
Human Resource Management Application

Maxime Borgeaud, SICA2a

CPNV, TPI 2024



Chef de projet : Mr. Pascal Benzonana

Expert 1: Mr. Ernesto Montemayor

Expert 2: Mr. Antoine Mveng Evina

Table des matières

1	Introduction.....	4
2	Analyse	4
2.1	Méthode de projet	4
2.2	Backup	4
2.3	Fonctionnalités prévues	4
2.4	Déroulement	5
2.5	Planification initiale	5
2.6	Maquettes	7
2.7	Modèle conceptuel de données.....	16
3	Conception	19
3.1	Concepts	19
3.2	Outils	19
3.3	Architecture.....	19
3.4	Gestion des erreurs par le serveur	19
3.5	Modèle Logique de Données.....	20
3.6	Diagrammes de classes.....	21
3.7	Diagrammes de flux.....	22
3.8	User cases.....	28
4	Réalisation	32
4.1	Démarrer l'application en local	32
4.2	Base de données.....	32
4.2.1	Structure.....	32
4.2.2	Mise en ligne de la base de données	34
4.2.3	Droits d'accès	34
4.2.4	Connexion.....	35
4.2.5	Données.....	35
4.3	Initialisation du projet	35
4.4	Génération des modèles	36
4.5	Structure du projet.....	36
4.6	Logique des endpoints	38
4.7	Stratégie de tests.....	39
4.7.1	Tests Backend	40
5	Conclusion	49

5.1	Erreurs/imprécisions restantes	49
5.2	Améliorations possibles.....	49
5.3	Conclusion personnelle	50
6	Annexes	50
6.1	Résumé du projet	50
6.2	Liste documents fournis	50
6.3	Planification effective.....	50
6.4	Glossaire	51
6.5	Sources	52
6.6	Archives - base de données.....	52

1 Introduction

Le projet consiste en un système de gestion des employés d'une entreprise. L'objectif est de développer une application web qui permet de gérer les informations des employés ainsi que les offres d'emploi, évaluations et candidatures. L'idée est d'optimiser les processus RH qui peuvent être longs et répétitifs afin de permettre aux utilisateurs d'éviter de générer des erreurs dans le système.

L'application comprend la consultation de ses données personnelles et professionnelles ainsi que ses demandes de congés et ses évaluations personnelles. Les employés RH ont évidemment des droits particuliers. Ils peuvent valider ou refuser des demandes de congé, consulter et modifier les informations des autres employés ainsi que gérer les offres d'emplois, candidatures et entretiens d'embauches. Cette application est développée dans le cadre du CPNV de Ste-Croix comme projet de TPI.

2 Analyse

2.1 Méthode de projet

La méthode utilisée est « AGILE » afin d'avoir une gestion flexible du projet et une vision d'ensemble du projet. La plateforme agile choisie est « Icescrum » et est disponible à l'adresse suivante : <https://icescrum.cpnv.ch/p/TPIRH/#/>

2.2 Backup

J'effectue des sauvegardes tous les jours de l'avancée du projet sur un repository Github, dont voici le lien : https://github.com/maxDevelopement/TPI_RH

2.3 Fonctionnalités prévues

Voici une description des fonctionnalités souhaitées pour l'application

Gestion des employés

Un employé RH doit pouvoir consulter la liste des employés actuels ou ex-employés avec toutes leurs informations personnelles et de leurs contrats. Il doit également pouvoir y ajouter ou supprimer un nouvel employé ainsi que modifier d'éventuelles informations

Gestion des demandes de congé

Un employé peut consulter à tout moment ses demandes de congés sur un calendrier dynamique. Il peut y voir le statut des demandes (accepté, refusé ou en attente) mais également faire une nouvelle demande de congé. Un employé RH peut consulter à tout moment la liste des demandes de congés en attente et leurs attribuer un nouveau statut, (refusé ou accepté).

Gestion des performances

Chaque employé reçoit une évaluation par année. Celle-ci contient une note (1 à 6), une note positive et une négative. Un employé RH peut Remettre une évaluation d'un employé pour l'année courante, s'il n'en a pas déjà reçu une.

Gestion du recrutement

Les employés RH peut également gérer les offres d'emplois et leurs candidatures. Ils peuvent ajouter/modifier/supprimer une candidature pour un poste, programmer un entretien et attribuer un statut à la candidature (accepté, refusé ou en attente).

2.4 Déroulement

Le projet débute le 30 avril 2024 à 8h et se termine le 29 mai à 11h35. Il se déroule sur 90 heures et va se diviser en 4 parties distinctes : 20% d'analyse, 45% d'implémentation, 10% de tests, 25% de documentation. Il y a également une découpe temporelle de 4 sprints d'environ une semaine dont voici une représentation visuelle :

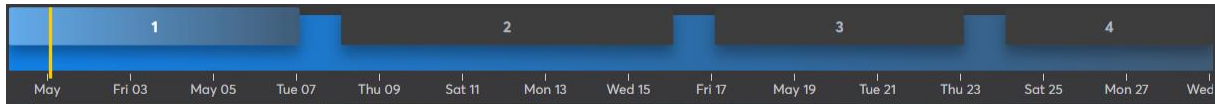


Figure 1 : Découpe temporelle

2.5 Planification initiale

Sprint 1

- Date : 30.04 ➔ 07.05
- Goal : *Analyse/conception du projet bien amorcée*

Sprint 2 :

- Date : 08.05 ➔ 16.05
- Goal : *Backend bien amorcée + adaptation conception*

Sprint 3 :

- Date : 17.05 ➔ 23.05
- Goal : *Backend terminée et frontend amorcé + adaptation conception*

Sprint 4 :

- Date : 24.05 ➔ 29.05
- Goal : *Frontend terminé + adaptation de la conception*

Planification du sprint 1

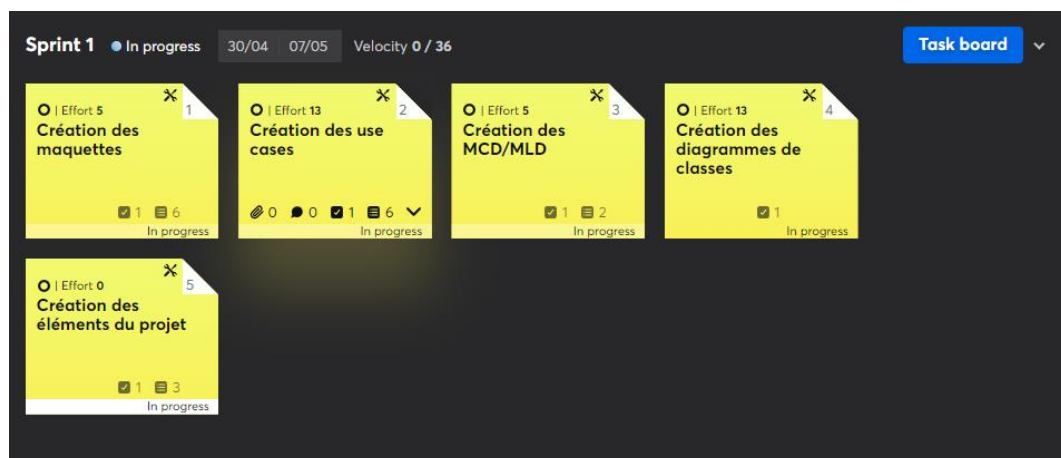


Figure 2 : Planification sprint 1

Planification du sprint 2

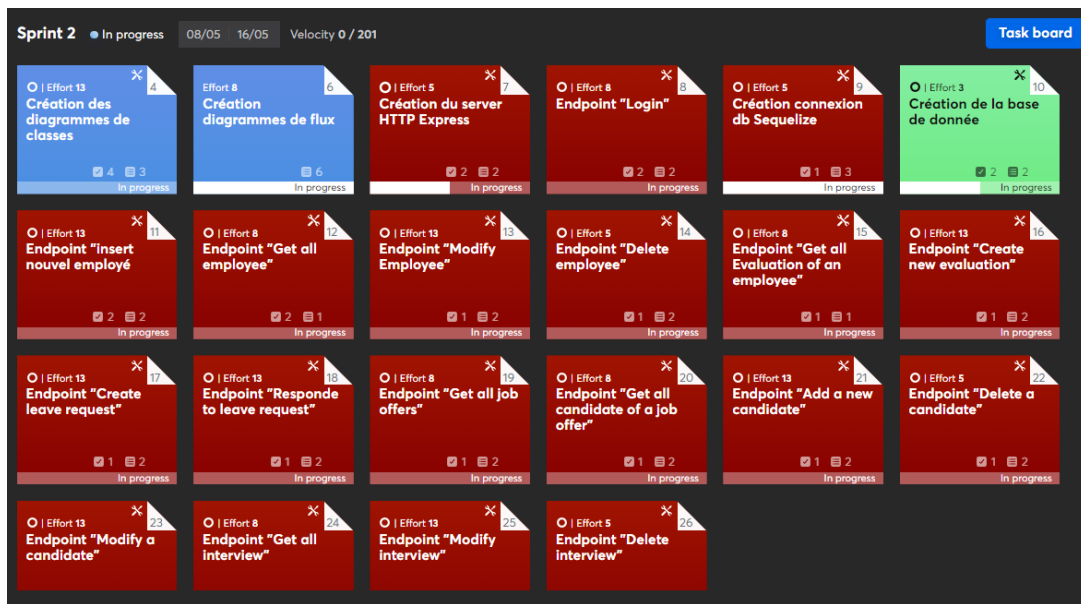


Figure 3 : Planification sprint 2

Les stories des diagrammes de classes et de flux n'ont pas été entièrement terminée lors du sprint précédent

Planification du sprint 3

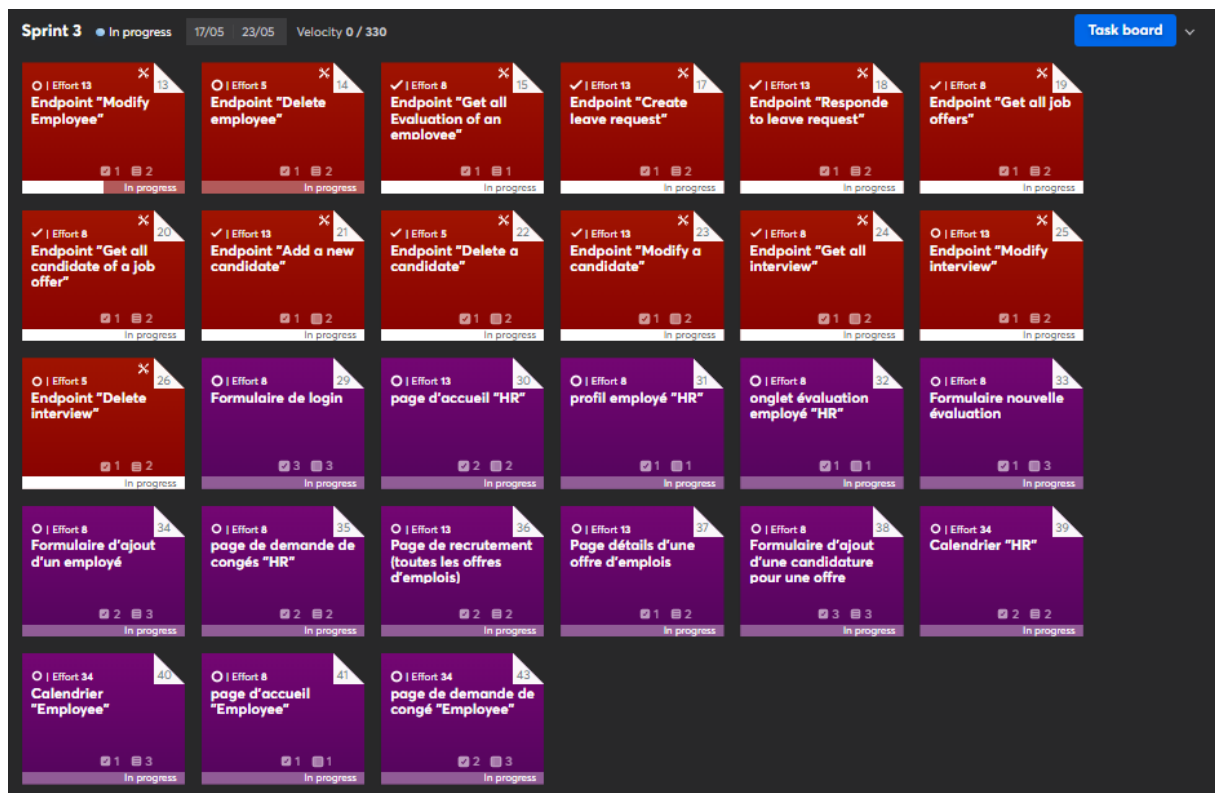


Figure 4 : Planification sprint 3

Un changement de la structure de la base de données peu avant la fin du sprint 2 engendra une légère régression du code backend. Ce qui déplaça les stories non impactées dans le sprint 3.

Planification du sprint 4

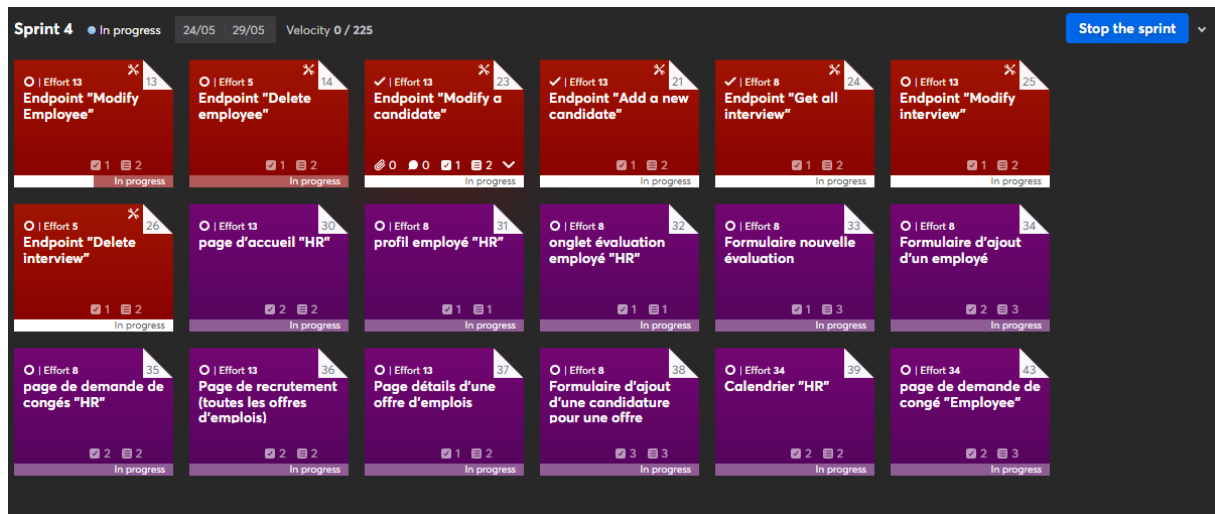


Figure 5 : Planification sprint 4

Ici se retrouvent toutes les stories qui n'ont pas pu être terminées durant le sprint précédent. Malheureusement toutes ne pourront pas être terminées dans les temps.

2.6 Maquettes

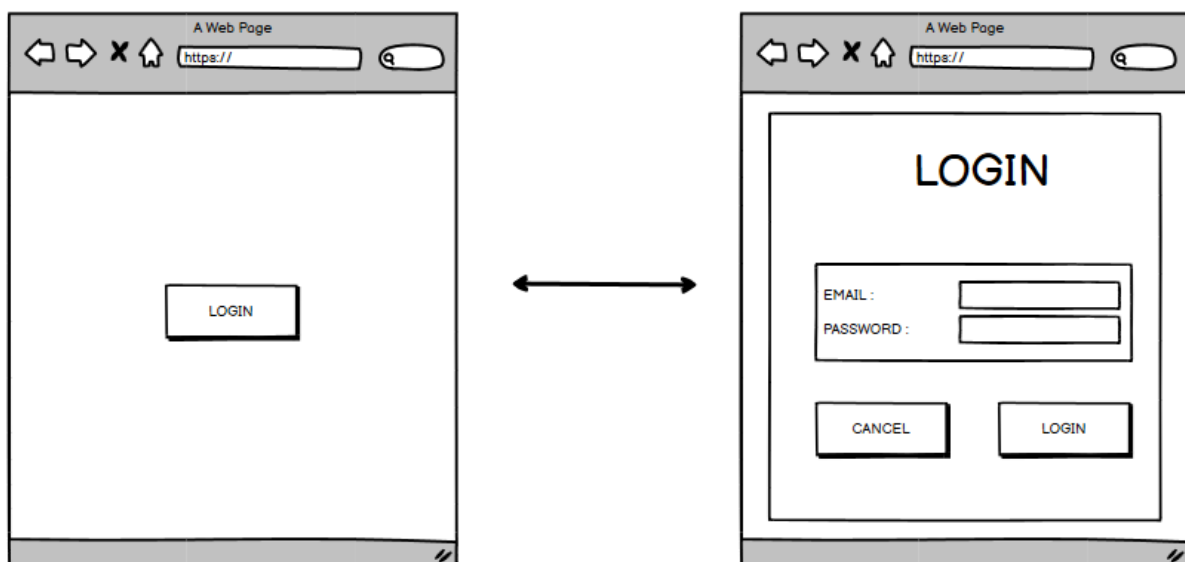


Figure 6 : login

Profile

Nom : yasin

Prenom : Maurer

Tél : +41 76 666 66 66

Email : yasin.maurer@entreprise.com

Rue : Chemin des lilas

Numéro : 999

NPA : 1066

Ville : Ville city

Poste : PDG

Type : CDI

Taux : 80%

Date de début : 19.06.24

Date de fin :

IBAN : 2344-2344-2344-44

Figure 7 : Profil employé - RH

Staff gestion

Ajouter un nouvel employé

Chercher un employé ... ->

Maxime	Borgeaud	Consulter
Ayami	Ogay	Consulter
Lucas	Machard	Consulter
Valentin	Maurer	Consulter

Logout
Etes-vous sur de vouloir vous déconnecter ?

