Faculté des Sciences Dhar El Mehraz de Fès

Département d'Informatique
Master Qualité du Logiciel
2024-2025

Rapport d'état d'avancement N° 1 du 12/05/2025

Hmidani Filali Amine, CGI TECHNOLOGIES ET SOLUTIONS – FÈS

# Conception et mise en œuvre d'un système intelligent de détection des données personnelles et d'anonymisation adaptée : application aux bases de données pour la conformité réglementaire

RÉALISÉ PAR : **Hmidani Filali Amine**

ENCADRÉ PAR : **M. CHENFOUR NOUREDDINE**

ENCADRANT À L'ENTREPRISE : **Bodor Ismail**

# Table des matières

Table des matières ..... 2
**1. Résumé du travail** ..... 4
1.1. Introduction ..... 4
1.2. Travail et tâches réalisées ..... 4
**2. Formation, Méthodologie et Processus** ..... 5
2.1. Introduction ..... 5
2.2. Formations ..... 5
2.2.1. Formations CGI ..... 5
2.2.2. Formations techniques spécifiques ..... 6
2.3. Méthodologie et Processus ..... 6
2.3.1. Introduction ..... 6
2.3.2. Approche Itérative ..... 7
2.3.3. Cycles de Développement ..... 7
2.3.4. Événements et Rituels ..... 8
**3. Technologies DevOps utilisées** ..... 8
3.1. Introduction ..... 8
3.2. Git et GitLab ..... 9
3.3. Maven ..... 10
3.4. Podman ..... 11
3.5. SonarQube et SonarLint ..... 12
**4. Technologies et outils utilisés** ..... 13
4.1. Introduction ..... 13
4.2. Java ..... 14
4.3. Spring ..... 14
4.4. Python (NER Service) ..... 15
4.5. PostgreSQL ..... 16

4.6. React JS ..... 17
4.7. IntelliJ IDEA et Visual Studio Code ..... 18
4.8. JUnit et Mockito ..... 19
4.9. Postman ..... 19
**5. travail individuel** ..... 20
**6. Composition de l'équipe et tâches attribuées** ..... 21
**7. Cahier des charges détaillé** ..... 21
7.1. Contexte et objectifs ..... 21
7.1.1. Contexte réglementaire ..... 21
7.1.2. Données personnelles et RGPD ..... 22
7.1.3. Défis actuels des entreprises ..... 23
7.1.4. Objectifs du projet PrivSense ..... 23
7.2. Architecture du projet ..... 26
**7.3.** Fonctionnalités actuelles ..... 27
7.3.1. Connexion aux bases de données ..... 27
7.3.2. Extraction de métadonnées ..... 28
7.3.3. Échantillonnage de données ..... 28
7.3.4. Détection de PII ..... 28
7.3.5. Génération de rapports ..... 29
7.4. Fonctionnalités à développer ..... 30
7.4.1. Module d'anonymisation ..... 30
7.4.2. Intégration API REST ..... 30
**8. Avancement et état actuel** ..... 31
8.1. État d'avancement du projet ..... 31
8.2. Résultats obtenus ..... 31
8.3. Difficultés rencontrées et solutions ..... 32
8.4. Prochaines étapes ..... 32
**9. Conclusion** ..... 34
Table des figures ..... 35

# 1. Résumé du travail

## 1.1. Introduction

Dans le cadre de notre stage de fin d'études chez **CGI**, nous avons commencé par une formation sur le code d'éthique et la culture de l'entreprise. Ensuite, en concertation avec mon encadrant, nous avons défini les besoins pour **PrivSense**, un projet interne destiné à renforcer la conformité au **RGPD**. Nous avons réalisé une étude fonctionnelle et technique, établi un cahier des charges, puis conçu et lancé le développement de la solution. Actuellement, nous travaillons sur la mise en place de son module d'anonymisation.

## 1.2. Travail et tâches réalisées

| Tâche | Statut |
|---|---|
| Formation sur le code d'éthique et la culture CGI | **Terminée** |
| Réunion de cadrage avec l'encadrant pour définir le besoin | **Terminée** |
| Réalisation de l'étude fonctionnelle et technique | **Terminée** |
| Rédaction du cahier des charges | **Terminée** |
| Élaboration du macroplanning | **Terminée** |
| Conception et définition de l'architecture de la solution | **Terminée** |

| | |
|---|---|
| Développement des modules Core, Connector, Metadata Extractor, Sampler, PII Detector et Reporter | **Terminée** |
| Présentation de la version MVP aux directeurs CGI | **Terminée** |
| Début du développement du module d'anonymisation | **En cours** |

# 2. Formation, Méthodologie et Processus

## 2.1. Introduction

La réussite du projet PrivSense repose sur une combinaison de formations adaptées et d'une méthodologie de travail structurée. Cette section détaille les formations suivies en début de mission ainsi que l'approche méthodologique adoptée pour gérer efficacement le développement de cette solution de détection et d'anonymisation de données personnelles.

## 2.2. Formations

### 2.2.1. Formations CGI

En début de stage, plusieurs formations m'ont été dispensées afin de m'intégrer dans le cadre de travail et les standards de CGI. Celles-ci couvraient à la fois des aspects réglementaires, éthiques et techniques, particulièrement importants dans le contexte d'un projet portant sur la protection des données personnelles.

Sur le plan institutionnel, j'ai suivi les modules suivants :

- Sensibilisation à la sécurité de l'information
- Code d'éthique et de conduite professionnelle de CGI
- Politique anti-corruption mondiale

- Sensibilisation au numérique responsable
- Principes fondamentaux de la protection des données personnelles et de la gestion de l'information
- Formation spécifique sur le RGPD et ses implications techniques
- Bonnes pratiques en matière de traitement de données sensibles

Ces formations étaient particulièrement pertinentes pour le projet PrivSense, dont l'objectif principal est justement la détection et l'anonymisation des données personnelles dans les bases de données et APIs REST.

