MICLEA Christian

ekino. 3iL INGENIEURS </les sciences informatiques> eFrei PARIS PANTHÉON-ASSAS UNIVERSITÉ

# Descriptif des missions Alternant

# Master Dev Manager FullStack

Introduction....4
Remerciements....6
**Présentation de l'entreprise....7**
Identification de l'entreprise:....7
Son corps de métier:....7
Les chiffres clés....7
Historique et évolutions clés....8
**Service d'affectation....9**
**Mon rôle....10**
**Infrastructure Matérielle et logicielle....12**
Aperçu général....12
Environnements et ressources matérielles....12
Architecture réseau....13
Plateforme logicielle principale....13
Poste de développement....14
Pipeline de tests et qualité....14
Supervision et support production....15
Conclusion....16
**Présentation succincte....17**
**Portefeuille de projets / affaires....18**
Visibilité globale....18
Outil principal....18
Processus....19
Efficacité constatée....19
Synthèse....20
**Gouvernance des projets....21**
Pilotage global....21
Chaîne de coordination quotidienne....21
Rituels et reporting....21
Affectation des ressources....22
Planification....22
Processus d'estimation et de charge....22
**Collaboration....24**
Gestion documentaire....24
Gestion de configuration & flux Git....24
Communication d'équipe....25
Accès planning & feuilles de temps....25
**Conclusion....27**
**Présentation succincte....29**
**Description du projet réalisé....30**
**Stratégie de tests....31**

Objectifs des tests....31
Moyens humains et techniques.... 31
Définition d’un plan de tests....32
Réalisation des tests....32
Intégration dans une démarche qualité....33
**Conclusion.... 33**

# Introduction

Je suis Christian MICLEA, actuellement en Master Dev Manager FullStack. j'ai enfin pu travailler dans le domaine qui m'a toujours intéressé, le développement informatique.

Mon premier foret dans l'informatique était en seconde , j'ai découvert la modélisation 3D et le développement en python et matlab en 2nde SSI, j'ai ensuite obtenu le bac STI2D ITEC avec mention Assez Bien, j'ai eu l'occasion d'avoir un avant goût du développement et de comprendre les base algorithmiques du développement informatique.

Je me suis d'abord inscrit en BTS SIO SLAM (**S**ervice **I**nformatique aux **O**rganisations **S**olutions **L**ogicielles et **A**pplications **M**étiers) où j'ai appris à développer des applications, gérer des bases de données et analyser les besoins des entreprises. J'ai également acquis des compétences en programmation, principalement en C, C++, C# et en cybersécurité, sans noter les langages orientés web comme PHP et ses frameworks, ainsi que Javascript entre autres.

Aujourd'hui, je suis en première année de Master Dev Manager FullStack, une formation qui me permet de renforcer mes compétences techniques et de management, avec un focus particulier sur la gestion de projet, l'architecture logicielle et le leadership. Mon objectif est de me préparer à des rôles de responsabilité dans le développement logiciel, où je pourrai combiner ma passion pour le code avec la gestion d'équipes techniques.

Actuellement, je travaille en tant qu'ingénieur Java Back-End chez ekino, où j'ai l'opportunité de mettre en pratique mes compétences et d'en apprendre plus en développement backend entouré de toute l'équipe d'ingénieurs ekino, notamment en travaillant sur un projet robuste et Scalable pour BPIFrance un de nos clients. Cette expérience me permet de perfectionner mon expertise technique tout en contribuant activement à des projets innovants.

# Remerciements

Je tiens d’abord à exprimer ma plus profonde gratitude aux membres de ma famille qui m’ont soutenu et encouragé tout du long que ce soit pour mes études comme pour mes fonctions une fois accepté chez ekino ainsi que pour leur bienveillance et la confiance qu’il m’ont accordé au fil des années.

Je tiens aussi à remercier mon tuteur **Jigar THAKKAR** pour son encadrement, ses conseils et son soutien continu tout au long de ma formation et de mes missions chez ekino ce qui m’apporte une réelle plus value que je constate même dans ma manière de penser au travail aujourd'hui.

Je tiens aussi à remercier **Romain FERRANTE**, **Keith AROUL**, **Florian MONTE**, et **Stéphane SILBANDE**, pour leur aide et disponibilité constante, leur partage de connaissance a grandement facilité ma progression.