Non Oui

Figure 8 : Gestion des employés – RH

https://

Calendrier

Evaluations

Supprimer

Profil

Nom:	yasin
Prénom:	Maurer
Tél :	+41 76 666 66 66
Email :	yasin.maurer@entreprise.com
Rue :	Chemin des lilas
Numéro:	999
NPA :	1066
Ville :	Ville city
Poste :	PDG
Type :	CDI
Taux :	80%
Date de début :	19.06.24
Date de fin :	
IBAN :	2344-2344-2344-44

Remettre évaluation 2024

Evaluation 2023

6

Points forts

Points à améliorer

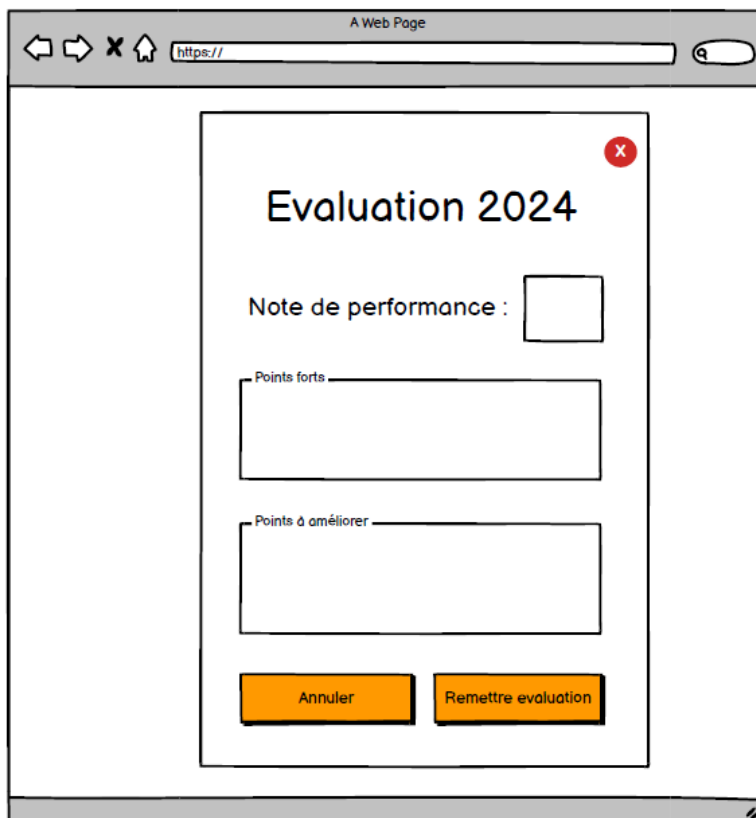
Evaluation 2022

5

Points forts

Points à améliorer

Figure 9 : profil employé avec évaluations ouvertes - RH



A Web Page

https://

Evaluation 2024

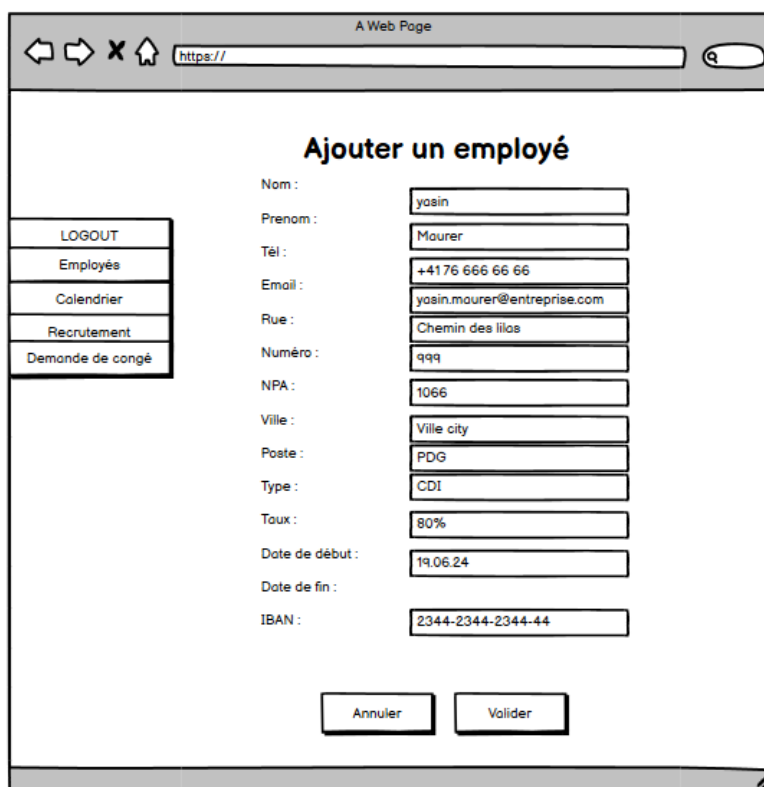
Note de performance :

Points forts

Points à améliorer

Annuler Remettre evaluation

Figure 10 : formulaire nouvelle évaluation



A Web Page

https://

Ajouter un employé

Logout
Employés
Calendrier
Recrutement
Demande de congé

Nom :

Prenom :

Tél :

Email :

Rue :

Numéro :

NPA :

Ville :

Poste :

Type :

Taux :

Date de début :

Date de fin :

IBAN :

Annuler Valider

Figure 11 : Formulaire nouvel employé

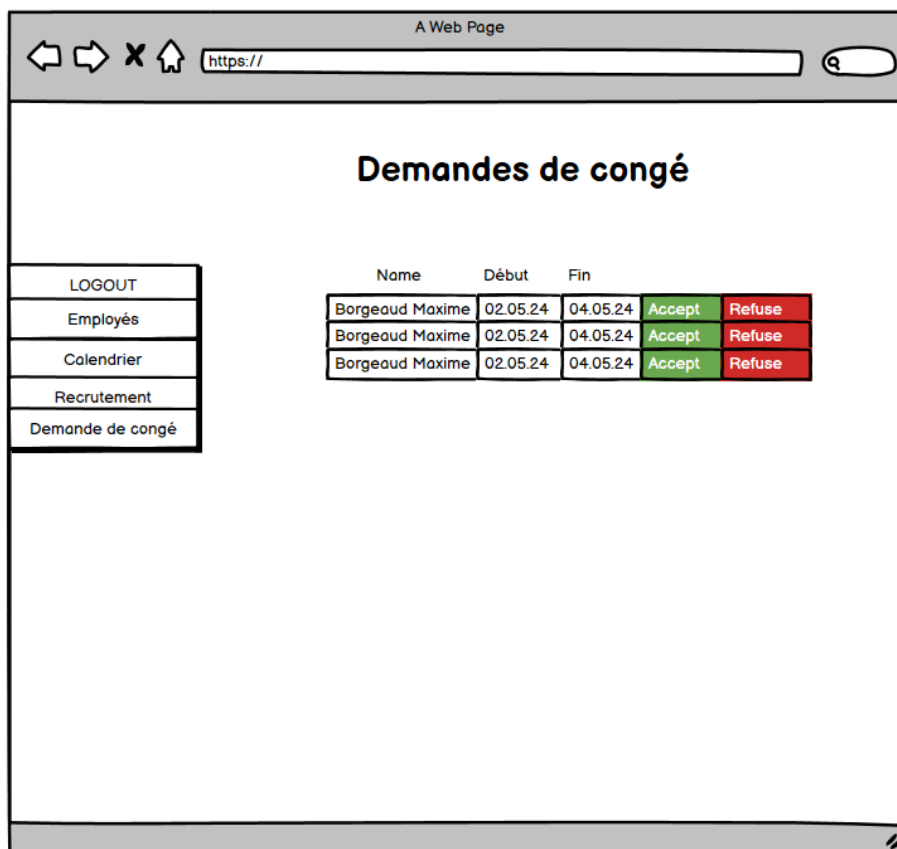


Figure 12 : Gestion des demandes de congés

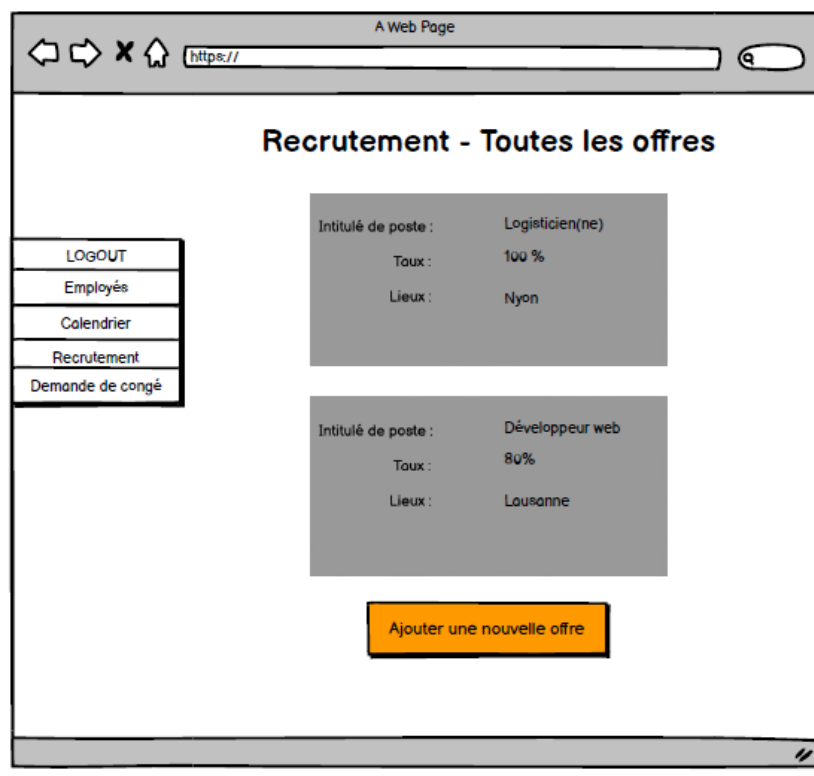


Figure 13 : Liste offres d'emplois

Recrutement - offre d'emploi

Intitulé de poste : Développeur web
Taux : 80%
Lieux : Lausanne

Nouveau candidat **Archiver**

Candidates

Firstname : Ayami
Lastname : Da silva
Postulation date : 02.05.24
Interview date : En attente
Decision : En attente

Suppression du candidat
Etes-vous sur de vouloir supprimer ce candidat ?
Non Oui

MAY 2024

S	M	T	W	T	F	S
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Figure 14 : détails offre d'emplois

Recrutement - Ajouter un candidat

Intitulé de poste : Développeur web
Taux : 80%
Lieux : Lausanne

Ajouter

Prenom : Antoine
Nom : Machard
Email : Anoine@candidat.ch
Tél : +41 78 666 66 66
Date de postulation : 03.05.24
Décision : pending

Figure 15 : Formulaire nouveau candidat

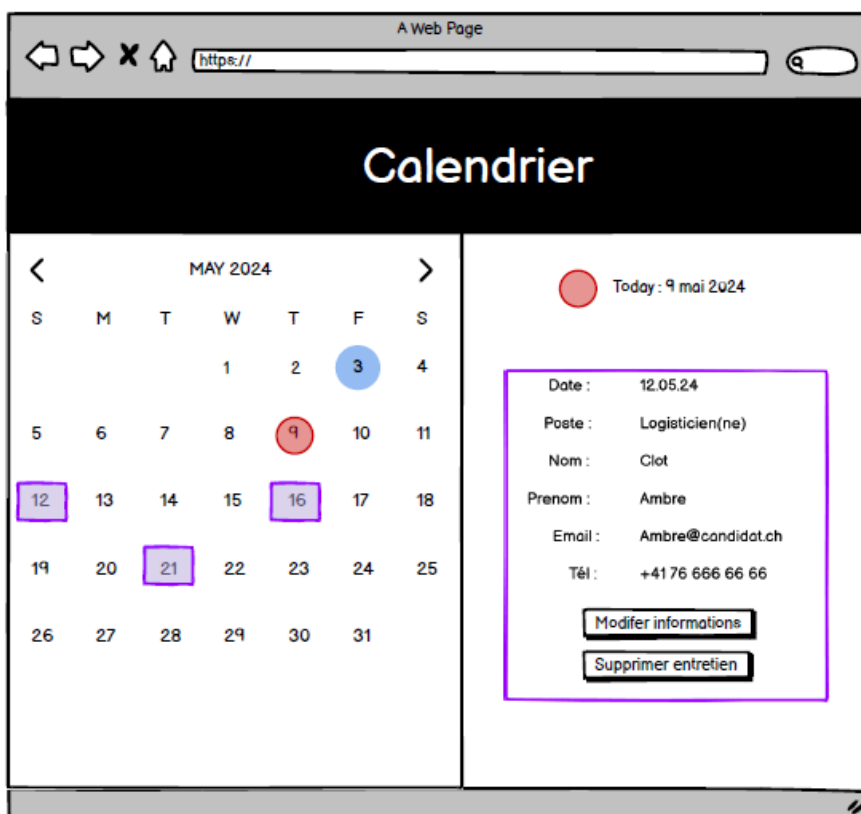


Figure 16 : calendrier RH

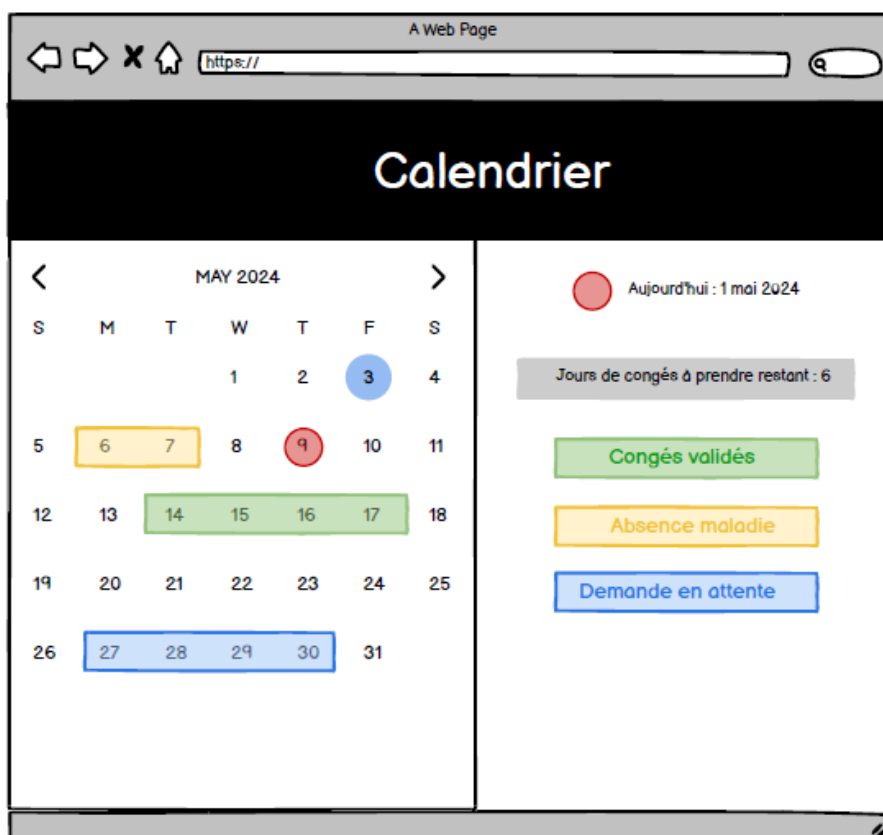


Figure 17 : Calendrier employé

A Web Page

https://

Profile

Nom : yasin

Prenom : Maurer

Tél : +41 76 666 66 66

Email : yasin.maurer@entreprise.com

Rue : Chemin des lilas

Numéro : 999

NPA : 1066

Ville : Ville city

Poste : PDG

Type : CDI

Taux : 80%

Date de début : 19 / 06 / 24

Date de fin :

IBAN : 2344-2344-2344-44

Calendrier

Evaluations

Figure 18 : profil employé - employé

A Web Page

⬅ ➡ ✕ 🏠

Q

Calendrier

Evaluations

Profile

Nom :	<input style="width: 90%;" type="text" value="yasin"/>
Prenom :	<input style="width: 90%;" type="text" value="Maurer"/>
Tél :	<input style="width: 90%;" type="text" value="+41 76 666 66 66"/>
Email :	<input style="width: 90%;" type="text" value="yasin.maurer@entreprise.com"/>
Rue :	<input style="width: 90%;" type="text" value="Chemin des lilas"/>
Numéro :	<input style="width: 90%;" type="text" value="999"/>
NPA :	<input style="width: 90%;" type="text" value="1066"/>
Ville :	<input style="width: 90%;" type="text" value="Ville city"/>
Poste :	<input style="width: 90%;" type="text" value="PDG"/>
Type :	<input style="width: 90%;" type="text" value="CDI"/>
Taux :	<input style="width: 90%;" type="text" value="80%"/>
Date de début :	<input style="width: 90%;" type="text" value="19.06.24"/>
Date de fin :	<input style="width: 90%;" type="text"/>
IBAN :	<input style="width: 90%;" type="text" value="2344-2344-2344-44"/>

Evaluation 2023

6

Points forts
Points à améliorer

Evaluation 2022

5

Points forts
Points à améliorer

Figure 19 : Profil employé avec évaluations ouvertes - employé

Date impression 29.05.2024

15 / 54

Maxime Borgeaud

The screenshot shows a web browser window with the title 'A Web Page'. The address bar contains 'https://'. The main content area is titled 'Calendrier'. On the left is a calendar for May 2024. The date 1st is highlighted in red, and the 3rd is highlighted in blue. On the right is a 'Demande de congé' form. It includes a red circle icon and the text 'Aujourd'hui : 1 mai 2024'. The form has a dropdown menu for 'Raison' with a tooltip showing four options: 'Vacances', 'Maladie', 'Urgence personnelle', and 'Autre :'. Below the dropdown are two radio buttons: 'Congé payé' and 'Congé non-payé'. At the bottom of the form are two buttons: 'Envoyer la demande de congé' and 'Annuler'.

Figure 20 : Formulaire nouvelle demande de congé

2.7 Modèle conceptuel de données

La table « Employés » contient les données personnelles d'un membre du personnel et ne sera pas supprimé même à la fin du contrat. Sa table associée, « Contrats », a les données complémentaires lié à son travail dans l'entreprise. Si des informations professionnelles doivent changer pour cet employé, un backup des anciennes données est automatiquement créé afin de garder trace de tous changements. Les tables « Evaluations » et « congés » sont également liées à « Contrats ».

