Dossier de conception

>> Diginamic.fr

Révision	Rédacteurs	Date	Objet
1	R. BONNAMY	17/02/2021	Création du document
2	A. CICCOLI	11/07/2023	Diagramme UML

1 INTRODUCTION

1.1 Objet du document

Ce document a pour objectif de présenter l'essentiel des questions techniques liées à la mise en place de l'application Gestion des abscences.

Ce document présente :

- Le diagramme de classes
- Le modèle physique de données

2 SOMMAIRE

1.1.1 Table des matières

3 ARCHITECTURE LOGICIELLE

3.1 Produits et versions

1.1.2 Langages, frameworks et librairies spécifiques

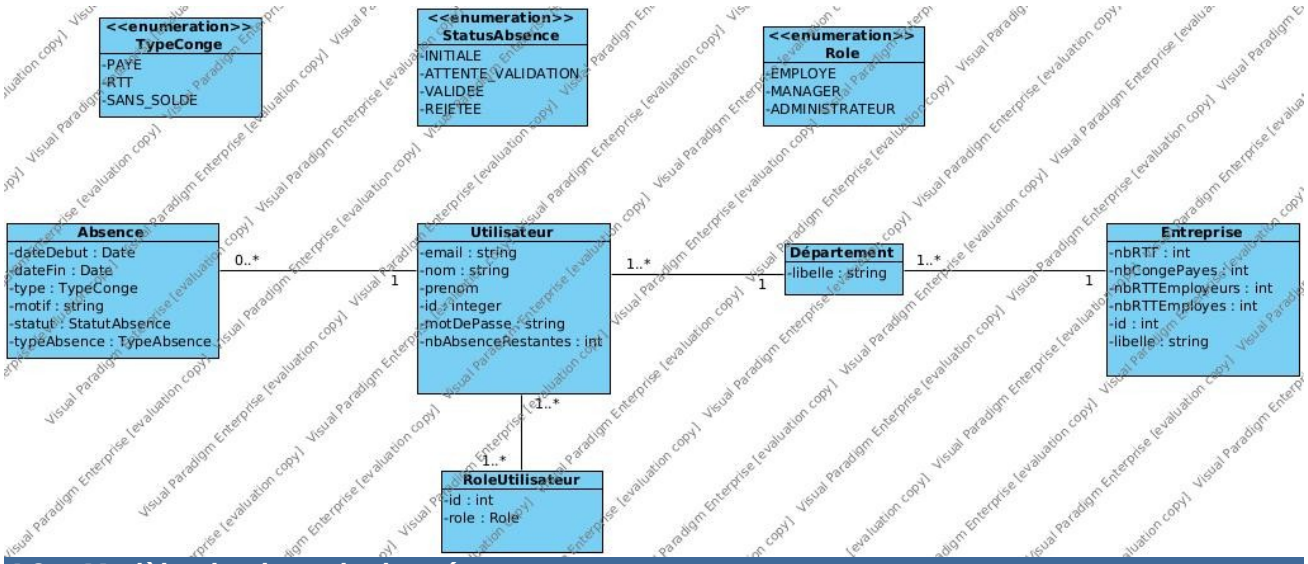
Nom	Version
Langage Java	11+
Spring Boot	3.0.5
Spring Data JPA	3.0.5
Angular	15
Apache POI (génération d'excel)	5.2.3