# Présentation de l’entreprise

## Identification de l’entreprise:

**Nom de l’entreprise:** EKINO

**Adresse:** 9 Rue de l'Ancien Canal, 93500 Pantin

**Date de Création:** 2010

**Fondateur:** Yann Doussot et Malo Gaudry

**Site:** https://www.ekino.fr/

## Son corps de métier:

Ekino est une agence digitale spécialisée dans la transformation numérique des entreprises. Elle propose des solutions innovantes en matière de développement logiciel, d’architecture technique, et de design d’expérience utilisateur. Ekino accompagne ses clients dans la création d’applications web et mobiles performantes, évolutives et sur mesure, tout en mettant l’accent sur l’intégration de nouvelles technologies et la gestion agile des projets. L’objectif principal d’Ekino est d’aider ses clients à relever les défis numériques complexes et à créer des produits qui répondent à leurs besoins spécifiques, tout en offrant une expérience optimale à leurs utilisateurs finaux.

## Les chiffres clés

**Nombre de salariés:** Entre 250 et 499 salariés

**Chiffre d’affaires:** 117,7 M€ en 2024

**Clients:** Canal+, Renault, BpiFrance, Volkswagen, Orange,  AXA, Nexity, ina, Henkel, Crédit Agricole, Eutelsat, Idemia, Euler Hermes, Bolloré Logistics

## Historique et évolutions clés

**2010** : Création d'**Ekino** par **Yann Doussot** et **Malo Gaudry**, avec pour ambition de combiner design, technologie et stratégie digitale pour accompagner les entreprises dans leur transformation numérique.

**2013** : Ekino rejoint le groupe **Havas**, l'un des leaders mondiaux de la communication. Cette intégration permet à l'entreprise de renforcer ses synergies avec d'autres entités du groupe, comme **BETC Digital** et **Fullsix**.

**2015 - 2018** : Expansion à l'international avec l'ouverture de bureaux en **Asie (Ho Chi Minh Ville, Singapour)** et en **Inde (Bangalore)**, permettant de soutenir les projets clients à l'échelle mondiale.

**2019** : Renforcement de l'offre autour des technologies émergentes : **cloud computing, intelligence artificielle, objets connectés, et blockchain**.

**2021** : Lancement de plusieurs initiatives internes autour de l'**accessibilité numérique**, de l'**écoconception** et du **développement durable**, inscrivant Ekino dans une logique d'innovation responsable.

**2023 - 2024** : Participation à des projets d'envergure avec des entreprises comme **Renault** et **BpiFrance**, tout en consolidant son expertise dans le domaine **bancaire**, de l'**assurance**, et des **services publics**.

# Service d’affectation

Je suis actuellement affecté à l’équipe de développement Java Back-End au sein d’ekino, dans le cadre d’un projet majeur pour le compte de BPIFrance. Plus précisément, je travaille sur un projet lié au suivi et à la **gestion des PGE (Prêts Garantis par l’État)**, un dispositif mis en place pendant la crise sanitaire pour soutenir financièrement les entreprises.

Ce projet a pour objectif de fournir une plateforme **robuste et scalable** permettant la gestion, le suivi et le traitement des données liées aux prêts accordés, en garantissant la **fiabilité, la sécurité** et la **traçabilité** des opérations. L’équipe est organisée selon les principes de l’agilité, avec des rituels réguliers tels que les stand-ups, les sprint plannings et les rétrospectives. Elle est composée de développeurs back-end, front-end, d’un Product Owner et d’un Scrum Master.

Mon rôle au sein de cette équipe est principalement orienté vers le développement Java / Spring Boot, la gestion de l’intégration avec les bases de données, ainsi que la participation à la mise en œuvre de tests automatisés et à l’amélioration continue des performances applicatives.

# Mon rôle

Au sein d'ekino, je tiens la fonction d'**ingénieur Java Back-End** dans la **team BEL** – une squad agile dédiée au projet **PGE** (Prêts Garantis par l'État) pour le client **BPI France**. Cette équipe compte quatre développeurs back-end, deux développeurs front-end et deux Product Owners. Je suis rattaché directement à **Jigar Thakkar**, Lead Engineer et tuteur de mon alternance, qui assure mon suivi technique et organisationnel.

