



**Rapport d'Alternance**  
**Développeur Fullstack JAVA**  
**PFI – ZEN**

**Lyes LAIMOUCHE**

Alternance Effectué du 02/09/2024 au 02/09/2025

Tuteur Entreprise : Mounir Bouibherne

Enseignant Réfèrent : Jacques Malenfant

Etablissement d'enseignement : Sorbonne Université

Entreprise d'accueil : Société Générale



## Remerciements

Je tiens à remercier toute l'équipe de PFI pour leur accueil chaleureux, leur soutien constant et leur confiance tout au long de ce stage.

Un merci particulier à mon tuteur Mounir, qui m'a guidé et a su me donner les conseils nécessaires pour surmonter les défis techniques. Merci également aux développeurs Mehdi, Daniel, David et Gilles pour leur soutien technique, leur collaboration et pour les échanges constructifs, qui ont enrichi mon expérience. Merci également à Baptiste, mon chef d'équipe qui m'a fait confiance et les responsabilités qui m'a confiée. Également merci au business analyste, Octavia, Ilhem et Wilfried, toujours présents et à l'écoute qui ont su répondre avec clarté à toutes mes questions fonctionnelles et m'aider à mieux comprendre les besoins métiers.

Enfin, je remercie sincèrement notre Scrum Master, Matthieu pour l'organisation efficace des sprints, mais aussi de m'avoir attribué des tâches adaptées à mon évolution, tout en assurant un suivi constant de mon intégration dans l'équipe.

Je remercie également l'équipe pédagogique du master STL et tout particulièrement M. Jacques Malenfant.

Je tiens enfin à exprimer ma reconnaissance à l'ensemble des camarades du master STL, avec qui j'ai partagé cette année riche en apprentissage, en entraide et en bons moments.

## Table des matières

<b>1. Introduction.....</b>	<b>1</b>
<b>2. Environnement de Travail .....</b>	<b>1</b>
2.1. Entreprise d'accueil .....	1
2.2. Organisation de l'équipe .....	1
2.3. Méthodologie de Travail .....	2
2.3.1. Outils utilisés.....	3
2.4. Présentation des Applications.....	3
2.4.1. PFI .....	3
2.4.2. ZEN .....	4
<b>3. Etat de l'art.....</b>	<b>5</b>
<b>4. Objectifs.....</b>	<b>6</b>
<b>5. Diagramme de Gantt .....</b>	<b>7</b>
<b>6. Missions Réalisées.....</b>	<b>7</b>
6.1. Intégration .....	7
6.2. Migration et passage Go to Cloud (Cloud interne) .....	8
6.2.1. Migration Java/Wildfly.....	9
6.2.2. Analyse et correction des vulnérabilités (SonarQube) .....	10
6.3. Correction de Bugs .....	12
6.3.1. Message d'erreurs ZEN .....	12
6.3.2. Health/Actuator et HealthCheck : .....	13
6.4. Indivision.....	15
6.4.1. Tri Indivision .....	17
6.4.2. Popin détails offres.....	17
6.5. Enrichissement des tables SQL.....	18
6.5.1. Tables Frais-Exante .....	18
6.5.2. Colonne PFI-Version .....	20
6.6. Evolutions fonctionnelles.....	21
6.6.1. Picto GSMD .....	21
6.6.2. Wording conditions particulières.....	22
6.6.3. Conseil Coach Financier .....	22
<b>7. Difficultés Rencontrées .....</b>	<b>23</b>
<b>8. Conclusion.....</b>	<b>23</b>
<b>9. Glossaire.....</b>	<b>25</b>
<b>10. Annexes.....</b>	<b>27</b>

## 1. Introduction

Ce rapport présente mon expérience d'alternance réalisée au sein de la Société Générale Private Bank, en qualité de Développeur Full stack Java, d'une durée d'un an, de septembre 2024 à septembre 2025. Cette expérience s'inscrit dans le cadre de ma deuxième année de master « Science et Technologie du Logiciel ». Au cours de cette période, j'ai pu intégrer une équipe agile spécialisée dans le développement d'application destinée aux conseillers en gestion de patrimoine avec pour objectifs d'améliorer l'expérience utilisateur des collaborateurs internes et de répondre aux exigences élevées du secteur bancaire (sécurité, conformité, performance).

Ce document détaille l'environnement technique et méthodologique dans lequel j'ai évolué, les différentes missions que j'ai réalisées, ainsi que les compétences développées tout au long de cette année.

## 2. Environnement de Travail

### 2.1. Entreprise d'accueil

Société Générale, fondée en 1864, est l'un des plus grands groupes bancaires européens. Présente dans 60 pays, la banque propose une large gamme de services, financiers aux particuliers, aux entreprises et aux institutions : banque de détail, services financiers spécialisés, banque d'investissement, gestion d'actifs et assurance. Elle compte plus de 117 000 collaborateurs à travers le monde et sert environ 25 millions de clients. En 2023, son produit net bancaire s'élevait à près de 25 milliards d'euros.

J'ai intégré la branche banque privée Société Générale Private Banking, cette dernière est la branche de gestion de fortune du groupe Société Générale. Elle s'adresse à une clientèle patrimoniale et propose des solutions personnalisées en matière de gestion financière, d'investissement, d'ingénierie patrimoniale et de planification successorale. Présente dans une dizaine de pays, elle gère plus de 110 milliards d'euros d'actifs (chiffres 2023) et accompagne une clientèle fortunée, entrepreneurs, dirigeants ou familles, en leur offrant un service sur mesure alliant proximité, expertise et confidentialité.

### 2.2. Organisation de l'équipe

L'équipe PFI/ZEN est composée de profils aux rôles complémentaires, organisés pour favoriser la collaboration entre les volets fonctionnels et techniques :

- **2 Business Analyste** : en charge de la rédaction des user stories, de la clarification fonctionnelle et du lien avec les utilisateurs métiers.
- **1 Chef d'équipe** : responsable de la planification globale et du delivery et de la coordination avec les autres équipes du programme.
- **1 Scrum Master** : garant de la méthodologie agile, animation des cérémonies Scrum, suivi des sprint et accompagnement de l'équipe.
- **5 Développeurs** : chargés de la conception, du développement et de la mise en production des évolutions techniques et fonctionnelles.

L'équipe PFI/ZEN fait partie du groupe **EFP** (Banque Privée / Epargne Financière), qui est structurés en deux tribus, elles-mêmes subdivisées en plusieurs équipes. Notre équipe appartient à une tribu qui comprend trois équipes principales :

- **PFI/ZEN** : Allocation et Propositions d'Investissements.
- **GCN** : Gestion Conseillé Numérique.
- **BOP** : Brique de collecte.

Ces équipes entretiennent une relation fournisseur/consommateur forte : par exemple PFI consomme les données de BOP/GCN, ou interagit avec les web services exposés par BOP. Cette organisation nécessite une coordination régulière, notamment lors des PI Plannings inter-équipes.

### 2.3. Méthodologie de Travail

Le cadre de travail suit les principes de la méthodologie Scrum, avec des sprints de trois semaines, Le **Product Owner (PO)**, bien qu'issu du métier, travaille en étroite collaboration avec l'ensemble de l'équipe pour affiner et prioriser les besoins utilisateurs.

Le cycle commence par la gestion du **Product Backlog**, alimenté par le PO et les BA à partir des besoins métiers, structurés sous forme d'epics et d'user stories. Après une phase de priorisation, les éléments sélectionnés alimentent les Sprint Backlog lors du Sprint Planning, réunion durant laquelle l'équipe s'engage sur les tâches à réaliser pendant le sprint.

Chaque jour, un **Daily Meeting** de 15minutes permet à l'équipe de synchroniser l'avancement, de lever les blocages éventuels et d'ajuster les priorités si nécessaire. En fin de sprint, une **Sprint Review** est organisée afin de présenter les fonctionnalités livrées au PO et aux parties prenantes. Elle est suivie d'une **Rétrospective**, durant laquelle l'équipe revient sur le déroulement du sprint pour identifier les axes d'amélioration et renforcer les pratiques collectives.

Tous les 4 à 5 sprints un **PI Planning** est organisé avec les autres équipes de la tribu pour planifier les dépendances, définir les objectifs globaux et coordonner les roadmaps entre équipes.

Enfin, en tant que membre interne de l'équipe, j'ai été convié aux séminaires de type All Staff, organisés environ tous les six mois. Ces événements rassemblent les collaborateurs autour de retours d'expérience, d'annonces stratégiques.

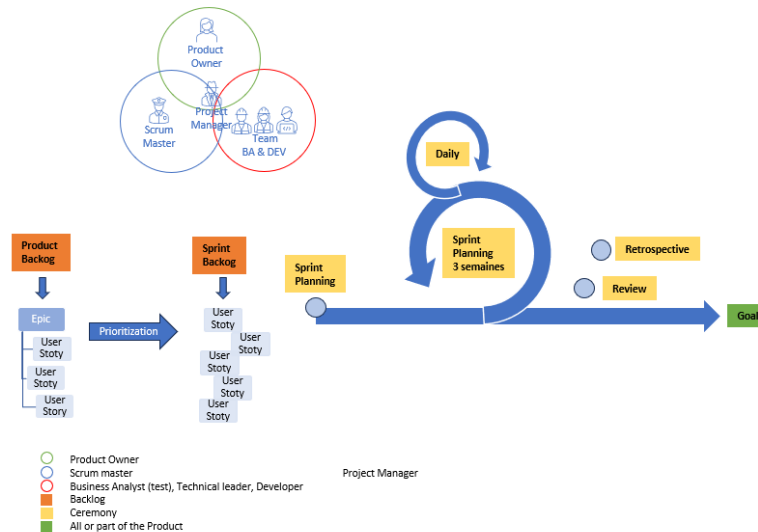


FIGURE 1 ORGANISATION DE L'EQUIPE PFI

### 2.3.1. Outils utilisés

#### Coté PFI

- Java EE version 8 et 11 : base du backend multi-modules.
- Jboss Wildfly version 10 et 26 : serveur d'application Java.
- Oracle SQL Développeur : SGBDR principal.
- Kubernetes : orchestration des conteneurs.
- JSP : frontend Java.

#### Coté ZEN

- Spring Boot: back-end microservices (architecture hexagonal).
- PostgreSQL : base de données dédié à ZEN.
- Angular : frontend.

#### Outils Transverses

- Jenkins : CI/CD, déploiements automatisés.
- Jira : gestion du backlog, suivi des tickets et cérémonies Scrum.
- SonarQube : analyse de la qualité du code, détection de vulnérabilités.
- Postman: Simulation Appels web services.

## 2.4. Présentation des Applications

### 2.4.1. PFI

L'application PFI (Allocation et Proposition Investisseur) est un outil central utilisé par les conseillers de Société Générale Private Banking dans le cadre de leurs missions de conseil patrimonial. Elle permet de construire un diagnostic complet du portefeuille du client (BP1 et BP2), d'identifier son profil investisseur, puis de lui proposer une combinaison optimale d'offres et de produits adaptés à sa situation. L'application respecte des exigences réglementaires strictes (profil investisseur) tout en offrant une interface métier riche et personnalisée selon le type de conseiller.

## Architecture de l'application

PFI repose sur une architecture Java EE avec plusieurs modules métier intégrés, initialement déployé sur un serveur Wildfly dans un environnement on-premise. À la suite d'une migration (Go to Cloud) l'application est déployée en environnement Cloud. Elle s'appuie sur une base de données Oracle et interagit avec de nombreux services externes comme RPF pour récupérer les informations clients, BOP pour récupérer les différents contrats (ASV, CTO, PEA.) liées aux clients, BVI par l'intermédiaire de ZEN pour récupérer les infos liées à chaque contrat (Note MIF, Devise, Type de gestion (GD, GC, GSM, GSMD), Montant minimum).

