

# Gestion d'un Supermarché



*2020/2021  
Projet de développement SGBD*

*Etudiant : HOTAK FAISAL  
Enseignant : M<sup>me</sup> D. PASSELECQ*

# **TABLE DES MATIERES**

<b>INTRODUCTION .....</b>	<b>3</b>
<b>OBJECTIFS.....</b>	<b>3</b>
<b>MODELE CONCEPTUEL .....</b>	<b>5</b>
<b>MODELE PHYSIQUE .....</b>	<b>8</b>
<b>DIAGRAMME DE CLASSES .....</b>	<b>9</b>
<b>DIAGRAMME DE USE CASE : PROCEDER A UNE VENTE .....</b>	<b>10</b>
<b>DIAGRAMME DE USE CASE : PAIEMENT D'UN TICKET.....</b>	<b>11</b>
<b>DIAGRAMME DE SEQUENCES : AUTHENTIFICATION.....</b>	<b>12</b>
<b>DIAGRAMME DE SEQUENCES : MVC .....</b>	<b>13</b>
<b>APPLICATION .....</b>	<b>14</b>
<b>LIBRAIRIE JFREECHART .....</b>	<b>22</b>
<b>METHODE DE CRYPTAGE/DECRYPTAGE .....</b>	<b>24</b>
<b>QUELQUES EXEMPLES DU CODE SOURCE JAVA .....</b>	<b>27</b>
<b>CONCLUSION.....</b>	<b>30</b>
<b>BIBLIOGRAPHIE .....</b>	<b>30</b>

## INTRODUCTION

*Dans le cadre de ce projet, j'ai été amené à réfléchir sur la création d'une base de données dont le but serait la gestion d'un supermarché.*

*Cette application concrète nous a immédiatement confronté aux différentes difficultés et interrogations qui peuvent se poser lors de la création d'une base de données.*

*Le client pour lequel cette application est dédié serait un employeur qui cherche à organiser sa liste de clients, employés, produits, gérer la vente de produits et avoir un aperçu des chiffres de ventes.*

*Le but est donc de collecter les données, les sauvegarder, les organiser, les modéliser pour faciliter leur gestion finale en tant qu'informations utiles à notre client.*