Mon rôle couvre l'ensemble du cycle de vie logiciel :

- **Conception et développement** des API REST du domaine PGE en **Java** et **Kotlin** sur **Spring Boot**, avec intégration **PostgreSQL** et orchestration **Docker/Kubernetes**.

- **Amélioration continue** de la performance et de la qualité du code : revue de pull requests, refactoring ciblé, optimisation des performances SQL et Kafka.

- **Mise en place et maintenance des tests** unitaires et d'intégration (JUnit 5, Mockito, Testcontainers), ainsi que l'enrichissement de la pipeline **GitLab CI/CD**.

- **Observabilité et production** : instrumentation et suivi via **Datadog** et **Kibana**, analyse de logs et dashboards pour garantir la stabilité des micro-services.

Mes interactions dépassent le cadre interne : je collabore quotidiennement avec les ingénieurs BPI France pour aligner les spécifications d'API, valider les contrats et organiser les déploiements. Bien que les User Stories me soient confiées en **autonomie quasi-totale**, je pratique le pair-programming ponctuel et des revues croisées pour fluidifier la transmission de connaissances et valider les solutions techniques.

Cette position, au carrefour des échanges entre les PO, les front-end, les DevOps et les équipes BPI France, me place au cœur de la **chaîne de livraison du produit** : je participe aux rituels Scrum (daily, sprint planning, revue, rétrospective) et assure, de la définition du besoin jusqu'au monitoring post-mise-en-production, une continuité technique essentielle à la réussite du projet.

# Infrastructure Matérielle et logicielle

## Aperçu général

Le projet **PGE** s'appuie sur une architecture **micro-services** containerisée qui permet de répondre aux exigences de robustesse et de scalabilité fixées par **BPI France**. Les environnements (développement, recette, production) sont tous orchestrés par **Kubernetes**, avec une chaîne CI/CD GitLab permettant de déployer automatiquement chaque micro-service après validation des tests.

## Environnements et ressources matérielles

| Environnement | Hébergement | Ressources principales | Usage |
|---|---|---|---|
| **Dev** | Cluster Kubernetes géré par ekino (EKS / AKS selon région) | 3 nœuds vCPU 4 – 8 Go RAM | Développement local, tests rapides, intégration continue |
| **Recette (UAT)** | Cluster dédié BPI France | 4 nœuds vCPU 8 Go RAM | Vérifications métier, homologation sécurité, démonstrations client |
| **Production** | Cluster Kubernetes BPI France (on-prem + cloud privé OVH) | 6 nœuds vCPU 16 Go RAM | Exploitation 24/7, haute disponibilité, PRA |

## Architecture réseau

- **Segmentation stricte** entre les VPC/namespaces ekino et ceux de BPI France.
- **Interconnexion sécurisée** assurée par **GlobalProtect** (solution VPN de Palo Alto Networks) : tunnel IPSec chiffré, authentification SAML + MFA côté BPI France.
- Trafic est-ouest (service-à-service) géré par **Istio Service Mesh** : quotas, rate-limiting et policies mutual TLS activés.
- Exposition publique via **API Gateway** sécurisée (WAF + OAuth 2 / Keycloak) ; terminaison TLS au load-balancer de l'infrastructure BPI France.

## Plateforme logicielle principale

| Couche | Technologies / outils | Rôle |
|---|---|---|
| **Langages** | Java 21, Kotlin, (Python 3 pour scripts ponctuels) | Développement business et scripts d'automatisation |
| **Framework** | Spring Boot 3, Spring Cloud, OpenFeign | Micro-services REST, clients inter-service |
| **Messaging** | **Kafka** (5 nœuds) + Schema Registry | Asynchronisme, diffusion d'événements PGE |
| **Base de données** | PostgreSQL 15 (cluster patroni) | Stockage transactionnel des prêts |

| | | |
|---|---|---|
| **CI / CD** | GitLab + GitLab Runner (auto-scalés) | Pipelines multi-stages orchestrant **Maven** : build → test → sonar → package → deploy … |
| **Observabilité** | **Datadog APM**, **Kibana** (Elastic Stack) | Logs, metrics, traces distribuées, alerting |
| **Sécurité** | Vault pour secrets, SonarQube, Snyk | Gestion des secrets, analyse statique & vulnérabilités |
| **Build & dépendances** | **Maven** (mvnw) + **JaCoCo** (plugin Maven) | Compilation Java/Kotlin, gestion des dépendances, **génération des rapports de couverture** intégrés ensuite dans SonarQube |