### 2.2.2. Formations techniques spécifiques

Pour répondre aux exigences techniques du projet, j'ai également suivi des formations spécifiques :

- Une autoformation sur l'implémentation d'un projet Maven multi-modules pour une architecture modulable et une séparation claire des responsabilités
- Une formation sur les techniques d'anonymisation et de pseudonymisation des données
- Une formation sur l'intégration de la bibliothèque ARX tools pour nos besoins d'anonymisation et d'analyse des risques
- Une autoformation sur des projets existants de l'équipe CGI France, comme PIMO : Private Input, Masked Output

Ces formations techniques ont permis d'acquérir rapidement les compétences nécessaires à la mise en œuvre des différents modules du projet PrivSense, en respectant les standards de qualité et de sécurité propres à CGI.

## 2.3. Méthodologie et Processus

### 2.3.1. Introduction

Le projet PrivSense est géré selon une approche itérative et incrémentale inspirée des méthodes agiles, mais adaptée au contexte d'une équipe réduite composée d'un développeur

stagiaire (moi-même) et d'un encadrant (Architecte). Cette organisation favorise une livraison progressive des fonctionnalités, une adaptation constante aux besoins du projet interne, et une communication directe et efficace.

### 2.3.2. Approche Itérative

L'approche itérative adoptée repose sur un ensemble de principes visant à rendre le développement logiciel plus flexible et réactif. Elle met l'accent sur :

- La livraison fréquente de versions fonctionnelles
- L'adaptation continue du produit en fonction des retours
- La communication directe entre le stagiaire et l'encadrant
- La priorisation régulière des fonctionnalités à développer

Cette approche a permis d'évoluer rapidement dans le développement du système PrivSense tout en s'assurant que chaque composant répond effectivement aux besoins de CGI.

### 2.3.3. Cycles de Développement

Le développement a été structuré autour de cycles de deux semaines, comprenant quatre phases distinctes :

- **Phase d'analyse et conception** : Durant cette première étape, je définis les spécifications techniques des fonctionnalités à développer, j'étudie les solutions possibles et je prépare les modèles et diagrammes nécessaires.
- **Phase de développement** : C'est la période durant laquelle le code est effectivement produit, en suivant l'architecture définie et en implémentant les fonctionnalités planifiées.
- **Phase de tests** : Cette phase consiste à vérifier le bon fonctionnement des composants développés par des tests unitaires et d'intégration, ainsi qu'à corriger les éventuels problèmes identifiés.

- **Session de revue** : À la fin de chaque cycle, une réunion avec l'encadrant permet de valider les développements réalisés et d'ajuster les priorités pour le cycle suivant.

### 2.3.4. Événements et Rituels

Malgré la taille réduite de l'équipe, plusieurs rituels ont été mis en place pour structurer le travail et assurer un suivi efficace du projet :

- **Points quotidiens** : En cas de blocage technique, des réunions brèves sont organisées avec l'encadrant pour résoudre rapidement les problèmes rencontrés.
- **GitLab Issues** : Utilisation systématique de l'outil pour documenter chaque fonctionnalité à développer, suivre son avancement et garder une trace des décisions techniques.
- **Présentations hebdomadaires** : Chaque vendredi, une démonstration des fonctionnalités développées durant la semaine est réalisée devant l'encadrant. Cette session permet de recueillir des retours immédiats, d'ajuster les priorités pour la semaine suivante et de résoudre collectivement les problèmes rencontrés.

Ces événements, bien qu'adaptés à une équipe de deux personnes, permettent de maintenir une dynamique de projet structurée, une visibilité continue sur l'avancement et une amélioration régulière du produit développé.

# 3. Technologies DevOps utilisées

## 3.1. Introduction

Pour assurer un développement efficace du projet PrivSense, nous avons mis en place un ensemble d'outils DevOps facilitant la collaboration, la gestion du code source, et le maintien d'une haute qualité de code. Ces outils ont été sélectionnés pour répondre aux exigences de qualité et de sécurité du projet, tout en respectant les contraintes et politiques de CGI.

## 3.2. Git et GitLab

Le système Git a été adopté comme outil de gestion de versions pour le projet PrivSense. Nous avons utilisé ses fonctionnalités de branches pour organiser le développement des différents modules du système.

La stratégie de branches suivante a été mise en place :

- **main** : Branche principale contenant le code stable et validé
- **develop** : Branche d'intégration des fonctionnalités en cours de développement
- **feature/xxx** : Branches spécifiques pour chaque nouvelle fonctionnalité
- **fix/xxx** : Branches dédiées à la correction des bugs

*Figure 1 : Logo GIT*

GitLab a servi de plateforme collaborative pour héberger le dépôt Git et gérer le cycle de vie du projet. Les fonctionnalités utilisées comprennent :

- **GitLab Issues** : Pour le suivi des tâches et la documentation des fonctionnalités
- **Merge Requests** : Pour la revue de code et l'intégration des nouvelles fonctionnalités
- **Wiki** : Pour la documentation technique du projet

Cette configuration nous a permis de maintenir un historique clair des évolutions du projet tout en facilitant le développement parallèle des différents composants du système PrivSense.

*Figure 2 : Logo Gitlab*

## 3.3. Maven

Maven a joué un rôle central dans la gestion de notre projet Java multi-modules. Grâce à sa structure de Project Object Model (POM), nous avons pu :

- Organiser efficacement notre architecture modulaire :
  - privsense-core
  - privsense-db-connector
  - privsense-metadata-extractor
  - privsense-sampler
  - privsense-pii-detector
  - privsense-reporter
  - privsense-api
- Gérer les dépendances de manière cohérente entre les modules
- Automatiser les processus de compilation, tests et packaging
- Standardiser le cycle de vie du développement

Le fichier POM principal définit l'architecture globale du projet, tandis que chaque module possède son propre POM qui hérite des propriétés communes tout en spécifiant ses dépendances particulières.