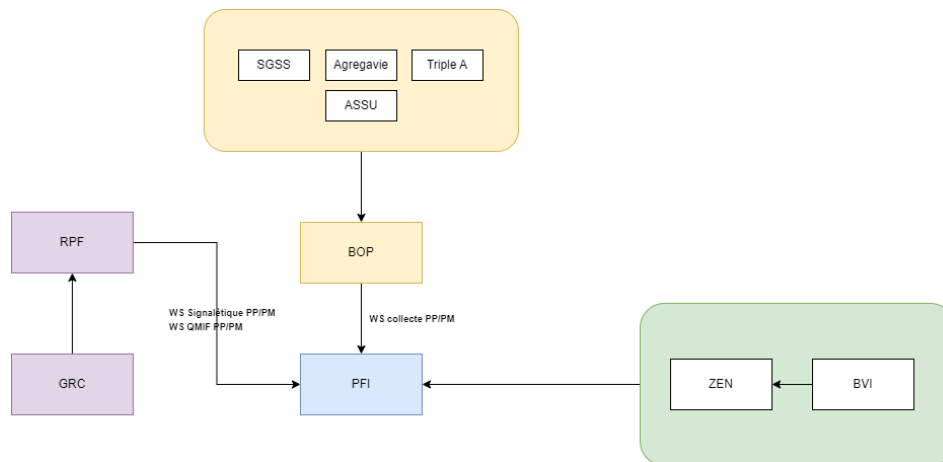


FIGURE 2 DIFFERENTS APPEL WS REÇUS PAR PFI

### 2.4.2. ZEN

ZEN (Référentiel de produits financiers) est une application dédiée au métier (Back-Office). Cette application permet de paramétrer un catalogue, appelé catalogue ZEN, qui référence tout un ensemble de produits bancaires dédiés à une clientèle PRIV.

Ce catalogue est aujourd'hui utilisé exclusivement par l'application API, qui permet aux banquiers comme aux gérants de conseiller leur client sur l'allocation de leur investissement, matérialisée, bien souvent, en fin de parcours, par une édition de la proposition API.

Le catalogue évoluant sans cesse (nouveaux produits commercialisés, produits en fin de commercialisation, caractéristiques à modifier ou à créer, anomalies catalogue en production), il est nécessaire que le paramétrage soit non seulement disponible en temps réel mais qu'il permette au métier d'anticiper de nouvelles demandes du marketing.

#### Les différents produits du catalogue :

- Les supports.
- Les mandats.
- Les groupes.
- Les enveloppes.
- Les offres

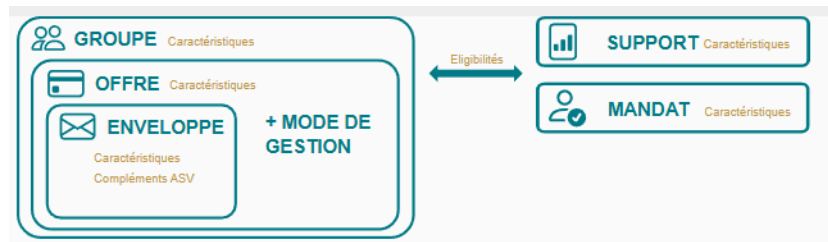


FIGURE 3 SCHEMA TYPE D'UN CONTRAT

Zen repose sur une architecture orientée micro-services avec un Backend Spring Boot structurés selon le modèle hexagonal. On y distingue un cœur métier (**core**), une API d'exposition front (**BFF**) et une API des communication inter-SI (**BFB**). Coté front ZEN utilise Angular, la base de données utilisée est PostgreSQL et l'application entièrement déployée sur le cloud interne.

PFI et ZEN sont donc deux applications étroitement liées et parfaitement complémentaires. Tandis que ZEN centralise la donnée produit, la rend exploitable et la maintient à jour de manière sécurisée, PFI en est le principal consommateur : il s'appuie sur le catalogue pour proposer aux clients des combinaisons cohérentes de contrats, mandats et supports. Ensemble, ces deux applications forment une brique essentielle du SI de Société Générale Private Banking.

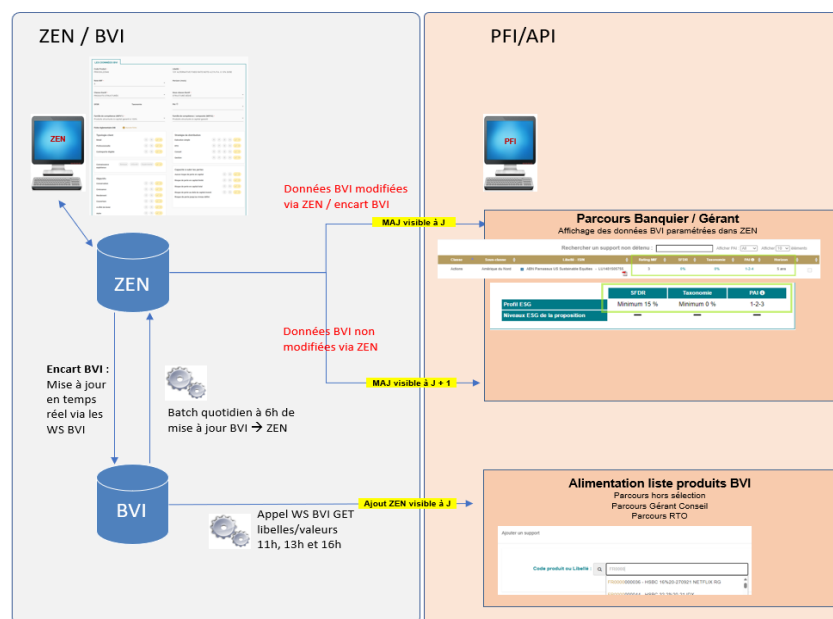


FIGURE 4 ARCHITECTURE ET COMMUNICATION DE ZEN

### 3. Etat de l'art

Le secteur bancaire fait face à une double exigence : moderniser ses systèmes d'information tout en maintenant un haut niveau de sécurité, de conformité réglementaire et de performance. Cette transformation se traduit par une adoption croissante de pratiques DevOps, d'approches Agile à l'échelle (Scrum), et de la migration vers des environnements cloud privés.



Les chaines d'intégration continue (CI/CD) reposent sur des outils comme Github, SonarQube, Nexus ou encore Kubernetes et Jenkins. Ils permettent d'automatiser les déploiements, de renforcer la qualité du code et d'assurer la traçabilité des livraisons.

Par ailleurs, les architectures monolithiques Java EE historiques évoluent vers des solutions modernes telles que Spring Boot pour les nouveaux modules, avec un passage vers les versions LTS de Java (Java 11 et 17). L'adoption du pattern hexagonal favorise une meilleure séparation des responsabilités dans le code et une testabilité accrue.

## 4. Objectifs

Les Objectifs définis au début de l'alternance étaient directement alignés avec les missions présentées initialement dans l'offre d'emploi. Plus concrètement, il s'agissait de :

- Maitriser l'ensemble du cycle de développement logiciel, depuis la compréhension du besoin utilisateur jusqu'au déploiement en production, en passant par l'analyse, le chiffage, les bonnes pratiques de développement, les revues de code, et les tests automatisés.
- Développer mes compétences techniques sur des technologies modernes utilisées dans l'entreprise : Java 11, Angular 15, Spring Boot, Jenkins, Maven.
- Comprendre et pratiquer les principes fondamentaux de la méthodologie Agile Scrum à travers la participation active aux cérémonies.
- Participer activement à l'amélioration continue de la qualité du code et de la sécurité applicative à travers des tâches d'analyse SonarQube et la gestion des vulnérabilités.
- Acquérir progressivement de l'autonomie sur les projets confiés, tout en développant une collaboration efficace avec l'équipe fonctionnelle (Business Analyste, Product Owner).
- Améliorer ma capacité à travailler dans un environnement professionnel stimulant, en apprenant aux côtés d'experts reconnus et en m'intégrant progressivement à la communauté de développeurs internes.

## 5. Diagramme de Gantt

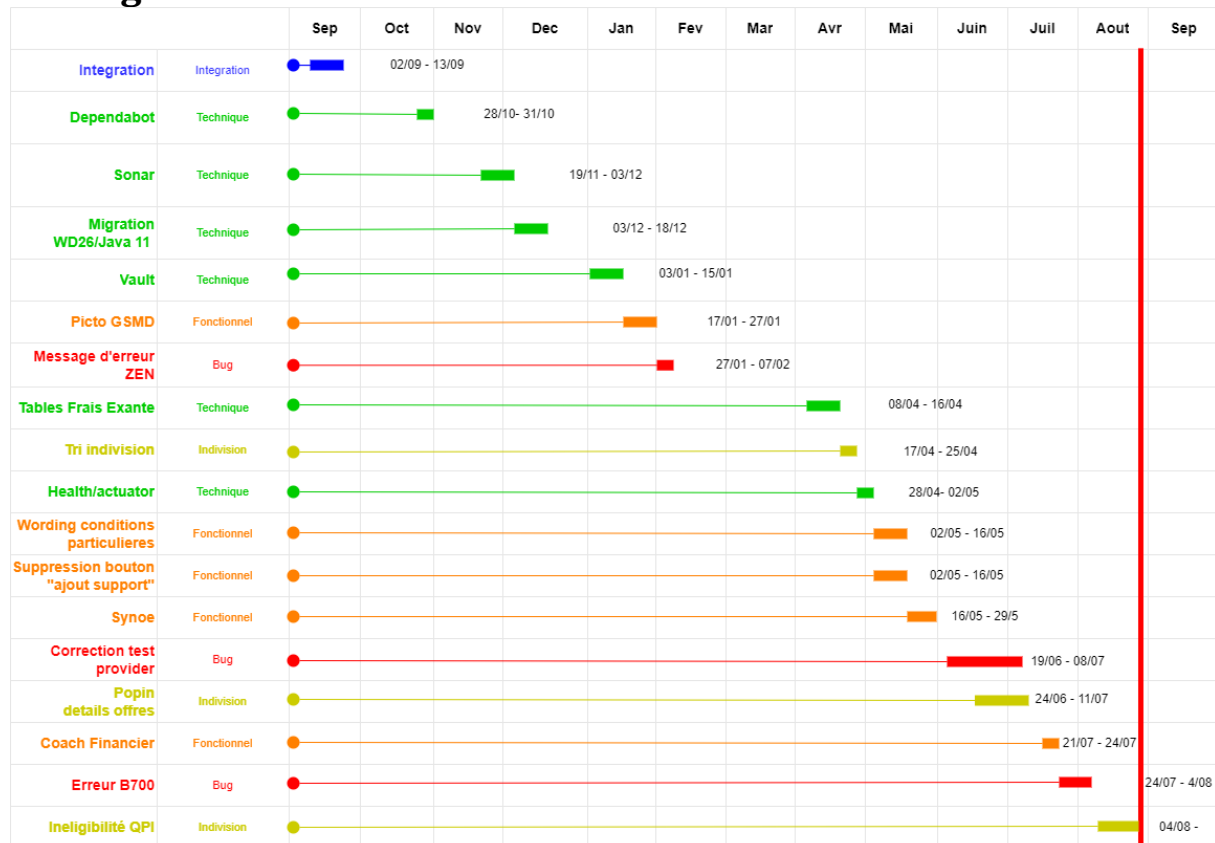


FIGURE 5 DIAGRAMME MICROSCOPIQUE DE GANTT

## 6. Missions Réalisées

### 6.1. Intégration

Mon intégration au sein de l'équipe s'est déroulée de façon progressive et structurée. Dès les premiers jours, j'ai été accompagné dans la configuration de mon environnement de travail : accès VPN, outillages DevOps (Git, Jira, Jenkins), et environnement de développement spécifique aux applications PFI et ZEN. Une phase d'onboarding fonctionnel a ensuite été organisée afin de me familiariser avec le contexte métier de la banque privée, les principaux parcours et écrans utilisateurs (étude, mandats, arbitrage...), ainsi que le rôle de chaque application dans l'écosystème technique.