## Poste de développement

- **IDE :** IntelliJ IDEA Ultimate (plugins Spring, Docker, Kotlin, SonarLint, Maven).
- **Containerisation locale :** Docker Desktop + Tilt pour le hot-reload.
- **Outillage quotidien :** GitLab Issues, Confluence, Slack/Teams pour la communication, Miro pour le cadrage fonctionnel.

### Pipeline de tests et qualité

1. **Tests unitaires** (JUnit 5, Mockito) lancés à chaque *push*.
2. **Tests d'intégration** (Testcontainers, embedded PostgreSQL/Kafka) exécutés en CI.
3. **Analyse statique** (SonarQube) : seuil « Quality Gate » bloquant si > 0 bug critique.
4. **Scan dépendances** avant déploiement.
5. **Tests end-to-end** lancés sur l'environnement UAT via **Cypress** + rapport Allure.
6. **Pipeline de tests et qualité** Le stage build de GitLab CI déclenche la commande « ./mvnw clean package » ; les stages suivants (tests, analyse Sonar, packaging Docker, etc.) s'appuient sur les artefacts générés par Maven.

## Supervision et support production

- **Tableaux de bord Datadog** : latence API, taux d'erreurs, saturation CPU/RAM, temps de GC.
- **Alertes** : seuils Dynamiques, escalade vers PagerDuty, rotation d'astreinte partagée entre ekino et BPI France.
- **Logs centralisés** : Filebeat → Logstash → Elastic ; recherche via Kibana, corrélation avec traces APM.

## Conclusion

Cette infrastructure combine la **flexibilité d’un environnement cloud-natif** (Kubernetes, CI/CD automatisée) et le **contrôle renforcé** exigé par un client bancaire comme BPI France (réseaux cloisonnés, monitoring exhaustif, pipeline Sécurité-by-design). Elle offre ainsi un socle robuste à la plateforme PGE tout en permettant une **évolution continue** du produit grâce à des mécanismes d’automatisation et de supervision avancés.

# Présentation succincte

Ekino est une agence digitale du groupe Havas fondée en 2010, spécialisée dans la transformation numérique. Elle conçoit et développe des plateformes web et mobiles sur mesure, combinant design UX, architecture logicielle et intégration technologique. L'entreprise, située à Pantin (9 Rue de l'Ancien Canal), compte entre 250 et 499 collaborateurs et accompagne des clients majeurs comme Canal+, Renault, BPI France ou encore Orange.

Je suis actuellement alternant en tant qu'ingénieur Java Back-End au sein de la squad BEL dédiée au projet PGE (Prêts Garantis par l'État) pour BPI France. L'équipe agile comprend quatre ingénieurs back-end, deux front-end, deux Product Owners, un Scrum Master, et est dirigée techniquement par mon tuteur Jigar Thakkar.

Mon rôle couvre tout le cycle de vie des micro-services liés aux PGE : conception et développement d'API REST en Java 21 et Spring Boot 3, intégration PostgreSQL, orchestration Docker/Kubernetes, automatisation des tests unitaires et d'intégration (JUnit 5, Mockito, Testcontainers), intégration continue et déploiement via GitLab CI/CD avancée, ainsi que la supervision et le monitoring de la production grâce aux outils Datadog et Kibana.

Cette position centrale dans la chaîne technique du projet m'offre autonomie et responsabilité directe sur les User Stories, ainsi qu'une forte exposition aux enjeux techniques et fonctionnels propres au secteur bancaire.

# Portefeuille de projets / affaires

## Visibilité globale

Sur le projet **PGE**, la **vision d'ensemble** est détenue par les deux **Product Owners** (fonctionnel & technique). Ils alimentent et organisent le backlog dans un **espace Notion** partagé avec toute l'équipe ; les Leads Delivery d'ekino et, côté client, le représentant BPI France y disposent d'un droit de consultation pour suivre l'avancement.