## OBJECTIFS

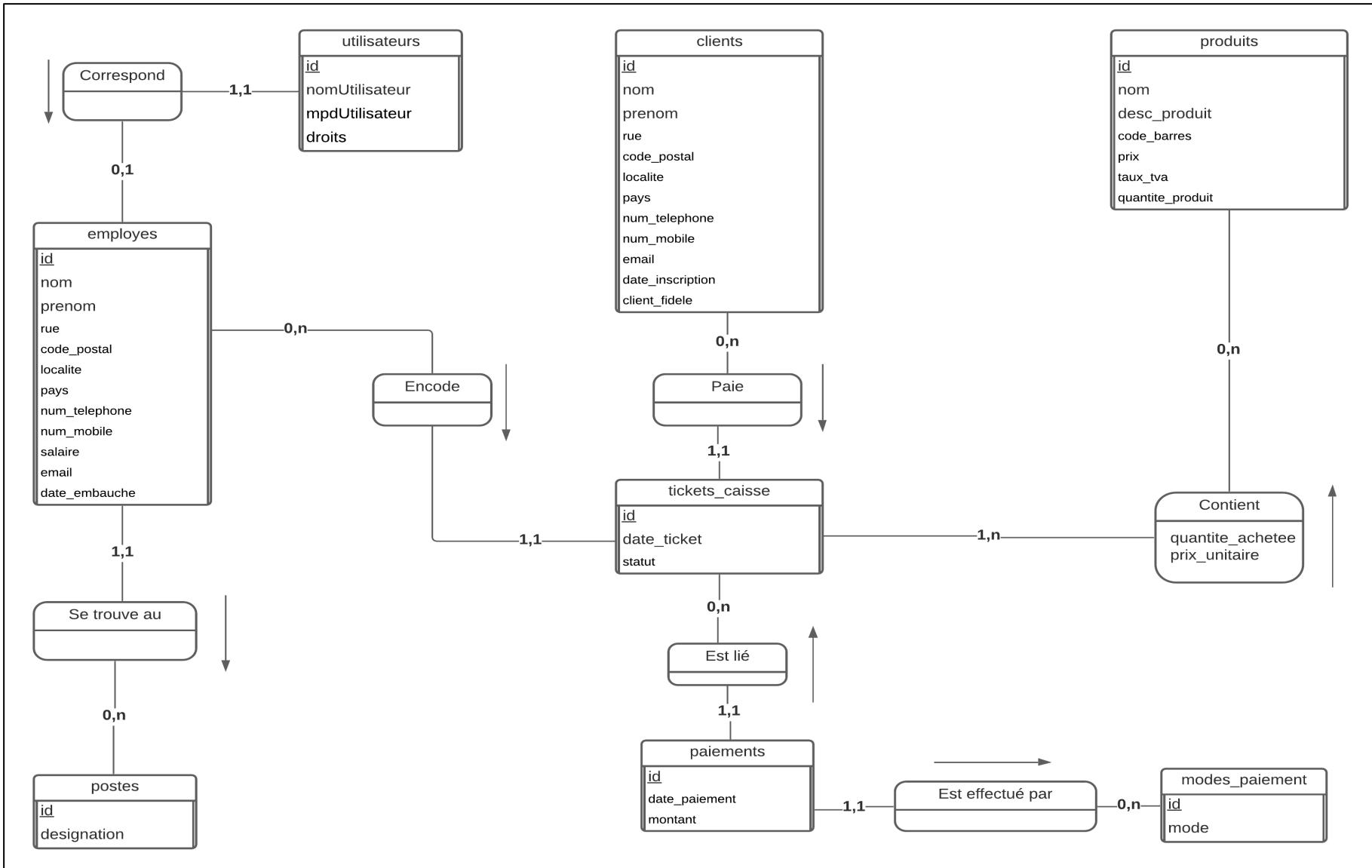
*L'application devra assurer plusieurs fonctionnalités :*

- **Connexion à la base de données** à l'aide d'un identifiant et d'un mot de passe (ce dernier étant crypté dans la base de données)
  
- **Gestion des clients**
  - Récupération des clients dans la base de données
  - Ajout, modification et suppression des clients dans la base de données
  - Rechercher des clients dans la base de données à partir d'un nom ou prénom donné
  
- **Gestion des employés**
  - Récupération des employés dans la base de données
  - Ajout, modification et suppression des employés dans la base de données
  - Rechercher des employés dans la base de données à partir d'un nom ou prénom donné

- **Gestion des produits**
  - Récupération des produits dans la base de données
  - Ajout, modification et suppression de produits dans la base de données
  - Rechercher des produits dans la base de données à partir d'un nom, d'un code-barres ou d'une description
- **Gestion des tickets de caisse**
  - Récupération des tickets de caisse dans la base de données
  - Possibilité de solder un ticket de caisse en plusieurs fois
  - Possibilité de générer un ticket de caisse au format PDF, celui-ci se lance automatiquement avec l'application par défaut
  - Suppression des tickets de caisse dans la base de données
  - Rechercher des tickets de caisse dans la base de données à partir d'un nom ou prénom (client ou employé) donné
- **Procéder à une vente**
  - Récupération des clients, employés et produits
  - Choix du client et de l'employé qui seront associés au ticket
  - Ajouter, retirer des produits du panier
  - Enregistrement du ticket pour paiements
- **Consulter les chiffres de ventes**
  - Choisir une date (un jour)
  - Affichage de la quantité de chaque produits vendus et le montant des paiements reçus dans la base de données
- **Gestion des paramètres de connexion à la base de données et des données des utilisateurs.**
  - Paramètres de connexion à la base de données
    - ⇒ Récupération des données contenues dans le fichier crypté (parametres.enc)
    - ⇒ Possibilité de modifier le nom, mot de passe, driver et serveur
    - ⇒ Génération d'un fichier crypté (parametres.enc) contenant ces paramètres
  - Gestion des utilisateurs
    - ⇒ Récupération des utilisateurs dans la base de données
    - ⇒ Ajout, modification et suppression des utilisateurs dans la base de données
    - ⇒ Possibilité d'afficher les mots de passe décryptés des utilisateurs