Durant cette phase, j'ai pu explorer le fonctionnement interne de PFI (architecture Java EE multi-modules, front JSP, interconnexions avec les services de données) et de ZEN (structure hexagonale Spring Boot, organisation frontend Angular). J'ai commencé par des lectures de documentation fonctionnelle et technique, ainsi que de l'analyse de code pour comprendre les logiques métiers déjà en place. J'ai également participé très tôt à des cérémonies agiles (Daily meeting, sprint review, planning), ce qui m'a permis de suivre l'activité globale et d'identifier les flux fonctionnels clés.

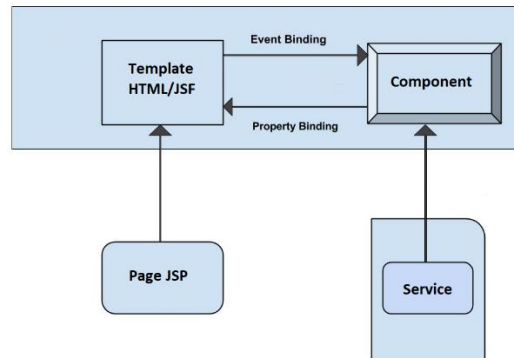


FIGURE 6 FONCTIONNEMENT FRONT JSP

## 6.2. Migration et passage Go to Cloud (Cloud interne)

Dans le cadre de la stratégie globale de modernisation des systèmes d'information de la Société Générale, un programme a été initié : **Projet Go To Cloud**. Il consiste à migrer progressivement l'ensemble des applications historiques vers une infrastructure de cloud interne, avec des objectifs clairs : meilleure résilience, sécurité renforcée, élasticité des services.

L'application PFI, outil central utilisé par les conseillers en gestion du patrimoine, a été l'une des premières ciblées. Elle reposait sur une architecture **Java EE/Wildfly 10** et une base de données Oracle on-premise (legacy), peu adaptée aux standards modernes.

La mission a été de participer activement à cette transformation technique à travers deux axes majeurs : migration de la stack Java et l'analyse des vulnérabilités, remontées par SonarQube et les alertes Github.

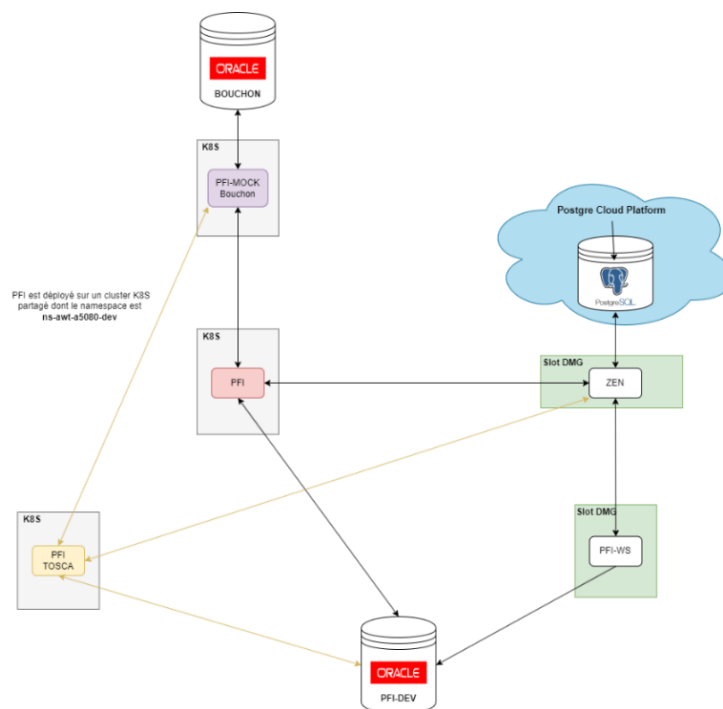


FIGURE 7 ARCHITECTURE PROJET AVANT G2C

### 6.2.1. Migration Java/Wildfly

La première étape a consisté à mener une étude comparative des versions stables Java afin d'identifier la version la plus adaptées aux besoins du projet.

Cette analyse s'est appuyée sur plusieurs critères techniques et stratégiques. Le premier critère concernait le support à long terme (**LTS**), indispensable pour garantir la pérennité et la maintenabilité de l'applications sur plusieurs années. Ensuite, la compatibilité avec le serveur d'application ciblé **Wildfly 26**. La compatibilité avec les librairies déjà présentes dans le projet a également été prise en compte afin de réduire les efforts de re factoring, qui lui aussi est un critère pris en compte. Les gains en performance attendus ont été étudiés, qu'il s'agisse d'optimisation mémoire, de vitesse d'exécution ou d'efficacité globale du runtime.

Critères évalués	Java 11	Java 17	Java 21
Support LTS	Oui	Oui	Oui
Compatibilité Wildfly 26	Très Bonne	Bonne	Moyenne
Compatibilités avec les librairies existantes	Très Bonne	Moyenne	Faible
Gains de performance attendus	Bons	Très Bons	Très Bons
Effort de migration nécessaire	Faible	Moyen	Elevée
<b>Recommandation finale</b>	Choisie	Possible	Déconseillée

L'ensemble de ces éléments a conduit au choix raisonné de **Java 11**, qui représentait un compromis optimal entre compatibilité, performance et facilité de migration dans le contexte de la Société Générale.

En parallèle, une migration de de **Wildfly 10 vers Wildfly 26** a été faite, cette version plus moderne et plus compatible avec les outils de déploiement en conteneurs. Cette migration s'est accompagnée de plusieurs adaptations techniques :

- Réécriture de la configuration (*standalone.xml*) pour intégrer les nouveaux modules.
- Mise à jour du système de log et des connecteurs **JBDC**.
- Résolution d'un bug bloquant : une incompatibilité avec la librairie Jackson empêchait l'accès aux classes du module *java.time*, ainsi la sérialisation des dates dans les DTO JSON. J'ai mis en place une classe de configuration personnalisée. Celle-ci permet d'enregistrer un *JavaTimeModule* dans un Object Mapper, avec un serializer sur mesure pour le type *LocalDateTime*.  
Ce serializer :
  - Tronque les dates aux seconds près.
  - Applique un format personnalisé utilisée généralement dans le projet (yyyy-MM-dd 'T' HH :mm :ss).
  - Garantit la cohérence d'encodage entre tous les micro-services producteurs et consommateurs de données temporelles.

```

javaTimeModule.addSerializer(LocalDateTime.class, new JsonSerializer<>() {
    @Override
    public void serialize(LocalDateTime localDateTime, JsonGenerator jsonGenerator,
        SerializerProvider serializerProvider) throws IOException {
        jsonGenerator.writeString(localDateTime.truncatedTo(ChronoUnit.SECONDS).format(DateTi
            meFormatter.ofPattern("yyyy-MM-dd'T'HH:mm:ss")));
    }
});

```

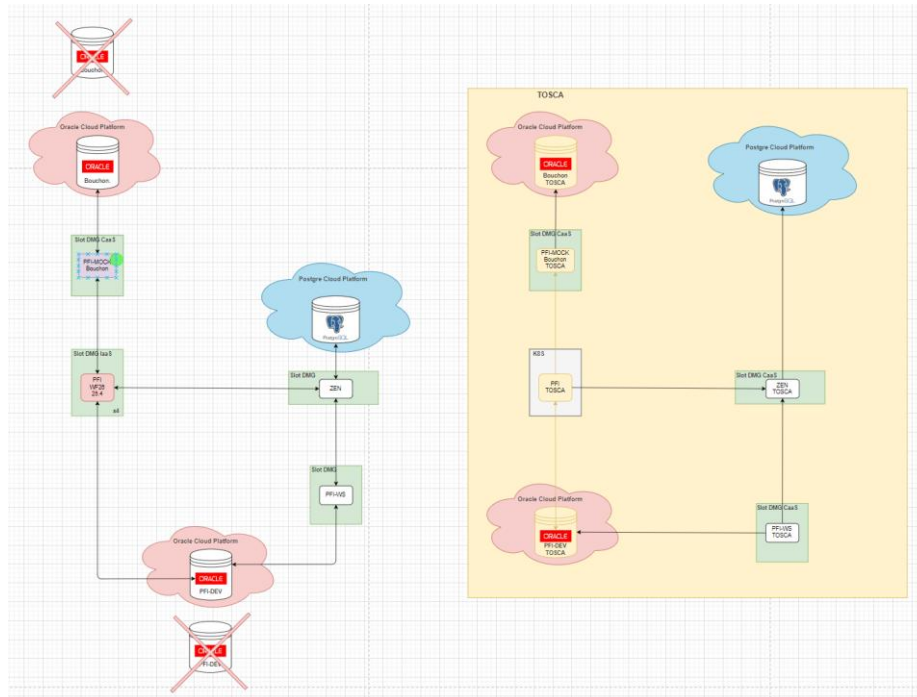


FIGURE 8 ARCHITECTURE DU PROJET POST G2C

### 6.2.2. Analyse et correction des vulnérabilités (SonarQube)

Une analyse de la dette technique et de la sécurité du code, a été initiée. J'ai pris en charge cette partie en utilisant **SonarQube** et les alertes de sécurité remontées par **Github**.

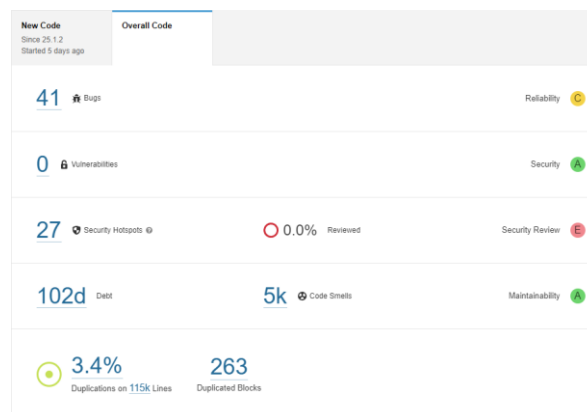


FIGURE 9 PAGE SONARQUBE AVANT CORRECTIONS

**Principaux problèmes détectés et actions correctives :**

- **Sécurité XML** : désactivation explicite des entités externes et interdiction de la déclaration DOCTYPE dans les parseurs XML afin de prévenir les attaques XXE (XML External Entity).
- **Authentification** : remplacement des systèmes d'authentification basiques (Authorization : Base64) utilisé dans le projet par des mécanismes modernes à base de JWT Tokens.
- **Vulnérabilité Path Traversal** : renforcement des contrôles sur les chemins utilisateur pour éviter toute lecture non autorisée.
- **Exception non traitée** : ajout de la gestion explicite des erreurs (IllegalArgumentException) dans les blocs try/catch.
- **Bonne pratique Java et Null Safety** : adoption de Optionnal et vérification systématique de la non-nullité pour éviter les NullPointerException, suppression de code redondant, initialisation correcte d'objets.

Parmi les vulnérabilités remontées par SonarQube quelque'une ont été considéré comme « Faux Positifs ». Par exemple les alertes concernant les exceptions liées à la concurrence (multithreading), les traitements en question étaient déclenché en contexte de single-thread, et toute tentative de modification pour satisfaire cette règle aurait introduit un risque de régression ou un comportement instable de l'application en production.

