**Projet d’Intégration JAVA**

**Institut Supérieur d’Enseignement Professionnel de Diamniadio**



**Un système de paiement automatique des employés en java.**

***Sujet:***

***Realisé par :***

**Abdoussalam TINE :** *Développeur BACK-END*

**Lamine DIEME :** *Développeur FRONT-END*

**Ramatoulaye Baldé :** *Développeur FRONT-END*

**Concevoir une application mettant en œuvre les connaissances acquises lors de l’enseignement en programmation orienté objet Java et aussi découvrir d’autres technologies utilisées dans le développement des logiciels**

**Table de matières**

**INTRODUCTION GENERALE**……...…………………. 3

1. **Architecture du Logiciel**……………………………...4
2. **Modélisation UML**……………………………………5
3. Diagramme de cas d’utilisation……..…….……..…….6
4. Identification des Acteurs…………………………..6
5. Description des Interactions………………………...7
6. Diagramme de classe……………….……….…………10
7. Concept de classe……………………...……….…10
8. Dictionnaire de class et des attributs…………...…11
9. Commandes de création de tables SQL……………..…12
10. Création des tables…………..…………..…..………12
11. Insertion des exemples proposés dans les tables……12
12. **Méthodes et Algorithmes**………….…………………13
13. **Conclusion**……………....…………….………………14
14. **Annexe**…………………..…………………………….14

**INTRODUCTION GENERALE**

De nos jours le monde est considérablement interconnecté. En effet, l’usage des différents outils de gestion poursuit sa croissance à travers le monde.

Un tel engouement s’explique par l’abolition des distances, la facilité du paiement, des achats, par tout le monde.

Cependant, la réalisation d’une application qui gère toutes ces taches impliquent la présence de certains critères pour mettre en valeur une bonne gestion des employés se trouvant dans l’entreprise **DAKARDISTRIBUTION** depuis le premier jour du contrat jusqu’à la fin.

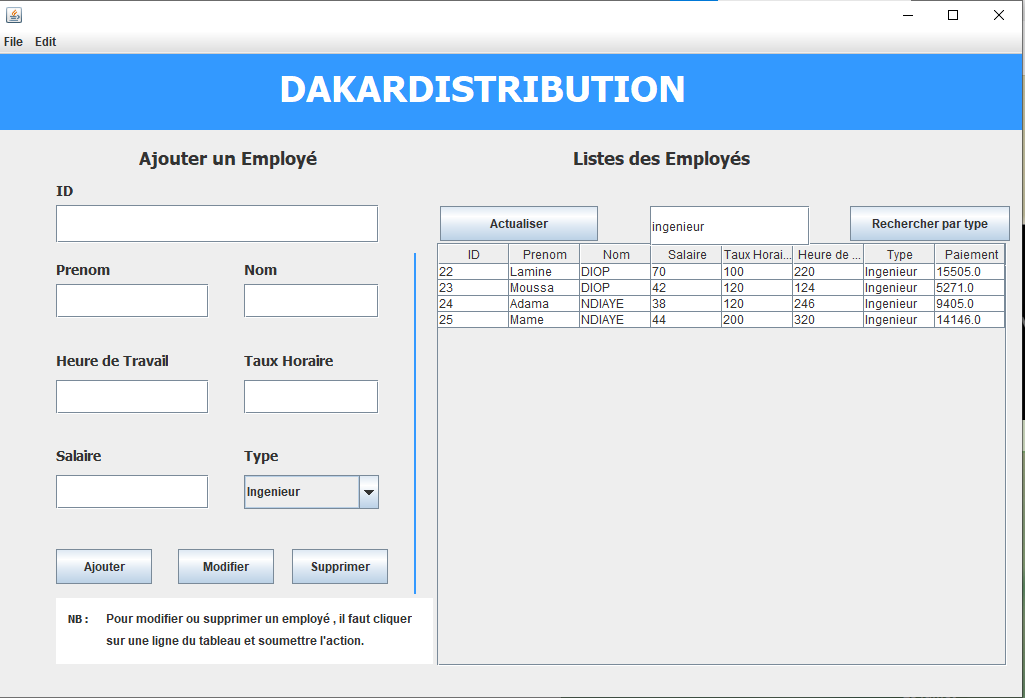
Dans le cadre de notre projet « d’intégration en java », nous étions invités à créer un programme qui permet de gérer le paiement des employés d’une société se nommant **DAKARDISTRIBUTION.**

1. **Architecture de Logiciel**

Ainsi, l'objectif de notre projet est de réaliser un système de paiement automatique des employés baptisé **DAKARDISTRIBUTION** interactive, fiable, conviviale et facile à s'intégrer dans l'environnement de travail d’une société et de multiples secteurs commerciaux et

Administratifs.