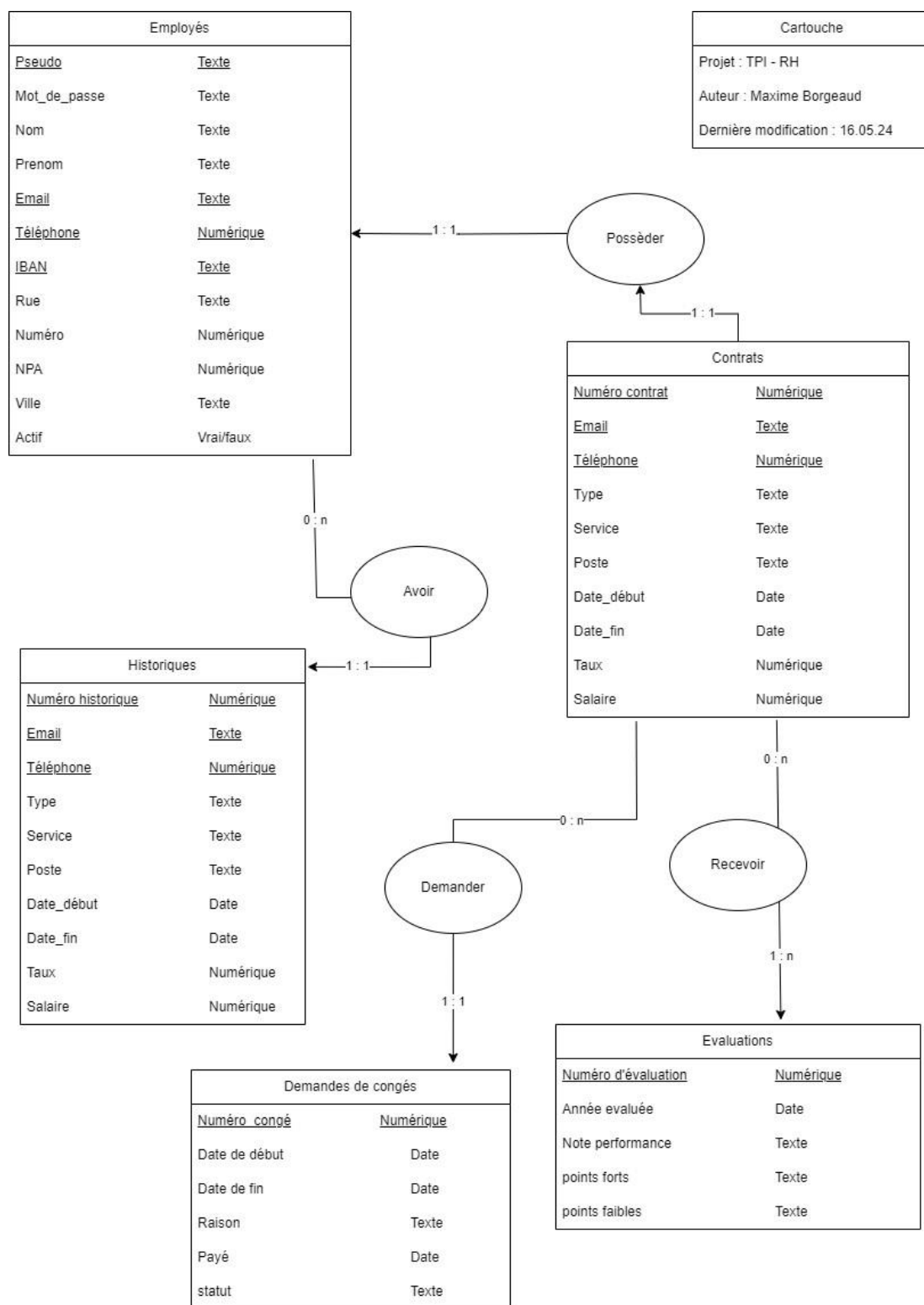


Figure 21 : MCD (1)

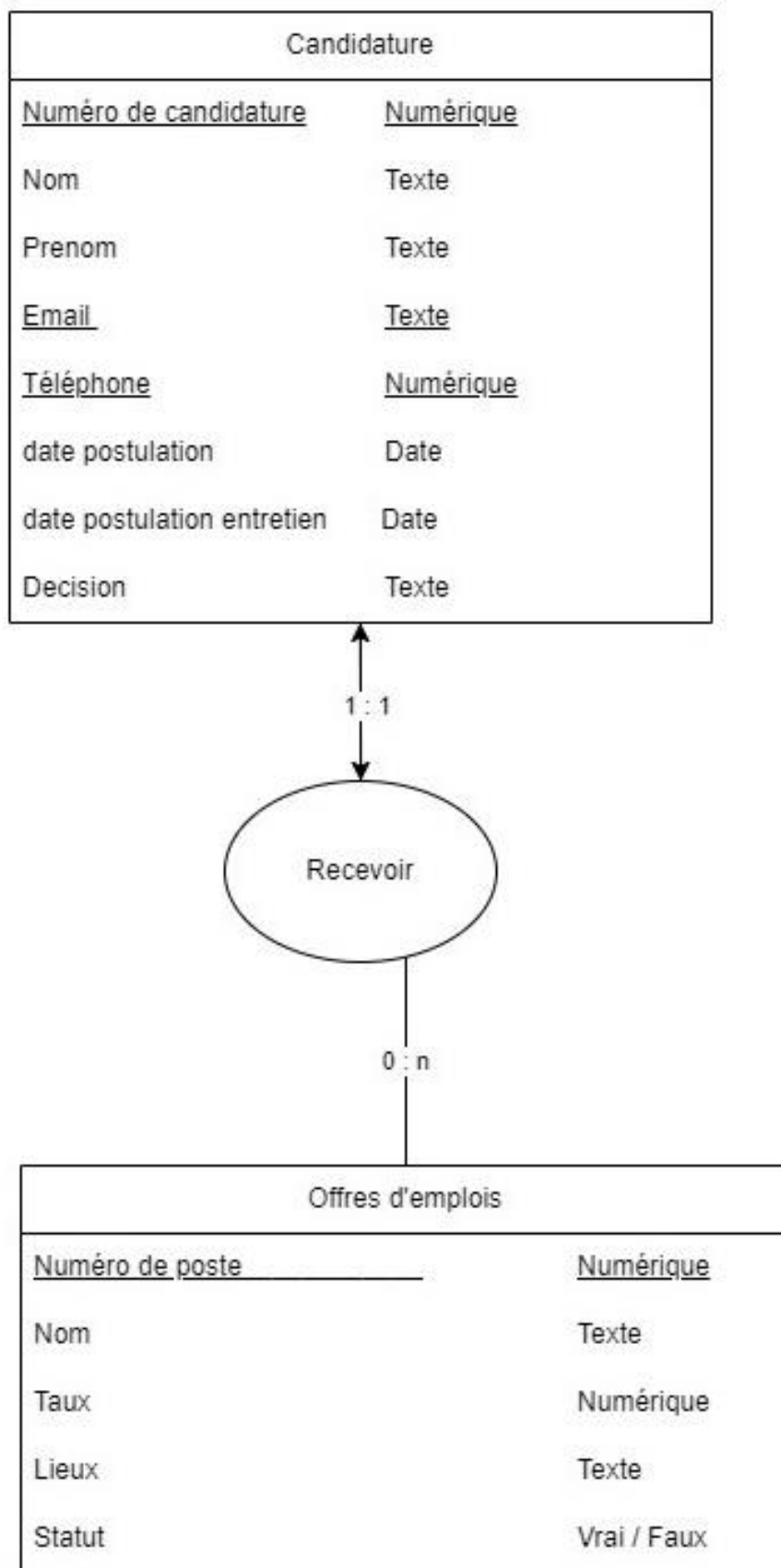


Figure 22 : MCD (2)

3 Conception

3.1 Concepts

Voici les principaux concepts qui composent le projet :

- ⇒ Modèle – Vue - Contrôleur
- ⇒ Programmation Orienté Objet
- ⇒ Un système d'authentification haché
- ⇒ Base de données MariaDb

3.2 Outils

Les outils utilisés pour la réalisation de l'application sont les suivants :

- ⇒ Langages de programmation : Javascript, Html-css
- ⇒ Frameworks/bibliothèques : NodeJS, VueJS, webPack
- ⇒ Editeur de texte : Visual Studio Code
- ⇒ Hébergement de la base de données : Apache (local), SwissCenter(online)
- ⇒ Requêtes http : Insomnia
- ⇒ Gestion de projet : Icescrum
- ⇒ Versioning : Github

3.3 Architecture

Cette application est conçue comme « client léger » dans le sens qu'il y a une architecture type client-serveur mais rien (ou presque) n'est installé sur le client. J'ai opté pour une architecture application web plutôt que « site web ». Le serveur, ne fournit qu'un seul fichier html qui évolue selon les actions de l'utilisateur, ce qui optimise les performances car le serveur renvoie beaucoup moins de fichiers. La création de bundle Webpack va également dans cette direction.

3.4 Gestion des erreurs par le serveur

Lorsque les utilisateurs envoient des requêtes au serveur, il peut se passer une multitude d'erreurs qui peuvent bloquer le serveur. Le but est que le serveur ne s'arrête jamais même lorsqu'un cas problématique se présente. Afin de gérer cela correctement j'attache un statut de requête à chaque réponse du serveur :

- ⇒ Les erreurs causées par les utilisateurs, par exemple lorsqu'il envoie des données invalides etc. (statut : 400)
- ⇒ Les erreurs causées par le serveur, par exemple s'il y a des problèmes de connexion avec la base de données ou un bug (statut : 500)
- ⇒ Une requête valide qui a réussi sa tâche (statut : 200)

J'attache également toujours un message à la requête afin d'afficher un message selon le résultat. Concernant les erreurs 400 il y en a 2 types « *error_unicity* » si l'on tente d'insérer des données qui contredit une règle d'unicité et « *error_data* » qui est retourné lorsque les données envoyées par le client sont invalides pour différentes raisons. Si le traitement des données a été correctement réalisé, le message de retour est : « *success_nomEndpoint* ». Si le traitement s'est exécuté sans problème mais qu'il n'y a aucune donnée à renvoyer le message est « *success_noData* ».