*Figure 3 : Logo Maven*

## 3.4. Podman

Podman a été choisi pour la conteneurisation des services du projet PrivSense, facilitant leur déploiement et leur intégration. Nous avons opté pour Podman plutôt que Docker en raison de sa nature open-source et de son approbation explicite pour une utilisation dans notre projet au sein de CGI.

Deux composants principaux ont été conteneurisés :

**Service NER (Named Entity Recognition)**

Le conteneur Podman encapsule l'environnement Python nécessaire au fonctionnement des modèles d'apprentissage automatique utilisés pour la détection avancée des données personnelles.

Le fichier Containerfile présent dans le répertoire ner-service définit les étapes de construction de l'image, incluant :

- L'installation des dépendances Python requises
- Le chargement des modèles de détection d'entités
- La configuration du serveur web qui expose l'API REST

**Base de données PostgreSQL**

La base de données PostgreSQL utilisée pour stocker les résultats des détections et les configurations a également été conteneurisée, offrant plusieurs avantages :

- Isolation complète de l'environnement de base de données

- Portabilité entre différents environnements de développement et de test
- Configuration reproductible via des scripts d'initialisation
- Persistence des données via des volumes montés
- Facilité de sauvegarde et de restauration

La configuration du conteneur PostgreSQL comprend :

- La définition des paramètres de connexion et d'authentification
- L'initialisation automatique du schéma de base de données au premier démarrage
- La configuration des sauvegardes automatiques
- L'optimisation des paramètres de performance pour l'usage spécifique de PrivSense

Cette approche de conteneurisation garantit la portabilité du système complet et simplifie considérablement son déploiement dans différents environnements, tout en respectant les politiques de sécurité et de licences de l'entreprise.


*Figure 4 : Logo Podman*

## 3.5. SonarQube et SonarLint

Pour garantir une qualité de code optimale et répondre aux normes de développement sécurisé, nous avons intégré SonarQube dans notre processus de développement :

- **SonarLint** : Intégré directement dans notre IDE (IntelliJ IDEA), ce plugin nous permet de détecter les problèmes de qualité et de sécurité en temps réel pendant le développement

- **Analyse de qualité** : L'outil vérifie la conformité du code avec les standards de qualité ISO 9126 (maintenabilité, fiabilité, efficacité)
- **Détection des vulnérabilités** : Identification automatique des risques de sécurité selon les recommandations OWASP Top 10
- **Dette technique** : Suivi et gestion proactive de la dette technique pour maintenir un code propre et évolutif

Les analyses régulières de SonarQube nous ont permis d'améliorer continuellement la qualité du code, de réduire les vulnérabilités potentielles et de maintenir une base de code saine pour les futures évolutions du projet.

Bien que la mise en place d'une intégration continue automatisée (CI/CD) soit prévue pour les phases ultérieures du projet, l'utilisation actuelle de ces outils nous permet déjà d'assurer un niveau élevé de qualité et de traçabilité dans notre développement.

*Figure 5 : Logo SonarQube*

# 4. Technologies et outils utilisés

## 4.1. Introduction

Le projet PrivSense s'appuie sur un ensemble de technologies modernes pour assurer une détection efficace et fiable des données personnelles dans les bases de données. Cette section détaille les langages, frameworks, bibliothèques et outils utilisés pour le développement de la solution.

## 4.2. Java

Java constitue le langage principal utilisé pour le développement du backend de PrivSense. Nous avons opté pour Java 21 afin de bénéficier des dernières améliorations du langage, notamment en matière de performances et de syntaxe.

Les caractéristiques de Java qui ont été particulièrement exploitées dans le projet sont :

- La programmation orientée objet pour une conception modulaire et extensible
- Le typage fort pour garantir la robustesse du code
- Les streams pour le traitement efficace des données
- La gestion des exceptions pour un traitement robuste des erreurs
- La concurrence pour optimiser les performances des opérations de scan

L'utilisation de Java nous a permis de créer une architecture solide, maintenable et évolutive pour notre système de détection de données personnelles.


*Figure 6 : Logo Java*

## 4.3. Spring

Spring a servi de cadre applicatif pour le développement de PrivSense, offrant une structure robuste et des composants prêts à l'emploi. Plusieurs modules Spring ont été utilisés :

- **Spring Core** : Pour l'injection de dépendances et la gestion des beans
- **Spring Web** : Pour la création des API REST exposant les fonctionnalités de PrivSense

- **Spring Data JPA** : Pour la persistance des résultats de détection et des configurations
- **Spring Security** : Pour sécuriser l'accès aux API avec authentification JWT
- **Spring WebFlux** : Pour les appels non-bloquants au service NER

Spring Boot nous a permis de nous concentrer sur le développement des fonctionnalités métier tout en bénéficiant d'une infrastructure technique solide et éprouvée.

*Figure 7 : Logo Spring*

## 4.4. Python (NER Service)

Le service de reconnaissance d'entités nommées (NER) constitue un élément clé de notre architecture pour la détection avancée des données personnelles. Développé en Python, ce microservice indépendant exploite des techniques d'intelligence artificielle pour identifier avec précision les informations personnelles dans des échantillons textuels.