1.1.3 Serveur de base de données

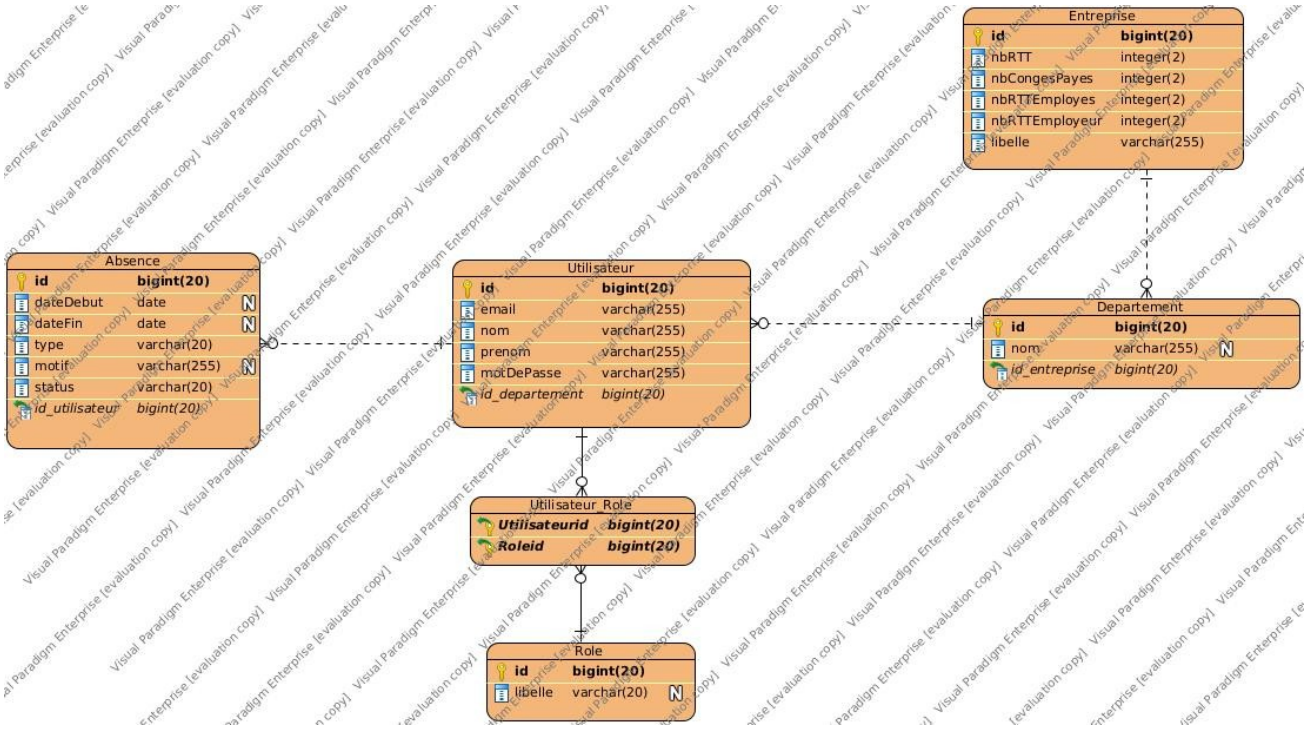
Nom	Version
MySQL	8
Clever Cloud	

4 FOCUS TECHNIQUE

4.1 Diagramme de classes métier



4.2 Modèle physique de données



4.3 Règles de développement coté back

Règles de développement :

- Javadoc au moins sur les classes et les méthodes

- Découpage du code en petites méthodes simples.
- Indentation du code
- Test unitaires au moins pour les services

Découpage en couches :
décrire les différentes couches avec contrôleurs, DAO, classes techniques, DTO, etc.

Plusieurs couches :
- Controller, qui récupère les données du front et en envoie
- Service, qui implémente la logique métier pour le transfert des données
- Repository, qui implémente la communication à la base de données via Spring Data JPA

Découpage en packages : étant donné le peu d'entités du projet, on va découper le projet en fonction des types de classes, et non des domaines des entités. Donc :

- un package controller
- un package service
- un package mapper
- un package entites
- un package DTO
- un package repository
- un package utils
- un package exception
- un package validator

Toute forme de post va demander beaucoup de validation coté back. Pour cela, il peut être intéressant de lester les différents services des validateurs pour en faire des classes à part entière

Règles de nommage :
- on suffixe les noms avec leur package
- pour les DTOs, s'il y en a plusieurs, on préfixe les DTOs avec un terme les qualifiant (par exemple, SimpleAbscenceDto)

5 TESTS ET INTÉGRATION

5.1 Stratégie de tests

Il y aura des tests unitaires à développer pour tester le code de toutes les classes de services et validateurs.

5.2 Indicateurs de qualité de code

Renseigner à 100% la javadoc.
Règles de nommage des classes, des packages : conventions standards à respecter

Nom du garant de la qualité : Abel Ciccoli