**Intégration des outils Github :**

- **Mises à jour des dépendances** : plusieurs bibliothèques utilisées dans le projet avaient des CVE critiques. Parmi elles :
  - **Log4j** (passage 2.17 vers 2.24) : faille critique sous la référence CVE-2021-44228. Cette faille permettait l'exécution de code arbitraire via des requête malveillante de journalisation. Monté de version annulé car incompatible avec le projet et qu'elle provoquait une régression, et le contexte d'exécution de la bibliothèque était restreint.
  - **Spring** (4.3 vers 6.x) : CVE-2020-36518, montée de version annulé, Spring4shell, une faille affectant les versions utilisant Tomcat en tant que conteneur de servlet. Toutefois, dans notre environnement, Spring était utilisé dans un contexte ne permettant pas l'utilisation l'exploitation de la faille, et une mise à jour aurait nécessité un grand re factoring dans le projet.
  - **Jackson Core**.
  - **Objenesis** : CVE-2019-25097, faille exposant un risque de Denial of Service non contrôlée. Elle a été remplacée par une nouvelle version de la bibliothèque (io.github.kostaskougios).
  - **Apache Commons IO** : bibliothèque décommissionné car elle n'était plus utilisée car elle a été remplacée par sa nouvelle version qui est sous le nom Apache IO.
  - **Struts** : 1 vers 2, CVE-2017-5638, version identifiée comme obsolète. Une montée de version aurait nécessité un re factoring total du code front JSP de l'application.

- **Secrets exposés** : Github Secret Scanning a détecté des jetons et mots de passe exposés (en clair) dans des fichiers .properties. Ces secrets ont été externalisé dans l'outil XVault, référentiel interne sécurisé.

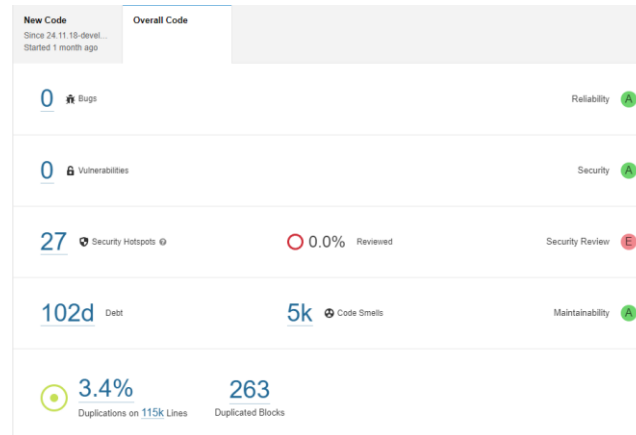


FIGURE 10 PAGE SONARQUBE POST CORRECTIONS

## 6.3. Correction de Bugs

### 6.3.1. Message d'erreurs ZEN

À la suite d'un bug remontés par les équipes métier qui ont signalé un bug concernant le message d'erreur affiché lors d'un refus d'accès à l'application ZEN, il a été demandé de modifier ce message pour le rendre plus explicite, plus clair et conforme aux demandes des utilisateurs.

Les étapes réalisées :

- Analyse du problème : Lorsqu'un utilisateur non habilité tente d'accéder à ZEN, une `AccessDeniedException` est levée coté backend, mais l'application retourne un statut **Http 500 (Internal Server Error) au lieu d'un 403 (Forbidden)**. Non conforme aux standard REST
- Mise en place d'un **ExceptionHandler** : La première solution a été de créer un handler globale avec l'annotation Spring **@ControllerAdvice** afin d'intercepter ces exceptions et de retourner un message explicite.
- Problème Persistant : Malgré la mise en place du handler, le frontend et l'application reçoivent toujours une erreur 500, après investigation, il s'avère qu'une librairie du **socle Société Générale (dgt-exception)** intercepte l'exception avant le handler, encapsulant l'erreur dans une `Internal Server Error`.
- Indentification du problème : le problème est que l'annotation Spring **@PreAuthorize** gérait mal la distinction des profils utilisateurs ce qu'il déclenchait l'exécution de la bibliothèque du socle
- Solution : Utilisation de l'annotation du socle **@DgtSafeRoles** pour différencier entre les profils (conseiller agence, backoffice) et prioriser la gestion des rôles.

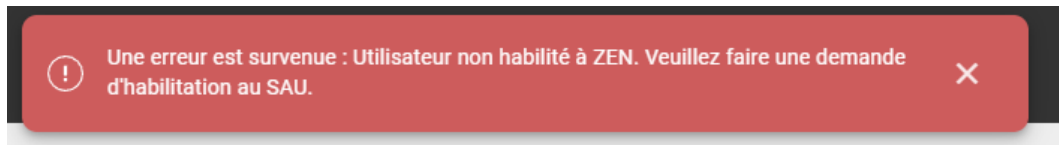


FIGURE 11 NOUVEAU MESSAGE D'ERREUR ZEN

### 6.3.2. Health/Actuator et HealthCheck :

- **ZEN :**

Dans la page Admin, une section sur les versions de ZEN déployés en environnement de **DEV, HOB et Prod**, à la suite de la migration **G2C** la récupération des versions n'était plus récupérées.

Pour résoudre ce problème, un contrôleur de healthCheck a été mis en place côté backend. Un contrôleur de healthCheck est un point d'entrée **REST** qui retourne une réponse JSON avec un statut, généralement "ok", indiquant que l'application est vivante et capable de traiter des requêtes **HTTP**.

Ce mécanisme permet à notre équipe de superviser le bon fonctionnement de l'application : si le contrôleur ne répond pas "ok", une alerte est déclenchée pour investigation.

Il est important de noter que ce contrôleur ne garantit pas à lui seul l'absence de tout problème : il vérifie principalement la capacité de l'application à répondre via le Framework (Spring). D'autres incidents peuvent survenir et nécessiter des contrôles complémentaires.

Les taches réalisées :

- a) Création d'un service **HealthCheckDataFrontSrv**, ce service expose deux Endpoint REST :

- /monitoring/healthCheck.json qui vérifie que l'application est vivante et fonctionnelle, il est exposé via **JAX-RS** avec les annotations **@Path**, **@GET** et **@Produces**, et utilise l'injection de dépendances pour accéder à un helper (*HealthCheckHelper*) chargé de récupérer les informations sur les versions de l'application (*getVersions*). Lorsque ce contrôleur est appelé il répond avec un « ok » de type **OdtResponseAvecDonnées** (le ok + informations sur les données). La réponse est automatiquement sérialisée en JSON grâce à la dépendance Jackson.

- /monitoring/version qui a pour but de récupérer les versions déployées sur les différents environnements et slots (**DEV, HOB, PROD, TOSCA**). Il est exposé via **JAX-RS** avec les annotations **@Path**, **@GET** et **@Produces**, il parcourt une liste statique de serveurs (URL et métadonnées). Pour chaque serveur il effectue un appel http avec **HttpURLConnection** afin de récupérer la version et le timestamp du build, puis sérialise la réponse en JSON. Un formatage du timestamp est effectué pour l'affichage grâce à **DateTimeFormatter**. L'Endpoint retourne une liste d'objets *VersionInfo* (URL, serveur, environnement, site, slot, version, date de build) encapsulée dans une réponse



**OdtReponseAvecDonnees**, avec sérialisation JSON automatique. Enfin, en cas d'erreur, le message d'exception est injecté dans la version pour faciliter le diagnostic.

- b) Adaptation du service Angular pour consommer le nouvel Endpoint backend monitoring/versions :

Grâce à l'utilisation de **HttpClient**, une requête GET est envoyée à ce point d'entrée, et la méthode *getAllVersions()* retourne un Observable permettant au composant de s'abonner à la réponse. Le type *VersionData* a été ajusté pour refléter fidèlement la structure JSON renvoyée par le backend, incluant les champs serveur, environnement, site, slot, version, etc. Toute la logique de récupération et d'agrégation des versions des serveurs étant désormais centralisée côté backend, le front se contente d'afficher les données reçues, ce qui simplifie la maintenance et assure la cohérence des informations présentées.

- c) Refactorisation du front Angular pour une récupération dynamique des versions serveurs :

Le front Angular a été refactorisé pour supprimer la gestion statique des serveurs dans le composant TypeScript, la variable contenant la liste des serveurs ayant été retirée afin de centraliser la source de vérité côté backend et d'éviter toute duplication. Désormais, l'affichage dans le composant html a été adapté : la colonne du tableau affiche directement l'URL interrogée (*{{element.url}}*) au lieu du nom du serveur, ce qui permet de visualiser précisément la cible de chaque requête. Une nouvelle méthode *versions()* a été ajoutée dans le composant pour appeler le service Angular, récupérer dynamiquement la liste des versions via HTTP, mettre à jour la source de données du tableau (*dataSourceServeurs*) avec la réponse du backend, et gérer l'état de chargement ainsi que les éventuelles erreurs.

Serveur	Env	Site	Slot	Statut	Version	DateBuild
https://main-hadynback-nsx.eu-fr-paris.dev.retail.socgen/zen/bff/main/data/monitoring/healthCheck.json	DEV	Paris	main	OK	9.0.2-test-token-SNAPSHOT	2025/07/24 16:09
https://main-hadynback-nsx.eu-fr-paris.dev.retail.socgen/zen/api/main/api/monitoring/healthCheck	DEV	Paris	main	OK	9.0.1	2025/07/01 14:19
https://main-hadynback-nsx.eu-fr-paris.dev.retail.socgen/zen/bff/tosca/data/monitoring/healthCheck.json	DEV	Paris	tosca	OK	9.0.1	2025/07/01 14:19
https://main-hadynback-nsx.eu-fr-paris.dev.retail.socgen/zen/api/tosca/api/monitoring/healthCheck	DEV	Paris	tosca	OK	9.0.1	2025/07/01 14:19
https://main-hadynback-nsx.eu-fr-paris.html.retail.socgen/zen/api/hob-main/api/monitoring/healthCheck	HOB	Paris	hob-main	OK	9.0.2	2025/07/25 12:13
https://main-hadynback-nsx.eu-fr-paris.html.retail.socgen/zen/web/hob-main/data/monitoring/healthCheck.json	HOB	Paris	hob-main	OK	9.0.2	2025/07/25 12:13
https://main-hadynback-nsx.eu-fr-paris.prd.retail.socgen/zen/web/prd-main/data/monitoring/healthCheck.json	PRD	Paris	prd-main	OK	9.0.2	2025/07/25 12:13
https://main-hadynback-nsx.eu-fr-paris.prd.retail.socgen/zen/api/prd-main/api/monitoring/healthCheck	PRD	Paris	prd-main	OK	9.0.2	2025/07/25 12:13

FIGURE 12 SECTION VERSION APPLICATION DANS ZEN

- **PFI :**

Dans l'application, un bouton d'aide en haut à gauche de la page permet d'afficher une popin, sur cette popin plusieurs informations sont indiquées, parmi elles la version de l'application sur laquelle on était.

Le problème est que sur les slots des environnements de HOB et PROD, cette popin ne s'ouvrait pas dû au blocage de l'Endpoint */actuator* sur la rampe du serveur **main-hadynbak-nsx** (blocage conseillé par la Société Générale).

Missions réalisées :

- Se connecter en local à l'environnement HOB.
- Exposition d'un nouveau web service REST dédié à la récupération de la version via la classe **ProvidersWsHealth** et le nouvel Endpoint */health*, référencée par la ressource **AppInfos**
- Adaptation de la page JSP responsable de la popin d'aide pour consommer ce nouvel Endpoint : lors du clic sur le bouton aide, un appel est effectué vers */health/info/getApplicationVersion* pour récupérer la version de l'application.
- Affichage dynamique de la version dans la popin, rétablissant ainsi le comportement attendu sur les slots HOB et PROD.