Cette application vise essentiellement à payer leur salaire par catégories (managers ou ingénieurs) en fonction de leur nombre d’heures de travail (Les managers considèrent les heures passées sur le terrain de golf comme des heures de travail et les ingénieurs considèrent les heures de communication  
sur les questions professionnelles comme des heures de travail).



1. **Modélisation UML**

Dans ce chapitre nous allons présenter la conception de notre système baptisé **DAKARDISTRIBUTION**, nous allons faire un appel à la boite à outils UML tout en respectant le processus d’unification UP à savoir : le diagramme des cas d’utilisation, le diagramme de séquences et le diagramme de classes.

• Diagramme de cas d’utilisation qui nous aide dans la spécification des besoins de notre système.  
• Diagramme de séquence qui nous donne le scénario des actions et interactions entre les acteurs et notre système.  
• Diagramme de classes qui définit l’architecture statique de notre système.

* **Définition :**

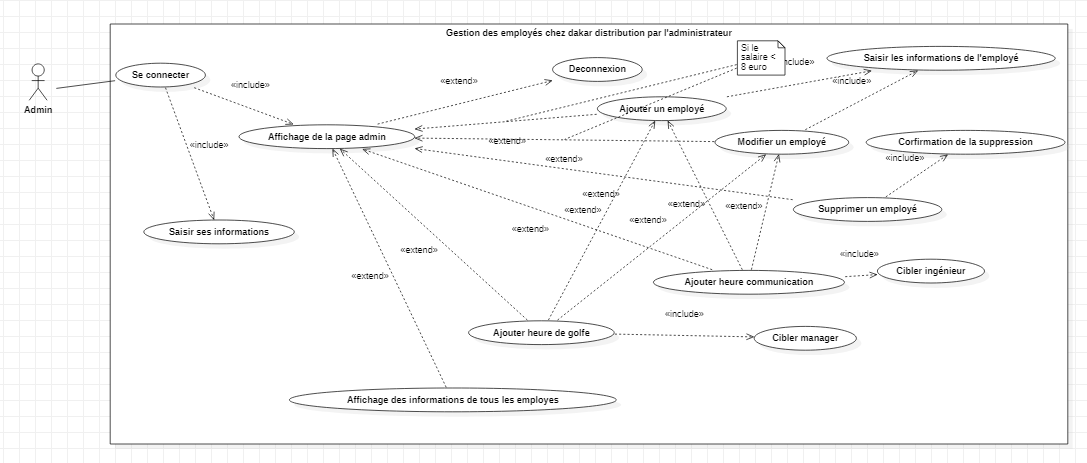
Elle est utilisée pour spécifier un logiciel et/ou pour concevoir un logiciel. Dans la  
spécification, le modèle décrit les classes et les cas d’utilisation, vus de l’utilisateur final du logiciel. Le modèle produit par une conception orientée objet est en général une extension du modèle issu de la spécification.

Ces ensembles d’outils techniques n’intéressent pas l’utilisateur final du logiciel mais seulement ses concepteurs. Il comprend les modèles des classes, des états et d’interaction. UML est également utilisée dans les phases terminales du développement avec les modèles de réalisation et de déploiement.