3.5 Modèle Logique de Données

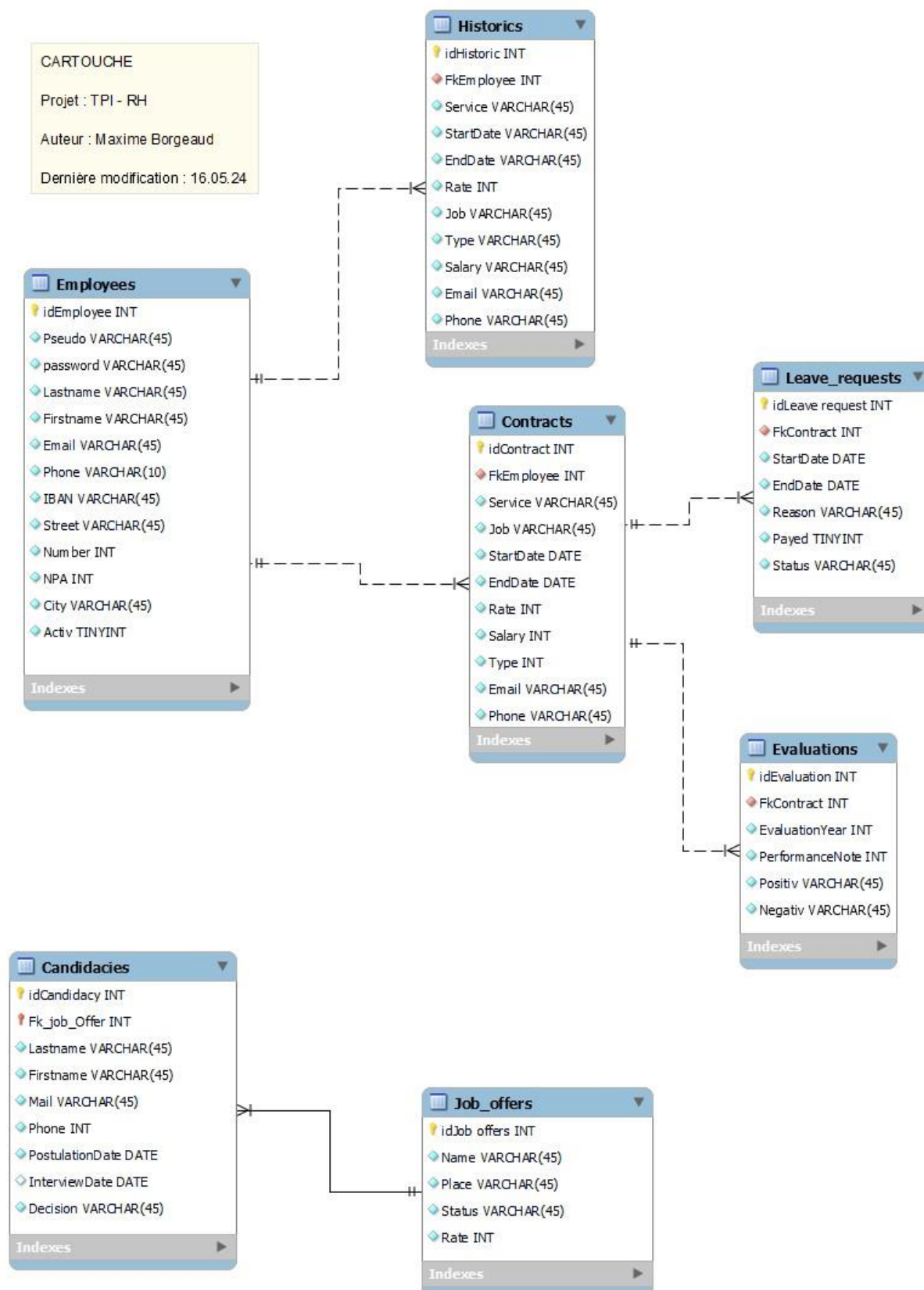


Figure 23 : MLD

3.6 Diagrammes de classes

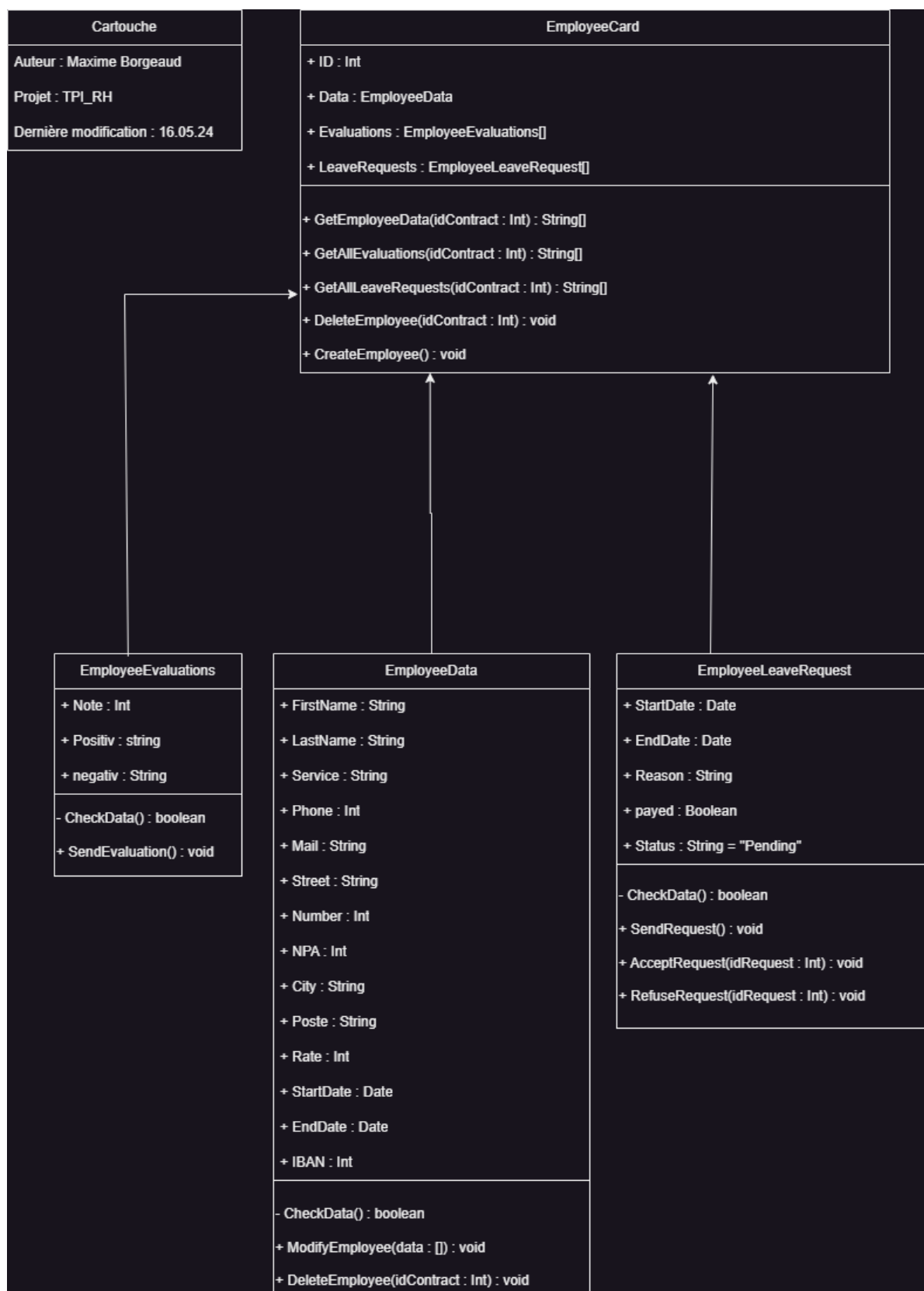


Figure 24 : Diagramme de classe

3.7 Diagrammes de flux

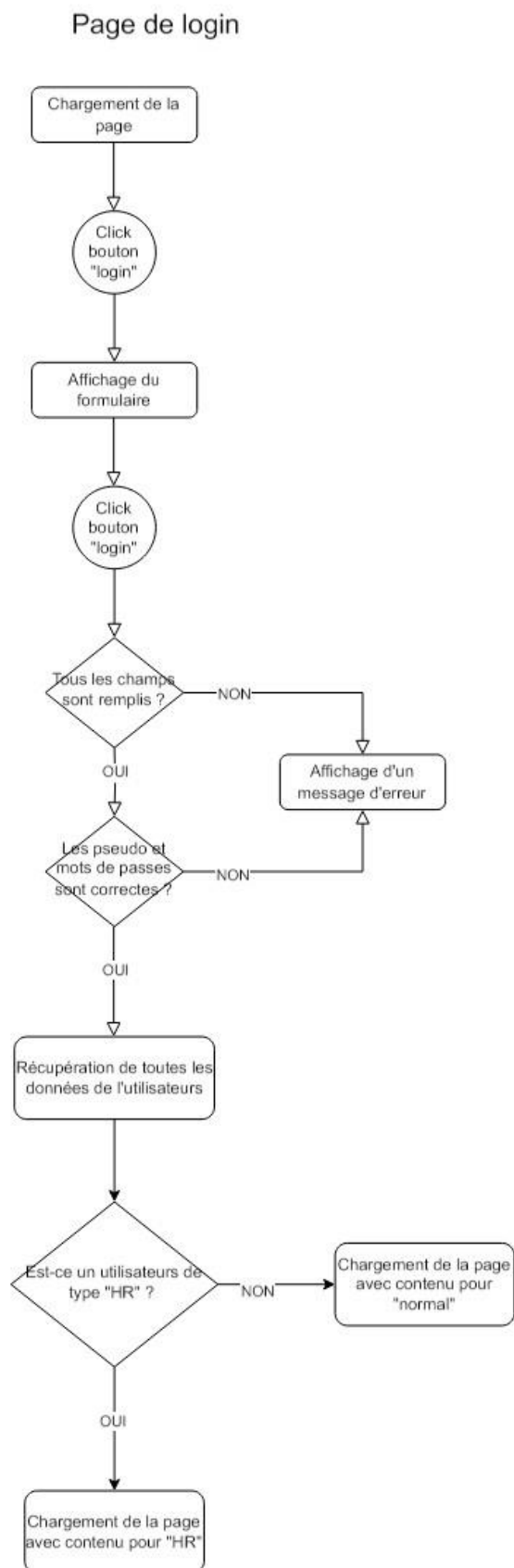


Figure 25 : login

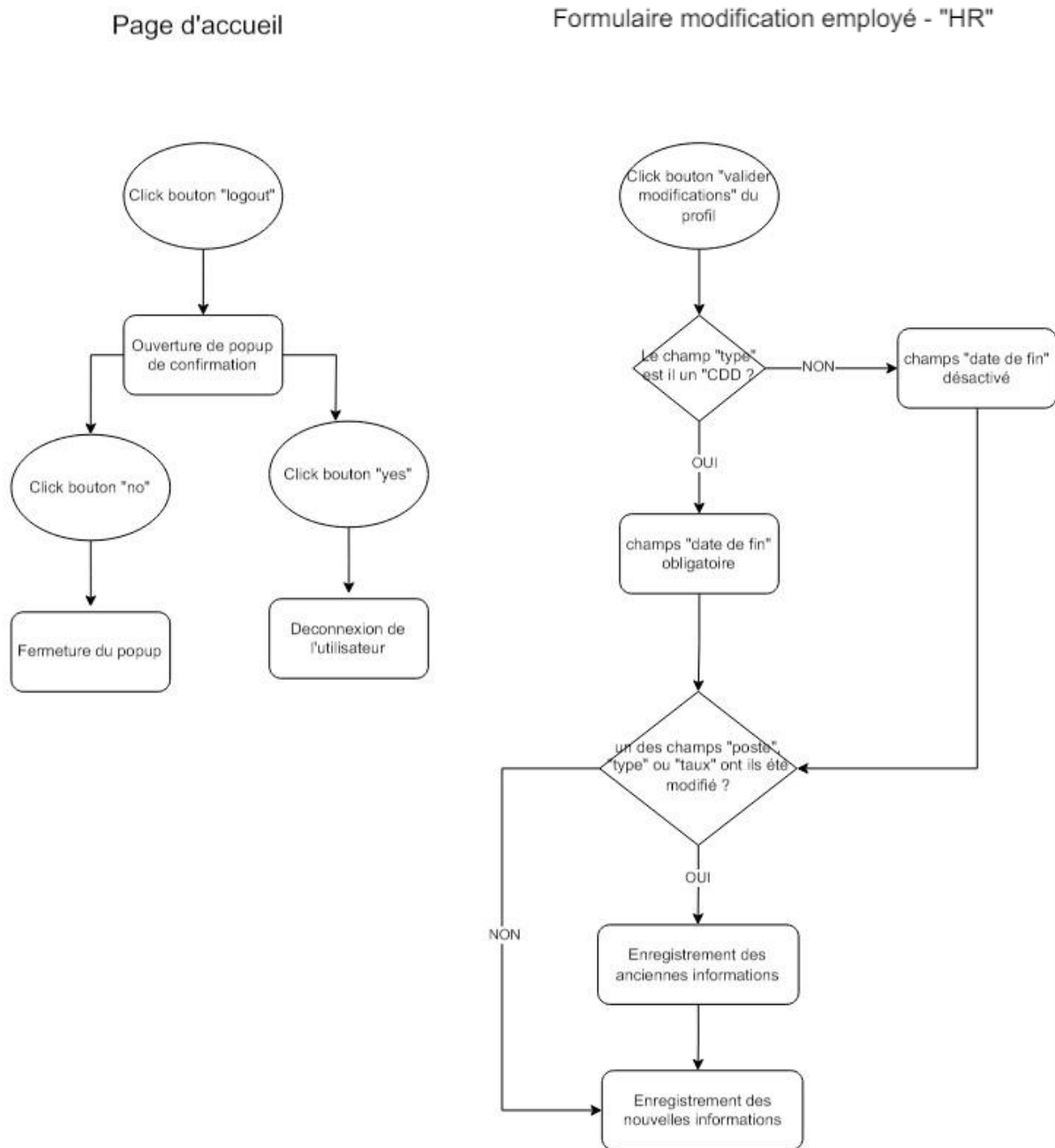


Figure 26 : logout et modification employé

Formulaire nouvel employé - "HR"

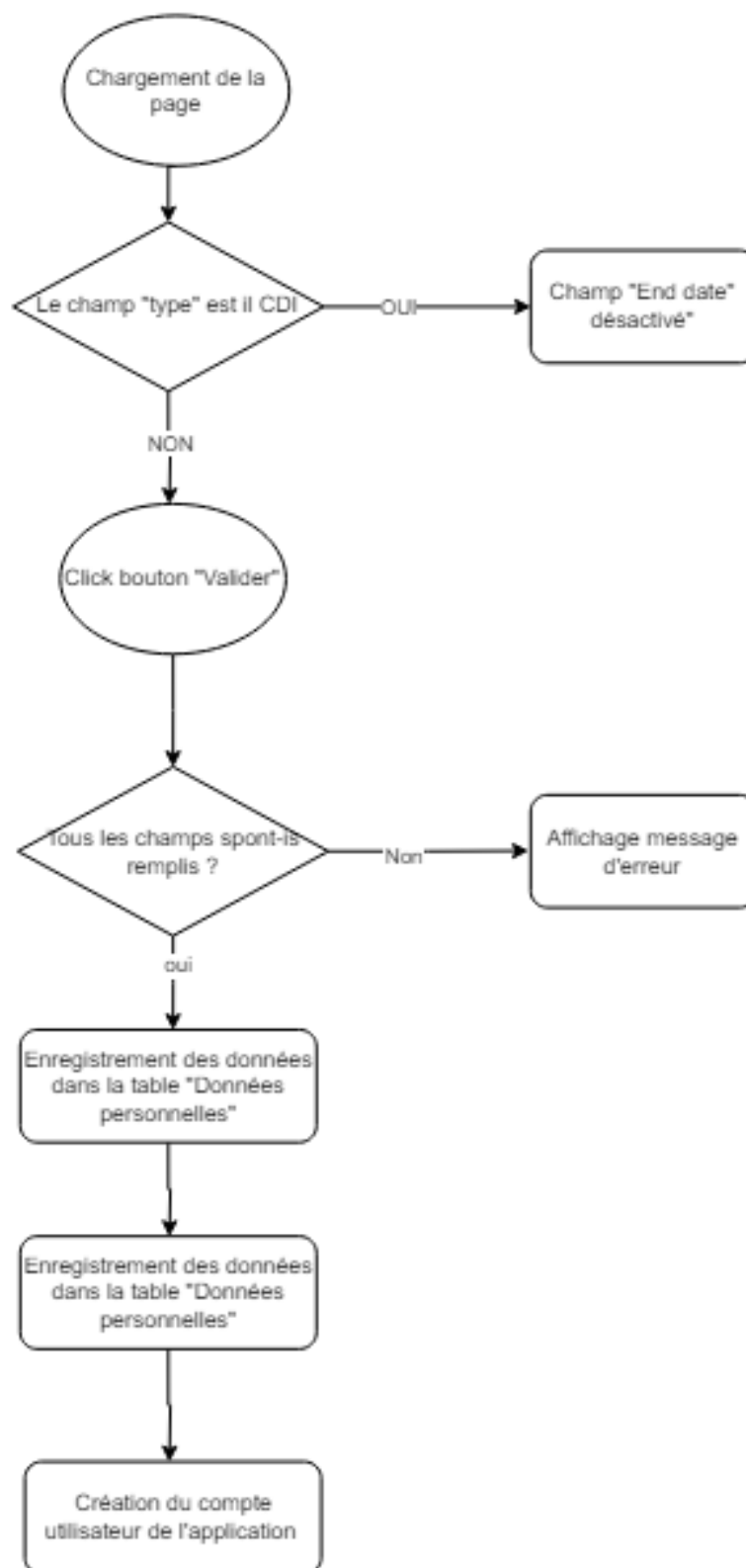
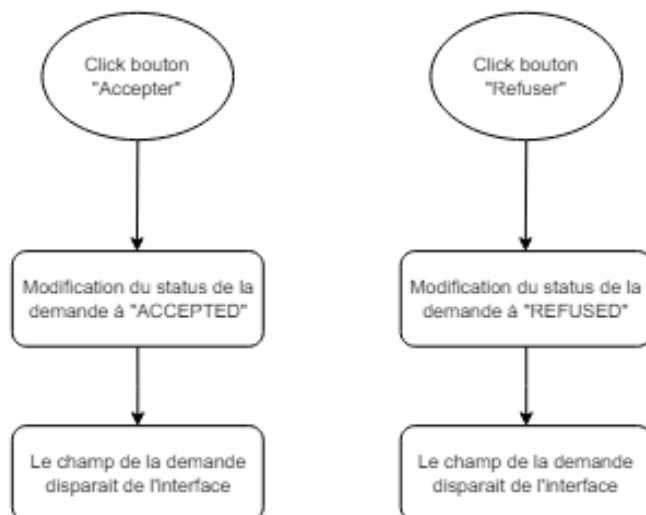


Figure 27 : Formulaire nouvel employé

Page "Gestion des demandes de congés" - "HR"



Page "Remettre une evaluation"

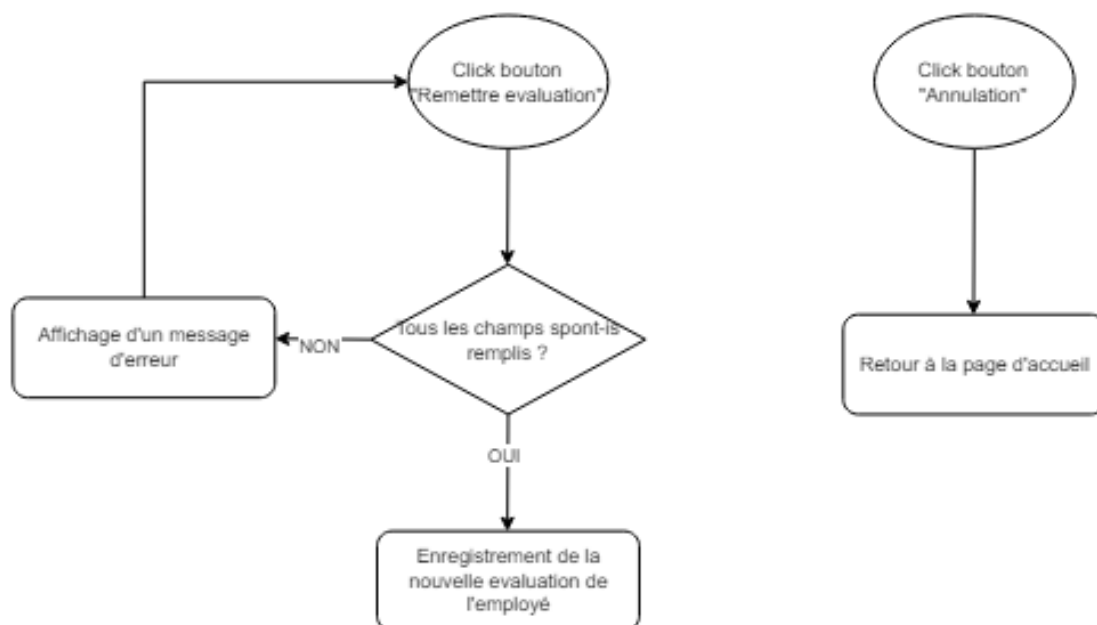


Figure 28 : nouvelle évaluation et gestion des demandes de congé

Profil employé – "HR"

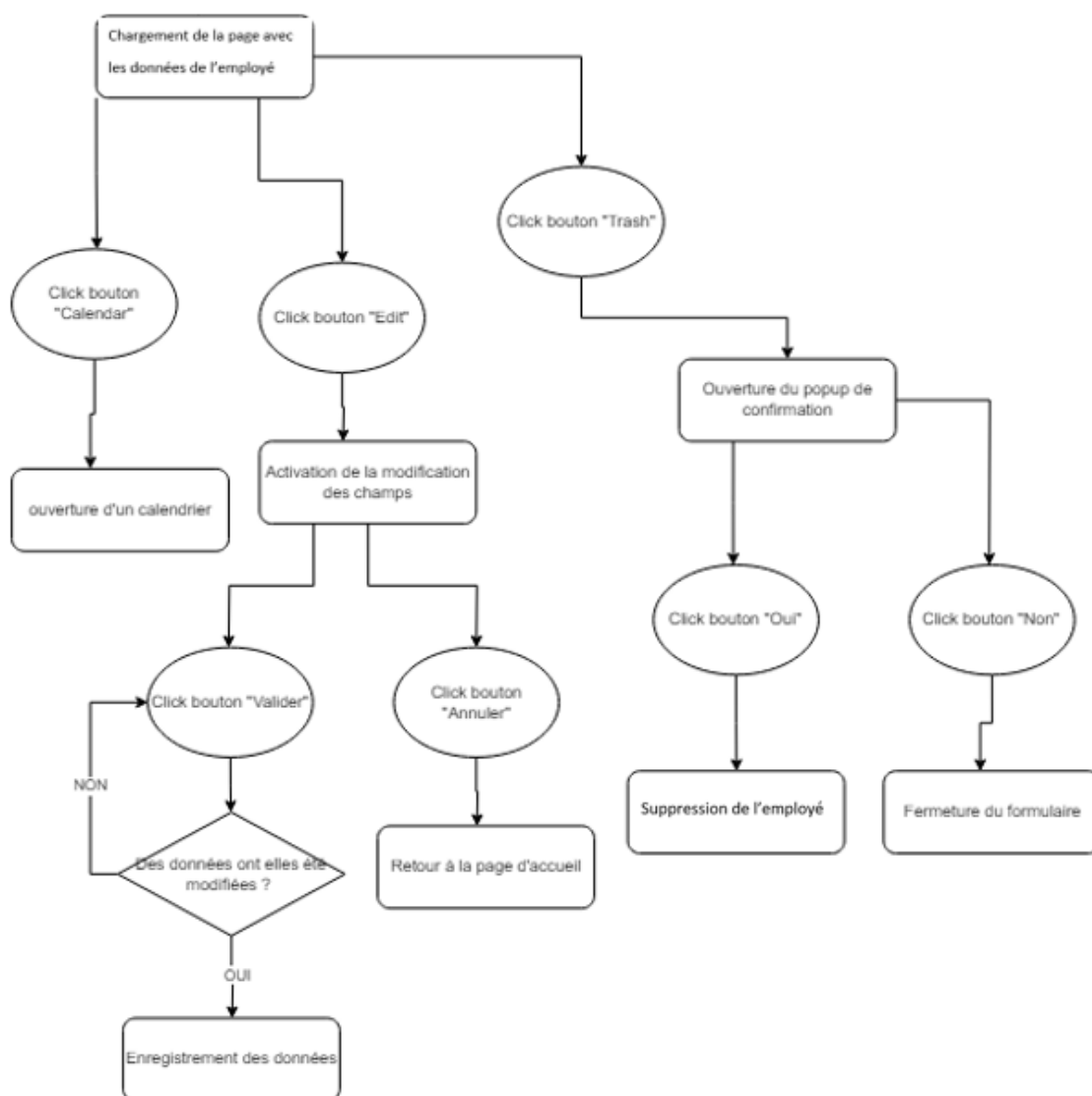


Figure 29 : Profil employé

Calendrier employé

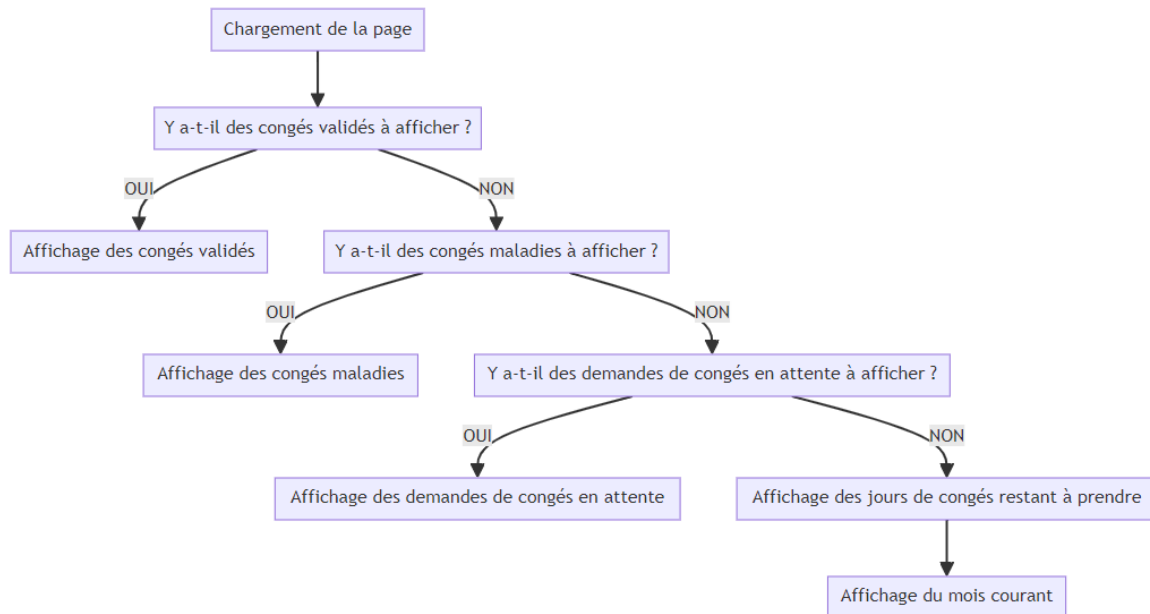


Figure 30 : Calendrier employé (généré par chat gpt à partir du drawio)

Faire une demande de congé - "employé"