FIGURE 13 POPIN BOUTON AIDE PFI

## 6.4. Indivision

Le chantier majeur de la banque privée cette année a été l'ajout de la gestion des comptes en indivision dans toutes les applications du périmètre (principalement PFI), afin de mieux identifier et traiter les différents types de comptes multi-titulaires. Historiquement, l'application ne distinguait pas clairement toutes les situations, ce qui était une limitation fonctionnelle.

L'objectif principal était d'étendre les données importées depuis les autres applications telles que RPF et BOP, afin de prendre en compte tous les types d'indivision, ensuite de mettre à jour tout l'applicatif, la logique et les calculs présents dans PFI (Arbitrage, Editique, QPI, actifs financiers). Enfin, permettre aux conseillers d'identifier plus facilement la nature d'un contrat et les actions à mener.

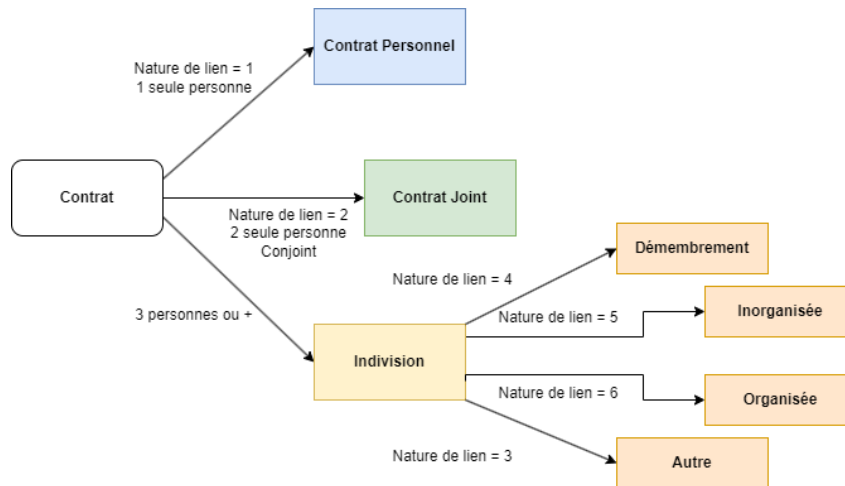


FIGURE 14 NOUVELLE ORGANISATION TYPE DE DETENTION CONTRAT

- **Les différents types de contrat concernés :**

Avant l'évolution, la nature d'un lien client-contrat se limitait à :

Nature de lien	Type
1	Contrat personnel
2	Contrat joint
3	Autre

Cette classification était insuffisante et ne permettait pas de distinguer les nouveaux cas.

À la suite du sujet, l'import des données inclut désormais six valeurs distinctes :

Nature de lien	Type
1	Contrat personnel
2	Contrat joint
3	Autre
4	Démembrement
5	Indivision inorganisée
6	Indivision Organisée

- **Définition des nouveaux types :**

- Indivision inorganisée : C'est le fait que plusieurs personnes sont propriétaires d'un même contrat, mais aucune convention d'organisation n'a été mise en place, toutes les décisions nécessitent un accord unanime des coindivisaires. Il est souvent rencontré dans les successions, lorsque le patrimoine n'a pas encore été réparti ou organisé.
- Indivision organisée : Même situation que l'indivision inorganisée, mais avec un accord formalisé (convention d'indivision). Une ou plusieurs personnes peuvent avoir délégué un pouvoir pour gérer le compte au nom de tous.
- Démembrement : Cas particulier où la propriété est scindée entre :
  - Nu-propriétaire : détient la propriété juridique, mais ne peut pas jouir des revenus du contrat.
  - Usufruitier : Détient le droit d'usage et les revenus. Par exemple : succession où un parent conserve l'usufruit d'un portefeuille et les enfants héritent de la nue-propriété.

- d) Autre (Club d'investissement) : Structures collectives qui ont un mode de fonctionnement et une fiscalité différente des indivisions. Toutefois, ces cas ne sont pas pris en compte dans ce sujet.

#### 6.4.1. Tri Indivision

Dans le cadre de ce sujet d'indivision, un besoin métier important a été exprimé pour améliorer la lisibilité des listes des contrats affichées dans les pages avoirs financiers et arbitrage de l'application PFI. Jusqu'alors, l'affichage suivait un tri purement alphabétique, sans prise en compte de la nature du contrat. L'objectif de ce ticket est de mettre en place un tri hiérarchisé :

D'abord en fonction du type de détention, puis selon le libellé du contrat dans l'ordre alphabétique croissant. L'ordre métier défini est le suivant : Personnel, Joint, Démembrement, Indivision inorganisée, Indivision organisée et enfin Autre. Ce nouvel ordre doit être appliqué de façon homogène sur toutes les pages concernées, y compris la page des actifs financiers.

Missions réalisées :

- Analyse et localisation du point de tri : débogage du code pour identifier le moment et la méthode de tri, La fonction *orderEnveloppes* a été localisée comme point central de la logique
- Création d'un enum métier : *TypeDetention* regroupe des types de détention avec ordre numérique, libellé et désignation technique.
- Développement d'une méthode de parsing : *parseLibelle* pour convertir les libellés bruts reçus de BOP en leur type métier, avec gestion des variations de casse et valeur par défaut (NOT\_A\_VALUE).

```
public TypeDetention parseLibelle(String libelle) {  
    if (StringUtilsPFI.hasLength(libelle)) {  
        return Arrays.stream(TypeDetention.values())  
            .filter(e -> e.getLibelle().equalsIgnoreCase(libelle))  
            .findFirst()  
            .orElse(NOT_A_VALUE);  
    }  
    return NOT_A_VALUE;  
}
```

- Adaptation de la fonction de tri : modification de *orderEnveloppes* pour appliquer le double critère.

Grace a ces évolutions, la navigation dans les listes de contrats est devenue plus claire et intuitive, permettant aux conseillers de gagner du temps et de réduire les erreurs de manipulation. De plus, la mise en place de l'enum permet de faciliter la tâche pour les prochains tickets du sujet.

#### 6.4.2. Popin détails offres

Dans le projet indivision, un besoin a été remonté concernant la sélection des offres de gestion (**gestion sous mandat, gestion conseillé, synoé...**). Lorsqu'un contrat est détenu par plusieurs personnes (Joint ou indivision). Il devient nécessaire de vérifier

la compétence des différents tiers rattaché au contrat avant de permettre certaines actions.

L'objectif principal était donc d'éviter que les offres inadaptées soient proposées dans des situations où l'un des tiers ne répond pas aux critères exigés. Cette évolution visait à renforcer la cohérence métier et la fiabilité du parcours client tout en respectant les contraintes de la banque.

Missions réalisées :

- a. Ajout de la gestion des contrats en indivision :
  - Création de la méthode utilitaire *isContratIndiv* dans *ZenUtils* pour identifier les contrats en indivision à partir du type de détenteur.
  - Intégration de cette logique dans *ZenRessources* pour appliquer un traitement spécifique lors de la préparation des clients visibles : parcours des indivisaires, récupération et filtrage des offres complémentaires selon l'âge, et adaptation du flux métier.
- b. Évolution des objets métiers :
  - Ajout de l'attribut *contratIndivsaire* (type boolean) dans la classe *OffresComplementsDto* pour indiquer si l'offre concerne un contrat en indivision.
  - Mise à jour de l'alimentation de cet attribut dans le flux de préparation des offres.
- c. Renforcement des contrôles de compétence et de compréhension :
  - Ajout dans *ZenCompetenceUtils* de la vérification que tous les indivisaires sont compétents ou ont compris le service d'investissement, via un parcours des clients visibles et de leurs services.
- d. Ajout d'outils de conversion et de filtrage :
  - Création de la méthode *prepareListAgeIndiv* dans *ZenUtils* pour convertir et filtrer les âges des indivisaires, facilitant le traitement des offres complémentaires.
  - Ces évolutions permettent une meilleure prise en charge des contrats en indivision, avec un traitement métier adapté, une traçabilité renforcée et une gestion fine des compétences et des offres.

## 6.5. Enrichissement des tables SQL

### 6.5.1. Tables Frais-Exante

Afin de pouvoir tester en environnement de développement le calcul des frais exante sur les produits financiers. La table réelle d'alimentation des frais ex-ante n'était pas remplie, rendant impossible l'affichage des slides frais exante dans les pdf générés.

Pour contourner ce manque, j'ai mis en place un bouchon de données via la création de 4 tables :

- Exante
- ExanteFrais
- ExanteRetrocession
- ExantePeriodeDetention

Ces tables sont reliées par la colonne ISIN et simulent le retour du webservice **EX-ANTE**.

### Les actions réalisées :

- Modélisation des entités : avec Spring Boot et Lombok pour la gestion automatique des getters/setters.
- Création des repositories en étendant **JpaRepository** pour bénéficier de l'implémentation automatique des accès aux données.
- Développement d'un contrôleur (**ExanteWs**) exposant des Endpoints simulant le webservice ex-ante, avec gestion du cas où le support n'est pas trouvé en base.
- Construction de la réponse via une interface service et son implémentation, respectant le format attendu par le front.

### Fonctionnement de Spring Data JPA :

Spring Data JPA facilite l'accès et la manipulation des données en base grâce au pattern **DAO**.

Il suffit de :

- Créer des classes d'entités qui représentent les tables de la base de données.
- Définir des interfaces repository qui étendent **JpaRepository** pour bénéficier automatiquement des méthodes **CRUD**.
- Spring se charge d'implémenter les accès aux données : il n'est pas nécessaire d'écrire le code SQL ou l'implémentation des **DAO**.

Cela permet de gagner du temps et d'assurer une architecture propre et maintenable.

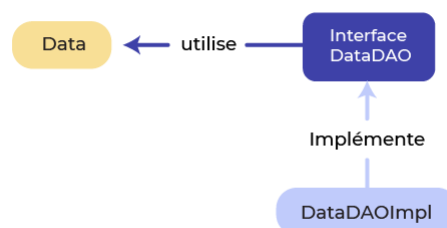


FIGURE 15 FONCTIONNEMENT DU JPARepository

### Exemple d'entity Java

Voici un exemple d'entity pour la table exante :

```
@Entity
@Table(name = "exante")
@Getter
@Setter
public class Exante implements Serializable {
    @Id
    @Column(name = "codeISIN")
    private String codeISIN;
```

```
@Column(name = "classe_actif")
private String classeActif;
@Column(name = "sens_ope")
private String sensOpe;
@Column(name = "famille")
private String famille;
@Column(name = "montant")
private Double montant;
@Column(name = "devise")
private String devise;
@Column(name = "nb_decimales")
private Integer nbDecimales;
@Column(name = "statut")
private String statut;
@Column(name = "libelle_statut")
private String libelleStatut;
@OneToOne(mappedBy = "exante" , cascade = CascadeType.ALL)
private ExanteFrais exanteFrais;
@OneToOne(mappedBy = "exante" , cascade = CascadeType.ALL)
private ExanteRetrocession exanteRetrocession;
@OneToOne(mappedBy = "exante" , cascade = CascadeType.ALL)
private ExantePeriodeDetention exantePeriodeDetention;
}
```

- **Exemple de réponse simulée**

La réponse du WS simule fidèlement le format attendu, incluant :

Les informations du support (ISIN, classe d'actif, etc.), les frais détaillés (taux et montants), les rétrocessions et la période de détention.

Cette démarche a permis de débloquent les tests et de valider le fonctionnement du module en environnement de développement, tout en respectant les bonnes pratiques de Spring Boot et de l'architecture DAO.