Voici les principales caractéristiques techniques du service NER implémenté :

- **Architecture REST** : Le service expose une API REST développée avec **FastAPI**, un framework moderne et performant qui facilite la création d'APIs asynchrones. Cette approche permet une intégration fluide avec le composant Java principal via des appels HTTP.
- **Modèle IA spécialisé** : Nous avons sélectionné le modèle **E3-JSI_gliner-multi-pii-domains-v1**, un modèle de langage spécifiquement entraîné pour la détection de données personnelles dans plusieurs langues et domaines. Ce choix nous permet d'identifier plus de 50 types différents de PII, dont :
    - Noms de personnes

- Adresses postales
- Numéros de téléphone
- Adresses email
- Numéros de cartes bancaires
- Identifiants de sécurité sociale
- Dates de naissance
- Numéros de passeport
- Informations médicales
- Et de nombreux autres types spécifiques


*Figure 8 : Logo Python*

## 4.5. PostgreSQL

PostgreSQL a été choisi comme système de gestion de base de données relationnelle pour stocker les résultats des détections, les métadonnées des bases analysées et les configurations du système. Ses avantages dans notre contexte sont :

- Support robuste des transactions ACID

- Capacités avancées pour la gestion des JSON et des tableaux

- Performance et stabilité pour les applications d'entreprise

- Compatibilité avec les normes SQL

Le schéma de base de données a été conçu pour optimiser les requêtes courantes tout en maintenant une bonne normalisation des données.


*Figure 9 : Logo PostgreSQL*

## 4.6. React JS

React a été choisi comme bibliothèque JavaScript pour le développement frontend de l'interface utilisateur de PrivSense. Cette technologie de pointe nous a permis de :

- Créer une interface utilisateur interactive et réactive
- Structurer l'application frontend en composants réutilisables
- Gérer efficacement l'état de l'application avec Redux
- Consommer les API REST exposées par le backend Spring Boot
- Mettre en place un système de visualisation des résultats avec des graphiques dynamiques
- Implémenter une expérience utilisateur intuitive avec Material-UI

L'architecture frontend a été conçue pour être modulaire et évolutive, permettant d'ajouter facilement de nouvelles fonctionnalités au fur et à mesure du développement du système.

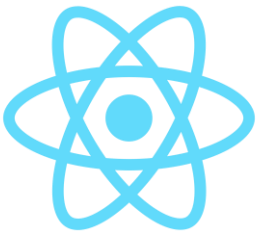


*Figure 10 : Logo ReactJS*

## 4.7. IntelliJ IDEA et Visual Studio Code

IntelliJ IDEA a été utilisé comme environnement de développement principal pour le code Java. Ses fonctionnalités avancées d'assistance au codage, d'inspection du code et d'intégration avec Maven ont considérablement amélioré notre productivité de développement.

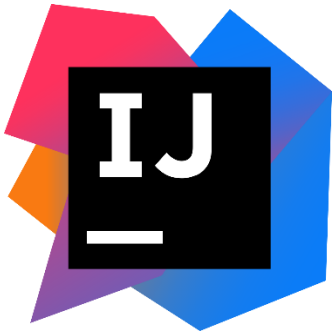



*Figure 11 : Logo Intellij*

Visual Studio Code a été utilisé principalement pour le développement du service NER en Python et pour l'édition des fichiers de configuration. Ses extensions pour Python et Docker ont facilité le développement et le débogage du service de reconnaissance d'entités.

*Figure 12 : Logo VsCode*

## 4.8. JUnit et Mockito

Pour garantir la qualité du code et sa robustesse, nous avons implémenté des tests unitaires et d'intégration à l'aide de :

- **JUnit 5** : Pour le framework de test
- **Mockito** : Pour simuler les dépendances externes
- **SpringBootTest** : Pour les tests d'intégration avec le contexte Spring

Cette approche de développement guidée par les tests a permis d'identifier et de résoudre les problèmes tôt dans le cycle de développement.

## 4.9. Postman

Postman a servi d'outil de test pour les API REST développées. Il nous a permis de :

- Tester manuellement les endpoints de l'API
- Créer des collections de requêtes pour les tests récurrents
- Valider les réponses et les codes d'état HTTP

- Documenter l'utilisation de l'API


*Figure 13 : Logo PostMan*

# 5. travail individuel

Le projet PrivSense a été principalement réalisé en travail individuel sous la supervision directe de l' encadrant de CGI « M .Ismail Bodor ». Cette configuration a permis une grande autonomie dans la prise de décisions techniques et dans l'organisation du travail au quotidien.

Néanmoins, plusieurs interactions essentielles ont enrichi le développement du projet :

- Réunions hebdomadaires avec l'encadrant pour valider les orientations et résoudre les problèmes complexes
- Consultations ponctuelles avec des experts techniques de CGI sur des sujets spécifiques (sécurité, performances, AI)

Cette configuration a favorisé à la fois l'autonomie nécessaire à un projet de recherche et développement, tout en bénéficiant de l'expertise collective de CGI.

# 6. Composition de l'équipe et tâches attribuées

Bien que le développement principal du projet PrivSense ait été réalisé de manière individuelle, deux acteurs ont contribué à son orientation et à sa réussite :

**1. Stagiaire (moi-même) - Développeur principal**

- Recherche et veille sur les techniques de détection de PII
- Conception de l'architecture logicielle
- Développement des différents modules du système
- Tests et validation des fonctionnalités
- Documentation technique
- Présentations d'avancement

**2. Encadrant CGI - Architecte**

- Définition des besoins et des priorités
- Validation des fonctionnalités développées
- Conseil technique et méthodologique
- Interface avec la direction
- Gestion des ressources et du calendrier

# 7. Cahier des charges détaillé

## 7.1. Contexte et objectifs

### 7.1.1. Contexte réglementaire

Le RGPD (Règlement Général sur la Protection des Données) est une réglementation européenne qui a fondamentalement transformé les exigences en matière de protection des

données personnelles. Cette réglementation, entrée en vigueur le 25 mai 2018, impose plusieurs obligations strictes aux entreprises :

- Obtention du consentement explicite des utilisateurs
- Transparence totale sur l'utilisation des données
- Sécurité renforcée des données personnelles
- Sanctions pouvant atteindre 4% du chiffre d'affaires mondial

Le caractère extraterritorial du RGPD est particulièrement important pour les entreprises marocaines : même basées au Maroc, si elles traitent des données de résidents européens, elles doivent respecter cette réglementation.

