Logiciel de gestion administrative de carrière et de paye

Application RH 3-tiers client C# WinForms (.NET 8) · API REST PHP 8.1+ · MySQL 8

Rapport de projet personnel

Dépôt : github.com/Jacque004/rastre

2. Résumé

Rastre est une application de gestion des ressources humaines destinée aux gestionnaires de carrière. Elle centralise le dossier administratif du collaborateur identité, contrats, affectations, notes RH et documents et y ajoute des outils d'aide au pilotage quotidien : alertes d'échéance de contrat, tableau de bord, checklist de complétude du dossier et échéancier de tâches.

Le projet s'appuie sur une architecture 3-tiers classique : un client lourd développé en C# WinForms (.NET 8) pour l'interface des gestionnaires, une API REST en PHP consommée en JSON avec authentification par jeton, et une base de données MySQL 8 pour la persistance. Cette première version (MVP+), développée en environ trois semaines, ne traite pas encore la paye, mais le modèle de données a été conçu pour pouvoir l'accueillir dans une itération future. Le développement de Rastre reste actif au-delà de ce MVP.

3. Contexte et problématique

3.1 Contexte

Dans de nombreux services RH, les informations administratives des collaborateurs identité, numéro de sécurité sociale, contrats, affectations restent dispersées entre fichiers bureautiques, tableurs et documents papier. Cette dispersion complique le suivi des échéances contractuelles, nuit à la traçabilité des actions et fragilise la conformité au regard de la protection des données personnelles.

Rastre est né de la volonté de reproduire, à l'échelle d'un projet personnel, les enjeux réels d'une application métier RH: gérer des données sensibles, structurer un dossier collaborateur cohérent, et construire une architecture technique proche de ce qu'on trouve en entreprise (client lourd + API + base relationnelle), plutôt qu'un simple exercice de CRUD.

3.2 Problématique

Comment centraliser, sécuriser et faciliter le suivi administratif des collaborateurs en particulier les agents contractuels, dont les contrats ont une date de fin à surveiller pour un gestionnaire RH qui doit garder une vision claire de ses dossiers et de ses échéances ?

3.3 Objectifs

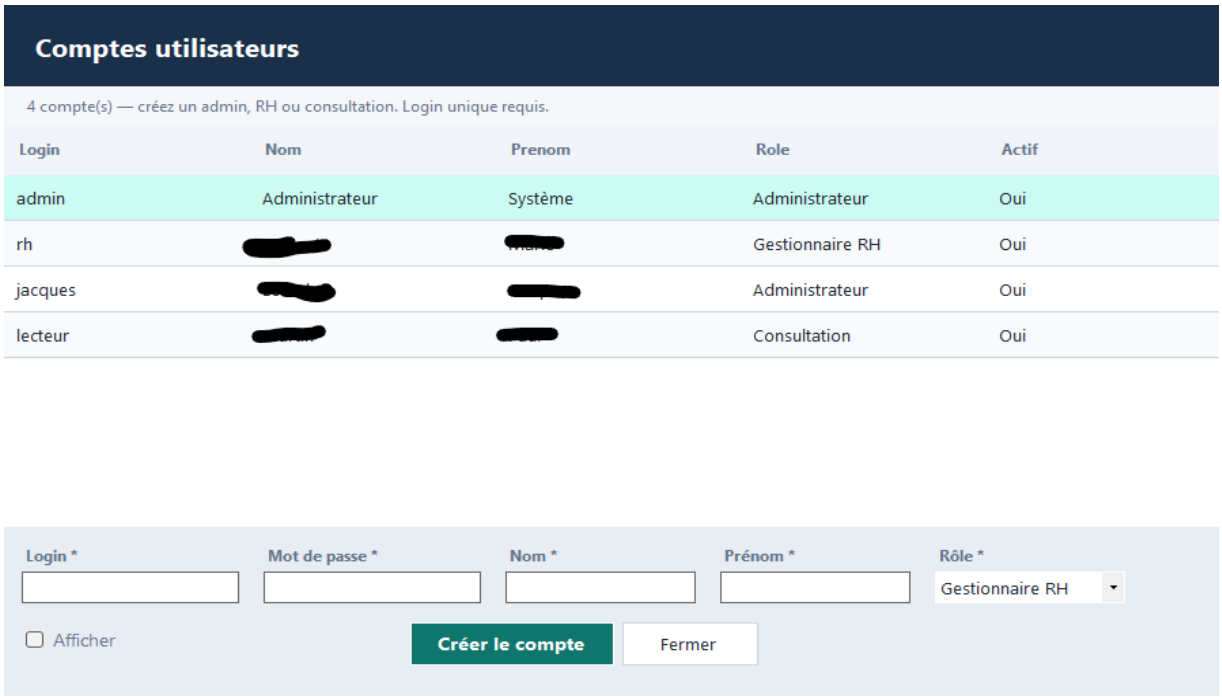
- Centraliser le dossier collaborateur (identité, contrats, affectations, notes, documents) dans un outil unique
- Suivre les contrats, les fonctions occupées et les affectations dans le temps
- Alerter automatiquement sur les échéances de contrat à venir
- Outiller le quotidien du gestionnaire RH avec un tableau de bord, une checklist de complétude et un échéancier de tâches
- Concevoir un modèle de données prêt à accueillir une future extension paye