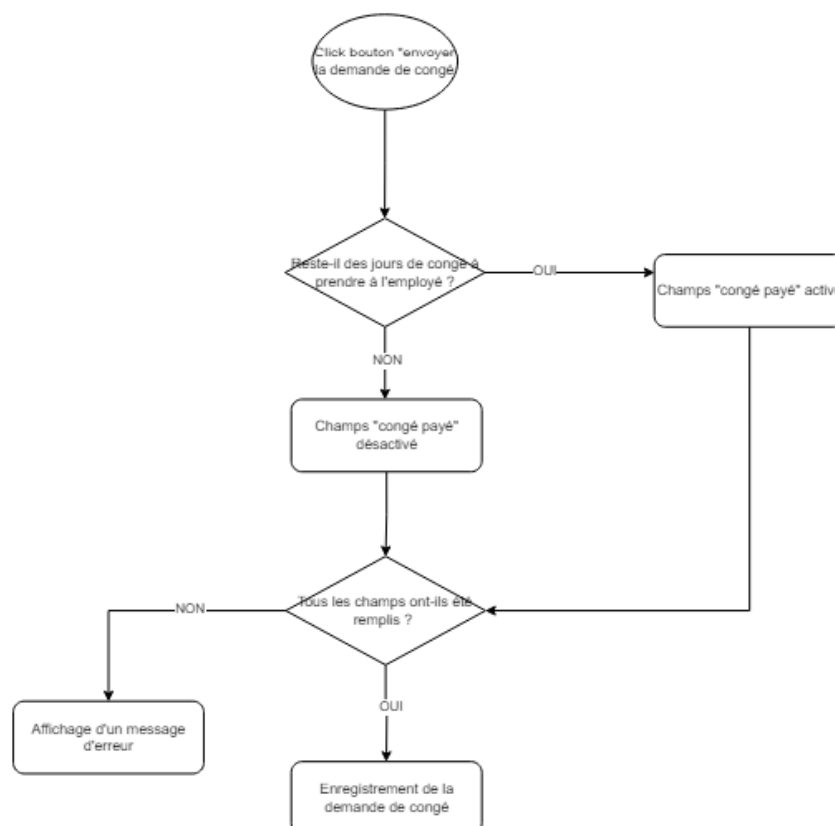


Figure 31 : nouvelle demande de congé

3.8 User cases

Page de login				
Objectif	Action	Condition	Scénario echec condition	Scénario réussite condition
Se connecter à l'application	Click bouton "login"	Avoir un pseudo et mot de passe correct	Un message s'affiche, "mot de passe ou pseudo incorrect"	La page d'accueil s'affiche selon le type d'utilisateurs ("HR" ou "employee")
		Avoir un compte "actif"	Un message s'affiche "Une erreur est survenue"	

Figure 32 : login

3.9

Page d'accueil - "HR"				
Objectif	Action	Condition	Scénario echec condition	Scénario réussite condition
Afficher la page d'un employé	Click sur bouton "staff"			La page de l'employé s'affiche
	Click sur bouton "Consulter" de l'employé voulu			
Ajouter un nouvel employé	Click sur bouton "Ajouter un nouvel employé"			Le formulaire s'ouvre
	Click sur bouton "valider"	Tous les champs doivent être remplis	Un message d'erreur s'affiche	Un message de confirmation s'affiche
				L'utilisateurs est enregistré dans la db
				Un compte utilisateurs est automatiquement créé pour le nouvel employé
Se déconnecter de l'application	Click sur bouton "logout"			Le formulaire se ferme
	Click bouton "oui" du popup de confirmation	Avoir cliqué sur le bouton "logout"		Le formulaire de confirmation s'ouvre
Rechercher un employé	Click sur bouton "->"	Avoir écrit des lettres dans le input de recherche	Un message d'erreur s'affiche	Le ou les employés trouvés s'affichent
		Au moins un employé contient la chaîne de caractères recherché dans son nom ou prenom	Un message "Aucuns employés trouvés" s'affiche	

Figure 33 : page d'accueil RH

Objectif	Action	Page de profile d'utilisateurs - "HR"		
Objectif	Action	Condition	Scénario echec condition	Scénario réussite condition
Afficher le profil d'un utilisateurs	Click sur bouton "Consulter" d'un utilisateurs sur la page "Staff"			Le profil utilisateur s'affiche
Modifier les informations d'un employé	Click sur bouton "Edit" de la page de profil utilisateur	Avoir modifier au moins un champs	Le formulaire se ferme	Un message "Les informations ont été modifiées avec succès"
Consulter les évaluations	Click sur bouton "Evaluations" du profil	Avoir au moins une évaluation	Un message s'affiche "Aucunes évaluations disponibles"	Les évaluations s'affichent classées par années décroissantes
Supprimer un employé	Click sur bouton "supprimer"	Etre sur le profil de l'employé en question		Retour sur la page d'accueil
Remettre une nouvelle évaluation employé	Click sur bouton "Remettre évaluation [année courante]" (profil employé)	Il n'y a pas encore d'évaluation pour l'année courante	Le bouton n'apparaît pas	Le formulaire s'ouvre
	Click sur bouton "remettre évaluation"	Touts les champs doivent être rempli	Un message d'erreur apparait	Un message de confirmation s'affiche
				L'évaluation s'enregistre dans la db
				Retour sur le profil employé

Figure 34 : Remettre une évaluation

Page de recrutement - "HR"				
Objectif	Action	Condition	Scénario echec condition	Scénario réussite condition
Ajouter une nouvelle offre	Click sur bouton "Ajouter une nouvelle offre"			Le formulaire s'ouvre
	Click sur bouton "Ajouter"	Avoir remplis tous les champs	Un message d'erreur s'affiche	L'offre est enregistrée dans la db Retour sur la page de toutes les offres
Ajouter un nouveau candidat pour une offre	Click sur bouton "Ajouter un nouveau candidat"			Le formulaire s'ouvre
	Click sur bouton "Ajouter"	Avoir remplis tous les champs	Un message d'erreur s'affiche	Le nouveau candidat est enregistré dans la db Retour à la page de l'annonce
Supprimer un candidat	Click sur bouton "Delete" (logo poubelle)			Le formulaire de confirmation s'ouvre
	Click sur bouton "oui"			Retour à la page de l'annonce Suppression du candidat de la db

Figure 35 : Page de recrutement (1)

Ajouter une date d'entretien	Click sur petit calendrier			Le popup calendrier s'ouvre
	Click sur date	Doit être une date dans le futur	La date n'est pas clickable	Fermeture du popup calendrier
				Enregistrement de la date dans la db
Ajouter une décision	Click sur input de décision	Avoir sélectionné une valeurs des radios boutons	Décision reste "pending"	La décision est enregistrée dans la db
Archiver une offre d'emplois	Click sur bouton "archiver"			Le statut de l'offre passe à "archived"
				Retour à la page de toutes les offres
Page de demandes de congés - "HR"				
Objectif	Action	Condition	Scénarion echec condition	Scénario réussite condition
Accepter une demande de congé	Click bouton "Accepter"	La demande de congé a un statut "pending"	La demande ne s'affiche pas	La demande disparaît
				La demande a maintenant un statut "accepted"
Refuser une demande de congé	Click bouton "Refuser"			La demande disparaît
				La demande a maintenant un statut "refused"

Figure 36 : Page de recrutement (2) et page de demande de congé

Page de calendrier - "HR"				
Objectif	Action	Condition	Scénario echec condition	Scénario réussite condition
Afficher calendrier	Click sur bouton "calendrier" de la page d'accueil			Le calendrier s'ouvre
Consulter les informations d'un entretien	Click sur date entourée d'une couleur			Les informations du rendez-vous s'affichent
Modifier les informations d'un entretien	Click sur bouton "Modifier informations"			Le formulaire s'ouvre
	Click sur bouton "Confirmer"	Avoir modifié au moins un champs	Le formulaire se ferme	Les modifications sont enregistrées dans la db
				Le formulaire se ferme
Consulter le calendrier d'un utilisateurs	Click sur bouton "calendrier" du profil d'un utilisateurs	L'utilisateur doit avoir un statut actif		Le calendrier s'ouvre

Figure 37 : Calendrier - RH

4 Réalisation

4.1 Démarrer l'application en local

Prérequis :

- 1) Il faut avoir 'NodeJS' installé sur l'ordinateur.
- 2) Avoir une connexion internet (pour se connecter à la base de données sur SwissCenter)
- 3) Le dossier « projet » du repository installé

Une fois les fichiers importés depuis le github, il faut ouvrir un bash dans le dossier du projet puis taper la commande "npm start". Vous devriez avoir un retour de ce genre dans la console :

```
server run on port 3000
(node:32736) [SEQUELIZE0006] DeprecationWarning: This database engine version is not supported, please update your database server. More information https://github.com/sequelize/sequelize/blob/main/ENGINE.md
(Use `node --trace-deprecation ...` to show where the warning was created)
connexion to db successful
```

Figure 38 : Console au démarrage du serveur

A ce moment le serveur tourne sur le port 3000 de l'ordinateur hôte, une fois démarré, l'application est disponible sur un navigateur à l'url : `http://< adresseIP | | localhost>:3000`. Je conseille l'utilisation de « Microsoft Edge » car parfois Google Chrome bloque les requêtes HTTP. Si vous obtenez une erreur de connexion à la base de données, faites un « ctrl + C » pour stopper le serveur puis refaire un « npm start ».

4.2 Base de données

4.2.1 Structure

Il s'agit d'une base de données « Mariadb » que j'ai généré via MySQLWorkbench à partir du MLD. La liaison avec NodeJS est grâce à l'ORM « Sequelize ». Voici la description de chaque table.




-  ⇒ Primary key
-  ⇒ Contrainte d'unicité
-  ⇒ Foreign key

Table « Employees »

Nom	Type de données	Taille/Ensemble	Non sig...	NULL aut...	ZERO...	Par défaut
idEmployee	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	AUTO_INCREM...
Pseudo	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
Password	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
Lastname	VARCHAR	45	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
Firstname	VARCHAR	45	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
Email	VARCHAR	45	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
Phone	VARCHAR	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
IBAN	VARCHAR	45	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
Street	VARCHAR	45	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
Number	INT	10	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Pas de défaut
NPA	INT	10	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Pas de défaut
City	VARCHAR	45	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
activ	TINYINT	3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Pas de défaut

Figure 39 : table employees

Table « Contracts »





#	Nom	Type de données	Taille/Ensemble	Non signé	NULL autorisé	ZEROFILL	Par défaut
 1	idContract	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
 2	FkEmployee	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
 3	email	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
 4	phone	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
5	Type	VARCHAR	3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
6	Service	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
7	Job	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
8	StartDate	DATE		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
9	EndDate	DATE		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
10	Rate	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
11	salary	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut

Figure 40 : Table Contracts

Table « LeaveRequests »



#	Nom	Type de données	Taille/Ensemble	Non signé	NULL autorisé	ZEROFILL	Par défaut
 1	idLeaveRequest	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
 2	FkContract	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
3	StartDate	DATE		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
4	EndDate	DATE		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
5	Reason	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
6	Payed	TINYINT	3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
7	Status	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut

Figure 41 : Table LeaveRequests

Table « Evaluations »



#	Nom	Type de données	Taille/Ensemble	Non signé	NULL autorisé	ZEROFILL	Par défaut
 1	idEvaluation	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
 2	FkContract	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
3	EvaluationYear	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
4	PerformanceNote	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
5	Positiv	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
6	Negativ	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
7	createdAt	DATE		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut

Figure 42 : Table Evaluations

Table « Historics »


#	Nom	Type de données	Taille/Ensemble	Non signé	NULL autorisé	ZEROFILL	Par défaut
 1	idHistoric	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
 2	FkEmployee	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
3	Service	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
4	StartDate	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
5	EndDate	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
6	Rate	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
7	Job	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
8	Salary	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
9	Email	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
10	Phone	VARCHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut

Figure 43 : Table Historics

Table « JobOffers »

#	Nom	Type de données	Taille/Ensemble	Non signé	NULL autorisé	ZEROFILL	Par défaut
1	idJobOffer	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
2	Name	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
3	Place	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
4	Status	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
5	Rate	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut

Figure 44 : Table JobOffers

Table « Candidacies »

#	Nom	Type de données	Taille/Ensemble	Non signé	NULL autorisé	ZEROFILL	Par défaut
1	idCandidacy	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
2	FkJobOffer	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
3	Lastname	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
4	Firstname	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
5	Mail	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
6	Phone	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
7	PostulationDate	DATE		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Pas de défaut
8	InterviewDate	DATE		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
9	Decision	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'pending'

Figure 45 : Table Candidacies

4.2.2 Mise en ligne de la base de données

Pour que toutes les parties prenantes du projet puissent tester la base de données avec un minimum d'installation, j'ai décidé de la mettre sur swisscenter. Pour cela j'ai simplement importé le fichier SQL généré par MySQLWorkbench.

J'ai également décidé de créer 2 bases de données, la première est pour le développement afin de pouvoir tester sans limites ni craintes. La deuxième est celle que je remettrai lors du rendu du projet, elle contiendra un jet de données qui permettra de tester toutes les fonctionnalités de l'application. C'était du moins mon idée de départ mais devant le manque de temps et de peur qu'il y ait une erreur je vous remettrai la version de développement.

4.2.3 Droits d'accès

Pour des questions de sécurité, j'ai créé un second utilisateur qui n'a que les droits lecture/écriture. C'est donc celui-ci qui sera utilisé par défaut dans l'application afin de ne pas utiliser 'root' à chaque requête.

Nom d'utilisateur	Accréditation
tpi_rh_db2	DB admin
tpi_rh_user2	Lecture / écriture

Figure 46 : Utilisateurs de la base de données

4.2.4 Connexion

Voici le code Sequelize de connexion en local et la version finale, qui se trouve sur le serveur de Swisscenter :

```
const { Sequelize } = require('sequelize')
const sequelize = new Sequelize(
  'rh',
  'root',
  '', {
    host: 'localhost',
    dialect: 'mariadb',
    dialectOptions: {
      timeZone: 'Etc/GMT-2'
    },
    logging: false
  }
)
module.exports = { sequelize }
```

Figure 47 : connexion - version initiale 1 (locale)

```
const { Sequelize } = require('sequelize')
const sequelize = new Sequelize(
  'tpi_rh_db',
  'tpi_rh_user',
  'tpiRHuser2024@', {
    host: 'web24.swisscenter.com',
    dialect: 'mariadb',
    port: 3306,
    dialectOptions: {
      timeZone: 'Etc/GMT-2'
    },
    logging: false
  }
)
module.exports = { sequelize }
```

Figure 48 : connexion - version finale 2 (online)

4.2.5 Données

J'ai inséré manuellement les données, qui ont été générées avec l'aide de Chat GPT, dont je fichier JSON se trouve en annexe. Il n'y a que les données de la table « JobOffers » que j'ai ajoutées manuellement depuis PhpMyAdmin car pour le moment le projet ne comprend pas une gestion complète des offres d'emploi.

4.3 Initialisation du projet

Une fois la base de données créée en local, j'ai initialisé NodeJS avec la commande « npm init ». Puis j'ai modifié le fichier « Package.json » comme ci-dessous afin d'insérer les dépendances minimums nécessaire au développement de l'application (bien que j'en ai installées d'autres modules plus tard). Puis en faisant la commande « npm install », tous les fichiers sont téléchargés dans le dossier « node_modules ».

```
{
  "devDependencies": {
    "jquery": "^3.7.1"
  },
  "dependencies": {
    "bcrypt": "^5.1.1",
    "body-parser": "^1.20.2",
    "cors": "^2.8.5",
    "dotenv": "^16.4.5",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.2",
    "mariadb": "^3.2.3",
    "path": "^0.12.7",
    "sequelize": "^6.37.1",
    "sequelize-auto": "^0.8.8"
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node app.js"
  }
}
```

Figure 47 : Package.json

4.4 Génération des modèles

J'ai utilisé l'utilitaire « sequelize-auto » afin de générer automatiquement tous les modèles qui correspondent aux tables de la base de données. J'ai pu donc gagner un certain temps et éviter des erreurs humaines. Voici la commande puis son résultat :

```
PS E:\TPI_RH\RH_projet> node node_modules/sequelize-auto/bin/sequelize-auto -o "./models" -d rh -u root -x --config ./sequelize-auto-config.json
Password:
```

Figure 48 : Sequelize auto

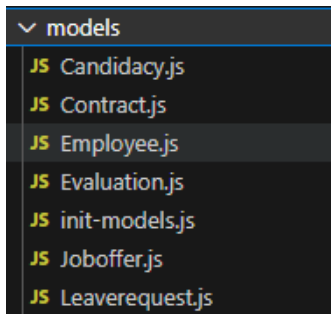


Figure 49 : dossier models

Le fichier « init-models.js » initie les modèles. J'y ai la déclaration des associations de tables (qui sont toutes reliées via une relation 1 à n).

Les autres fichiers sont donc les modèles utiles à Sequelize pour se synchroniser avec la base de données. J'ai du légèrement les modifier,

4.5 Structure du projet

Voici à quoi ressemble l'arborescence du projet :

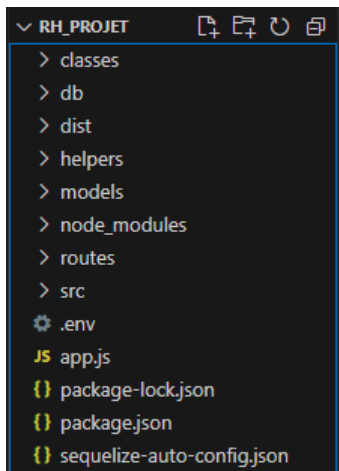


Figure 50 : Dossier de projet

« app.js » : Il s'agit du fichier racine du server Express. Il gère toutes les requêtes http du type « subscribe » « login » etc. Si aucune route n'est

« package.json » et « package-lock.json » : sont des fichiers qui montrent les dépendances du projet, ce dernier montre plus de détails.

« node_modules » : contient toutes les librairies nécessaires au bon fonctionnement de l'application.

« .env » : fichier qui contient les variables d'environnement "ACCESS_TOKEN_SECRET" & "REFRESH_TOKEN_SECRET", qui servent de clefs d'identifications des utilisateurs.

Dossier « db »

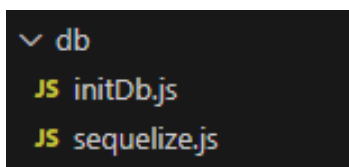
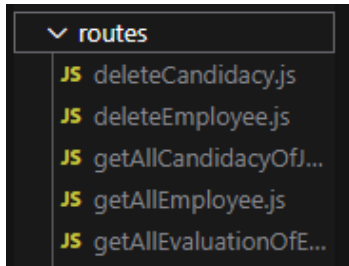


Figure 51 : dossier db

« sequelize.js » : Présente les informations de connexion à la base de données comme le nom d'utilisateur, mot de passe etc.

« initDb » : S'authentifie auprès de la base de données grâce aux informations présentes dans « sequelize.js ».

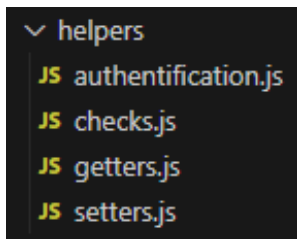
Dossier « routes »



Ce dossier contient les fichiers serveurs qui réceptionne les requêtes http, effectuent le traitement adéquat des données puis renvoie une réponse au client. Les fichiers sont nommés selon l'action du endpoint (insert – get – update – delete).

Figure 52 : Dossier routes

Dossier « helpers »



« authentications » : Ce fichier fait la gestion des TokenJWT des requêtes, afin d'autoriser ou non l'accès aux endpoints.