## Outil principal

| **Période** | **Outil** | **Fonctionnalités exploitées** | **Rôle dans la visibilité** |
|---|---|---|---|
| Jusqu'à fin 2024 | **Jira Software** | Tableaux Scrum, vélocité automatique, rapports burn-down | Suivi standard Agile (story points, burndown). |
| Depuis janv. 2025 | **Notion** (base de données Kanban + vues Sprint) | • Propriétés personnalisées : *Status*, *Story Points*, *Priorité*, *Date cible*.• Vues « *Backlog Grooming* », « *Sprint Board* », « *Done* ». | Backlog unique accessible à tous ; tri & filtres pour la planification. |

## Processus

- **Backlog** – Les POs créent ou déplacent les tickets Notion lors des séances *refinement*.

- **Choix des tâches** – Pendant la *sprint planning* ou les *dailys* (stand-up), chaque ingénieur “se sert” dans la pile *To Do* selon son affinité technique ou bien est attribué par défaut son ticket si c’est sur la continuité d’un précédent ticket.

- **Suivi** – L’avancement est discuté quotidiennement en *daily*, puis formalisé en *retro* à la fin du sprint. Les écarts prévus/réels sont calculés manuellement.

## Efficacité constatée

| | |
|---|---|
| **Accessibilité** | Notion centralise backlog, docs fonctionnelles et RACI ; visible par ekino & BPI France. |
| **Flexibilité** | Ajout/édition de champs à la volée, templates réutilisables par squad. |
| **Autonomie équipe** | Pull-system : chacun choisit ses tickets → meilleur engagement et répartition naturelle des expertises. |
| **Gestion des écarts** | Roll-ups & tableur suffisent pour un suivi basique (story points). |

## Synthèse

La migration vers **Notion** a **simplifié la collaboration** (backlog + documentation réunis) et renforcé l’engagement grâce au système de sélection libre des tickets.

# Gouvernance des projets

Sur le projet **PGE** réalisé pour **BPI France**, la gouvernance est volontairement légère et très orientée client :

## Pilotage global.

Aucune cellule PMO interne n'a été identifiée côté ekino ; les grandes orientations et la priorisation des lots fonctionnels sont décidées directement par le **comité projet BPI France**. Lorsque le client valide un nouveau besoin, il le confie au Product Owner fonctionnel, qui le traduit aussitôt en tickets Notion puis l'intègre au backlog de la squad.

## Chaîne de coordination quotidienne

L'activité opérationnelle est orchestrée par le tandem **Product Owners / Lead Engineer**. Jigar Thakkar, Lead Engineer, supervise la cohérence technique et sert de premier point d'escalade ; c'est également à lui que je rends compte de l'avancement de mes tâches. Les Product Owners, eux, assurent le reporting fonctionnel et la démonstration des incréments auprès de BPI France.

## Rituels et reporting

La gouvernance formelle repose exclusivement sur les cérémonies **Scrum**

**Daily** : suivi de l'avancement et détection des blocages.

**Sprint Planning** et **Refinement** : projection de la charge et clarification des prochains tickets.

**Sprint Review** lorsque des livrables sont prêts, pour présenter le travail au client.

**Rétrospective** : réflexion sur l'amélioration continue.

Aucun steering mensuel, quarterly business review ni reporting financier n'a pour l'instant été mis en place au niveau de l'équipe technique.

## Affectation des ressources

La composition de la squad (quatre back-end, deux front-end, deux PO) est restée stable depuis le démarrage et aucun conflit de staffing n'a encore été rencontré ; le mécanisme officiel d'arbitrage entre projets ekino n'a donc pas été sollicité à ce jour.

### Planification

Nos itérations suivent un **cadre Scrum** parfaitement régulier :

- **Durée des sprints** : 2 semaines fixes, du lundi matin au vendredi soir J-10.
- **Outil** : un **tableau Notion** disposant d'une vue regrouppant *Backlog* et *Sprint Board*

## Processus d'estimation et de charge

1. **Sprint Planning** : chaque ticket éligible est présenté par le Product Owner (PO).
2. **Auto-estimation** : le **développeur qui prend la tâche évalue lui-même** le nombre de Story Points (SP) après discussion rapide avec le PO et le Lead Engineer.
3. **Capacité** : la somme des SP retenus est comparée à la vélocité moyenne des trois derniers sprints ; en cas de dépassement, le PO ajuste le scope.
4. **Suivi** : la progression est visualisée grâce à un **roll-up Notion** qui totalise les SP « Done ».

