|  |
| --- |
| Emilie Bonnin & Morgane LOPVET – Apping2 2014 |
| Rapport du projet J2EE |
| Intranet de la Poste: Gestion des erreurs de caisse |

|  |
| --- |
| Morgane LOPVET  [Choisir la date] |

Table des matières

[1. Introduction 2](#_Toc342420974)

[1.1. Contexte du projet 2](#_Toc342420975)

[1.2. Equipe 2](#_Toc342420976)

[1.3. Ce qui a été fait 2](#_Toc342420977)

[2. Réalisation 2](#_Toc342420978)

[2.1. Conception 2](#_Toc342420979)

[2.2. Développement 2](#_Toc342420980)

[3. Mode d’emploi 2](#_Toc342420981)

[3.1. Compilation, déploiement et exécution 2](#_Toc342420982)

[3.2. Exemples 2](#_Toc342420983)

[3.3. Tests 2](#_Toc342420984)

[4. Conclusion 2](#_Toc342420985)

# Introduction

## Présentation du projet

Le projet consiste en un intranet permettant la gestion des erreurs de caisse dans les bureaux de Poste. Il est destiné aux agents de la Poste. Il doit permettre de signaler et de régulariser totalement ou partiellement des erreurs de caisse ainsi que d’en assurer le suivi. La gestion des droits est également indispensable et l’authentification est obligatoire.

## Equipe

Notre équipe se compose de Emilie Bonnin et Morgane Lopvet, toutes deux apprenties et suivant une formation en ingénierie informatique à Epita.

Morgane travaille chez Eureva, société pour laquelle elle développe des applications mobiles, notamment utilisant la reconnaissance d’image.

Emilie…

## Ce qui a été fait

Toutes les fonctionnalités requises dans le sujet ont été implémentées. C’est-à-dire :

* Gestion des droits
* Déclaration d’une erreur de caisse
* Consultation des erreurs de caisses
* Régularisation partielle ou totale d’une erreur de caisse
* Consultation des régularisations des erreurs de caisse
* Calcul des bilans des erreurs de caisse journaliers et périodiques
* Envoi d’un mail d’avertissement en cas de dépassement d’un certain montant du bilan (en déficit ou excédent)

## Technologies utilisées

Nous avons choisi d’utiliser une base de données Oracle accessible au code java grâce à la plateforme JDBC. Nous nous sommes servies des servlets ainsi que de pages jsp. La technologie JSTL ainsi que les EL nous ont permis de faire communiquer le contrôleur avec la vue. Nous avons respecté un modèle MVC. Le serveur Tomcat a été utilisé pour déployer l’application. L’API Java mail est également de la partie. Du côté de l’affichage, nous avons utilisé des ccs et du JavaScript via les bibliothèques JQuery et TableSorter.

