragraphe d'au-dessus et de s'y abonner pour appeler un design pattern Observer et qui permet de notifier l'application dès qu'un changement s'effectue. Enfin, on encode le tout en UTF-8 afin d'avoir tous les caractères encodables et on utilise une méthode importée d'une dépendance appelée « saveAs » afin de nommer le fichier qui va être disponible en téléchargement. On inclut un message dans la console dans le cas où le téléchargement échouerait.

b. Tests

Afin de réaliser les tests unitaires, il faut écrire une méthode de test pour une méthode écrite dans le code. J'ai par la suite dû écrire deux méthodes de test dans le « project.service.spec.ts ». Ces méthodes vont servir à s'assurer que le comportement des méthodes écrites précédemment est correct et que les informations retournées correspondent à un échantillon écrit en « dur ».

Je vais ici décrire le test que j'ai écrit pour la méthode d'export de masse. À la fin de cette méthode, on s'attend à récupérer une chaîne de caractères formatée pour correspondre à un fichier CSV et être reconnu en tant que tel. J'ai également créé une variable contenant des informations sur les projets récupérés et qui reprennent les informations attendues en sorties. Une fois que l'on a exécuté la méthode pour récupérer tous les projets sous forme de CSV, on regarde si ces informations attendues sont présentes dans la chaîne de caractères récupérée. On s'attend également à ce que la requête http soit exécuté une fois seulement et qu'elle ne soit pas arrêté par une action tierce. Si toutes ces informations sont valides alors le test passe et la méthode fonctionne convenablement.

V. Livraison

Lorsque les fonctionnalités qui ont été demandés ont été développées, testées et fonctionne correctement à ce qui est attendu, on va pouvoir livrer le tout afin qu'il soit accessible de tous. Pour cela, on va utiliser une nouvelle fois Jenkins. Dans l'entreprise, le processus de déploiement est entièrement automatisé via des actions Jenkins. Il faut simplement signifier quelques informations comme par exemple le type de livraison que l'on souhaite faire.

Il est possible de faire une RELEASE ou une SNAPSHOT. La différence entre les deux est légère mais très importante car lorsqu'on livre une SNAPSHOT, on livre un environnement de développement qui est par définition instable. C'est pour cela que les SNAPSHOT sont utilisés exclusivement pour la partie développement (donc branche develop) et rarement en production.

Une fois, le build de l'image à livrer lancée, on passe à nouveau par un pipeline qui va vérifier que tout est bon et qu'aucune faille n'est trouvée pour assurer la sécurité de l'application. Si la construction réussie, l'image est envoyée sur Artifactory. On retrouve toutes les versions dans cette application. Par la suite, l'image est déployé sur un environnement cloud propre à la Banque de France appelé ECLAIR.

Enfin, une dernière précision, les images sont hébergées sur des pods Kubernetes. Un pod est un environnement de déploiement le plus petit et compact de Kubernetes. Une fois l'image livrée sur le pod, elle est accessible par tous.

VI. Bilan

Ma mission a été réalisée rapidement et j'ai aisément réussi à faire le backend. Pour le frontend c'était également facile puisque j'en faisais également à côté, mais je n'avais jamais pratiqué Angular. Les technologies utilisées, que ce soit pour le backend ou frontend, sont assez similaires et la hiérarchie relativement similaire permet de ne pas trop se perdre lorsque l'on passe du back au front ou inversement.

Le résultat obtenu correspond aux attentes et est également prêt à être mis en production. Il correspond aux besoins de l'utilisateur et fourni le résultat attendu. La possibilité d'avoir pu réaliser l'entièreté du ticket est extrêmement agréable et favorise une véritable prise de compétence à tous les niveaux. Les interactions avec l'équipe m'ont également permis d'assimiler beaucoup de choses très rapidement. Néanmoins, je me suis rendu compte que j'arrivais à être autonome, mais que je devais aussi savoir demander de l'aide plus rapidement parfois surtout quand j'ai la chance d'être entouré de personnes si compétentes et aimables.

V. Conclusion

Pour terminer, j'aimerai encore une fois remercier toutes les personnes que j'ai pu côtoyer au cours de ce stage et toutes les merveilleuses rencontres que j'ai pu faire au cours de ce stage en entreprise.

J'ai énormément appris pendant ce stage et je souhaiterai même recommencer un projet personnel afin de reprendre les mêmes technologies pour approfondir mon apprentissage et me permettre d'acquérir encore plus de compétences dans le domaine du développement web que j'affectionne beaucoup.