### 6.5.2. Colonne PFI-Version

Dans le cadre de l'amélioration de la traçabilité et du support applicatif, il a été demandé d'indiquer la version de l'application PFI ayant généré chaque étude dans la base de données en ajoutant une colonne dans la table **API\_ETUDE**.

Cette évolution permet d'identifier précisément la version utilisée lors de la création d'une étude, facilitant ainsi le suivi, l'analyse et la maintenance.

Missions réalisées :

- Évolution du schéma de base de données : Ajout de la colonne **PFI\_VERSION** (type : VARCHAR2(20)) dans la table **API\_ETUDE**, ainsi que la documentation de la colonne via un commentaire SQL.

Exemple de script :

```
ALTER TABLE API_ETUDE ADD PFI_VERSION VARCHAR2(20);  
COMMENT ON COLUMN API_ETUDE.PFI_VERSION IS 'Version applicative de PFI qui a généré l'étude.' ;  
COMMIT;
```

- Mise à jour de la couche d'accès aux données (**DAO**) : Prise en compte de la nouvelle colonne dans l'entité Java et le **DAO** associé.
- Implémentation de l'alimentation de la colonne : À la fin du parcours de génération d'une étude, j'ai identifié le point d'insertion des données en base. À ce moment précis, la version applicative de PFI est récupérée dynamiquement depuis le fichier `infoApplication.properties` et affectée à la colonne **PFI\_VERSION** lors de l'insertion dans la table **API\_ETUDE**, assurant ainsi la traçabilité de la version utilisée pour chaque enregistrement.

## 6.6. Evolutions fonctionnelles

### 6.6.1. Picto GSMD

L'un des tickets fonctionnels qui m'a été confié consistait à afficher un pictogramme spécifique **GSMD** (Gestion sous mandat déléguée) sur les contrats répondant à un ensemble de conditions métiers. Le but était de faciliter l'identification de ces contrats au sein de l'application.

Les critères métiers étaient les suivants :

- Le contrat est un contrat d'assurance vie
- Il est rattaché à l'enveloppe **SG29** (les GC ne sont pas concernés)
- Le typeCode du portefeuille est égal à **PTF10**

Pour répondre à ce besoin :

Le champ *typeCode* a été ajouté dans la table **MandatSG29** ainsi que dans le projet du bouchon **PFI-Mock**

- Les entités et mappings coté backend, ont été mis à jour utilisant Lombok pour générer les getters/setters
- Un flag *gsmdTypeCodeSg29* a été ajouté et il est calculé dynamiquement dans la classe métier
- Le front a également été modifié afin d'inclure ce flag dans la condition d'affichage du pictogramme, en complément de la règle existante.

Cette tâche m'a permis d'intervenir sur toute la chaîne technique : Backend, services, mapping, mock et affichage front.



SGGP CAPI GESTION DÉLÉGUÉE - MANDAT DYNAMIQUE	2	8 %
SGGP VIE EVOL. 1 GESTION LIBRE -	2	13 %

FIGURE 16 CONTRAT AVEC LA PICTOGRAMME GSMD

### 6.6.2. Wording conditions particulières

Dans le cadre d'un changement de raison sociale, il a été nécessaire de mettre à jour les différents documents contractuels (**CTO**, **PEA**, **ASV**) pour les parcours Banquier (PP/PM) en remplaçant toutes les occurrences de "**SG 29 Haussmann**" par "**Société Générale Investment Solutions (France)**". Cette modification impacte plusieurs sections des documents, notamment les articles spécifiques selon les codes offres, et la section signatures.

- **Explication du fonctionnement de la génération des PDF :**
  - Les Template **SVG** définissent la mise en page statique.
  - JasperReports compile les fichiers **JRXML**.
  - Les données dynamiques sont injectées via des beans Java (**JRBeanCollectionDataSource**).
  - Le PDF final est généré en fusionnant Template et données.
  - Une phase d'optimisation est appliquée pour la signature électronique.

#### Les actions réalisées :

- Mise à jour des Template **SVG** contenant les textes à modifier.
- Adaptation des espacements et mise en page pour éviter les chevauchements.
- Impact sur les codes offres : **CTO (1,2)**, **PEA (3,4)**, **ASV (5,6,7,8)**.

### 6.6.3. Conseil Coach Financier

À la suite de l'évolution des flux de BOP et de GCN, un besoin a été remonté, afin de ne pas casser la logique métier et provoquer une cascade de bugs on adapter le traitement d'une donnée métier : le coach financier. Initialement, la présence de cette donnée était fournie via un appel au web service GCN qui retournait la valeur *OfferType*.

L'objectif de la mission est de décommissionner cet appel et d'exploiter une nouvelle donnée fournie par BOP **conseilCoachFinancier** dans le bloc **Sogecap**.

Cette évolution concerne uniquement les contrats Sogecap pour les clients personnes physiques (PP) et pour les modes de gestion 00 et 39 (gestion libre). La valeur de la balise **conseilCoachFinancier** peut être : *ACTIVATED*, *COUNSEL2\_0*, *FREE*, *HISTORICAL\_PREMIUM*, ou *PREMIUM*.

Lorsque la valeur reçue est **COUNSEL2\_0**, un pictogramme **CF** doit s'afficher dans l'interface utilisateur, à la fois sur la page des avoirs financiers et sur la page arbitrage.

#### Étapes réalisées :

- Mise à jour du **Swagger** : Remplacer l'ancien swagger fournie par BOP par un nouveau avec la valeur **conseilCoachFinancier** puis on relance le build pour qu'elle soit prise en compte

- Mise à jour du modèle de données : Ajout de l'attribut **conseilCoachFinancier** dans la classe *Contrat.java* pour stocker la nouvelle donnée issue du flux BOP.

```
Private String conseilCoachFinancier
```

- Propagation de la donnée dans l'API : Dans *PortefeuilleAPIImpl.java*, ajout de l'affectation pour transmettre la donnée du backend vers l'API
- Exploitation de la donnée côté UI : Dans *EnveloppePanierUtilis.java*, adaptation de la logique pour afficher le pictogramme **CF** si la valeur reçue est **COUNSEL2\_0** :

```
enveloppePanier.setConseil20(isConseil20(contrat))
```

- Décommissionnement de l'appel GCN : Suppression du fichier *ConseilService.java* et de tout appel à ce service, puisque la donnée est désormais reçue directement dans le flux BOP.



FIGURE 17 CONTRAT AVEC PICTOGRAMME COACH FINANCIER

## 7. Difficultés Rencontrées

- Apprentissage du métier : nécessité de s'approprier rapidement les règles financiers et patrimoniaux pour bien comprendre les besoins fonctionnels
- Compréhension du code existant : difficulté liée à la taille de la complexité des applications (plus de 200.000 lignes de code et beaucoup de legacy code), avec parfois un code peu documenté.
- Courbe d'apprentissage technique : montée en compétence sur de nouveaux Framework (Angular, Spring Boot), sur les outils internes de la Société Générale, et sur l'architecture BFB, BFF et Multi-services.
- Analyse des tickets : certains tickets nécessitaient une connaissance fonctionnelle approfondie avant de pouvoir être traités.
- Tickets dépendants : impossibilité d'avancer sur certains tickets tant qu'un ticket n'était pas résolu (Sujet Indivision).
- Sécurité et conformité : obligation stricte de respecter les règles de sécurité internes
- Documentation manquante et dépassée : nécessité de reconstituer le fonctionnement de certaines parties du code à partir des logs ou en échangeant avec les collègues.
- Gestion du temps : nécessité de respecter les deadlines fixés par le sprint de chaque ticket.

## 8. Conclusion

Mon expérience d'alternance à la Société Générale a constitué une étape importante dans mon parcours professionnel sur le plan technique que personnel. Elle m'a permis

de travailler dans un environnement exigeant et stricte ou la rigueur et le sens de collaboration sont essentiels au bon fonctionnement du projet.

Sur le plan technique, j'ai eu l'opportunité de participer à des missions variées, telle que la correction de bugs, l'intégration de nouvelles fonctionnalités, l'enrichissement des bases de données, ou encore la migration vers le cloud interne. Ces réalisations m'ont permis d'approfondir mes connaissances en Java, Angular, SQL, Github et aussi d'en apprendre de nouvelles technologies (Spring Boot, SonarQube, Architecture multi-service, K8s, Jsp) et de nouvelles compétences (sécurité applicative, gestion des dépendances et optimisation des performances).

Sur le plan fonctionnel, cette alternance m'a appris les enjeux stratégiques de la transformation des systèmes d'information dans le secteur bancaire (migration des architectures en cloud, sécurisation des SI et adaptation continue des outils par rapport aux besoins des conseillers). L'un des sujets majeurs de l'année, l'indivision, m'a permis de comprendre la complexité des règles métiers propres au domaine patrimonial, tout en développant des solutions concrètes pour automatiser et fiabiliser leur traitement dans les applications existante.

Au-delà de l'aspect technique et fonctionnel, cette expérience m'a permis d'apprendre le travail en environnement agile dans une grande équipe, et à devoir s'adapter et de s'intégrer à cette équipe pour le bon déroulement du travail.

Cette alternance a été une opportunité unique de mettre en pratique mes différents acquis de mon master STL. En particulier GPSTL et PC3R se sont révélés particulièrement utiles : GPSTL pour l'organisation, la méthodologie et le suivi de travail en mode agile et PC3R pour la compréhension et la gestion des problématiques de concurrence et des applications avec une architecture multi-services. Ces enseignements m'ont fourni des bases solides qui m'ont largement aidé durant mon travail.

Enfin, cette alternance m'a aidé à clarifier mon orientation professionnelle et de prendre une décision importante pour la suite : poursuivre dans la voie du développement full stack, un domaine où je peux mobiliser mes compétences techniques, mon intérêt pour la sécurité et l'architecture logicielle, ainsi que mon appétence pour les projets complexes à forte valeur ajoutée.

## 9. Glossaire

**QPI** : Questionnaire profil investisseur

**MIF** : Marché des Instruments Financier

**BP1** : Client banque privée 1 (>200 000€ d'actifs)

**BP2** : Client banque privée 2 (>2 000 000€ d'actifs)

**EFP** : Epargne financière Banque privée

**RPF** : Entrepôt de données

**GSM** : Gestion sous mandat

**GD**: Gestion déléguée

**GSMD** : Gestion sous mandat déléguée

**GC** : Gestion conseillée

**ASV** : Assurance vie

**CTO** : Compte-Titres Ordinaire

**PEA**: Plan épargne actions

**BVI**: Base Valeur des supports

**BFF**: Back for front

**BFB**: Back for back

**HOB** : Environnement d'homologation

**PP** : Personne physique

**PM** : Personne morale

**TOSCA** : Environnement de test en dev pour logiciel TOSCA

## 10. Bibliographie

- [java.net.URLConnection Class in Java - GeeksforGeeks](https://www.geeksforgeeks.org/java/java-net-urlconnection-class-in-java/) :  
<https://www.geeksforgeeks.org/java/java-net-urlconnection-class-in-java/>