# Réalisation

## Conception

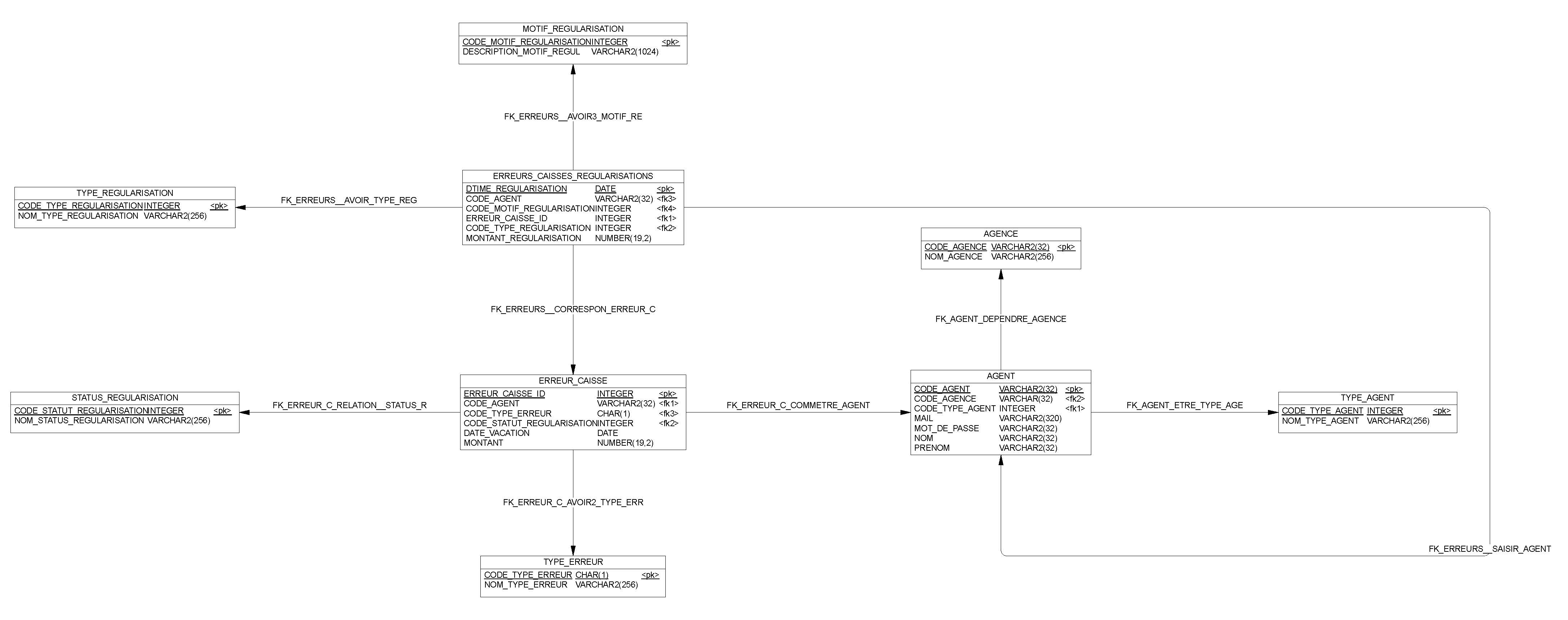
### Modèle conceptuel de données

Après avoir étudié le sujet, nous avons réalisé ce modèle conceptuel de données. Afin d’avoir une organisation propre et permettre une évolution des fonctionnalités, nous avons séparé au maximum les différentes entités.

### D:\Users\Morgane LOPVET\Documents\Cours\Epita\Projet\Apping 2\Intranet de la Poste\Diagrammes\Diagramme conceptuel de données.jpg