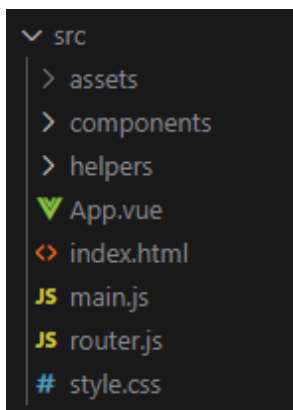
« checks » : Ce fichier contient des fonctions qui checkent la validité des données avant de les insérer.

« getters » : Ce fichier contient quelques fonctions qui facilitent la récupération de données de la base de données.

« setters » : Contient des fonctions qui servent à fixer des valeurs.

Figure 53 : Dossier helpers

Dossier « SRC »



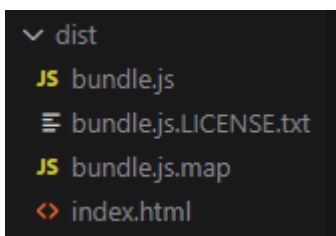
Il s'agit des fichiers de code frontend sur lesquels j'ai travaillé et qui ne servent qu'au développement. Ce dossier a été généré automatiquement par NPM. « assets » dans lequel devrait se trouver normalement les fichiers CSS et images. Le dossier « components » contient tous les composants VueJS du projet. Le dossier helpers ne contient qu'une seule fonction très utilisée qui vérifie que l'utilisateur est connecté et renvoie toutes les données.

« App.vue » : Il s'agit du composant racine de Vue pour notre application. On y utilise notamment le routeur de Vue pour afficher le bon composant

« main.js » : C'est le fichier qui sert de point d'entrée de l'application. Pour créer son bundle, Webpack part de ce fichier.

Figure 54 : Dossier src

Dossier « dist »



Il s'agit du dossier finalement renvoyé au client, c'est le bundle de WebPack. Tous le code javascript est minifié et optimisé afin d'être plus léger.

Figure 55 : dossier dist

4.6 Logique des endpoints

Dans cette section je décris chaque endpoint et la logique qui lui est associée.

Login

Ce endpoint va checker si les informations reçues correspondent bien à un pseudo et mot de passe valides. Si c'est le cas, le serveur renvoie toutes les données personnelles, contractuelles ainsi que les évaluations, demandes de congés de l'utilisateur ainsi qu'un jeton d'accès.

Insert employee

Ce endpoint insère les données d'un nouvel employé dans les tables « employees » et « contracts ». Il crée dynamiquement un pseudo pour l'utilisateur avec son prenom et son ID de la table « employees ». Si un autre employé a déjà la même email, téléphone ou IBAN le serveur n'insère rien et renvoie une erreur.

Insert leave request

Ce endpoint récupère les informations d'une demande de congé, vérifie les informations avant de les insérer ou de renvoyer une erreur. Pour le moment le endpoint vérifie qu'il n'y ait pas d'autre demande de congé avec une date de début identique.

Insert Candidacies

Ce endpoint vérifie l'existence de l'offre d'emploi pour laquelle la candidature se présente, puis check que la candidature n'existe pas déjà pour cette offre (par son email et numéro de téléphone).

Insert évaluation

Ce endpoint vérifie s'il existe déjà une évaluation existante pour l'année de la requête et bloque l'insertion si c'est le cas. L'idée est de ne pas avoir deux évaluations pour une même année.

Get all job offers

Ce endpoint va chercher toutes les offres d'emplois qui ont un statut « pending » (en attente). S'il n'y en a aucunes, il renvoie un tableau vide

Get all candidacies of a job offer

Ce endpoint retourne toutes les candidatures associées à une offre d'emploi existante.

Get all employees

Ce endpoint renvoie l'ID, le nom et prénom de tous les employés actifs. Je voulais utiliser ce endpoint pour la page de recherche d'employés des RH. Afin d'économiser les requêtes au serveur, au chargement de la page le programme charge ces données afin de pouvoir filtrer les employés sans devoir communiquer avec la base de données.

Get all evaluations of an employee

Ce endpoint retourne toutes les données qui concernent les évaluations d'un employé existant.

Get all interviews dates

Ce endpoint retourne toutes les candidatures qui ont une date d'interview fixée dans le futur.

Update candidacy

Ce endpoint remplace les anciennes données par les nouvelles. La vérification que au moins un champ a été modifié est faite dans le frontend avant d'envoyer la requête.

Update leave request

Ce endpoint sert aux RH afin de refuser ou d'accepter une demande de congé. Actuellement il check si la demande existe bien et si le statut est valide (« accepted », « refused » ou « pending »).

Update employee

Ce endpoint enregistre les nouvelles informations d'un employé existant.

Update Contract

Ce endpoint enregistre les nouvelles informations du contrat d'un employé. A chaque modification d'un record, un nouvel enregistrement est fait dans la table « historique » qui contient toutes les données de l'ancien contrat.

Delete candidacy

Ce endpoint supprime une candidature existante d'une offre d'emploi existante.

Delete Employee

Ce endpoint passe le champs « activ » de la table « employee » à false puis génère un backup de la table « contracts » en fixant la date du jour comme date de fin si rien n'est spécifié. De plus il va supprimer les demandes de congés et évaluations liées à cet employé.

4.7 Stratégie de tests

J'ai utilisé l'outil gratuit « Insomnia » afin de tester les endpoints des requêtes. Vous trouverez la collection de requêtes (compatible avec Postman) dans les annexes ainsi qu'une série de données en json afin de pouvoir tester tous les endpoints.

Si vous souhaitez tester le login sans devoir créer un nouvel employé vous pouvez utiliser les identifiants du compte ci-dessous. Il y aura des demandes de congés ainsi que des évaluations qui lui seront associées.

```
"pseudo": "Emily57",  
"password": "vt04cKuGIwhJ"
```

Figure 56 : Identifiants de login

4.7.1 Tests Backend

Login

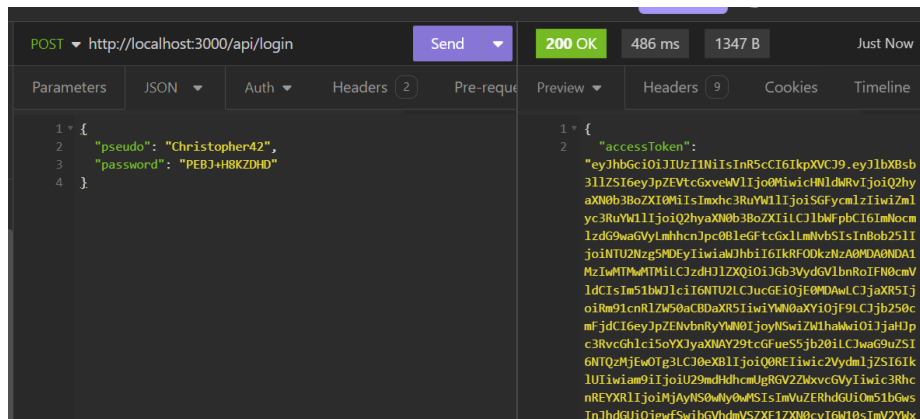


Figure 57 : test login (1)

Avec des identifiant incorrects :

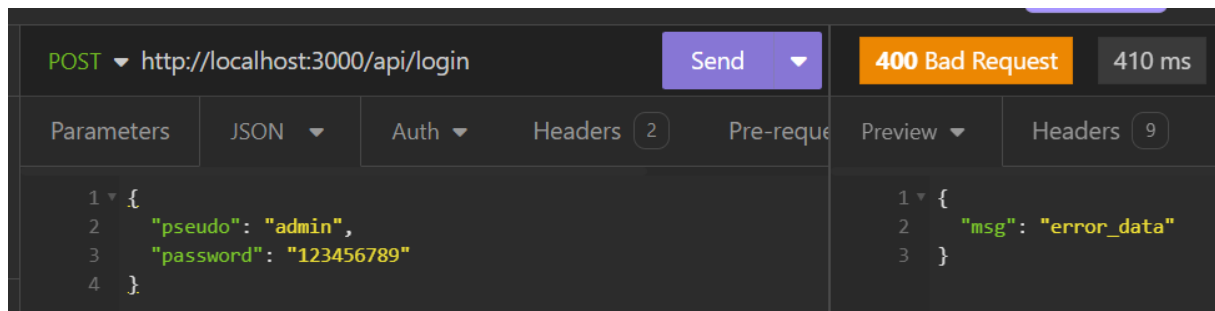


Figure 58 : test login (2)

Insert leave request

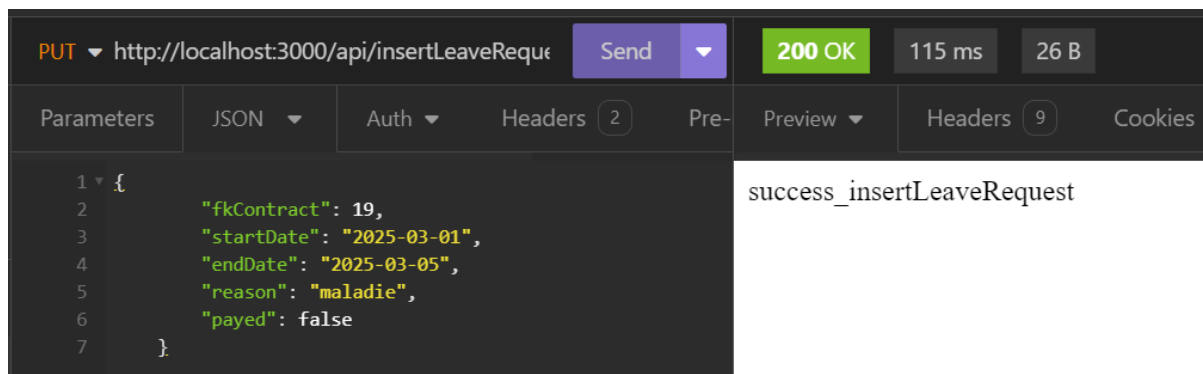


Figure 59 : test insert leaveRequest (1)

Si on tente de réinsérer une deuxième fois la même « startDate » :

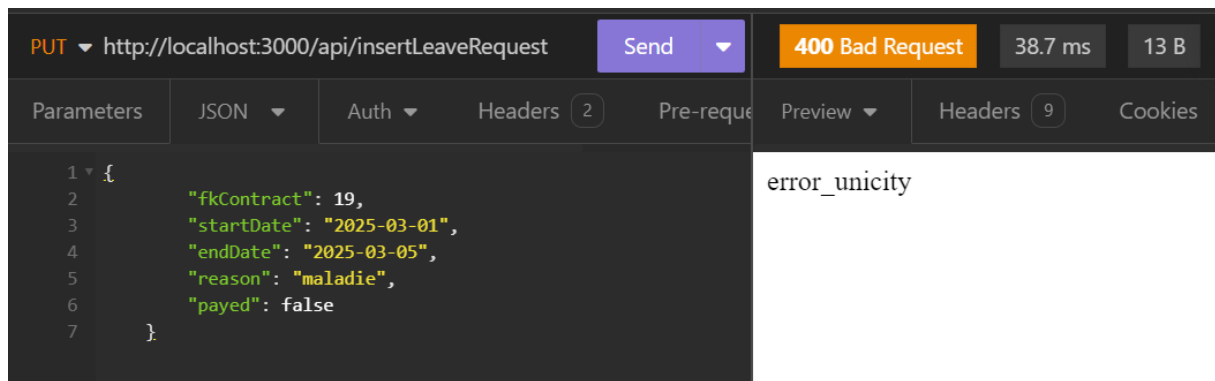


Figure 60 : test insert leaveRequest (2)

Update leave request

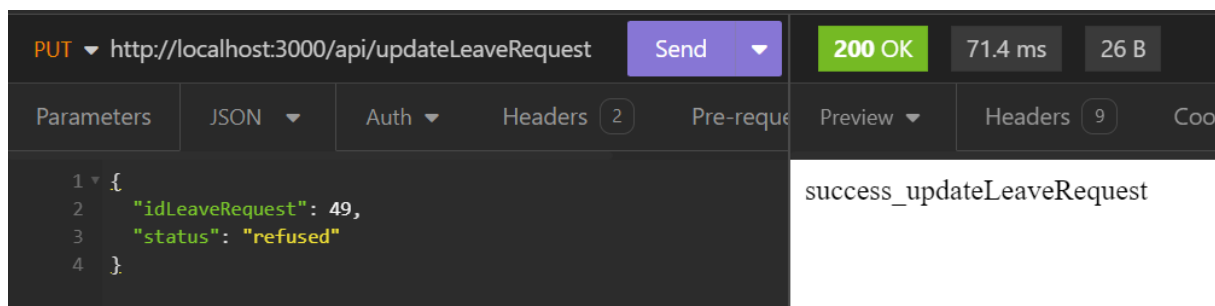


Figure 61 : test update LeaveRequest (1)

Si on tente de modifier une demande de congé qui n'existe pas :

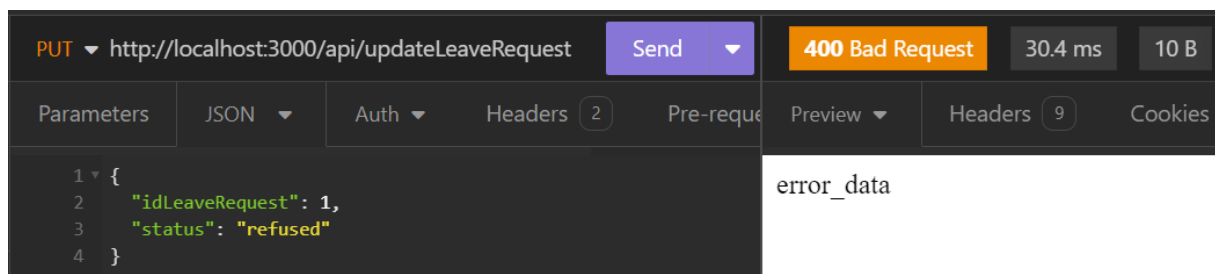


Figure 62 : test update LeaveRequest (2)

Si on tente de modifier une demande de congé avec un autre statut que « accepted », « refused » ou « pending » :

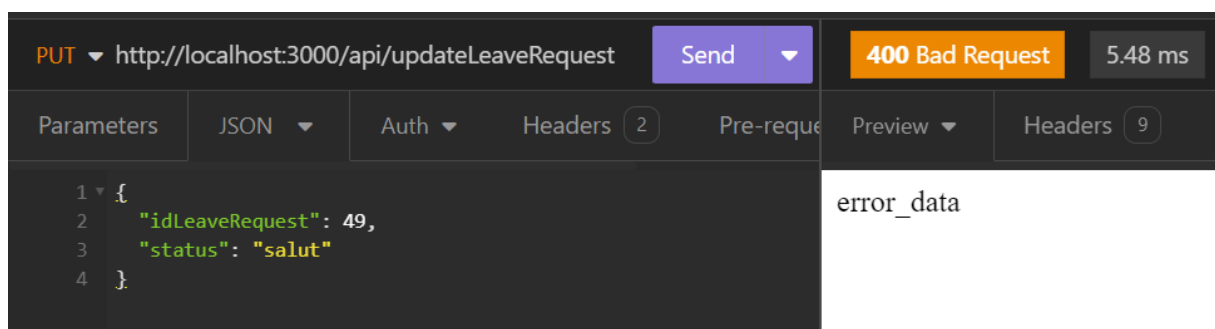


Figure 63 : test update LeaveRequest (3)

Insert candidacy

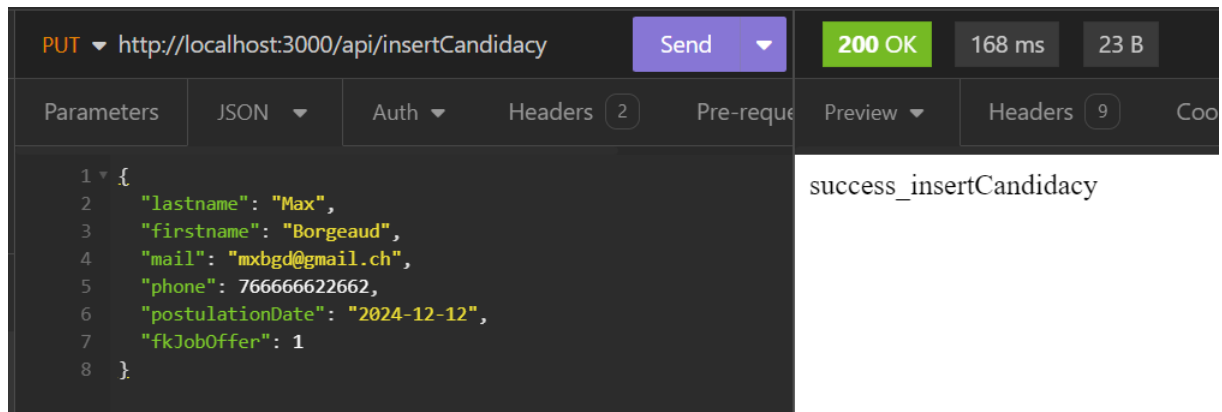


Figure 64 : test insertCandidacy (1)

Si je tente d'insérer les mêmes données pour une offre d'emploi différente (fkJobOffer) :

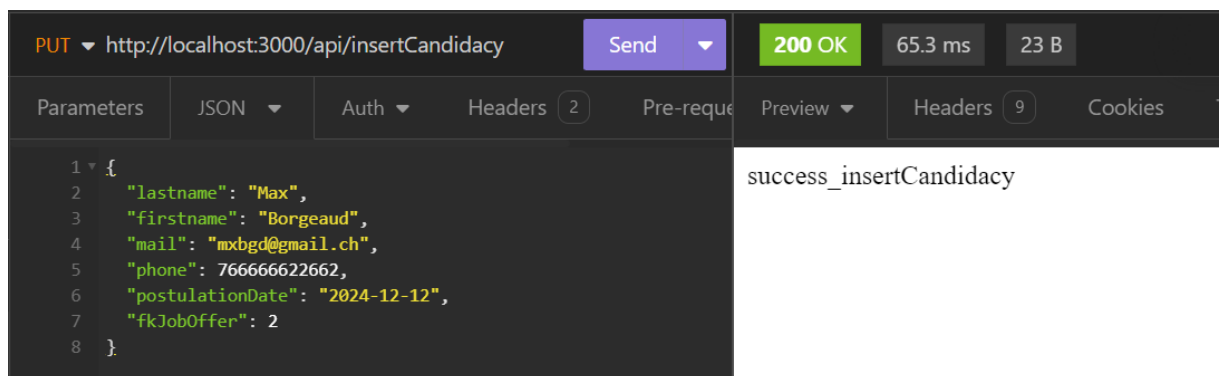


Figure 65 : test insertCandidacy (2)