4. Acteurs et gestion des droits

L'application distingue quatre rôles, avec des droits différenciés appliqués côté API. Il n'existe pas, dans cette première version, de portail dédié permettant à l'agent de consulter lui-même son propre dossier.

Rôle	Code	Droits
Administrateur	admin	Tous les droits, y compris la gestion des comptes utilisateurs

Rôle	Code	Droits
Gestionnaire RH / Carrière	rh_carriere	CRUD sur les dossiers, contrats, affectations et tâches
Consultation	lecture	Lecture seule
Responsable paye	paye	Rôle réservé, non actif dans le MVP (préparation de l'extension paye)



Écran de gestion des comptes utilisateurs (réservé au rôle administrateur)

La capture ci-dessus illustre l'écran d'administration des comptes : création d'un login unique par utilisateur, attribution d'un rôle parmi les quatre définis, et visibilité sur l'ensemble des comptes actifs. Pour les besoins de démonstration, trois comptes types sont disponibles (admin, rh, lecteur).

5. Périmètre fonctionnel

5.1 Fonctionnalités réalisées (MVP+)

- Authentification par jeton Bearer, valable 8 heures
- Liste des collaborateurs avec filtres (statut, type de contrat, entité, échéance) et export CSV
- Dossier collaborateur structuré en onglets : Identité, Organisme (Affectation + Contrat), Notes RH, Documents
- Renouvellement de contrat
- Alertes automatiques sur les contrats arrivant à échéance (ex. sous 90 jours)
- Tableau de bord RH : indicateurs clés et listes de dossiers prioritaires
- Checklist et score de complétude du dossier
- Échéancier de tâches RH
- Gestion des comptes utilisateurs (réservée à l'administrateur)
- Journal d'audit des actions sensibles (audit_log)

5.2 Hors périmètre (backlog)

- Module paye complet (bulletins, cotisations, déclarations)
- Titularisation complète (corps, grade, échelon)
- Gestion des congés et absences
- Portail agent (consultation par le collaborateur lui-même)
- Organigramme avancé
- Installateur Windows pour le client

Le choix de circonscrire le MVP à la gestion administrative de carrière en laissant la paye et le portail agent de côté a permis de concentrer l'effort sur la structuration du dossier collaborateur et sur les outils d'aide au suivi (alertes, checklist, tableau de bord), qui constituent le cœur de la valeur ajoutée du projet. La paye, plus lourde en règles métier (cotisations, calculs, déclarations), est traitée comme une extension à part entière, rendue possible par un modèle de données déjà pensé pour l'accueillir.