### 7.1.2. Données personnelles et RGPD

L'article 4(1) du RGPD définit les données à caractère personnel comme "toute information relative à une personne physique identifiée ou identifiable". Une personne est considérée comme identifiable si elle peut être reconnue, directement ou indirectement, que ce soit par un nom, un numéro, des données de localisation, un identifiant en ligne, ou par des éléments spécifiques liés à son identité physique, physiologique, génétique, psychique, économique, culturelle ou sociale.

Ces données personnelles peuvent être classées en deux catégories :

**Identifiants directs** : Permettent d'identifier immédiatement une personne sans informations supplémentaires

- Nom complet, adresse email, numéro de téléphone

- Numéro de sécurité sociale, passeport

- Données biométriques (empreintes digitales)

- Identifiants numériques (adresse IP fixe)

**Identifiants indirects** : Semblent anonymes isolément, mais peuvent identifier une personne lorsqu'ils sont combinés

- Date de naissance

- Ville de résidence

- Profession

- Code postal

Le défi majeur pour les entreprises réside dans l'identification et la protection de ces différentes catégories de données, particulièrement les identifiants indirects dont le potentiel d'identification n'est visible que par recoupement.

### 7.1.3. Défis actuels des entreprises

Les entreprises font face à plusieurs défis majeurs concernant la gestion des données personnelles :

- Manque de visibilité : 83% des entreprises ne savent pas où se trouvent toutes leurs données sensibles
- Impact financier considérable : 4,35 millions d'euros en moyenne par incident de fuite de données
- Complexité croissante des systèmes d'information interconnectés
- Équilibre délicat entre protection des données et valeur business

### 7.1.4. Objectifs du projet PrivSense

Face aux défis réglementaires et techniques présentés ci-dessus, le projet PrivSense a été conçu avec des objectifs précis visant à transformer ces contraintes en opportunités :

1. **Automatisation de la détection des données personnelles** : Développer un système capable d'analyser de manière automatique et exhaustive les bases de données pour identifier les différents types de données personnelles, qu'elles soient des identifiants directs ou indirects. Cette détection repose sur une approche multi-niveaux combinant :
    - Analyse heuristique des noms et métadonnées des colonnes
    - Détection par expressions régulières pour les formats standardisés
    - Reconnaissance d'entités nommées (NER) basée sur l'intelligence artificielle pour l'analyse contextuelle
2. **Détection avancée des quasi-identifiants** : Mettre en œuvre des algorithmes complexes pour identifier les combinaisons d'attributs qui, ensemble, peuvent conduire à la réidentification des individus :
    - Analyse statistique de la singularité des combinaisons de valeurs entre colonnes
    - Calcul de l'unicité des enregistrements basé sur différentes combinaisons d'attributs
    - Évaluation de la k-anonymité pour chaque groupe potentiel de quasi-identifiants
    - Utilisation de techniques de clustering pour découvrir automatiquement les corrélations cachées
    - Calcul des scores de risque basés sur l'entropie de l'information et la diversité des données
3. **Évaluation précise des risques liés aux données personnelles** : Mettre en place un système de scoring permettant de quantifier le niveau de risque associé à chaque donnée personnelle détectée et aux combinaisons potentielles de quasi-identifiants. Cette évaluation doit prendre en compte :
    - Le niveau de sensibilité de chaque type de donnée

- Le potentiel de réidentification des individus
- Les exigences spécifiques du RGPD concernant chaque catégorie de données

4. **Génération de rapports de conformité exploitables** : Créer des rapports détaillés et personnalisables qui fournissent une visibilité complète sur les données personnelles présentes dans les systèmes d'information. Ces rapports doivent :
    - Présenter les résultats sous différents formats (PDF, CSV, JSON)
    - Offrir différents niveaux de lecture (synthétique pour la direction, détaillé pour les équipes techniques)
    - Inclure des recommandations concrètes pour améliorer la conformité
5. **Développement de capacités d'anonymisation adaptatives** : Concevoir un module capable de suggérer et d'appliquer différentes techniques d'anonymisation en fonction du type de données personnelles et de leur contexte d'utilisation. Les techniques incluent :
    - K-anonymisation pour les quasi-identifiants
    - Hachage ou tokenisation pour les identifiants directs
    - Masquage partiel pour conserver l'utilité des données tout en protégeant la vie privée
6. **Extension aux flux de données dynamiques** : Élargir la protection au-delà des bases de données statiques pour couvrir également les échanges de données en temps réel via les API REST. Cet objectif implique de :
    - Créer un middleware d'interception transparent
    - Assurer une détection et une anonymisation en temps réel
    - Maintenir les performances des API malgré le traitement supplémentaire
7. **Création d'une interface utilisateur intuitive** : Développer un frontend moderne en React permettant aux utilisateurs de :
    - Configurer facilement les scans de détection

- Visualiser graphiquement les résultats et les rapports
- Paramétrer les règles d'anonymisation selon leurs besoins spécifiques

En résumé, PrivSense vise à fournir une solution complète et intégrée qui simplifie considérablement la mise en conformité avec le RGPD tout en transformant cette obligation réglementaire en avantage compétitif. Par son approche automatisée et intelligente, le projet permet aux entreprises de gagner en efficacité tout en réduisant significativement les risques liés à la gestion des données personnelles.

## 7.2. Architecture du projet


*Figure 14 : Logo du projet PrivSense*

PrivSense repose sur une architecture modulaire orientée services, qui permet une grande flexibilité et extensibilité. Le système est composé de plusieurs modules interconnectés :

1. **privsense-core** : Module central qui définit les interfaces, les contrats de service et les modèles de données partagés entre tous les autres modules. Il établit les fondations de l'application.