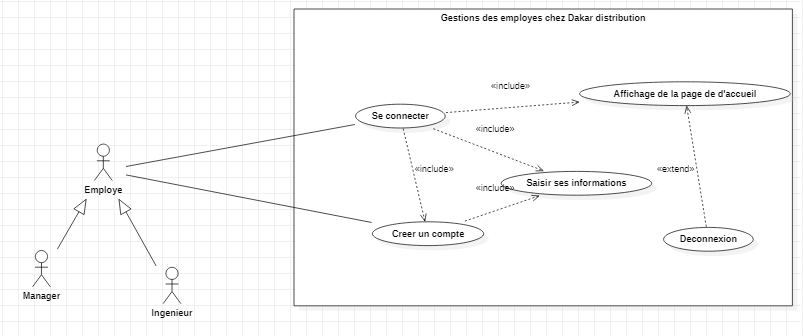
1. **Diagramme de cas d’utilisation**

Un diagramme de cas d’utilisation permet de représenter graphiquement les cas d’utilisations. C'est le diagramme principal du modèle UML, celui qui s'assure la relation  
entre l'utilisateur et les objets que le système met en œuvre.

* **Identification des acteurs :**Un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système) qui interagissent directement avec le système. Les  
  principaux profils qui auront à utiliser le Système sont les suivants :Un **admin** et les **employés** désireux de voir son état de paiement

****

**Diagramme de cas d'utilisation d’un Admin système DAKARDISTRIBUTION**

****

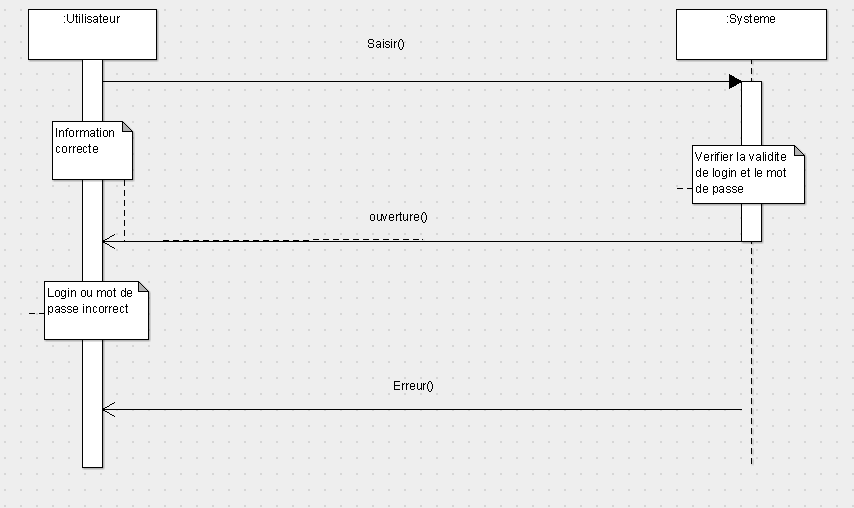
**Diagramme de cas d'utilisation d’un Employé système DAKARDISTRIBUTION**

* **S’authentifier**Processus par lequel le système s'assure de l'identité de l’utilisateur, il doit s'authentifier par un login et un mot de passe. L’authentification se fait qu’une seule fois. L’utilisateur va avoir un identifiant et un mot de passe. La prochaine connexion sur le site sera prise en compte avec son identifiant.
* **Ajouter – Modifier – Supprimer**

L’utilisateur peut ajouter un employé via le formulaire, il peut

Aussi modifier ou supprimer un employé.

* **Diagramme de séquence**Les diagrammes des séquences permettent de représenter les interactions entre objet selon un point de vue temporel l’accent est mis sur la chronologie des envois de messages.  
  • Scénario : une liste d’actions qui décrivent une interaction entre un acteur et le système.  
  • Interaction : Un comportement qui comprend un ensemble de messages échangés par un ensemble d’objets dans un certain contexte pour accomplir une certaine tache.  
  • Message : Un message est une transmission d'information unidirectionnelle entre deux objets, l'objet émetteur et l'objet récepteur.
* **Authentification**Voilà ci-dessous le diagramme d’authentification d’un utilisateur:



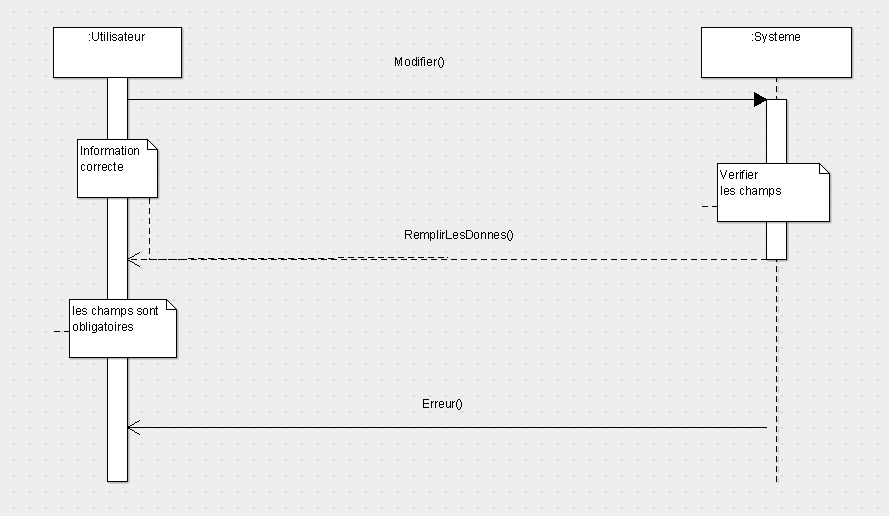
**Diagramme de séquence << Authentification>>**

* **Ajouter un employé :**Après l’authentification l’utilisateur peut ajouter le paiement d’un employé



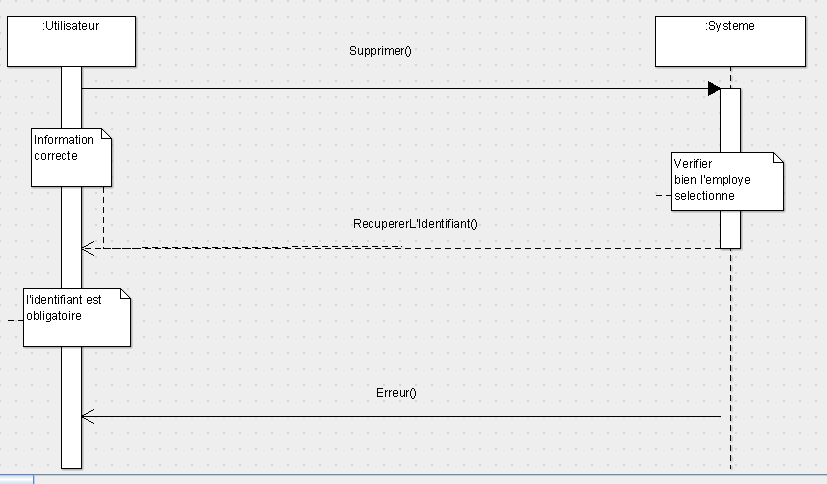
**Diagramme de séquence << Ajouter un employé >>**

* **Modifier un employé :**Après l’enregistrement, l’utilisateur a le droit de modifier le paiement d’un ou plusieurs employés, le diagramme suivant représente cette partie



**Diagramme de séquence << Modifier un employé >>**

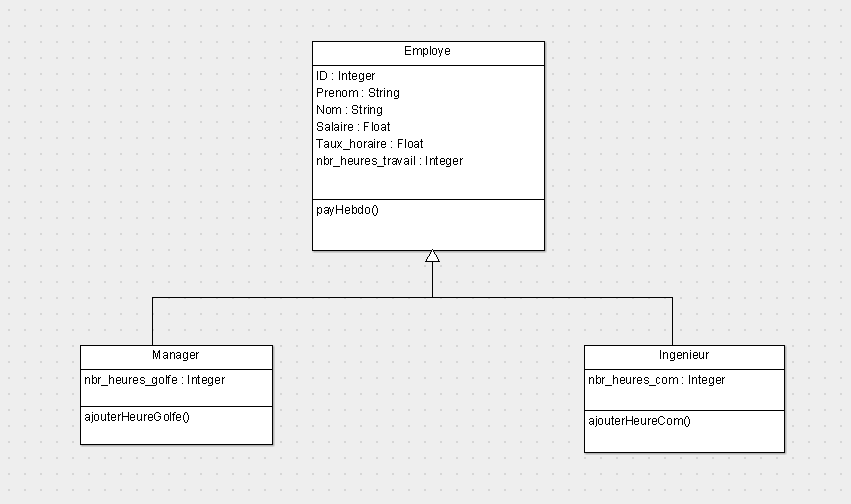
* **Supprimer un employé :**Le diagramme ci-dessous représente la partie de suppression.