6. Architecture technique

L'application repose sur une architecture 3-tiers: un client lourd WinForms consomme une API REST en PHP, qui elle-même interroge une base de données MySQL. Les échanges entre le client et l'API se font en HTTP/JSON, sécurisés par un jeton Bearer.

Couche	Technologie	Rôle
Présentation	WinForms (.NET 8)	Interface utilisateur des gestionnaires RH
Application	PHP 8.1+ (sans framework lourd)	Logique métier, gestion des rôles, audit
Données	MySQL 8, utf8mb4	Persistance des données

L'URL locale de l'API en environnement de développement est http://localhost/rastre/api/public, avec unpoint de santé exposée en GET /health. Le choix d'une API PHP « sans framework lourd » a été fait pour garder le contrôle complet sur le routage, la gestion des rôles et l'audit, sans la surcouche d'un framework comme Laravel ou Symfony — un compromis pédagogique qui a permis de mieux comprendre le fonctionnement d'une API REST de bout en bout, au prix d'un peu plus de code d'infrastructure à écrire soi-même.

6.1 Sécurité

- Mots de passe hashés (bcrypt / password_hash)
- Authentification par jeton (api_token)
- Contrôle des droits par rôle, appliqué côté API
- Journal d'audit des actions sensibles
- HTTPS requis en production (l'environnement de développement local, en XAMPP, fonctionne en HTTP)

7. Modèle de données

Le modèle de données s'articule autour de l'entité Collaborateur, à laquelle se rattachent l'ensemble des informations administratives et de suivi.

Table	Rôle
utilisateur, api_token	Comptes et sessions applicatives
collaborateur, enfant	Identité du collaborateur et de ses enfants à charge
type_contrat, contrat, fonction	Carrière contractuelle (type, dates, fonction occupée)

Table	Rôle
entite, affectation	Organisation et rattachement du collaborateur
note_rh, document	Suivi RH et pièces jointes
tache_rh	Échéancier de tâches RH
audit_log	Traçabilité des actions
situation_familiale	Table de référence

Les relations clés sont les suivantes : un collaborateur peut avoir plusieurs enfants, contrats, affectations, notes, documents et tâches (relations 1-N) ; un contrat peut être rattaché à une ou plusieurs fonctions ; les entités forment une hiérarchie via une relation parent_id, permettant de représenter une organisation à plusieurs niveaux (direction, service, sous-service…).

Les scripts SQL du projet (rastre.sql pour le schéma, seed.sql pour les données de démonstration, ainsi que deux scripts de migration) permettent de recréer la base intégralement, y compris via un script d'installation dédié (install.php).

8. Conception applicative

8.1 API

L'API est structurée autour d'un point d'entrée unique (api/public/index.php) qui fait office de routeur vers un ensemble de contrôleurs dédiés : authentification, collaborateurs, contrats, affectations, notes, documents, alertes, tableau de bord, tâches, utilisateurs et référentiels. Des services transverses (authentification, audit, calcul de complétude, accès base de données, gestion des requêtes/réponses) sont mutualisés entre ces contrôleurs.