Le **Lead Engineer** joue un rôle d'arbitre technique : il valide la faisabilité des estimations élevées et, si un risque de dérive apparaît, propose un recalage (déplacement de tickets ou découpage plus fin).

# Collaboration

## Gestion documentaire

Tout le contenu fonctionnel et technique est **centralisé dans Notion** (pages specs, ADR, comptes-rendus). L'absence d'outils multiples limite la dispersion et facilite la recherche.

## Gestion de configuration & flux Git

**GitLab** héberge les dépôts :

1. Développement sur branche feature/* → MR vers preprod (tests CI) → revue & approbation (≥ 1) → merge main.
2. Création d'une **Merge Request** vers main nécessitant **au moins un approbateur** (souvent le Lead Engineer).
3. **Merge vers preprod** dès qu'une fonctionnalité est prête ; une pipeline d'intégration tourne automatiquement.
4. **Tests en preprod** avant validation finale de la feature et autorisation de merge

Ce double passage garantit à la fois des tests d'intégration précoces et une revue de code systématique.

## Communication d’équipe

Les échanges passent quasi exclusivement par **Slack** :

- **Canaux projet** : #bpifrance_bel et #team-bpi-brp-pge-garantie-dgl rassemblent l’ensemble de la squad (ekino + BPI France) pour le suivi opérationnel quotidien.

- **Canaux techniques** : #team-bpi-brp-tech-back pour les questions d’implémentation et #team-bpi-brp-tech-back-code-reviews pour les revues de pull requests ; #team-java sert de guilde interne ekino pour la veille technologique.

- Chaque **User Story** possède son fil de discussion dédié dans Notion afin de conserver l’historique des décisions.

- Les besoins de pair-programming ou de déblocage rapide se règlent via des appels « Slack Huddle ».

- Les échanges externes avec BPI France passent par courriel ou réunions Teams planifiées.

## Accès planning & feuilles de temps

- **Board Notion** : visible en lecture/écriture par toute l'équipe et par les parties prenantes BPI France.

- **Pointage sur Napta** : chaque développeur déclare ses heures quotidiennement ; les exports hebdomadaires servent de base au suivi de charge et à la facturation.

Ces pratiques garantissent une **planification souple mais contrôlée** et une **collaboration fluide** : un seul outil pour le backlog, un workflow GitLab clair pour le code, Slack pour la communication instantanée et Napta pour la réalité des temps passés – autant de briques cohérentes qui soutiennent la performance de la squad.

# Conclusion

Les outils et pratiques adoptés sur le projet **PGE** forment une chaîne de gestion de projet simple mais efficace : le backlog unifié dans **Notion** offre une visibilité immédiate sur les User Stories ; le workflow **GitLab → preprod → main** encadre la qualité du code ; **Slack** fédère les échanges techniques et fonctionnels ; enfin **Napta** consigne le temps passé. Ce socle léger favorise l'engagement des développeurs (auto-sélection des tickets, feedback continu) tout en donnant au Product Owner une vue claire de la vélocité et du reste à faire.

Pour aller plus loin, trois pistes d'amélioration se dégagent :

1. **Automatiser le reporting temps / coût**

   Connecter **Napta** ou Harvest à Notion (via Zapier ou l'API native) permettrait de remonter automatiquement le temps réel sur chaque User Story et de suivre les écarts budgétaires sans recours aux exports manuels.

2. **Industrialiser les indicateurs d'avancement**

   Remplacer le burndown « Google Sheets » par un tableau de bord Looker Studio / Power BI alimenté par l'API Notion donnerait une consolidation instantanée (SP consommés, vélocité, RAG) visible par le client et la direction Delivery.

3. **Créer un tableau de bord multi-squads**

   Si d'autres équipes ekino interviennent sur le programme BRP, un "Portfolio View" transverse (Notion Dashboards ou un module PMO léger) aiderait à anticiper les conflits de ressources et à partager les risques entre squads.