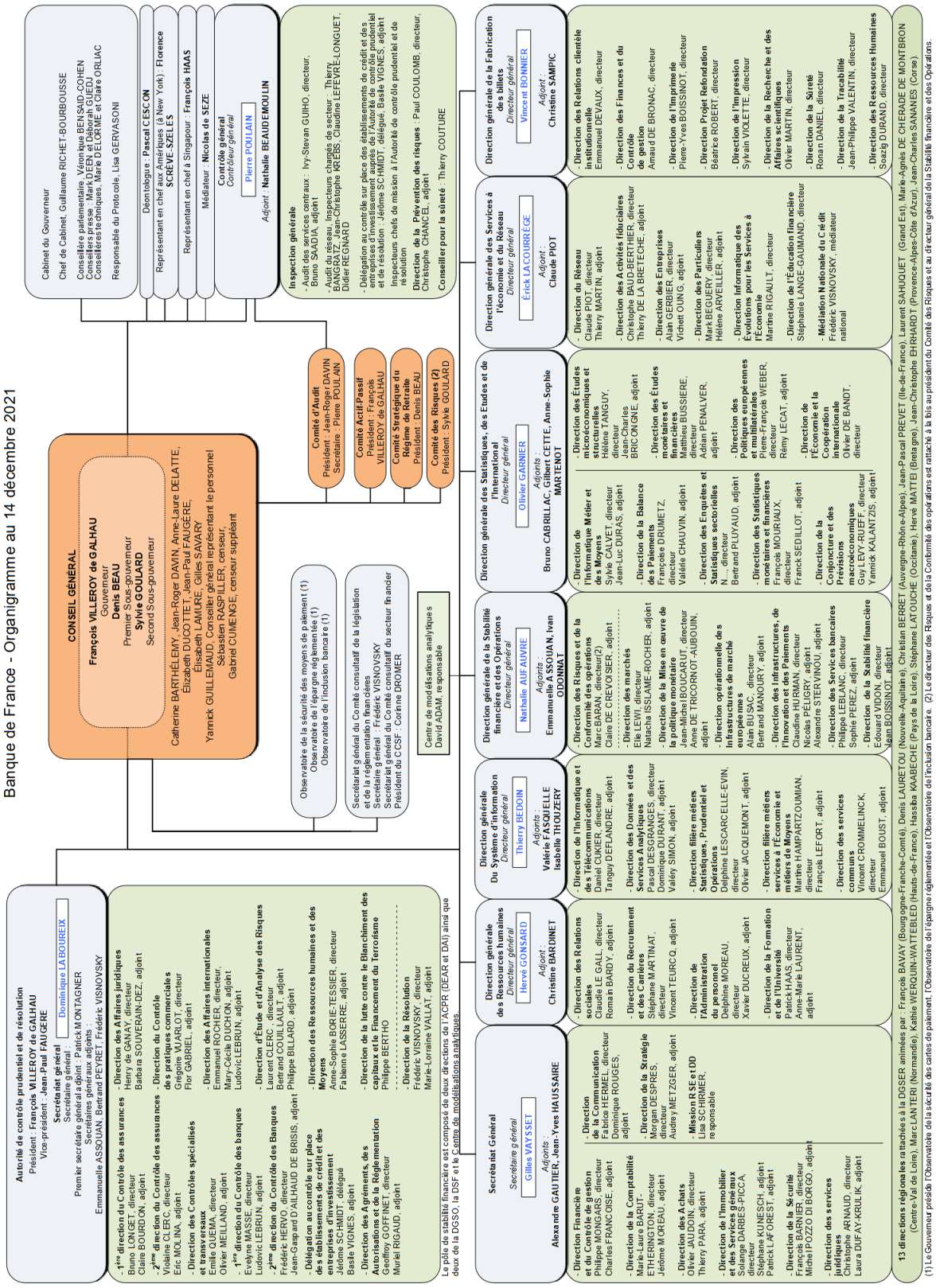
Je n'ai pas eu énormément de grosses difficultés au long de ce stage et c'est surement grâce au fait que l'équipe qui m'entourait et même, plus globalement le service, était très accessible et était prête à aider peu importe le problème et leur niveau de compétences sur le sujet. J'ai énormément appris sur le plan compétences que sur le plan organisationnel d'une grande entreprise et d'une équipe divisée sur plusieurs sites.