2. **privsense-db-connector** : Responsable de l'établissement des connexions avec différents types de bases de données (PostgreSQL, MySQL, Oracle, etc.) de manière dynamique et sécurisée.
3. **privsense-metadata-extractor** : Analyse les schémas des bases de données pour extraire les métadonnées (tables, colonnes, types de données, relations) nécessaires à l'analyse.
4. **privsense-sampler** : Réalise des échantillonnages intelligents des données pour permettre une analyse efficace sans traiter l'intégralité des bases, ce qui optimise les performances.
5. **privsense-pii-detector** : Cœur de la détection des données personnelles, ce module implémente différentes stratégies de détection (heuristique, regex, reconnaissance d'entités nommées) pour identifier avec précision les PII.
6. **privsense-reporter** : Génère des rapports détaillés sur les données personnelles détectées, avec différents formats de sortie (JSON, CSV, PDF, texte).
7. **privsense-api** : Expose les fonctionnalités du système via des API REST sécurisées, permettant l'intégration avec d'autres systèmes.
8. **ner-service** : Service Python indépendant qui utilise des techniques d'intelligence artificielle (modèles de langage) pour la détection avancée de données personnelles dans le texte.

Cette architecture décomposée permet une évolution indépendante de chaque module et facilite la maintenance du système.

## 7.3. Fonctionnalités actuelles

### 7.3.1. Connexion aux bases de données

Le module privsense-db-connector gère la connexion à différents types de bases de données :

- Support de PostgreSQL, MySQL, Oracle et SQLite
- Configuration dynamique des paramètres de connexion

- Gestion sécurisée des identifiants de connexion
- Tests de connectivité et validation de l'accès aux données

### 7.3.2. Extraction de métadonnées

Le module privsense-metadata-extractor analyse les schémas des bases de données pour :

- Identifier les schémas disponibles
- Énumérer les tables accessibles
- Extraire les métadonnées des colonnes (noms, types de données, contraintes)
- Cartographier les relations entre tables (clés primaires/étrangères)

Ces métadonnées fournissent le contexte nécessaire pour optimiser les stratégies de détection.

### 7.3.3. Échantillonnage de données

Le module privsense-sampler implémente plusieurs stratégies d'échantillonnage :

- Échantillonnage aléatoire simple
- Échantillonnage stratifié (pour représenter toutes les catégories de données)
- Échantillonnage adaptatif (qui ajuste la taille de l'échantillon en fonction de la diversité des données)

Ces techniques permettent d'analyser un sous-ensemble représentatif des données, optimisant ainsi les performances sans compromettre la précision de la détection.

### 7.3.4. Détection de PII

Le module privsense-pii-detector constitue le cœur du système avec trois approches complémentaires :

**1. Détection heuristique** :

- Analyse des noms de colonnes pour identifier les champs potentiellement sensibles
- Analyse des commentaires associés aux colonnes dans le schéma

- Attribution de scores de confiance basés sur le type de correspondance

**2. Détection par expressions régulières** :

- Application de patterns regex prédéfinis pour détecter des formats spécifiques (email, téléphone, SSN, etc.)
- Paramétrage des seuils de confiance pour chaque type de pattern
- Classification des colonnes selon le type de PII détecté

**3. Reconnaissance d'entités nommées (NER)** :

- Utilisation du service Python avec un modèle d'IA spécialisé
- Détection de noms, adresses, organisations et autres entités personnelles
- Analyse contextuelle pour réduire les faux positifs

Le système inclut également une détection avancée des quasi-identifiants :

- Identification des colonnes qui, combinées, peuvent permettre de réidentifier des personnes
- Analyse des corrélations entre colonnes pour détecter les groupes à risque
- Évaluation du risque de réidentification basée sur la k-anonymité et l'entropie

### 7.3.5. Génération de rapports

Le module privsense-reporter produit des rapports détaillés qui incluent :

- Statistiques globales sur les bases analysées (tables scannées, PII détectées)
- Détail des colonnes contenant des données personnelles, avec le type de PII et le niveau de confiance
- Évaluation des risques liés aux quasi-identifiants
- Recommandations pour l'anonymisation ou la pseudonymisation des données sensibles

Ces rapports peuvent être générés dans différents formats (JSON, CSV, PDF, texte) pour faciliter leur exploitation et leur partage.

## 7.4. Fonctionnalités à développer

### 7.4.1. Module d'anonymisation

Le développement du module d'anonymisation est actuellement en cours. Ce module s'appuiera sur les résultats de la détection pour :

- Proposer des techniques d'anonymisation adaptées à chaque type de données personnelles
- Appliquer ces techniques de manière automatisée ou semi-automatisée
- Générer une version anonymisée de la base de données originale
- Vérifier l'efficacité de l'anonymisation par des tests de réidentification

L'intégration avec une sollution existante développé par CGI France est prévue pour accélérer cette phase.

### 7.4.2. Intégration API REST

La seconde phase majeure consistera à étendre les capacités du système pour :

- Créer un middleware capable d'intercepter les réponses API REST
- Détecter en temps réel les données personnelles dans les réponses JSON/XML
- Appliquer des règles d'anonymisation avant de transmettre les données au client
- Configurer dynamiquement les règles d'anonymisation sans nécessiter de redéploiement

Cette extension permettra d'appliquer la protection des données personnelles non seulement aux bases de données statiques, mais aussi aux flux de données dynamiques transitant par les API.