Si je tente d'insérer les mêmes données pour la même offre d'emploi que précédemment :

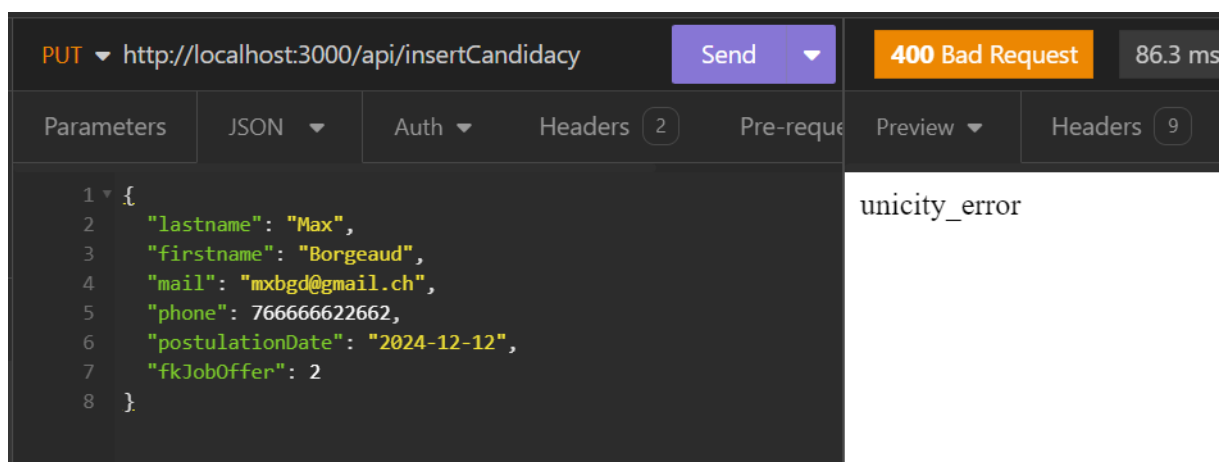


Figure 66 : test insertCandidacy (3)

Maintenant lorsque je tente d'insérer des données pour une offre d'emploi qui est inexistante ou qui a un statut différent de « en attente » :

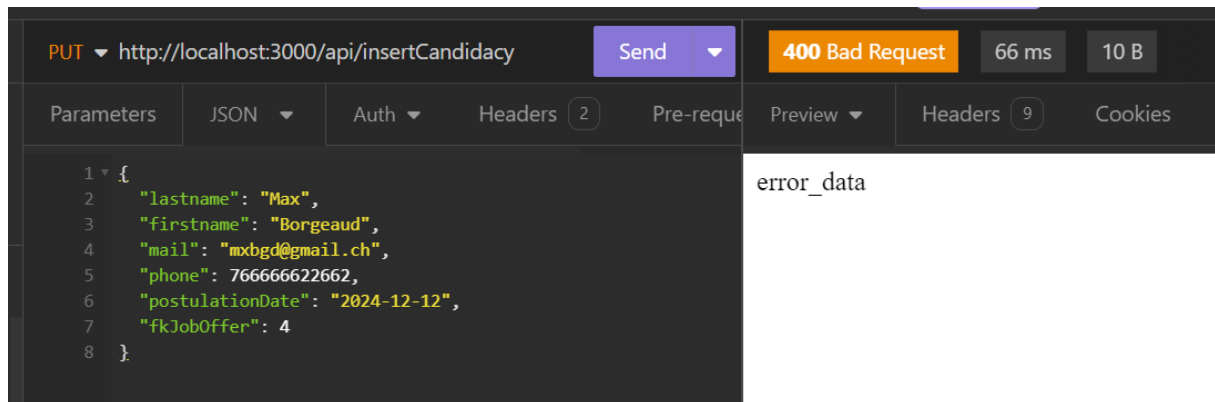


Figure 67 : test insertCandidacy (4)

Get all interviews

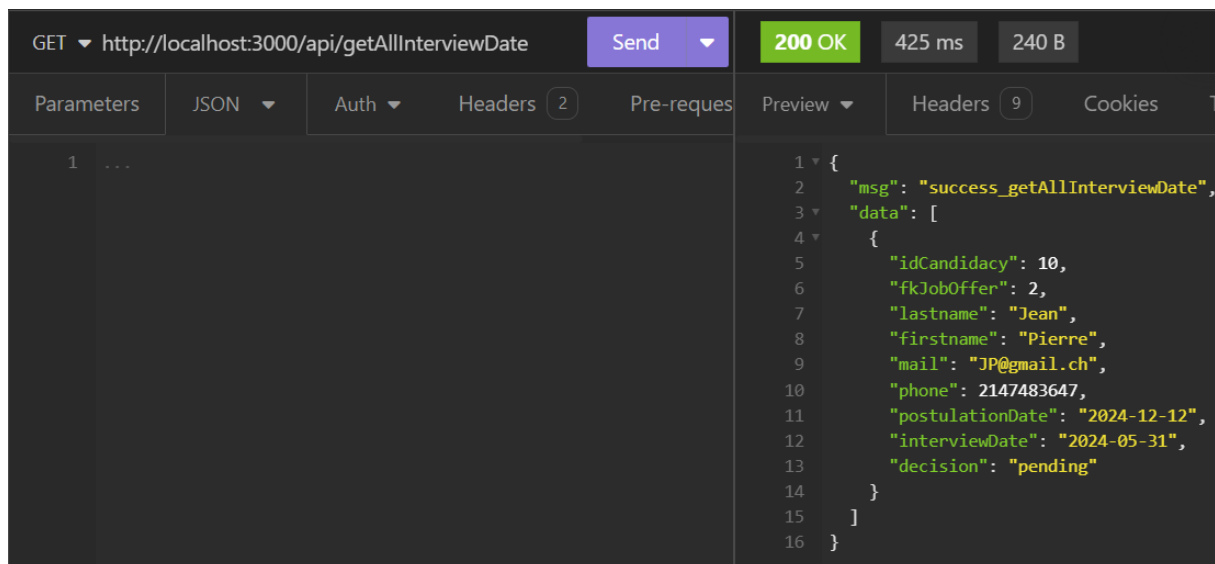


Figure 68 : test GetAllInterviews (1)

Même requête mais sans données à renvoyer :

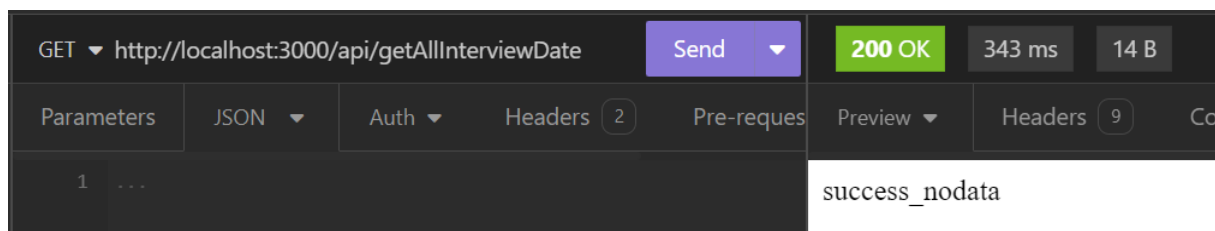


Figure 69 : test GetAllInterviews (2)

Get all candidacies of job offer

GET `http://localhost:3000/api/getAllCandidacyOfJobOffer` **Send** **200 OK** 153 ms 239 B Just Now

Parameters JSON Auth Headers (2) Pre-request Script Preview Headers (9) Cookies Timeline

```
1 {
2   "idJobOffer": 2
3 }
```

```
1 {
2   "msg": "success_getAllCandidaciesOfJobOffe",
3   "data": [
4     {
5       "idCandidacy": 10,
6       "fkJobOffer": 2,
7       "lastname": "Jean",
8       "firstname": "Pierre",
9       "mail": "JP@gmail.ch",
10      "phone": 2147483647,
11      "postulationDate": "2024-12-12",
12      "interviewDate": null,
13      "decision": "pending"
14    }
15  ]
16 }
```

Figure 70 : test GetAllCandidacies (1)

S'il n'y a aucune candidature :

GET `http://localhost:3000/api/getAllCandidacyOfJobOffer` **Send** **200 OK** 131 ms 14 B

Parameters JSON Auth Headers (2) Pre-request Script Preview Headers (9) Cookies

```
1 {
2   "idJobOffer": 1
3 }
```

success_noData

Figure 71 : test GetAllInterviews (2)

Si l'offre d'emploi n'existe pas :

GET `http://localhost:3000/api/getAllCandidacyOfJobOffer` **Send** **400 Bad Request** 109 ms

Parameters JSON Auth Headers (2) Pre-request Script Preview Headers (9) Cookies

```
1 {
2   "idJobOffer": 211
3 }
```

error_data

Figure 72 : test GetAllInterviews (3)

Update candidacies

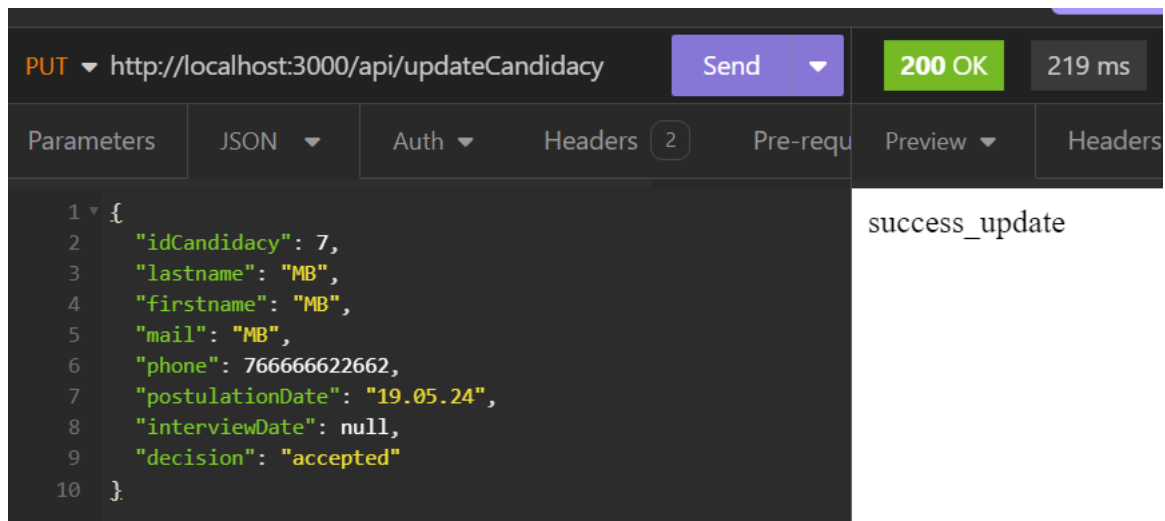


Figure 73: test updateCandidacy (1)

Lorsque l'on tente de modifier la décision avec autre chose que « pending », « accepted » ou « refused » :

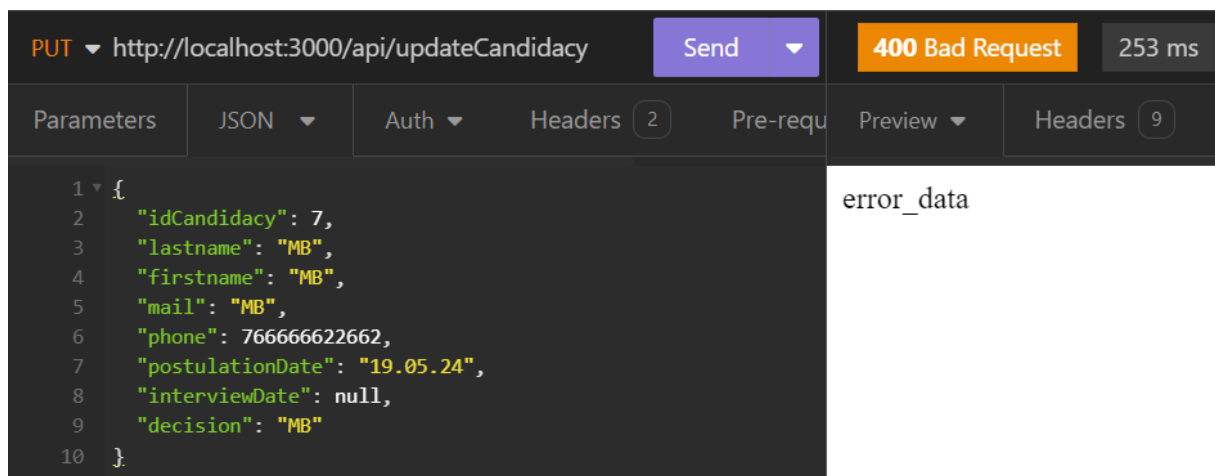


Figure 74: test updateCandidacy (2)

Lorsque l'on tente de modifier une candidature inexistante :

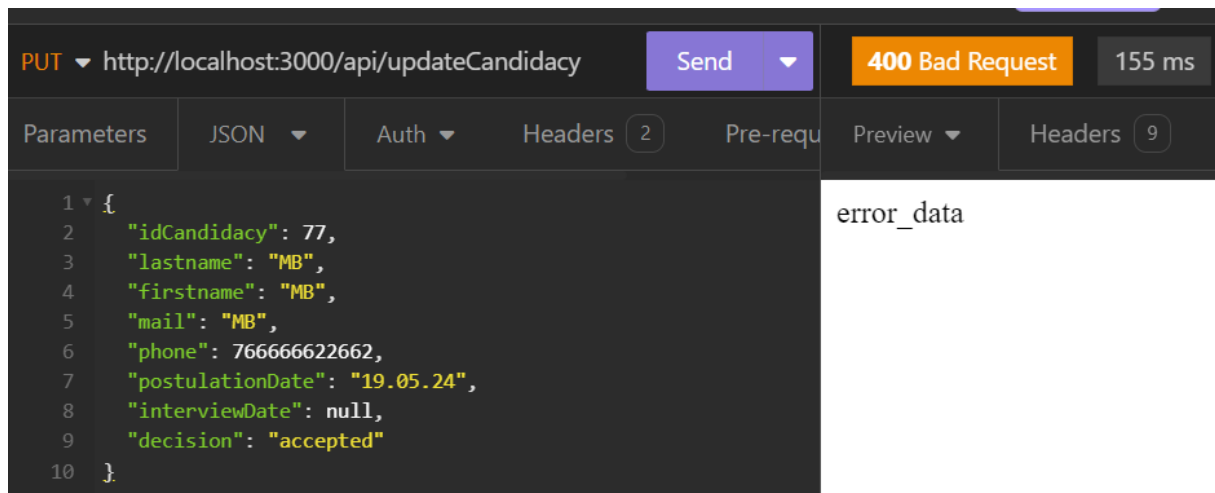


Figure 75: test updateCandidacy (3)

Delete candidacy

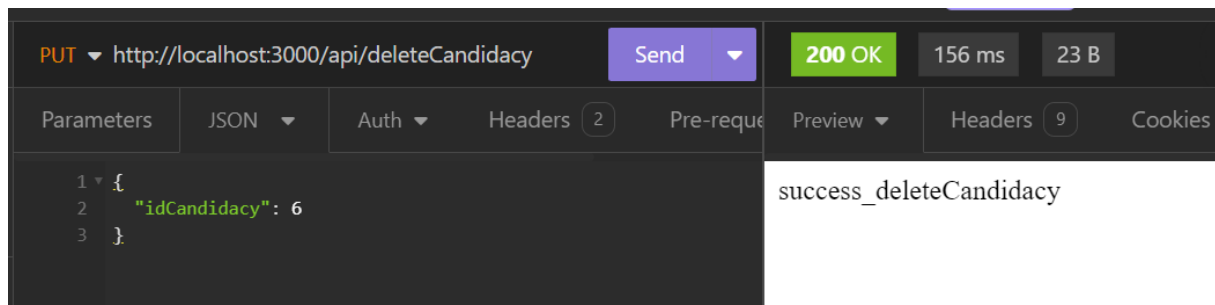


Figure 76 : test deleteCandidacy (1)

Si je tente de supprimer une candidature qui n'existe pas :

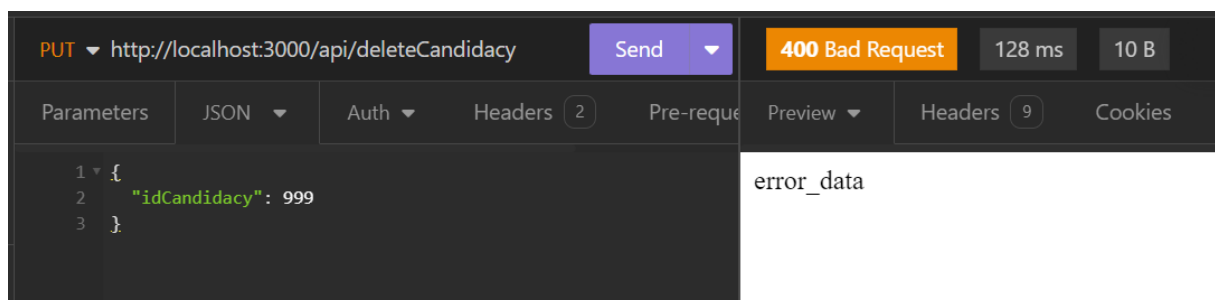


Figure 77: test deleteCandidacy (2)

Insert employee

The screenshot shows a REST client interface with a PUT request to `http://localhost:3000/api/insertEmployee`. The request body is a JSON object containing employee details. The response is a 200 OK status with a success message and user information.

```
PUT http://localhost:3000/api/insertEmployee
Send 200 OK 1.24 s 92 B

Parameters JSON Auth Headers (2) Pre-reqs Preview Headers (9) Cookies

1 {
2   "lastname": "Harris",
3   "firstname": "Christopher",
4   "email": "christopher.harris@example.com",
5   "job": "Software Developer",
6   "phone": "556789012",
7   "iban": "DE89370400440532013013",
8   "street": "Fourteenth Street",
9   "number": "556",
10  "npa": "14000",
11  "city": "Fourteenth City",
12  "activ": true,
13  "type": "CDD",
14  "email_prof": "christopher.harris@company.com",
15  "phone_prof": "543210987",
16  "startDate": "2025-07-01",
17  "endDate": "2026-07-01",
18  "service": "IT",
19  "rate": 80,
20  "salary": 59000
21 }
```

```
1 {
2   "msg": "success_insertEmployee",
3   "user": {
4     "pseudo": "Christopher40",
5     "password": "uzLt5eBd8w85"
6   }
7 }
```

Figure 78 : test insertEmployee (1)

Si on tente de réinsérer une deuxième fois les mêmes données :