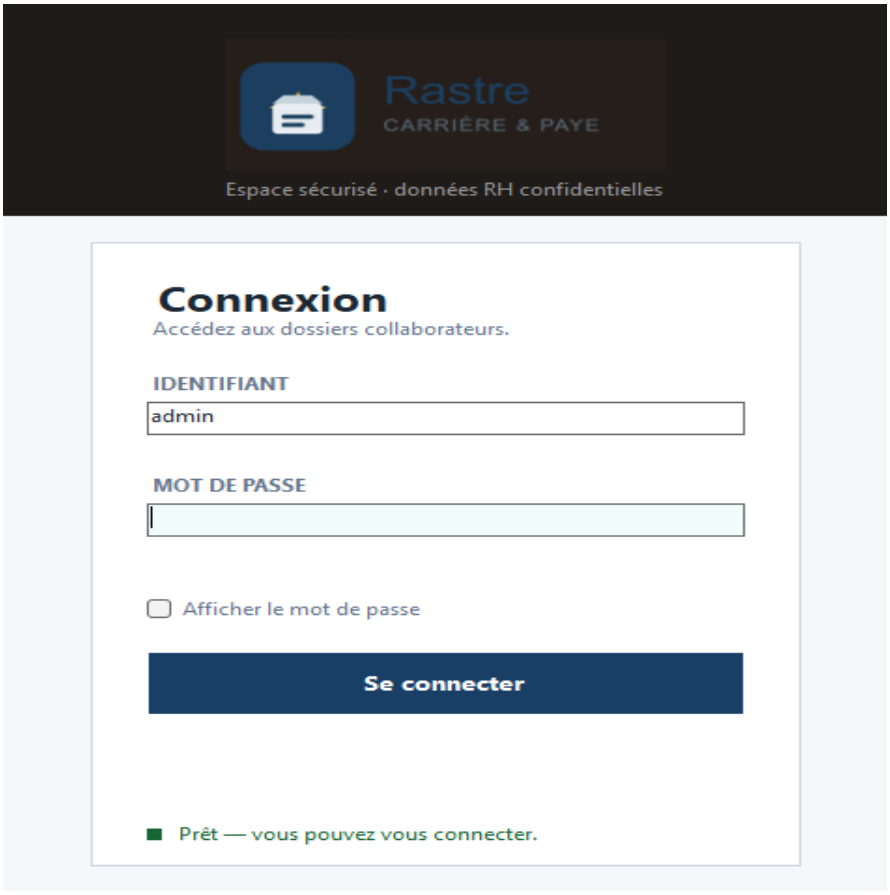
8.2 Client WinForms

Écran	Rôle
LoginForm	Connexion et vérification de la disponibilité de l'API
MainForm	Liste des dossiers, filtres, indicateurs, alertes
DossierForm	Fiche dossier détaillée et checklist de complétude
DashboardForm	Tableau de bord RH
EcheancierForm	Gestion des tâches RH
UsersForm	Gestion des comptes (réservé à l'administrateur)

La communication entre le client et l'API est centralisée dans une classe ApiClient, qui gère la sérialisation JSON (convention snake_case côté API) et l'ajout automatique du jeton Bearer à chaque requête authentifiée.