# 8. Avancement et état actuel

## 8.1. État d'avancement du projet

Le projet PrivSense a atteint un jalon important avec la livraison d'une version MVP (Minimum Viable Product) qui a été présentée aux directeurs de CGI. Cette version comprend l'ensemble des fonctionnalités de base pour la détection des données personnelles dans les bases de données :

- Connexion dynamique aux bases de données
- Extraction des métadonnées
- Échantillonnage intelligent
- Détection multi-niveaux des PII (heuristique, regex, NER)
- Détection des quasi-identifiants
- Génération de rapports de conformité

L'avancement global du projet est estimé à environ 70%, les 30% restants concernant principalement le module d'anonymisation et l'intégration avec le passe-plat API REST.

## 8.2. Résultats obtenus

Les tests réalisés sur des jeux de données synthétiques et réels ont démontré l'efficacité du système :

- Détection de plus de 20 types différents de données personnelles (emails, téléphones, noms, adresses, etc.)
- Taux de détection supérieur à 95% pour les PII directes avec un faible taux de faux positifs
- Identification pertinente des quasi-identifiants et évaluation des risques associés
- Génération de rapports clairs et exploitables pour les équipes techniques et juridiques

## 8.3. Difficultés rencontrées et solutions

Plusieurs défis techniques ont dû être relevés durant le développement :

1. **Optimisation des performances d'échantillonnage**
    - *Problème* : Lenteur des requêtes d'échantillonnage sur de très grandes tables
    - *Solution* : Implémentation d'un mécanisme de requêtes optimisées et d'échantillonnage adaptatif qui ajuste la taille des échantillons selon la volumétrie
2. **Intégration du service NER**
    - *Problème* : Latence élevée lors des appels au service Python depuis Java
    - *Solution* : Mise en place d'appels asynchrones et d'un mécanisme de mise en cache des résultats pour les patterns récurrents
3. **Détection des quasi-identifiants**
    - *Problème* : Complexité algorithmique élevée pour l'analyse des corrélations entre colonnes
    - *Solution* : Implémentation d'un algorithme de clustering optimisé et établissement de seuils configurables pour limiter l'espace de recherche
4. **Gestion des différents dialectes SQL**
    - *Problème* : Variations syntaxiques entre les différents SGBD pour les requêtes d'échantillonnage
    - *Solution* : Création d'une couche d'abstraction et de traduction des requêtes adaptée à chaque type de base de données

## 8.4. Prochaines étapes

Les prochaines phases du projet se concentreront sur :

1. **Finalisation du module d'anonymisation**
    - Intégration avec la sollution  existante PIMO de CGI France

- Implémentation de techniques d'anonymisation spécifiques à chaque type de PII
- Développement d'un mécanisme de validation de l'efficacité de l'anonymisation

2. **Construction d'interfaces utilisateur intuitives**
   - Développement d'un frontend moderne en React
   - Création de tableaux de bord interactifs pour la visualisation des données détectées
   - Implémentation d'interfaces permettant de configurer les règles d'anonymisation
   - Conception de formulaires intuitifs pour la configuration des scans
   - Création de visualisations graphiques pour les rapports de conformité

3. **Développement du passe-plat API REST**
   - Création du middleware d'interception
   - Adaptation des algorithmes de détection pour un traitement en temps réel
   - Mise en place d'un système de cache pour optimiser les performances

4. **Optimisation globale et documentation**
   - Amélioration des performances générales du système
   - Rédaction de la documentation technique détaillée
   - Élaboration de guides d'utilisation

5. **Tests et validation finale**
   - Tests de charge pour évaluer les limites du système

Le planning prévoit la finalisation de ces étapes d'ici la fin du stage, avec une priorité sur le module d'anonymisation qui représente un enjeu majeur pour la valeur ajoutée du système.