# MODELE CONCEPTUEL

## *Schéma conceptuel*



## *Description des entités :*

- Utilisateurs : Un utilisateur est défini par un identifiant unique. Il possède un nom, un mot de passe ainsi que des droits.
- Employés : Un employé est défini par un identifiant unique. Il a un nom, prénom, rue, code postal, localité, pays, numéro de téléphone, numéro de gsm, email, salaire et une date à laquelle l'employé est embauché.
- Postes : Un poste est défini par un identifiant unique. C'est un rôle que l'employé endossera. Il pourra par exemple être un caissier, réassortisseur, gérant... Son poste n'est pas permanent et peut changer.
- Clients : Un client est défini par un identifiant unique. Il a un nom, prénom, rue, code postal, localité, pays, numéro de téléphone, numéro de gsm, une date à laquelle il s'est inscrit, une carte de fidélité si le client est fidèle.
- Modes de paiement : Un mode de paiement est défini par un identifiant unique. Le mode concerne tous les types de paiement comme par exemple le cash, Bancontact, la carte Visa, ...
- Paiements : Un paiement est défini par un identifiant unique. Il contiendra le montant du paiement ainsi qu'une date générée au moment du paiement.
- Produits : Un produit est défini par un identifiant unique. Le produit a un nom, une description, un code-barres, un prix, un taux de tva et son stock en quantité disponible.
- Tickets de caisse : Un ticket de caisse est caractérisé par un identifiant unique. Une date sera générée lors de sa création. Le statut définit le statut de paiement du ticket, si le ticket est payé dans son entièreté ou non.

## *Les associations :*

- Un utilisateur **correspond** à un employé.
- Un employé **se trouve** à un poste.
- Un employé **encode** un ticket de caisse.
- Un client **paie** un ticket de caisse.
- Un paiement **est effectué par** un mode de paiement.
- Un paiement **est lié** à un ticket de caisse.
- Un ticket de caisse **contient** un produit.

## *Les contraintes :*

- Un utilisateur correspond à **1 seul** poste. Un employé correspond à **0 ou 1** utilisateur.
- Un employé se trouve à **1 ou N** poste(s). Un poste est attribué à **0 ou N** employé(s).
- Un employé encode **0 ou N** ticket(s). Un ticket est encodé par **1 seul** employé.
- Un client paie **0 ou N** ticket(s) de caisse. Un ticket de caisse est payé par **1 seul** client.
- Un paiement est effectué par **1 seul** mode de paiement. Un mode de paiement appartient à **0 ou N** paiement(s).
- Un paiement est lié à **1 seul** ticket de caisse. Un ticket contient **0 ou N** paiements.
- Un ticket contient **1 ou N** produit(s). Un produit est contenu dans **0 ou N** ticket(s).

## *Remarques :*

Les tickets de caisse étant liés à des employés et clients, si nous devions supprimer un employé ou client, le ticket de caisse associé serait supprimé.

Dans notre cas, nous n'allons pas supprimer les clients ou employés. Nous allons simplement ne plus les afficher dans le programme. Ainsi, le ticket ne disparaîtra pas.