**Diagramme de séquence << Supprimer un employé >>**

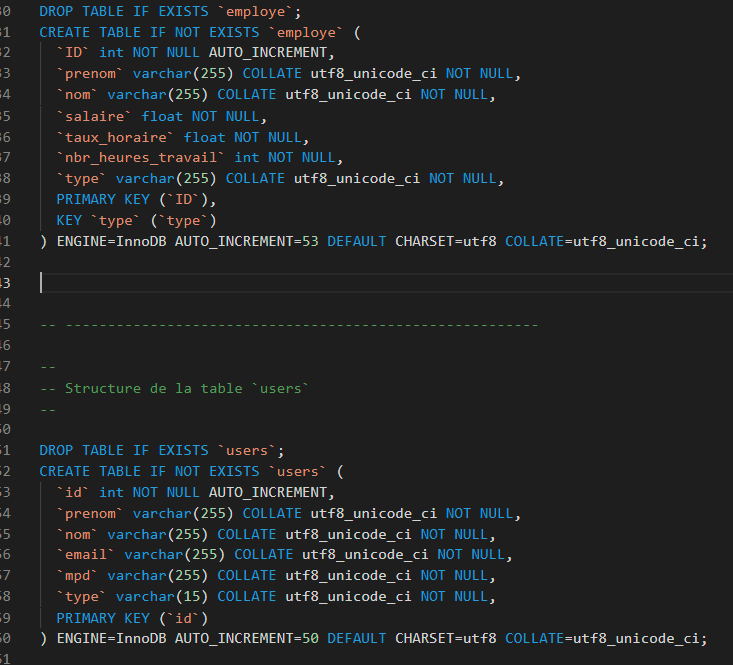
1. **Diagramme de classe :**Les diagrammes de class sans doute c’est les diagrammes les plus utiliser d’UML ils décrivent les types des objets qui composent un système et les différents types de relation  
   statique qui existent entre eux.
   1. **Le concept de la classe :**Une classe est une description d'un groupe d'objets partageant un ensemble commun de propriétés (les attributs), de comportements (les opérations) et de relations avec d'autres  
      objets (les associations et les agrégations)  
      La classe est définie par son nom, ses attributs et ses opérations.  
      Les classes sur lesquelles se porte notre système sont les suivantes :