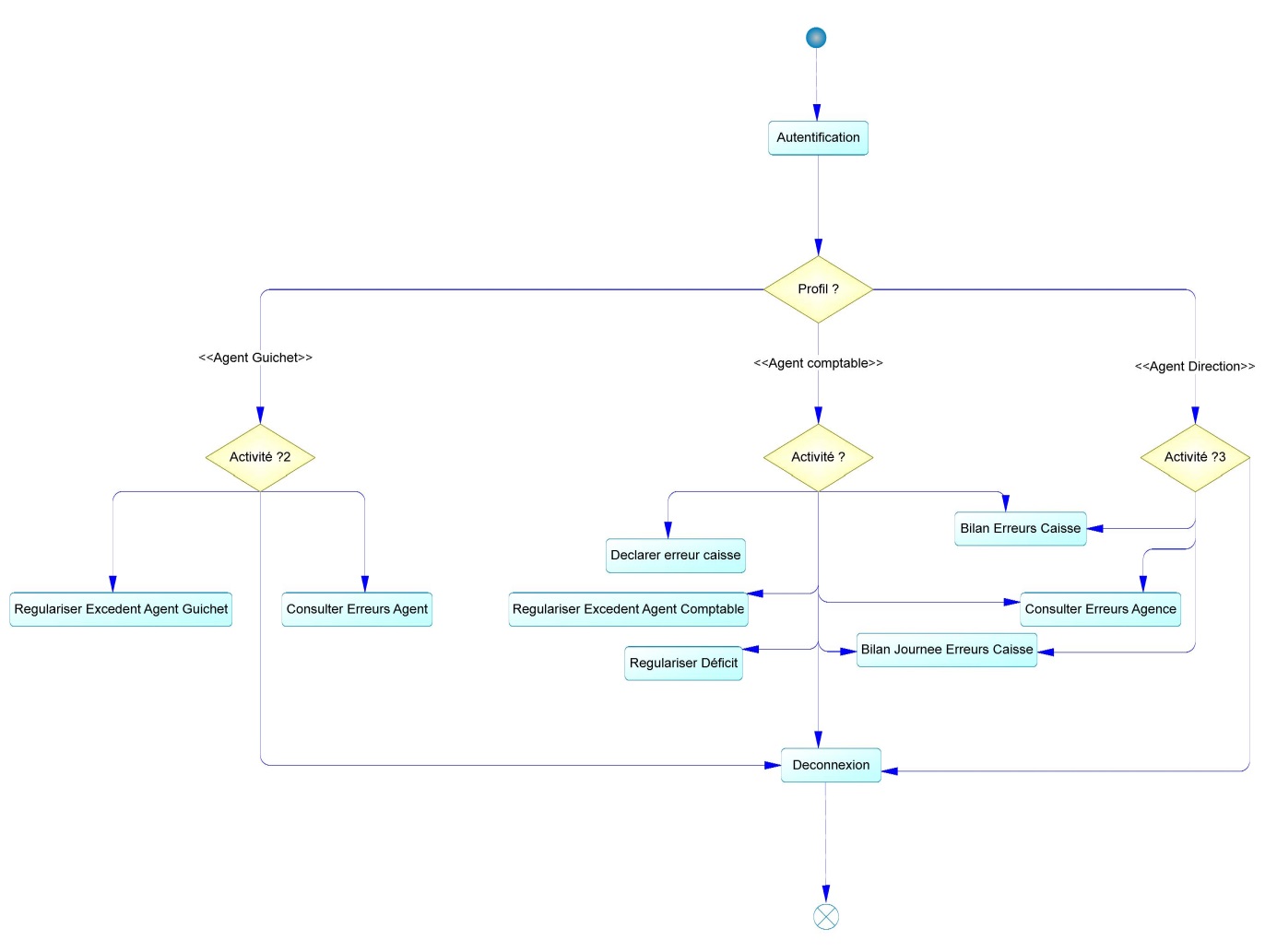
### Modèle physique de données

Le modèle conceptuel de données nous a alors permis de concevoir le modèle physique de données. C’est ce modèle qui représente notre base de données. Le moindre détail, la moindre contrainte a été indiquée dans ce modèle afin que le logiciel PowerAMC génère directement le code SQL de création de la base de données.



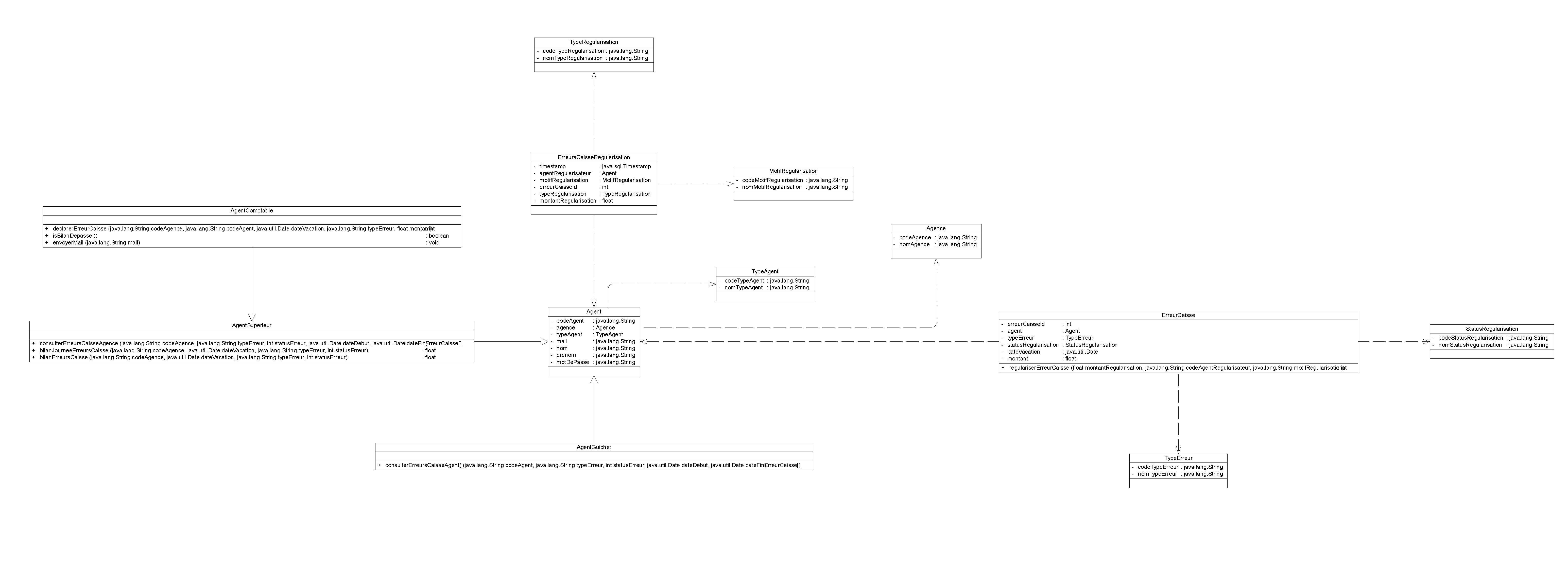
### Diagramme d’activités UML

A ce stade de la conception, nous avons voulu avoir une vision plus précise des fonctionnalités à implémenter ainsi que des différents chemins de navigation possibles. A cette fin, nous avons conçu le diagramme d’activités UML suivant.



### Diagramme de classes UML

Une fois la base de données mise en place et le digramme d’activité défini, nous avons réfléchis au diagramme de classes UML. Seules les classes métier ont été spécifiées mais quelques classes utilitaires ainsi que des DAO (uniques classes accédant directement à la base de données) ont été implémentées.



Afin de gérer les différents agents ainsi que leurs droits, nous avons décidé de créer une classe agent de laquelle hérite des agents de type spécifique en fonction des fonctionnalités qui lui sont accessibles. Ainsi, les agents de direction et comptables ont accès à des fonctionnalités communes mais l’agent de direction n’a aucune fonctionnalité exclusive. La classe AgentSupérieur réunit dont les fonctionnalités communes aux agents comptables et de direction tandis que les agents de guichet et comptables ont chacun leur propre classe implémentant les fonctionnalités qui leurs sont réservées.

## Développement

# Mode d’emploi

## Compilation, déploiement et exécution

## Exemples

## Tests

# Conclusion