Il faudra toutefois nettoyer les champs contenant des données personnelles dans le cadre des obligations du Règlement Européen sur la Gestion des données personnelles :

### Clients

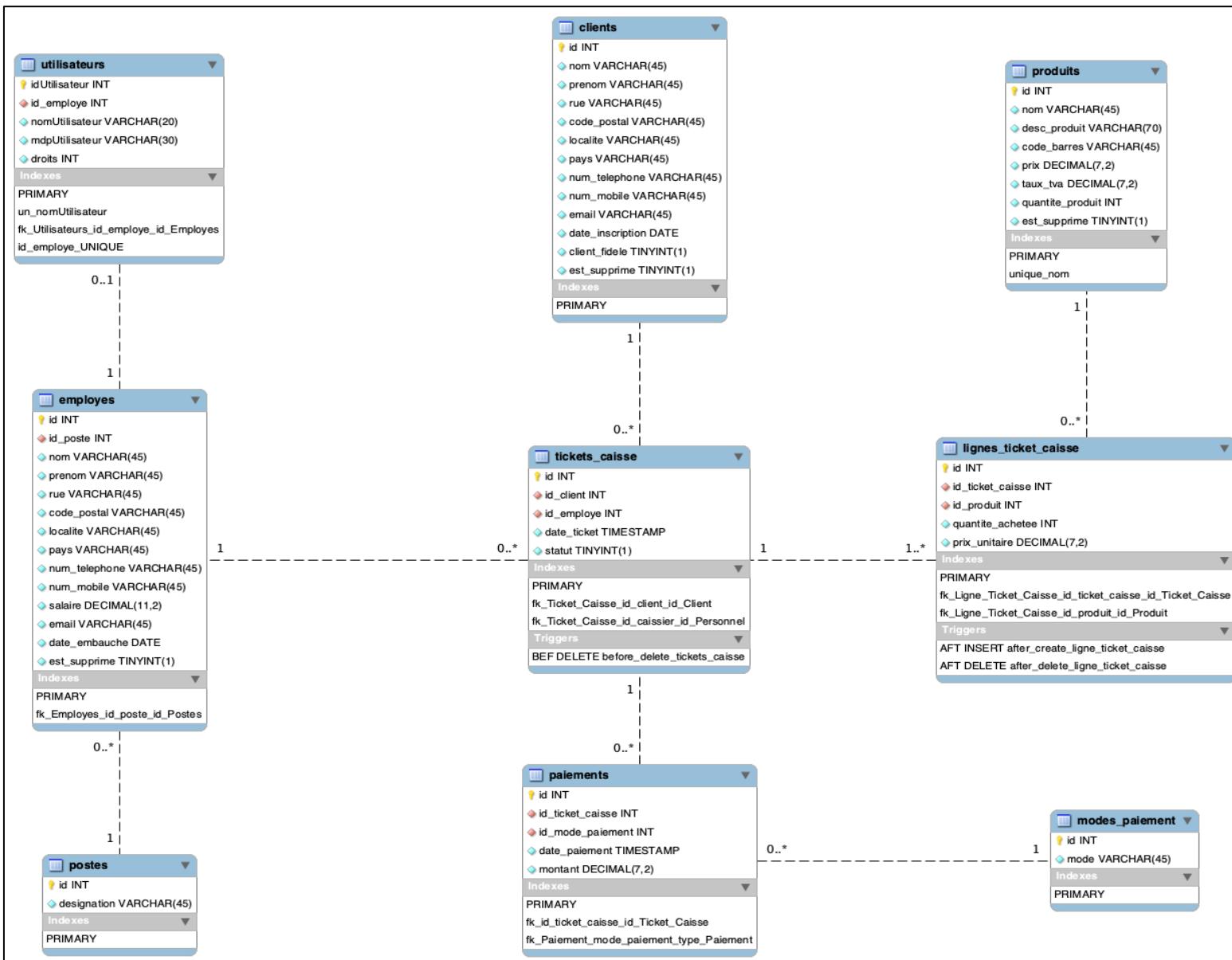
Vous devez supprimer les données personnelles des personnes inactives depuis **3 ans** de votre base de données. Il s'agit ici du droit à l'oubli. Vous pouvez rendre anonyme ces données afin de les conserver pour leur valeur statistique.