* **Employé**
* **Manager**
* **Ingénieur**

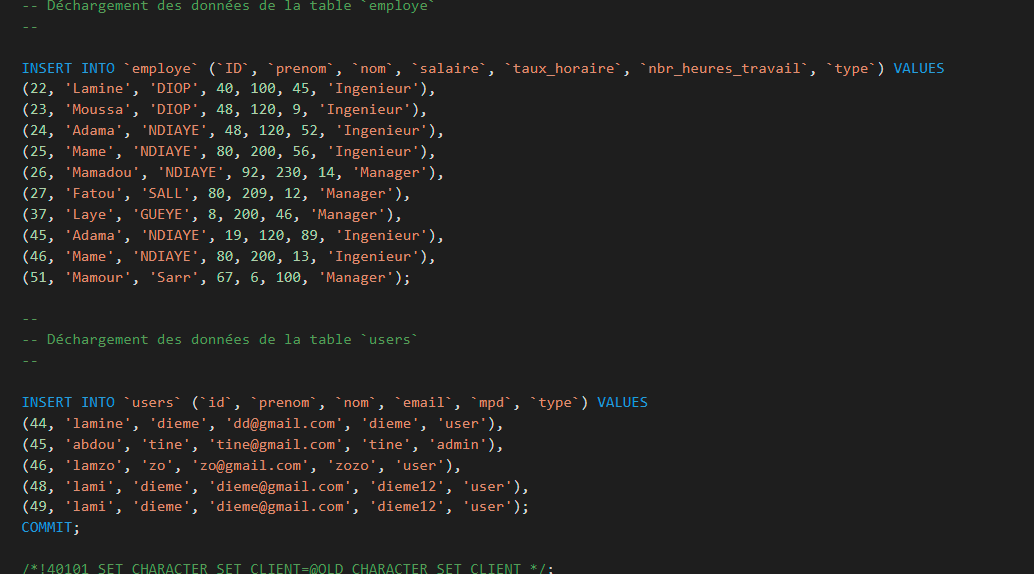
****

|  |  |  |  |
| --- | --- | --- | --- |
| **Classes** | **Attributs** | | **Méthodes** |
| **Champ** | **Type** |
| **Employe** | **ID** | **Integer** | **payHebdo()** |
| **Prenom** | **String** |
| **Nom** | **String** |
| **Salaire** | **Float** |
| **Taux\_horaire** | **Float** |
| **Nbr\_heures\_travail** | **Integer** |
| **Manager** | **Nbr\_heure\_golfe** | **Integer** | **ajouterHeureGolfe** |
| **Ingenieur** | **Nbr\_heure\_com** | **Integer** | **ajouterHeureCom** |

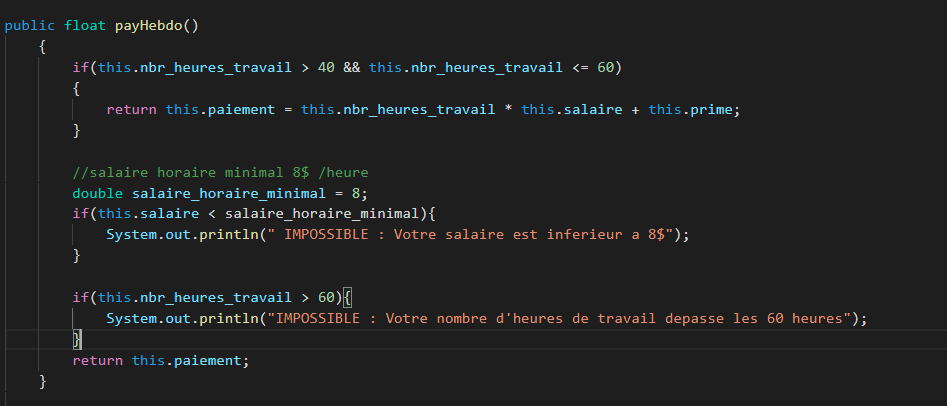
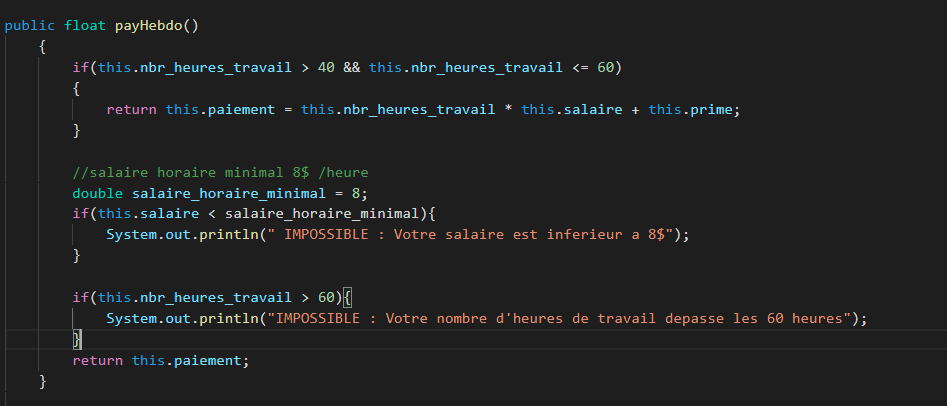
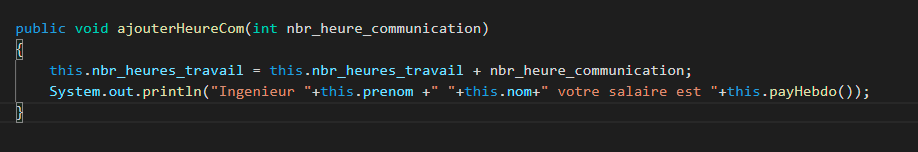
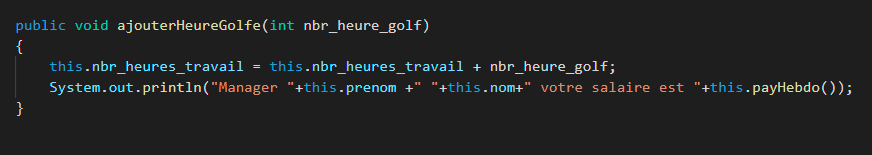
1. **Dictionnaire de class et des attributs**
2. **Commandes de création de tables SQL**
   1. **Création des tables**