C'était également une très bonne expérience au niveau de l'organisation de mon travail. Je gérais mes horaires seul et l'organisation de mon travail dans le temps. Bien sûr, il fallait aller le plus vite possible, mais cela m'a appris à mieux gérer mon temps dans mon travail. De même, j'ai appris à gérer le fait de travailler en « Remote » (3 jours en télétravail, 2 jours sur site). J'étais toujours au moins un jour sur site par semaine et donc cela m'a appris à m'organiser au niveau de mes trajets et de bien gérer mon temps pour arriver à l'heure, etc.

Enfin, j'ai été très heureux de faire mon stage dans une entreprise si grande et intéressante que la Banque de France. J'ai travaillé sur un projet très instructifs et j'ai appris énormément de nouvelles choses et suis monté en compétences rapidement au cours de ces 3 mois et demi. J'ai appris le fonctionnement d'une grande entreprise et suis heureux d'avoir pu participer à cette aventure à la Banque de France.

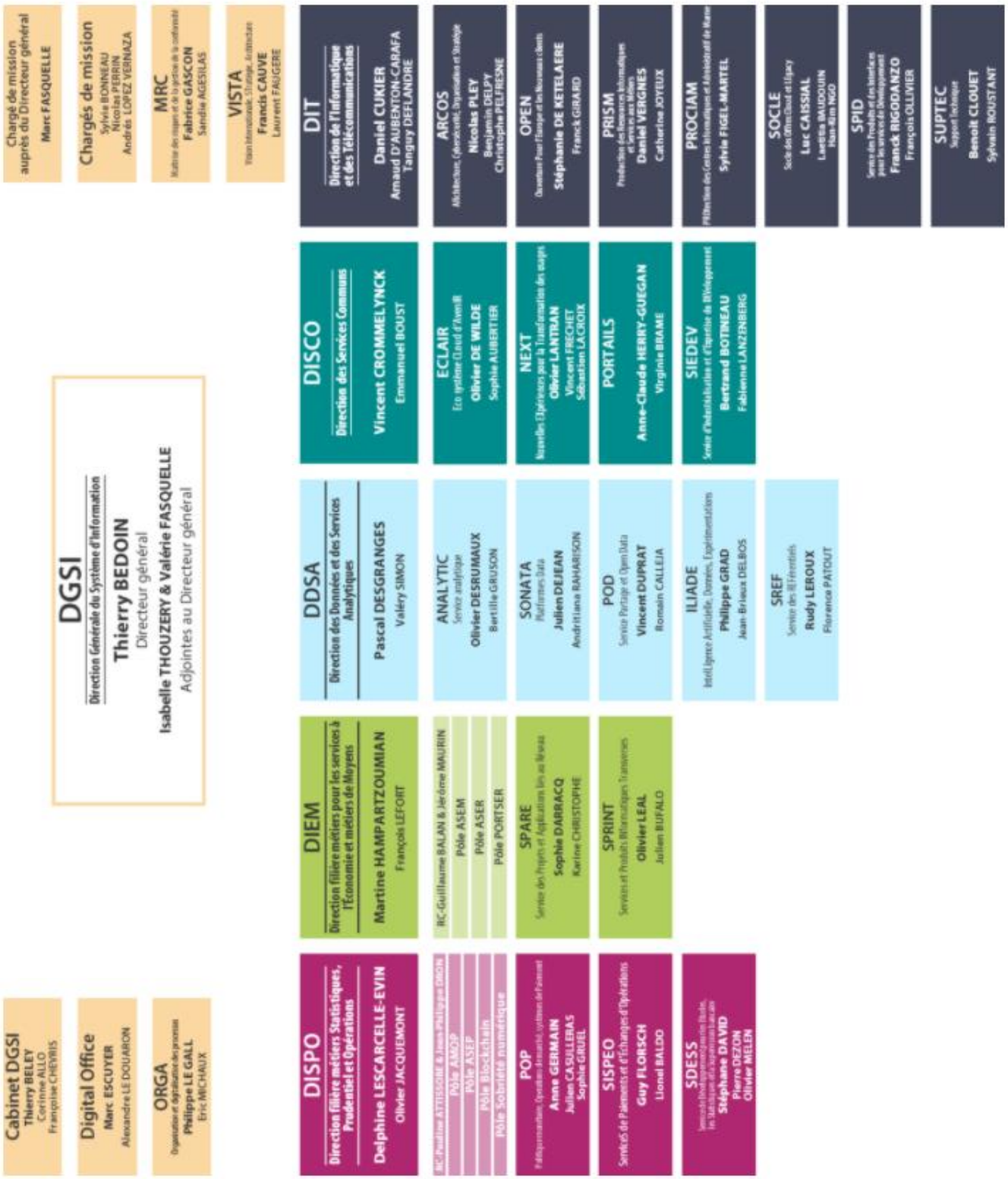
VI. Annexes

I. Organigramme de la Banque de France



BANQUE DE FRANCE
EUROSYSTÈME

II. Organigramme de la DGSI



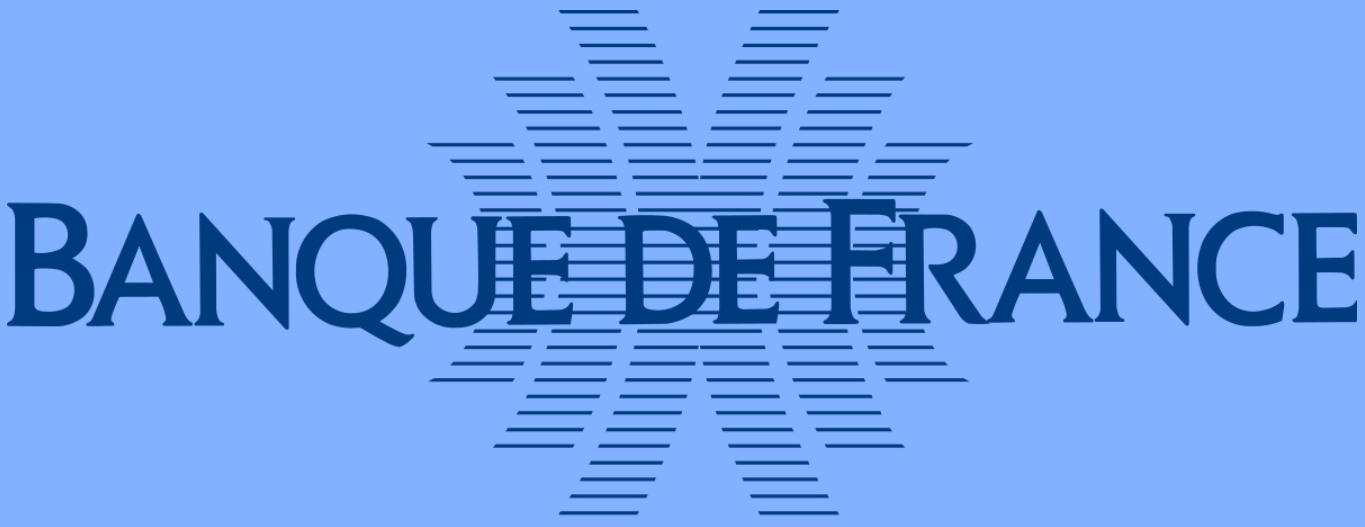
BANQUE DE FRANCE
EUROSYSTÈME

III. Glossaire