- *Finding the Redirected URL of a URL in Java | Baeldung :*  
<https://www.baeldung.com/java-find-redirected-url>
- *Do a Simple HTTP Request in Java | Baeldung :*  
<https://www.baeldung.com/java-http-request>
- *Java URLConnection and HttpURLConnection Examples :*  
<https://www.codejava.net/java-se/networking/java-urlconnection-and-httpurlconnection-examples>
- *Exemple de niveaux Log4j - Ordre, Priorité, Filtres personnalisés :*  
<https://fr.linux-console.net/?p=6159>
- *Maven Repository: Search/Browse/Explore :*  
<https://mvnrepository.com/>
- *LES FRAIS EX-ANTE C'EST QUOI ? - Groupe La Française :*  
<https://www.la-francaise.com/fr/nous-connaître/les-actualités/detail/les-frais-ex-ante-cest-quoi/>
- *Présentation - Société Générale :*  
<https://www.societegenerale.com/fr/le-groupe/presentation>
- *Société Générale Private Banking (SOCIETE GENERALE) :*  
<https://climate-transparency-hub.ademe.fr/dossier/societe-generale-private-banking-societe-generale/>
- *Société générale — Wikipédia :*  
[https://fr.wikipedia.org/wiki/Soci%C3%A9t%C3%A9\\_g%C3%A9n%C3%A9rale](https://fr.wikipedia.org/wiki/Soci%C3%A9t%C3%A9_g%C3%A9n%C3%A9rale)
- *Spring Data JPA :: Spring Data JPA :*  
<https://docs.spring.io/spring-data/jpa/reference/#jpa.query-methods>
- *Développons en Java - Les JSP (Java Server Pages) :*  
<https://www.jmdoudoux.fr/java/dej/chap-jsp.htm>
- *Java 8 vs Java 11 vs Java 17 vs Java 21 | GUVI-Blog :*  
<https://www.guvi.in/blog/java-8-vs-java-11-vs-java-17-vs-java-21-comparison/>
- *OptaPlanner - How much faster is Java 17? :*  
<https://www.optaplanner.org/blog/2021/09/15/HowMuchFasterIsJava17.html>
- *Démembrement et Indivision : Comprendre les Subtilités Juridiques | Ouestfrance-immo :*  
<https://actu.ouestfrance-immo.com/demembrement-et-indivision-comprendre-les-subtilites-juridiques.htm> o

## 11. Annexes

SOCIETE GENERALE  
Private Banking

API : Allocation Proposition d'Investissement

Recherche en date du

Personne Physique | Personne Morale

Identifiant \* ou Nom \*

Prénom

Date de naissance

Rechercher

Le lancement de l'application API doit se faire uniquement depuis le dossier client à partir de DASHBOARD.  
Lorsque l'application API est ouverte, vous pouvez revenir à ce menu en cliquant en haut à droite sur « Accueil ».

FIGURE 18 PAGE D'ACCUEIL PFI

SOCIETE GENERALE  
Private Banking

API : Allocation Proposition d'Investissement

Dossier client > MME CONJOINTE MDC1302 > MDA1282 > Sélection parcours

API : Profil supervision

Sélectionner un profil

Sélectionner un profil

Géant conseil

Banquier Privé

FIGURE 19 PAGE CHOIX DU PROFIL PFI

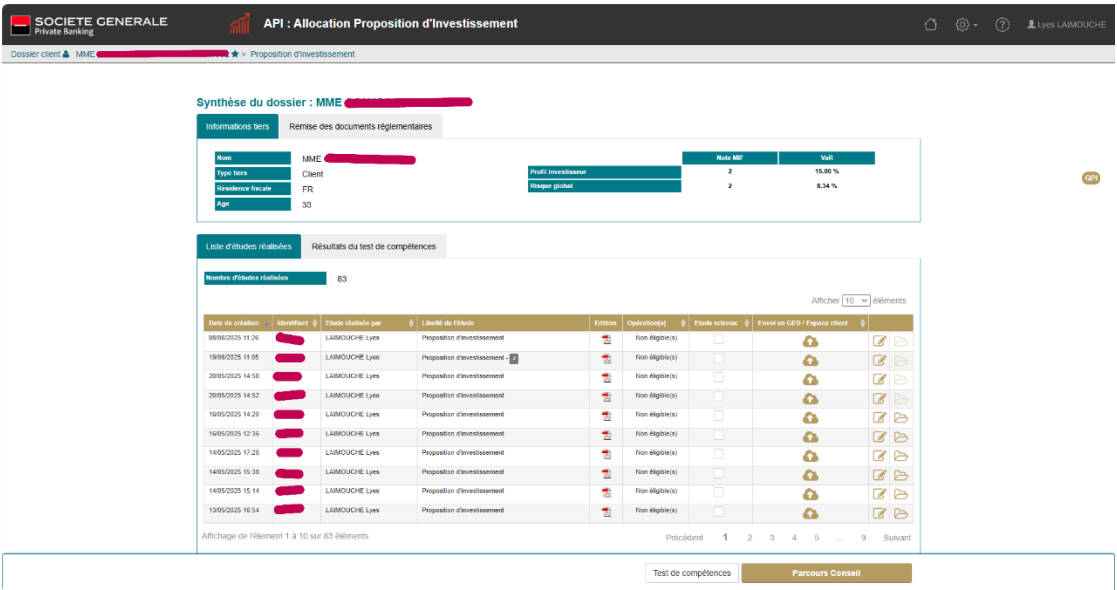


FIGURE 20 SYNTHESE PROFIL PFI



























































Compétences Financières	ESG		Objectifs et horizons	
<b>Instruments financiers</b>				
Support en euros, fonds monétaires				
Obligations et fonds / ETF Obligataires				
Obligations (fonds / ETF) convertibles				
Obligations (fonds / ETF) haut rendement, émergents				
Actions et fonds / ETF actions				
Actions (fonds / ETF) pays émergents				
Produits structurés à capital garanti				
Produits structurés à capital non garanti				
Produits financiers immobiliers				
Produits de gestion alternative, fonds spéculatifs				
Produits de Capital Investissement		Non Ev.	Non Ev.	
Produits de défiscalisation				
Matières premières et métaux précieux				
Change à terme				
Produits dérivés, ETF avec effet de levier				
<b>Services d'investissement</b>				
Gestion conseillée				
Gestion sous mandat				
<b>Assurance vie et capitalisation</b>				
Niveau de connaissance	Novice	Novice	Novice	