****

1. **Insertion des données**

****

**Méthodes et Algorithmes:**

****

**Conclusion**

En somme nous pouvons dire que la création de l’application avec comme langage programmation **Java**  est une chose tellement enrichissant et a nécessiter des heures de travail. Par contre ça n’était pas une chose facile par le fait qu’on avait du mal à savoir comment se faisait le paiement au seins de l’entreprise, en fait de compte après avoir demander à des professionnels du métier nous avons pu implémenter ça dans notre code. Nous avons perspective de continuer à apporter plus de fonctionnalité à cette application qui est **DAKARDISTRIBUTION.**

**Annexe**

-- phpMyAdmin SQL Dump

-- version 5.0.2

-- https://www.phpmyadmin.net/

--

-- Hôte : 127.0.0.1:3306

-- Généré le : sam. 24 juil. 2021 à 18:10

-- Version du serveur : 8.0.21

-- Version de PHP : 7.3.21

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Base de données : `gestion\_paiement`

--

-- --------------------------------------------------------

--

-- Structure de la table `employe`

--

DROP TABLE IF EXISTS `employe`;

CREATE TABLE IF NOT EXISTS `employe` (

`ID` int NOT NULL AUTO\_INCREMENT,

`prenom` varchar(255) COLLATE utf8\_unicode\_ci NOT NULL,

`nom` varchar(255) COLLATE utf8\_unicode\_ci NOT NULL,

`salaire` float NOT NULL,

`taux\_horaire` float NOT NULL,

`nbr\_heures\_travail` int NOT NULL,

`type` varchar(255) COLLATE utf8\_unicode\_ci NOT NULL,

PRIMARY KEY (`ID`),

KEY `type` (`type`)

) ENGINE=InnoDB AUTO\_INCREMENT=35 DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

--

-- Déchargement des données de la table `employe`

--

INSERT INTO `employe` (`ID`, `prenom`, `nom`, `salaire`, `taux\_horaire`, `nbr\_heures\_travail`, `type`) VALUES

(22, 'Lamine', 'DIOP', 70, 100, 220, 'Ingenieur'),

(23, 'Moussa', 'DIOP', 42, 120, 124, 'Ingenieur'),

(24, 'Adama', 'NDIAYE', 38, 120, 246, 'Ingenieur'),

(25, 'Mame', 'NDIAYE', 44, 200, 320, 'Ingenieur'),

(26, 'Mamadou', 'NDIAYE', 32, 230, 225, 'Manager'),

(27, 'Fatou', 'DIOP', 45, 250, 120, 'Manager'),

(28, 'Mama', 'WADE', 34, 345, 234, 'Manager'),

(29, 'Abdou Salam', 'TINE', 45, 560, 239, 'Ingenieur'),

(31, 'Ramatoulaye', 'Balde', 456, 220, 44, 'Ingenieur');

-- --------------------------------------------------------

--

-- Structure de la table `users`

--

DROP TABLE IF EXISTS `users`;

CREATE TABLE IF NOT EXISTS `users` (

`id` int NOT NULL AUTO\_INCREMENT,

`prenom` varchar(255) COLLATE utf8\_unicode\_ci NOT NULL,

`nom` varchar(255) COLLATE utf8\_unicode\_ci NOT NULL,

`email` varchar(255) COLLATE utf8\_unicode\_ci NOT NULL,

`mpd` varchar(255) COLLATE utf8\_unicode\_ci NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=40 DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

--

-- Déchargement des données de la table `users`

--

INSERT INTO `users` (`id`, `prenom`, `nom`, `email`, `mpd`) VALUES

(1, 'Abdou Salam', 'Tine', 'ast@gmail.com', 'password'),

(2, 'Ramatoulaye', 'Balde', 'rb@gmail.com', 'password'),

(3, 'Lamine', 'Dieme', 'ld@gmail.com', 'password'),

(4, 'Lamzo', 'DIEME', 'zo@gmail.com', 'zo12'),

COMMIT;