### Employés

La conservation des données personnelles des salariés de l'entreprise est limitée à **5 ans** après la fin de la relation contractuelle.

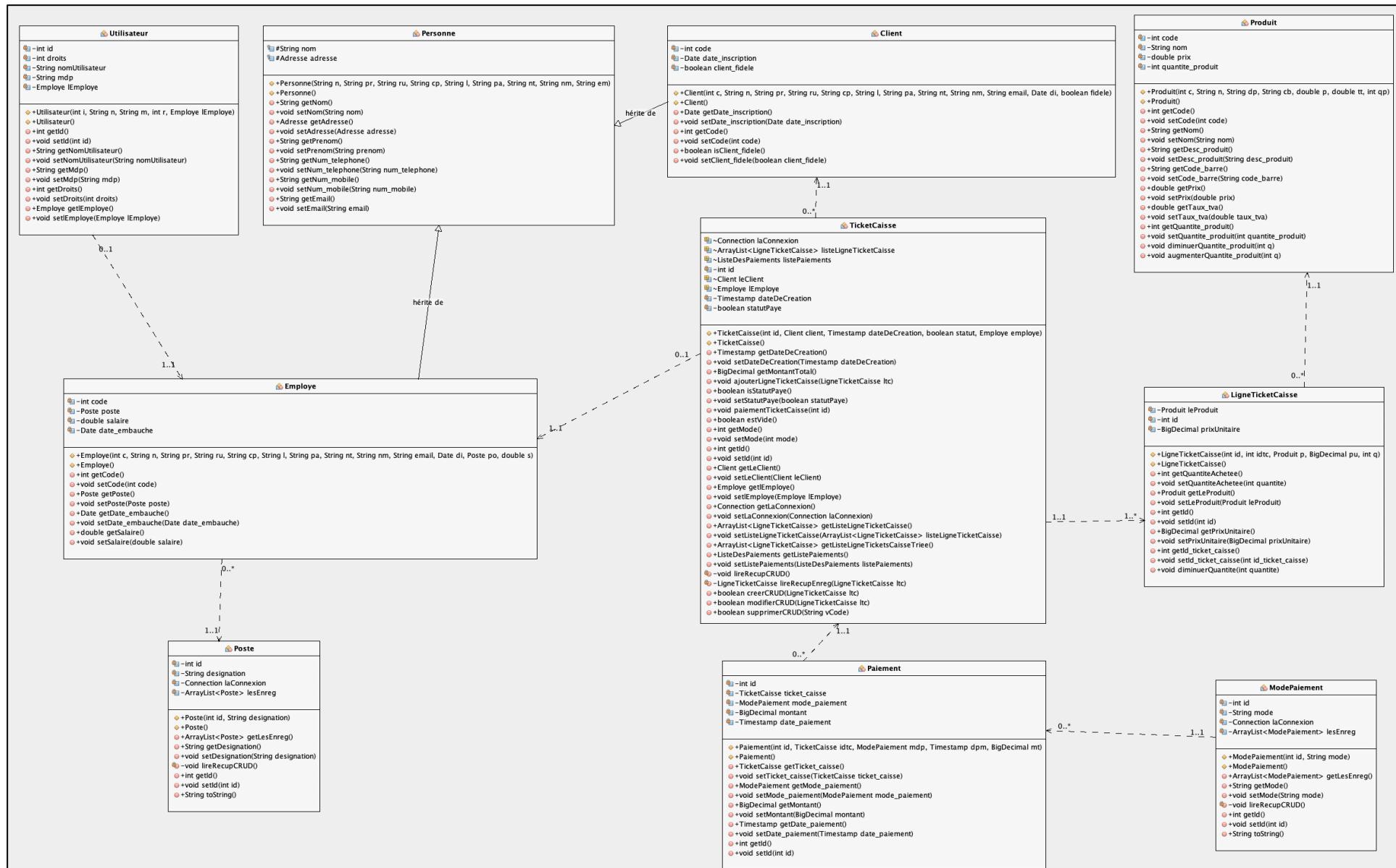