The screenshot shows a REST client interface with a PUT request to `http://localhost:3000/api/insertEmployee`. The request body is the same JSON object as in Figure 78. The response is a 400 Bad Request status with a `unicity_error` message.

```
PUT http://localhost:3000/api/insertEmployee
Send 400 Bad Request 191 ms

Parameters JSON Auth Headers (2) Pre-reqs Preview Headers (9) Cookies

1 {
2   "lastname": "Harris",
3   "firstname": "Christopher",
4   "email": "christopher.harris@example.com",
5   "job": "Software Developer",
6   "phone": "556789012",
7   "iban": "DE89370400440532013013",
8   "street": "Fourteenth Street",
9   "number": "556",
10  "npa": "14000",
11  "city": "Fourteenth City",
12  "activ": true,
13  "type": "CDD",
14  "email_prof": "christopher.harris@company.com",
15  "phone_prof": "543210987",
16  "startDate": "2025-07-01",
17  "endDate": "2026-07-01",
18  "service": "IT",
19  "rate": 80,
20  "salary": 59000
21 }
```

```
unicity_error
```

Figure 79: test insertEmployee (2)

Delete Employee

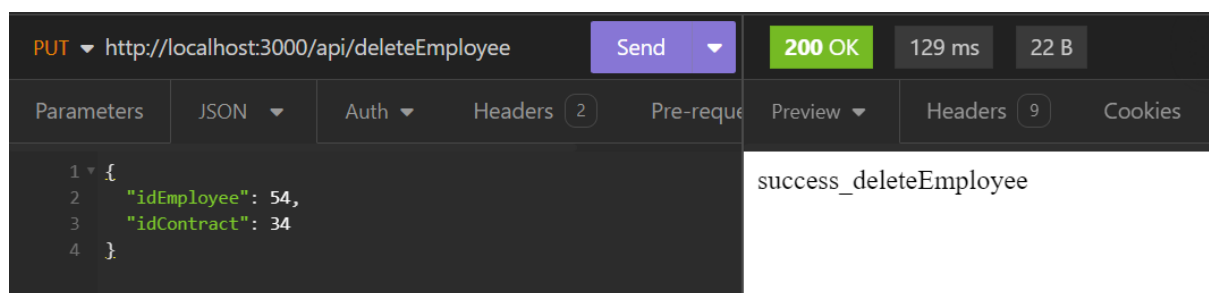


Figure 80 : test deleteEmployee (1)

Si on tente de supprimer un employé qui n'existe pas :

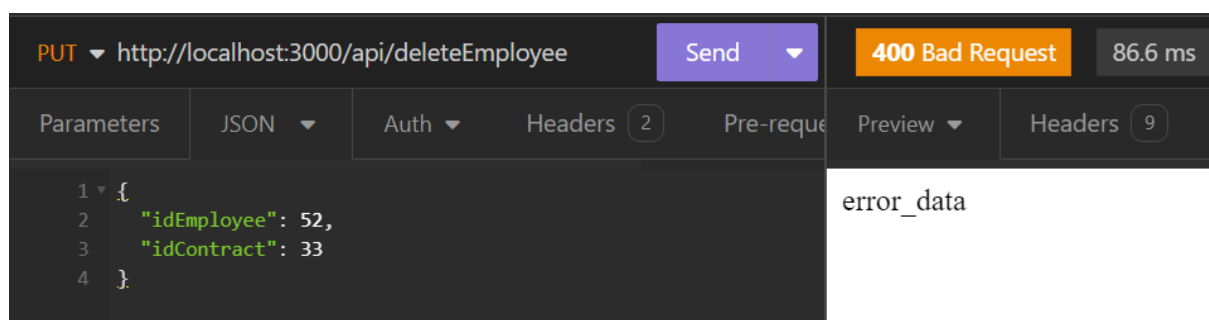


Figure 81 : test deleteEmployee (2)

5 Conclusion

5.1 Erreurs/imprécisions restantes

Endpoint de « deleteEmployee »

Il ne fonctionne que partiellement, tout se fait sans soucis excepté l'insertion dans la table « historics » qui ne se réalise pas.

Fonctionnalités frontend

La première chose à signaler ici est la faite que beaucoup de fonctionnalités ne sont pas présentes dans l'application frontend. Je me suis donc concentré sur la partie des demandes de congés avec le temps que j'avais à ma disposition.

Check du endpoint « insertLeaveRequest »

S'il y a une plage de date, il Faudrait checker la validité pour chacune et tout insérer si tout est valide et ne rien insérer si une date est invalide.

Modification d'une évaluation existante

Pour rajouter cette petite fonctionnalité, il faudrait rajouter un nouveau endpoint et de plus je rajouterais les champs « updatedAt » et « updatedBy » dans la table « evaluations » qui contiendraient la date de modification ainsi que l'id de l'employé RH. Il faudrait également faire que les employés RH ne puisse pas modifier leurs propres évaluations.

Mise à jour des données du calendrier

Lorsqu'un utilisateur envoie une demande de congé valide qui s'inscrit dans la base de données, il faudrait ajouter la nouvelle demande de congé dans les données de l'utilisateurs

Autres

- ⇒ Erreur de dépréciation au démarrage du serveur
- ⇒ Endpoint « updateEmployee » non-terminé et instable
- ⇒ Commentaires du code frontend

5.2 Améliorations possibles

- ⇒ Il serait intéressant d'ajouter la gestion des salaires, avec par exemple l'établissement de grilles salariales, le calcul des charges sociales, calculs de bonus de performances/d'ancienneté.
- ⇒ Il serait également possible de rajouter une plateforme où les offres d'emplois sont disponibles sans login et avec des formulaires de postulation. Dans ce cas il serait utile de renommer la table « candidacy » en « candidate » et créer une table intermédiaire entre elle et « joboffers ». Ainsi un même candidat pourra postuler à plusieurs sans pour autant créer de redondance dans la base de données
- ⇒ Centraliser tous les « query » Sequelize dans un seul fichier pourrait éventuellement rendre le code plus propre.
- ⇒ Actuellement, le système génère automatiquement un mot de passe aléatoire, que l'employé RH transmettrait à l'employé concerné. Il faut absolument mettre en place un système qui force l'utilisateur à changer de mot de passe à la première connexion et éventuellement par la suite une fois par années pour la sécurité

5.3 Conclusion personnelle

En tant que développeur j'ai pris beaucoup de plaisir à réaliser cette application bien qu'il y ait eu quelques frustrations. Premièrement, j'ai pris soin de ne pas bâcler la phase d'analyse comme j'ai pu le faire par le passé, ce qui m'a permis de ne pas avoir trop de régression dans mon avancement et donc ne pas perdre un temps précieux. J'ai rencontré quelques difficultés techniques, notamment l'implémentation de VUE avec Webpack mais j'ai surmonté ce genre de problèmes à l'aide de ChatGPT notamment. Je suis cependant fier du travail que j'ai accompli, ayant acquis les connaissances NodeJS et VUE en autodidacte.

6 Annexes

6.1 Résumé du projet

Comme situation de départ, je dois créer une petite application qui doit permettre de gérer les points principaux qui composent les ressources humaines. Principalement la gestion des employés, demandes de congés, gestion des performances et des recrutements avec évidemment de la sécurité sur l'authentification des utilisateurs.

J'ai commencé la mise en œuvre par l'implémentation du backend. J'ai créé le serveur express, puis configuré les endpoints qui répondent aux requêtes http correspondantes. Ces requêtes font un CRUD complet sur les différents points évoqués au paragraphe précédent. On y insert, lit, modifie et supprime des éléments, ou non selon certains critères. Une fois que les endpoints ont été configuré, j'ai commencé le frontend avec VUE. C'est à cette étape que j'ai compris que finir toutes les interfaces serait impossible et donc nous avons convenu avec mon chef de projet que me concentrer sur la partie frontend des demandes de congés serait suffisant.

Comme résultat je présente une API NodeJS qui répond presque à toutes les demandes de l'applications (voir section « Erreurs/imprécisions restantes »). Les interfaces répondent à une partie des fonctionnalités demandées mais le temps à disposition ne permettait pas de coder l'ensemble du frontend. Les règles des checks furent une réelle difficulté car n'ayant aucunes compétences métiers je n'avais parfois pas trop idée quelles sont les bonnes pratiques.

6.2 Liste documents fournis

Employees.json → Jet de données à insérer

requests.GamingRoot → Requêtes http à importer dans Insomnia ou Postman

db.sql → Fichier de base de données

/projet → Dossier qui contient tout le code final de l'application

Journal_de_travail.pdf → Résumé des tâches pour chaque jour

6.3 Planification effective

J'ai constaté en refaisant les calculs d'heures du journal de travail que certaines estimations sont approximatives et il manque donc certaines heures, bien que le travail ait été réalisé en 90 heures. Voici la répartition de temps réalisées dans projet :

⇒ Analyse	:	11%
⇒ Conception	:	30%
⇒ Réalisation	:	46%
⇒ Documentation	:	13%

J'ai passé plus de temps de prévu dans la conception car étant un de mes point faible, j'ai bien pris le temps de réfléchir au projet, ce qui n'empêcha pas de remodifier par la suite.

6.4 Glossaire

Bibliothèque

Un ensemble de fonctions regroupées pour être utilisées par des programmes informatiques.

Bundle

Dans le contexte de Webpack, un bundle est un dossier de code du projet compilé et minifié. Ce sont les fichiers qui seront finalement renvoyé aux clients en production.

Endpoint

Un point d'accès à une API permettant l'interaction avec des services web. C'est là que se fera le vrai traitement de la requête.

Framework

Il s'agit d'un "environnement de développement ». Plus concrètement il s'agit d'outils, bibliothèques etc

ORM (Object Relational Mapping)

Il s'agit du "pont" qui relie le programme et la base de données, « Sequelize » dans ce projet
query

Dans le contexte de Sequelize, un query est une requête SQL envoyée à la base de données.

Record

Une ligne dans une table de base de données représentant un ensemble unique de données.

Repository

Un stockage centralisé où le code source et les fichiers de projet sont conservés.

Sprint

Dans le contexte de la méthode Agile, un sprint est une période délimitée pendant laquelle une équipe spécifique travaille à compléter une ou plusieurs tâches.

Table

Il s'agit d'une « entité » physique ou non-physique représentée dans une base de données.

6.5 Sources

Sequelize auto :

<https://ratneshpandey.hashnode.dev/auto-generate-models-for-sequelize-using-sequelize-auto>

Associations sequelize :

<https://sequelize.org/docs/v6/core-concepts/assocs/>

Recherches Javascripts diverses :

<https://www.syncfusion.com/blogs/post/js-commonjs-vs-es-modules>

Transactions Sequelize :

<https://sequelize.org/docs/v6/other-topics/transactions/>

Recherches/debuggage divers, notamment VUE

<https://chat.openai.com>

6.6 Archives - base de données

J'ai fait quelques modifications entre la dernière version de la base de données présentée plus haut et l'ancienne version présentée ici. J'ai déplacé les champs « Job » et « Rate » de la table « Employees » vers la table « Contracts ». J'y ai également ajouté la table « historics ». Lorsqu'un utilisateurs modifiera une donnée de la table « contracts », il y aura une sauvegarde automatique des anciennes données dans la table « historics ». J'ai également mis la relation de la table « Historics » avec « Employees » plutôt qu'avec « contracts » sinon il y avait des erreurs SQL lors de la suppression de la table contract.

MCD

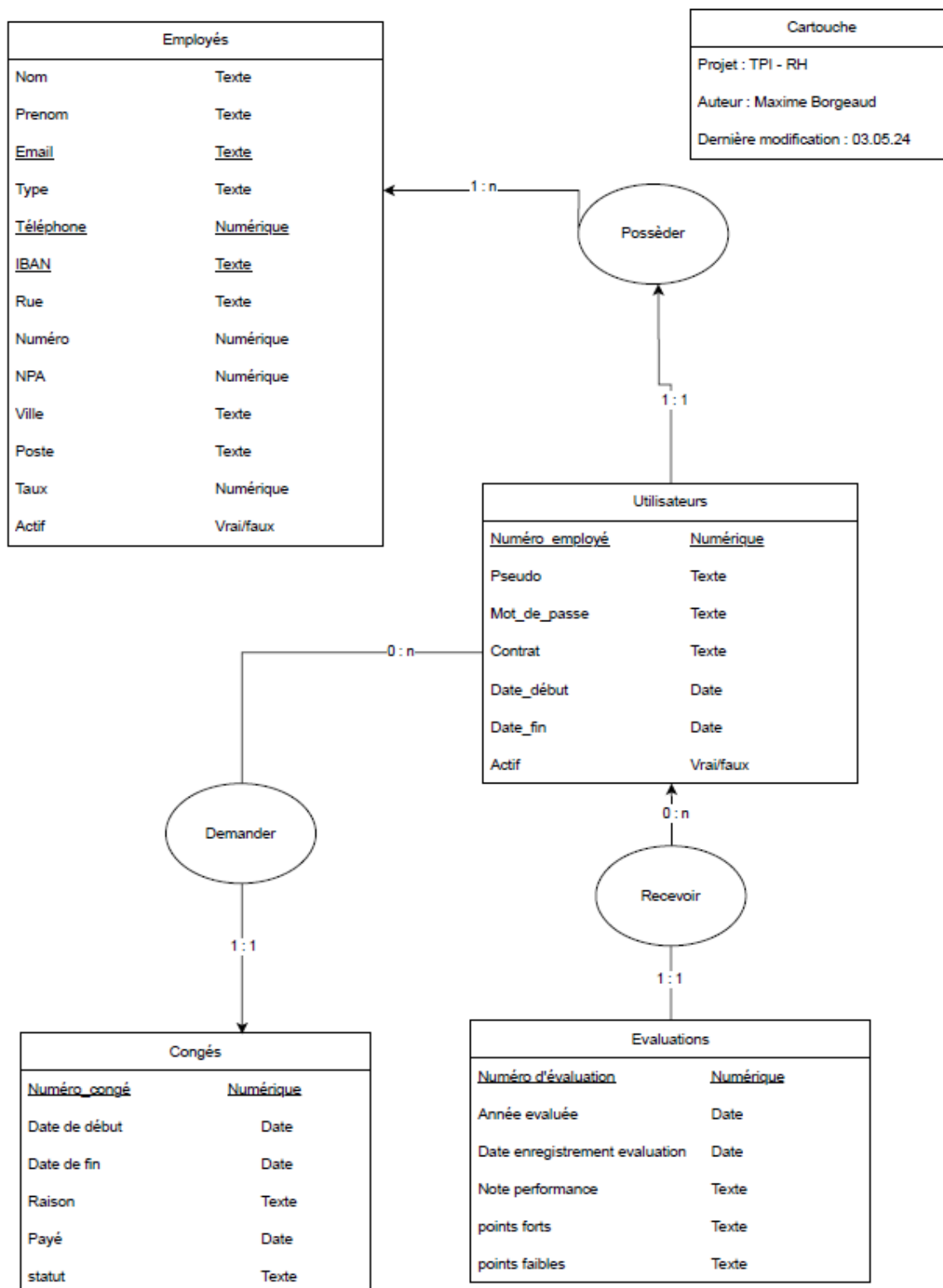


Figure 82 : MLD archives

MLD

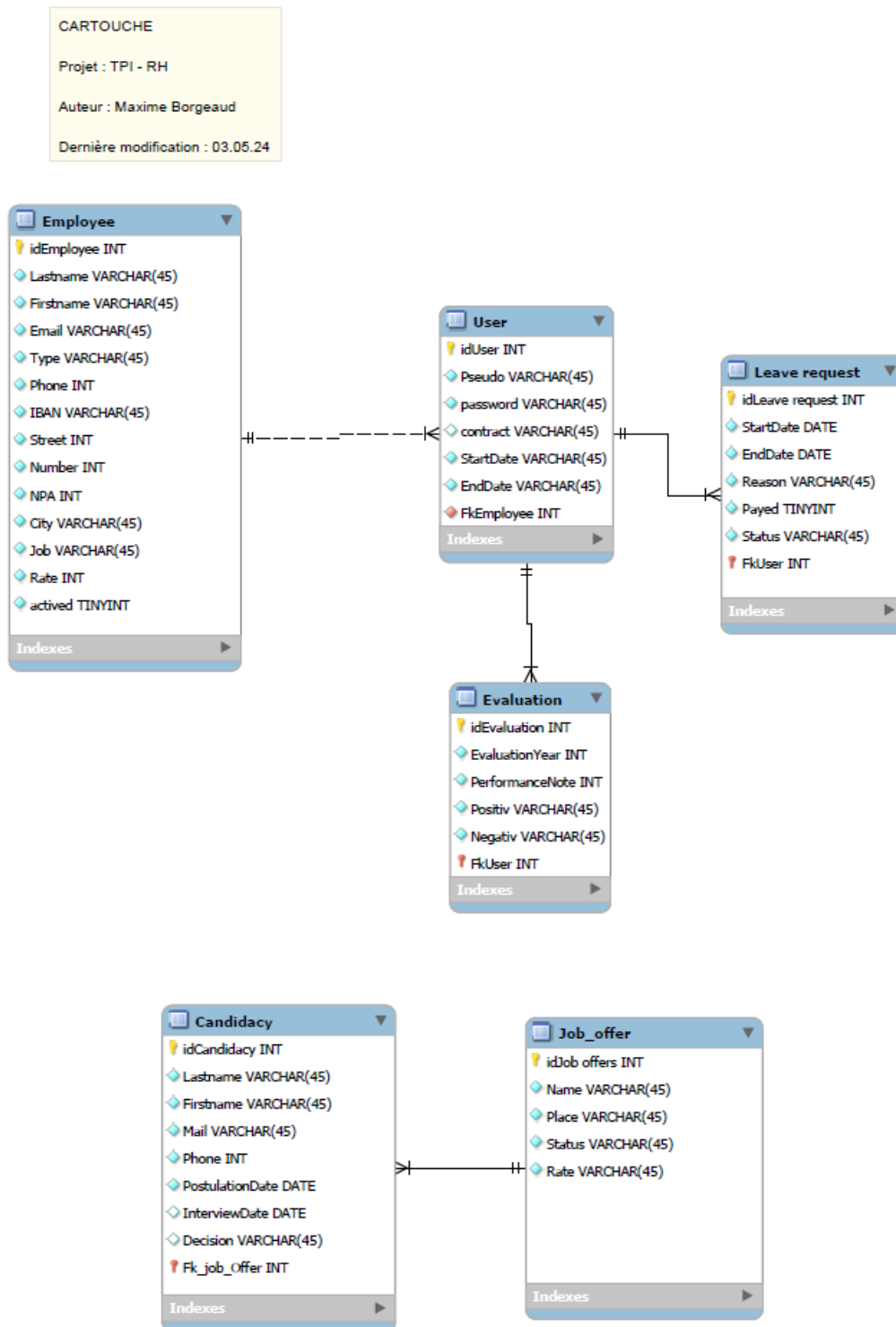


Figure 83: MLD archives