9. Fonctionnalités détaillées

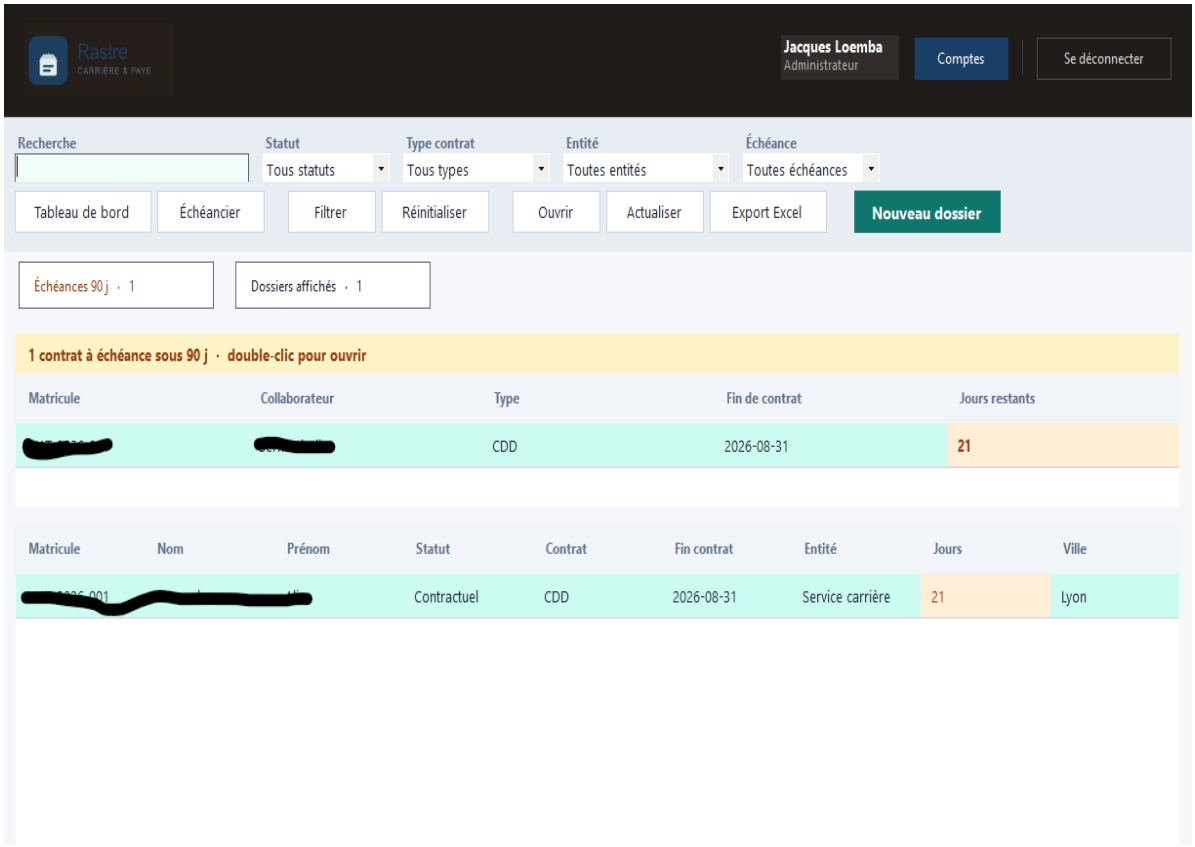
9.1 Connexion



Écran de connexion, avec vérification de la disponibilité de l'API en temps réel

L'écran de connexion affiche un indicateur de disponibilité de l'API (« Prêt — vous pouvez vous connecter ») avant même la saisie des identifiants, ce qui évite à l'utilisateur de tenter une connexion si le service est indisponible.