# MODELE PHYSIQUE

## *Schéma physique*



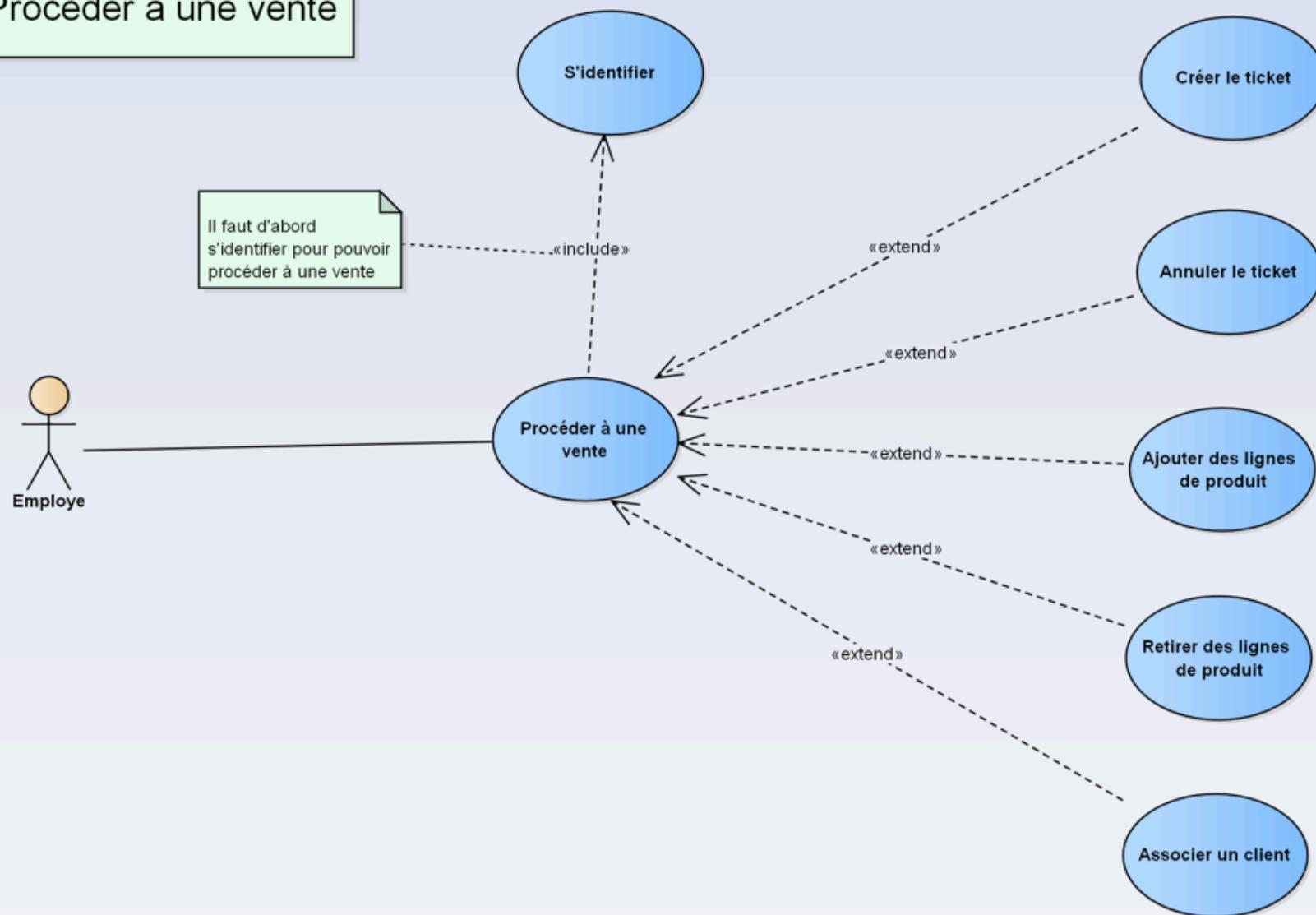
# DIAGRAMME DE CLASSES

## Schéma UML de classes



# DIAGRAMME DE USE CASE : PROCÉDER A UNE VENTE

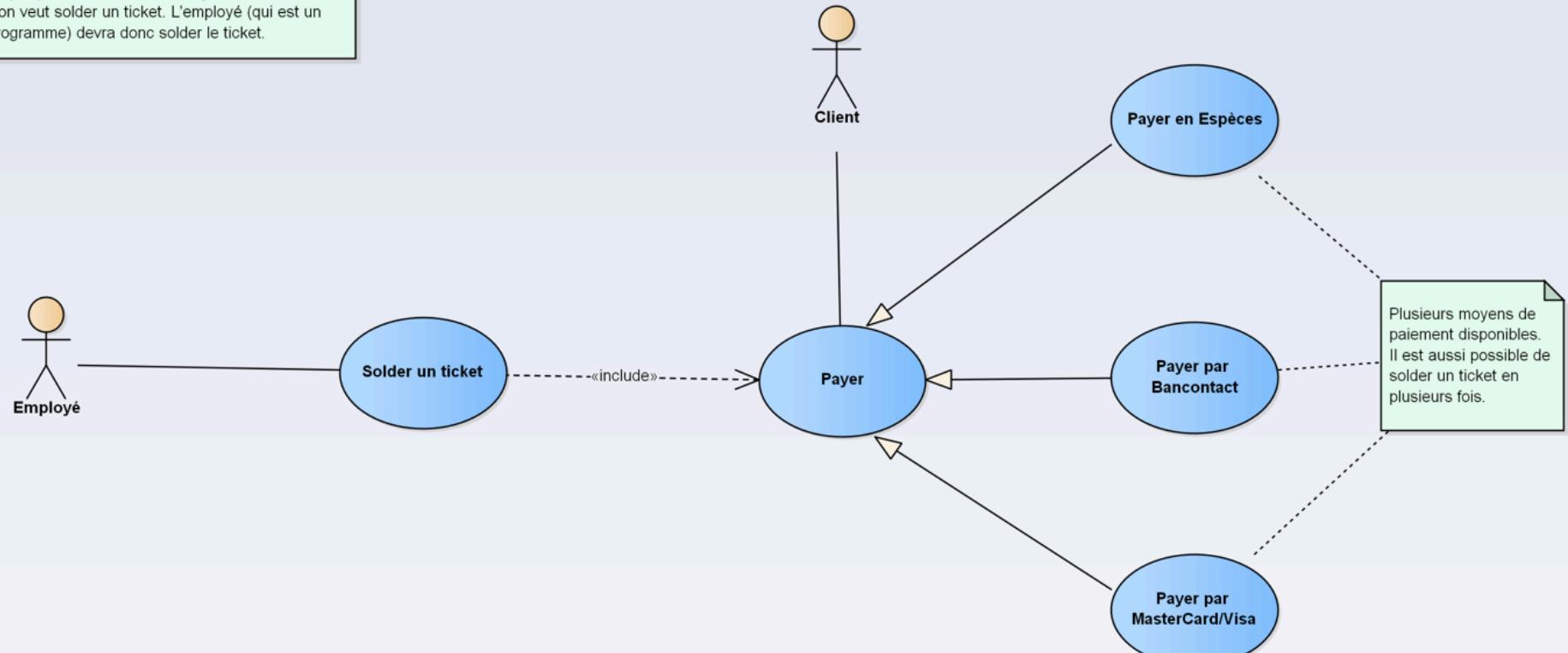