# Description du projet réalisé

Le projet PGE vise à mettre en œuvre une plateforme robuste et scalable destinée à la gestion et au suivi des prêts garantis par l'État, un dispositif financier essentiel pour soutenir les entreprises en difficulté post-crise sanitaire.
La plateforme repose sur une architecture micro-services, conteneurisée et orchestrée via Kubernetes, avec des déploiements continus automatisés (GitLab CI/CD). Les services échangent principalement via Kafka, et les données sont stockées dans des clusters PostgreSQL. L'environnement de production, hébergé sur un cloud privé hybride, impose des contraintes fortes en matière de sécurité, de traçabilité et de fiabilité.
Le projet est conduit selon une méthodologie Scrum, avec des sprints de deux semaines, un backlog géré sur Notion, et des échanges synchrones via Slack, Teams et des daily stand-ups. Chaque fonctionnalité passe par une phase de développement, de validation technique, de recette fonctionnelle, puis de mise en production planifiée.
Ma contribution se situe principalement sur :

- le développement des micro-services liés à la gestion des PGE,
- l'écriture des tests automatisés (unitaires et d'intégration),
- la configuration des pipelines de déploiement,
- la surveillance post-prod et le diagnostic des anomalies.

# Stratégie de tests

## Objectifs des tests

La stratégie de tests vise à garantir la stabilité, la qualité du code, et la conformité fonctionnelle des micro-services, en s'alignant sur les exigences métier strictes du secteur bancaire. Elle s'inscrit dans une logique "shift-left" avec un maximum de validations en amont, dès le push des développeurs.

## Moyens humains et techniques

**Ressources humaines** : chaque développeur est responsable des tests unitaires et d'intégration de ses fonctionnalités. La QA métier est assurée côté BPI France lors des phases UAT.

**Outils :**

- JUnit 5, Mockito : pour les tests unitaires.
- Testcontainers : pour simuler des environnements PostgreSQL et Kafka en CI.
- Cypress : pour les tests end-to-end lancés sur l'environnement UAT.
- GitLab CI/CD : pour l'automatisation de l'ensemble des tests.
- SonarQube + Snyk : pour l'analyse statique et la détection de vulnérabilités.
- Allure : pour la génération des rapports de tests fonctionnels.

## Définition d’un plan de tests

La stratégie suit un plan structuré :

1. Phase locale : exécution des tests unitaires avant push.
2. CI GitLab : exécution automatique des tests d’intégration et analyse statique.
3. UAT : validation métier et sécurité par les équipes BPI France.
4. Mise en production : déclenchée uniquement si toutes les étapes sont validées.

Chaque User Story est livrée avec un plan de tests associé : cas nominaux, cas limites, erreurs attendues. Les résultats sont archivés via GitLab et Allure.

## Réalisation des tests

- Les tests sont exécutés à chaque push, à chaque MR, puis à chaque merge vers la branche main.
- Les tests unitaires couvrent les logiques métiers (validation, mapping, contrôleurs).
- Les tests d’intégration valident la communication avec les bases de données et Kafka.
- Les bugs détectés sont remontés dans Notion et corrigés dans le sprint courant ou suivant.

- Le “Quality Gate” SonarQube impose zéro bug critique pour accepter une merge request.
- Le temps consacré aux tests est estimé entre 20 et 30 % du sprint.

### Intégration dans une démarche qualité

Les tests sont pleinement intégrés à une démarche qualité continue :

- Revue de code obligatoire (≥ 1 approbateur).
- Linting et scan de vulnérabilités automatisés.
- Dashboards Datadog pour la surveillance post-prod.
- Pas de mise en prod sans validation complète du pipeline CI/CD.
- En cas de faille détectée, la correction est priorisée dans le backlog.

## Conclusion

La stratégie de tests mise en place sur le projet PGE s’inscrit dans une démarche DevOps mature, qui combine automatisation, collaboration et qualité logicielle. L’usage d’outils modernes comme Testcontainers, Cypress ou SonarQube, intégré à GitLab CI/CD, permet de garantir un haut niveau de confiance dans les livrables.

Cette approche permet non seulement de détecter rapidement les erreurs, mais aussi d’améliorer la résilience du système en production. Elle illustre comment une stratégie de tests bien pensée peut devenir un véritable levier de performance, de transparence et de fiabilité, surtout dans un contexte aussi exigeant que celui du secteur bancaire.