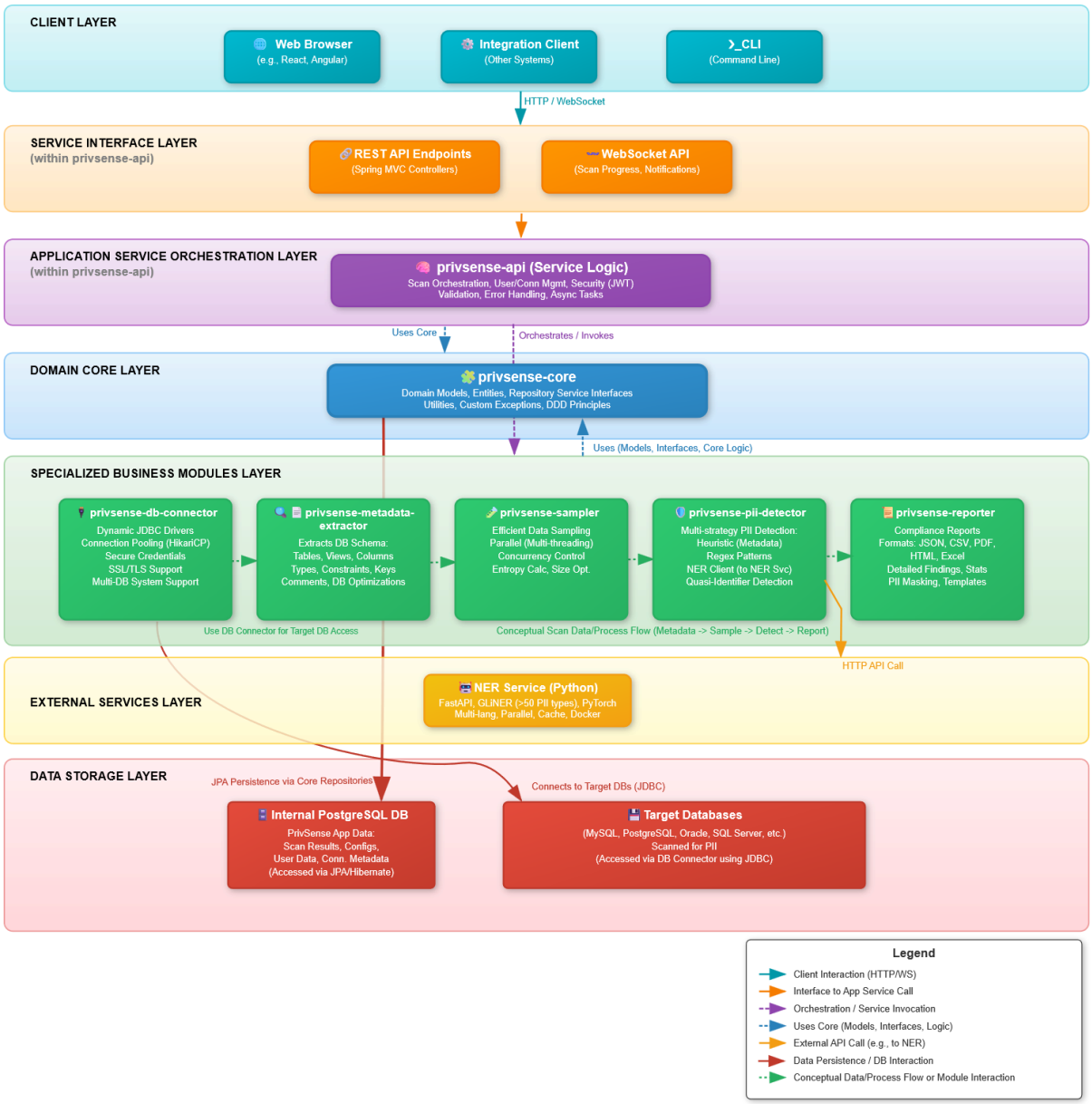
# 9. Conclusion

Le projet PrivSense représente une avancée significative dans la démarche de conformité au RGPD pour CGI. Avec un taux d'avancement de 70%, nous avons réussi à développer une solution robuste capable d'analyser automatiquement les bases de données, d'identifier avec précision les données personnelles et de générer des rapports détaillés. L'architecture modulaire du système, son approche multi-niveaux de détection et sa capacité à traiter divers types de bases de données en font un outil polyvalent et performant.

PrivSense illustre parfaitement comment l'innovation technologique peut être mise au service de la conformité réglementaire et de la protection des données personnelles, un enjeu majeur pour les entreprises aujourd'hui.

# Table des figures

**Figure 1 : Logo GIT** ..... 9
**Figure 2 : Logo Gitlab** ..... 10
**Figure 3 : Logo Maven** ..... 11
**Figure 4 : Logo Podman** ..... 12
**Figure 5 : Logo SonarQube** ..... 13
**Figure 6 : Logo Java** ..... 14
**Figure 7 : Logo Spring** ..... 15
**Figure 8 : Logo Python** ..... 16
**Figure 9 : Logo PostgreSQL** ..... 17
**Figure 10 : Logo ReactJS** ..... 18
**Figure 11 : Logo Intellij** ..... 18
**Figure 12 : Logo VsCode** ..... 19
**Figure 13 : Logo PostMan** ..... 20
**Figure 14 : Logo du projet PrivSense** ..... 26


CLIENT LAYER
Web Browser
(e.g., React, Angular)
Integration Client
(Other Systems)
>_CLI
(Command Line)
HTTP / WebSocket
SERVICE INTERFACE LAYER
(within privsense-api)
REST API Endpoints
(Spring MVC Controllers)
WebSocket API
(Scan Progress, Notifications)
APPLICATION SERVICE ORCHESTRATION LAYER
(within privsense-api)
privsense-api (Service Logic)
Scan Orchestration, User/Conn Mgmt, Security (JWT)
Validation, Error Handling, Async Tasks
Uses Core
Orchestrates / Invokes
DOMAIN CORE LAYER
privsense-core
Domain Models, Entities, Repository Service Interfaces
Utilities, Custom Exceptions, DDD Principles
Uses (Models, Interfaces, Core Logic)
SPECIALIZED BUSINESS MODULES LAYER
privsense-db-connector
Dynamic JDBC Drivers
Connection Pooling (HikariCP)
Secure Credentials
SSL/TLS Support
Multi-DB System Support
privsense-metadata-extractor
Extracts DB Schema:
Tables, Views, Columns
Types, Constraints, Keys
Comments, DB Optimizations
privsense-sampler
Efficient Data Sampling
Parallel (Multi-threading)
Concurrency Control
Entropy Calc, Size Opt.
privsense-pii-detector
Multi-strategy PII Detection:
Heuristic (Metadata)
Regex Patterns
NER Client (to NER Svc)
Quasi-Identifier Detection
privsense-reporter
Compliance Reports
Formats: JSON, CSV, PDF,
HTML, Excel
Detailed Findings, Stats
PII Masking, Templates
Use DB Connector for Target DB Access
Conceptual Scan Data/Process Flow (Metadata -> Sample -> Detect -> Report)
HTTP API Call
EXTERNAL SERVICES LAYER
NER Service (Python)
FastAPI, GLiNER (>50 PII types), PyTorch
Multi-lang, Parallel, Cache, Docker
DATA STORAGE LAYER
JPA Persistence via Core Repositories
Connects to Target DBs (JDBC)
Internal PostgreSQL DB
PrivSense App Data:
Scan Results, Configs,
User Data, Conn. Metadata
(Accessed via JPA/Hibernate)
Target Databases
(MySQL, PostgreSQL, Oracle, SQL Server, etc.)
Scanned for PII
(Accessed via DB Connector using JDBC)
Legend
Client Interaction (HTTP/WS)
Interface to App Service Call
Orchestration / Service Invocation
Uses Core (Models, Interfaces, Logic)
External API Call (e.g., to NER)
Data Persistence / DB Interaction
Conceptual Data/Process Flow or Module Interaction