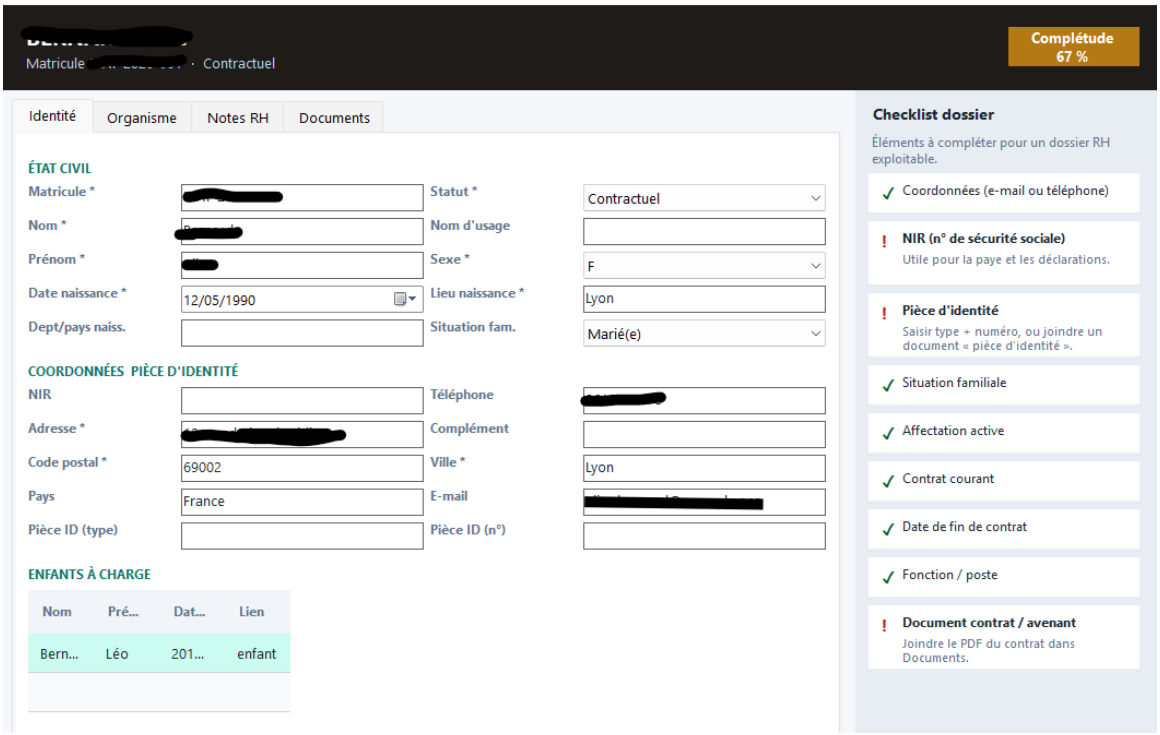
9.2 Liste des dossiers et alertes



Liste des dossiers collaborateurs avec filtres et alertes de contrats à échéance

L'écran principal présente la liste des dossiers avec des filtres combinables (statut, type de contrat, entité, échéance) et un export CSV. Un bandeau met en évidence les contrats arrivant à échéance sous 90 jours, avec un accès direct au dossier concerné par double-clic — une fonctionnalité pensée pour éviter qu'un renouvellement de contrat ne soit oublié.

9.3 Dossier collaborateur et checklist de complétude



Fiche dossier — onglet Identité, avec score de complétude et checklist associée

Le dossier collaborateur est organisé en quatre onglets : Identité, Organisme (Affectation et Contrat), Notes RH et Documents. L'onglet Identité regroupe l'état civil, les coordonnées, les informations de pièce d'identité et les enfants à charge. Le statut de l'agent (contractuel ou non) conditionne l'affichage de l'onglet Contrat : masqué pour les agents non contractuels, il est actif et obligatoire pour les contractuels.

Un score de complétude, affiché en tête de fiche (67 % dans l'exemple ci-dessus), est calculé à partir d'une checklist de critères jugés essentiels à un dossier RH exploitable : coordonnées renseignées, numéro de sécurité sociale, pièce d'identité, situation familiale, affectation active, contrat courant, date de fin de contrat, fonction occupée et document de contrat joint. Chaque critère manquant est signalé visuellement, ce qui guide le gestionnaire vers les informations à compléter en priorité.

10. Installation et démonstration

10.1 Prérequis

- XAMPP (Apache + MySQL)
- PHP 8.1 ou supérieur
- SDK .NET 8

10.2 Installation de la base de données

Deux méthodes sont possibles : exécuter le script php database/install.php, ou passer par phpMyAdmin en créant une base rastre puis en important successivement rastre.sql (schéma) et seed.sql (données de démonstration).

10.3 Lancement

Après démarrage d'Apache et de MySQL via XAMPP, le client se lance depuis le dossier client avec la commande dotnet run --project Rastre.Client. La connexion de démonstration s'effectue avec l'identifiant admin et le mot de passe admin123.