FIGURE 21 POPIN QPI





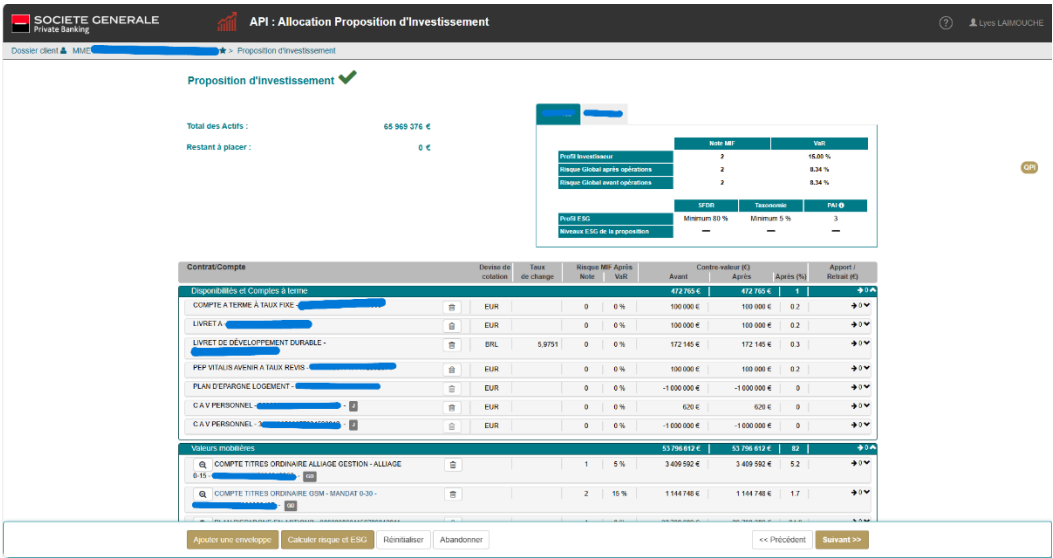


FIGURE 24 PAGE ARBITRAGE

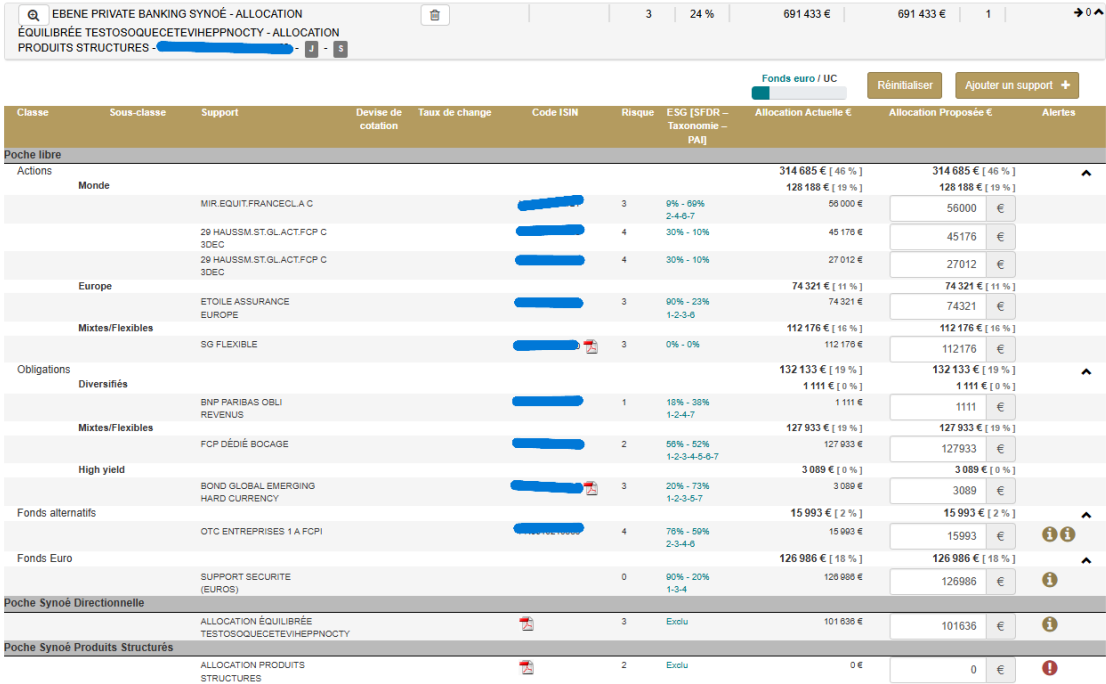


FIGURE 25 DETAIL CONTRAT

## Proposition d'investissement ✓

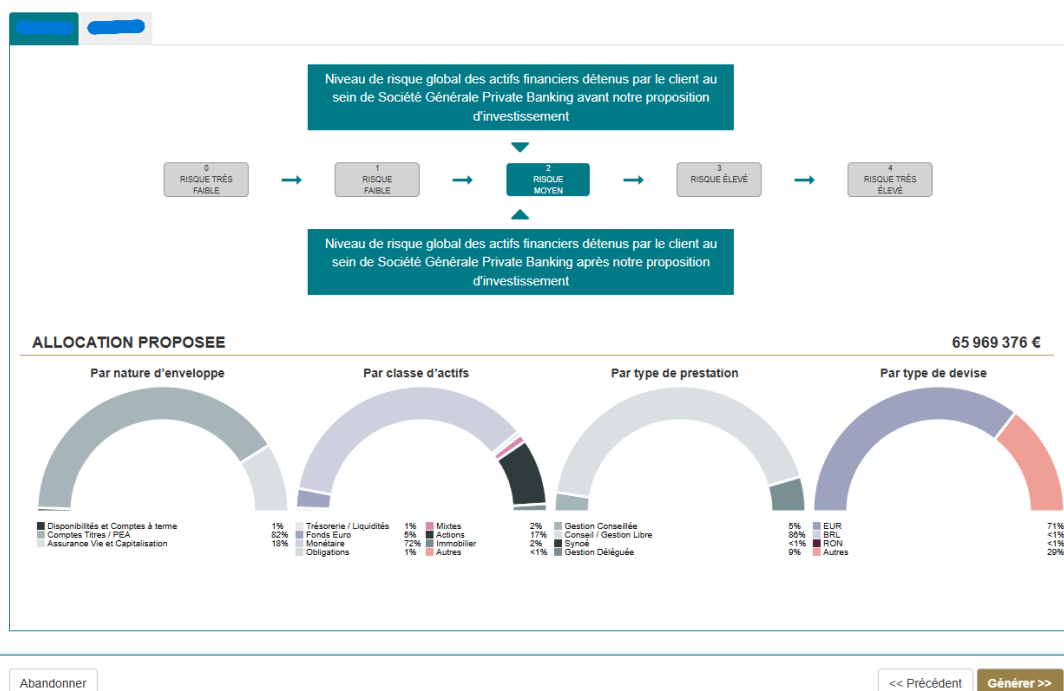


FIGURE 26 PAGE RECAPITULATIF ETUDE

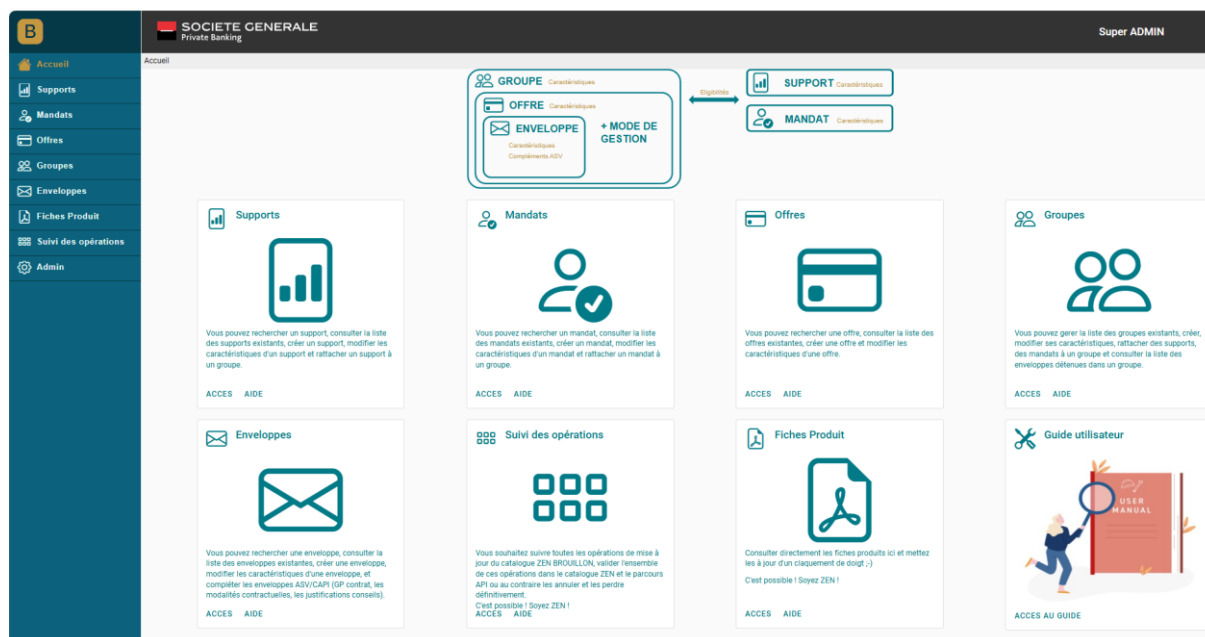


FIGURE 27 PAGE D'ACCUEIL ZEN

SOCIETE GENERALE Private banking							
Super ADMIN							
Accueil > Supports > Liste des supports							
RECHERCHE							
Code produit ou Libellé							
2778 éléments trouvés / 2778 éléments							
Libellé	Code produit	Début commercialisation	Fin commercialisation	Création	Modification		
10Y EUR FIXED RATE CALLABLE 5% 3.9% PI NCA 20082035 ADV PI		03/10/2024	30/12/2029	06/03/2023	23/07/2025		
1112-TEST-SUPPRESSION-SUPPORT		09/02/2021	31/03/2023	06/03/2023	04/06/2025		
123 CLUB PME 2016		14/04/2016	03/06/2016	21/06/2021	23/07/2025		
123 CLUB PME 2017		28/03/2022		21/06/2021	30/06/2025		
123 EXPANSION A FIP				21/06/2021	23/07/2025		
123 EXPANSION H A FIP			31/03/2023	21/06/2021	23/07/2025		
123 HOLDING ISF 2009				21/06/2021	23/07/2025		
123 ISF 2012 A FERRET ZER ZER ZA REZE				21/06/2021	16/05/2025		
123 MULTINOVA IV DYNAMI FCP				21/06/2021	15/04/2025		
29 HAUSMANN CROISSANCE EUROPE		21/01/2014	03/07/2014	21/06/2021	15/04/2025		
29 HAUSMANN EUROPE OPPORTUNITÉS				21/06/2021	23/07/2025		
29 HAUSMANN EUROPE OPPORTUNITÉS				21/06/2021	16/05/2025		
29 HAUSMANN EURO RENDEMENT		26/01/2014		21/06/2021	15/04/2025		
29 HAUSMANN EURO RENDEMENT - CAP				21/06/2021	15/04/2025		
29 HAUSMANN FLEXIBLE MONDE		03/02/2014		21/06/2021	28/07/2025		
29 HAUSMANN FLEXIBLE MULTI-STRATÉGIES		10/07/2020		21/06/2021	23/07/2025		
29 HAUSMANN FLEXIBLE STRUCTURÉS		23/09/2020		21/06/2021	15/04/2025		
29 HAUSMANN MULTI-MANAGERS PEA PME		24/12/2024		21/06/2021	15/04/2025		
29 HAUSMANN SÉLECTION EUROPE		03/02/2014		21/06/2021	16/05/2025		
29 HAUSMANN SÉLECTION FRANCE		21/01/2014		21/06/2021	15/04/2025		
29 HAUSMANN SÉLECTION MONDE		21/01/2014		21/06/2021	15/04/2025		
29 HAUSMANN SÉLECTION MONDE - CAP (I)				21/06/2021	16/05/2025		
29 HAUSMANN SÉLECTION MONDE D				21/06/2021	16/05/2025		
29 HAUSMANN SÉLECTION US		21/01/2014	23/11/2017	21/06/2021	16/05/2025		
29 HAUSMANN SÉLECTION US				21/06/2021	15/04/2025		
29 HAUSMANN SÉLECTION US				21/06/2021	16/05/2025		
36 FUND 3F GENERATIONIC				21/06/2021	23/07/2025		
7Y EUR REVERSE FLOATER CALLABLE 5.55% [5.55%-EUM] PI 200820				10/06/2025	10/06/2025		
BY PHOENIX NOTE XSXE 6.40%				21/06/2021	23/07/2025		
Affichage de 1 à 30 sur 2778							

FIGURE 28 PAGE DES SUPPORT

SOCIETE GENERALE Private banking												
Super ADMIN												
Accueil > Supports > Modification caractéristiques support												
LES DONNÉES ZEN BROUILLON												
Code Produit	Code ISIN	Libellé	10Y EUR FIXED RATE CALLABLE 5% 3.9% PI NCA 20082035 ADV PI									
		Max 60 caractères	60 / 60									
Date de début de commercialisation		Date de fin de commercialisation										
Montant minimum (€)		Montant maximum (€)										
Blocage seul		Blocage partiel										
Ratio encours max ADV/CAP (%)			5									
Message alerte			1									
			Max 300 caractères									
			1 / 300									
Classe d'actif			Autre / Non classé Industriels									
Fonds euro dédié		Exception ESG										
FICHES PRODUIT												
Fiches marketing												
Fiche réglementaire												
Tout changement de fiche produit est effectif en production immédiatement (un délai d'une heure pour être nécessaire pour qu'il soit pris en compte dans API)												
LES DONNÉES BVI												
Code Produit		Libellé	10Y EUR FIXED RATE CALLABLE 5% 3.9% PI NCA 20082035 ADV PI									
Note MBF		Horizon (mois)	12									
		Débit	EUR - EURO									
Classe d'actif		Sous-classe d'actif	STRUCTURÉE DÉCÉE									
SFR		Taux nominal										
Famille de compléance (MFV1)		Famille de compléance / composite (MFV2)	Produits structurés à capital garanti à 100%									
Fiche réglementaire DIB			Autre fiche									
Typologie client												
Retail												
Professionnelle												
Contingente éligible												
Connaissance expérience			Basique Intermédiaire Expertement									
Objectifs												
Conservation												
Croissance												
Rendement												
Couverture												
A effet de levier												
Autre												
Stratégie de distribution												
Exclusion simple												
BTO												
Conseil												
Désion												
Capacité à subir les pertes												
Aucun risque de perte en capital												
Risque de perte en capital limité												
Risque de perte en capital total												
Risque de perte au-delà du capital investi												
Risque de perte jusqu'au niveau défilé												
Le paramétrage de la gouvernance produit de ce support doit être effectué directement dans BVI												
Supprimer Annuler Valider												

FIGURE 29 CARACTERISTIQUES D'UN SUPPORT

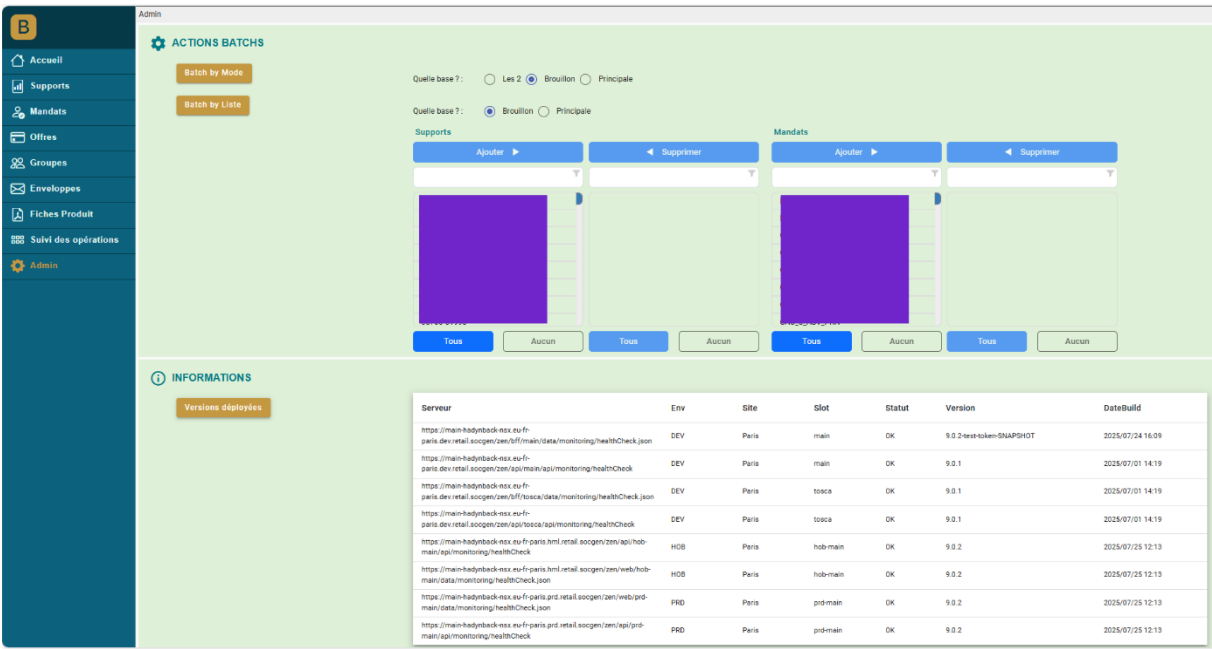


FIGURE 30 PAGE ADMIN