Release	(Anglicisme) Révision, version, distribution (nouvelle version d'un logiciel, en particulier)
Front | Frontend	Fait référence à la couche de présentation, l'interface d'un logiciel
Back | Backend	Fait référence à la couche d'accès aux données d'un logiciel, ou l'infrastructure physique ou le matériel
Product Owner	Responsable de la définition et de la conception d'un produit ou d'un service digital
User story	Description simple d'un besoin ou d'une attente exprimée par un utilisateur
Build	Version exécutable d'une partie du système ou du système entier
ID | Identifiant	Numéro unique permettant d'identifier un élément précis
Getter	Méthode accordant l'accès en lecture à un attribut d'objet
Token	Dispositif matériel ou logiciel nécessaire à un utilisateur pour accéder à une application ou à un système réseau de manière plus sécurisée
Lifespan	Durée de vie
Design pattern	Arrangement caractéristique de modules, reconnu comme bonne pratique en réponse à un problème de conception d'un logiciel

IV. Bibliographie

Banque de France	https://www.banque-france.fr/
Draw.io	https://app.diagrams.net/
Wikipédia	https://fr.wikipedia.org/
Scribens	https://www.scribens.fr/
Larousse	https://www.larousse.fr/
Google Image	https://www.google.fr/imghp?hl=fr


BANQUE DE FRANCE
EUROSYSTÈME