10.4 Scénario de démonstration

- Connexion avec le compte administrateur
- Ouverture d'un dossier existant ou création d'un nouveau collaborateur
- Ajout d'un contrat et d'une affectation
- Observation de la mise à jour automatique de la checklist de complétude
- Consultation du tableau de bord et des échéances
- Création d'une tâche dans l'échéancier
- Export CSV de la liste des dossiers

11. Difficultés rencontrées, tests et limites

11.1 Difficultés rencontrées

Le développement du projet a fait face à plusieurs difficultés techniques, réparties sur les différentes couches de l'application :

- Mise en place de l'environnement de développement : installation et configuration croisée du SDK .NET et de XAMPP.
- Gestion de la base MySQL : contraintes d'intégrité référentielle, imports de données, écriture et enchaînement des scripts de migration.
- Stabilité du client WinForms: gestion du cycle de vie des formulaires et des appels asynchrones vis-à-vis du thread d'interface (async/UI).
- Intégration de l'API REST : authentification par jeton Bearer, sérialisation/désérialisation JSON, application cohérente des droits par rôle.

Ces difficultés ont été traitées par des correctifs itératifs au fil du développement, accompagnés de la rédaction d'une documentation d'installation détaillée et de scripts de réinstallation de base permettant de repartir d'un état propre en cas de problème — une pratique qui a nettement fiabilisé les phases de test.

11.2 Tests

Les tests menés au fil du développement ont porté sur les scénarios suivants, sans qu'un plan de tests formel ni des indicateurs chiffrés (taux de couverture, nombre de cas exécutés) n'aient été établis à ce stade :

- Connexion réussie et échouée (identifiants valides / invalides)
- Opérations CRUD sur un dossier collaborateur selon le rôle connecté
- Vérification de l'accès en lecture seule pour le rôle « lecture »
- Déclenchement de l'alerte sur un contrat proche de l'échéance
- Évolution du score de complétude après ajout du NIR ou d'un document
- Création et clôture d'une tâche dans l'échéancier

11.3 Limites connues

- Environnement de développement local (XAMPP), sans HTTPS en dev
- Absence de module paye à ce stade
- Absence d'installateur Windows pour le client
- La checklist de complétude repose sur des règles métier heuristiques, et ne constitue pas un workflow réglementaire officiel

12. Méthodologie et organisation

Le projet a été mené en autonomie, selon une progression en six phases : analyse et rédaction du cahier des charges, conception (modèle de données, architecture 3-tiers, contrat d'API), réalisation (base de données, API,

client WinForms), extensions (notes, documents, tableau de bord, checklist, échéancier), recette (jeux de tests, corrections d'ergonomie) et livraison (documentation et dépôt GitHub). La première version fonctionnelle (MVP+, décrite dans ce rapport) a été développée en environ trois semaines. Le projet reste néanmoins en développement actif au-delà de cette première version : les fonctionnalités listées en section 5.2 (module paye, titularisation complète, portail agent, etc.) constituent la feuille de route des prochaines itérations.

13. Conclusion et perspectives

En trois semaines, Rastre a atteint son objectif initial : offrir aux gestionnaires RH un outil centralisant le dossier administratif des collaborateurs, avec une réelle aide au pilotage grâce aux alertes de contrat, au tableau de bord, à la checklist de complétude et à l'échéancier de tâches. Sur le plan technique, le projet a permis de mettre en œuvre une architecture 3-tiers complète, de la base de données jusqu'au client lourd, en passant par une API REST sécurisée par rôles. Le projet reste en développement actif au-delà de ce MVP. Plusieurs pistes d'évolution ont été identifiées pour la suite :

- Développement du module paye (bulletins, cotisations, déclarations)
- Gestion complète de la titularisation (corps, grade, échelon)
- Génération de courriers types au format PDF
- Import de données depuis Excel
- Droits d'accès plus fins sur les données les plus sensibles
- Déploiement en HTTPS

14. Annexes

- Cahier des charges complet (docs/CDC.md)
- Catalogue de l'API (docs/API.md)
- Schéma SQL et modèle conceptuel de données
- Captures d'écran : connexion, liste des dossiers, fiche dossier, gestion des comptes
- Guide d'installation (README.md)

Dépôt GitHub: https://github.com/Jacque004/rastre