USE CASE : Procéder à une vente



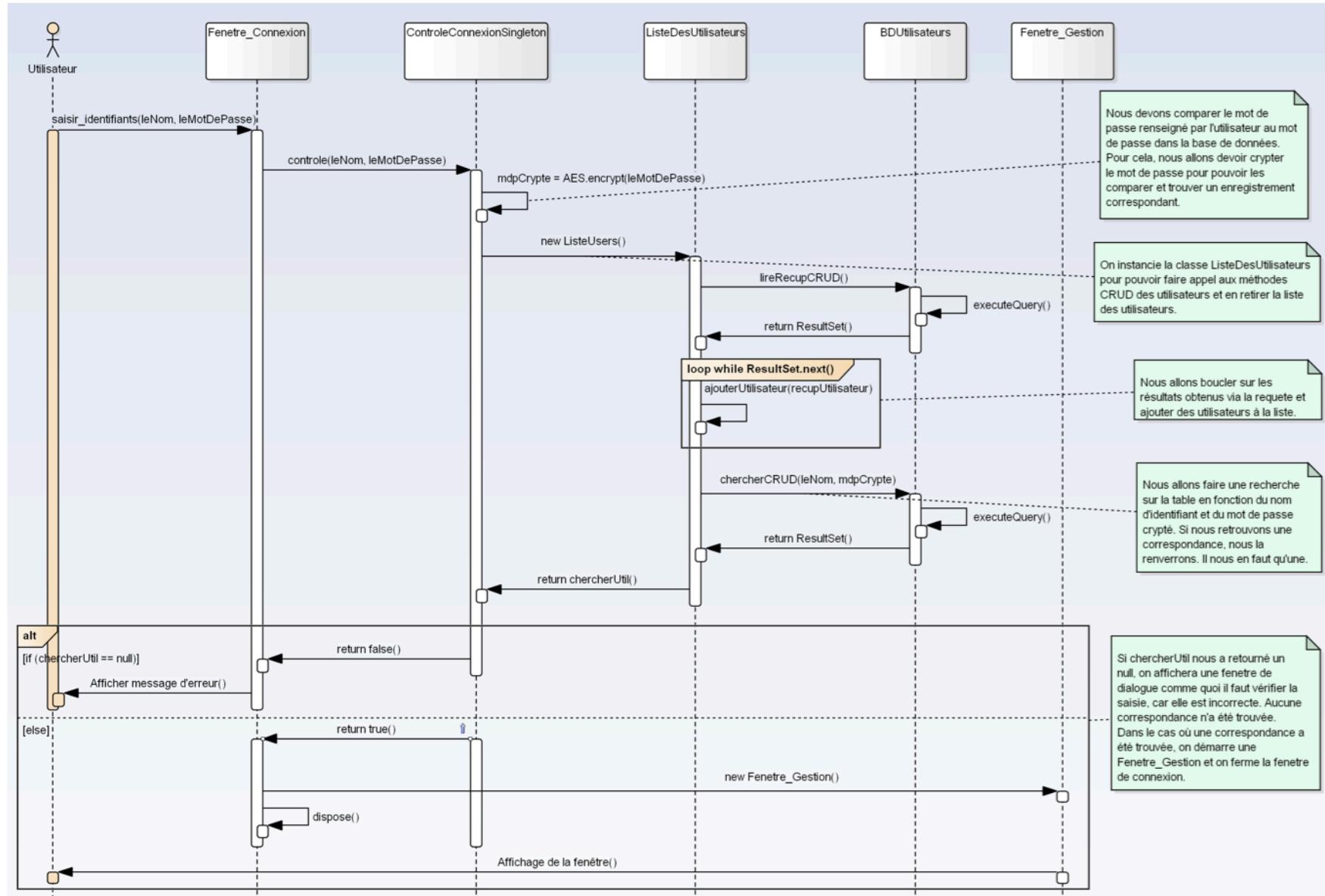
# DIAGRAMME DE USE CASE : PAIEMENT D'UN TICKET

## USE CASE : Solder un ticket

Ici nous omettons le fait de devoir s'authentifier au programme. On part du principe qu'on se trouve dans le gestionnaire de tickets et que l'on veut solder un ticket. L'employé (qui est un utilisateur du programme) devra donc solder le ticket.

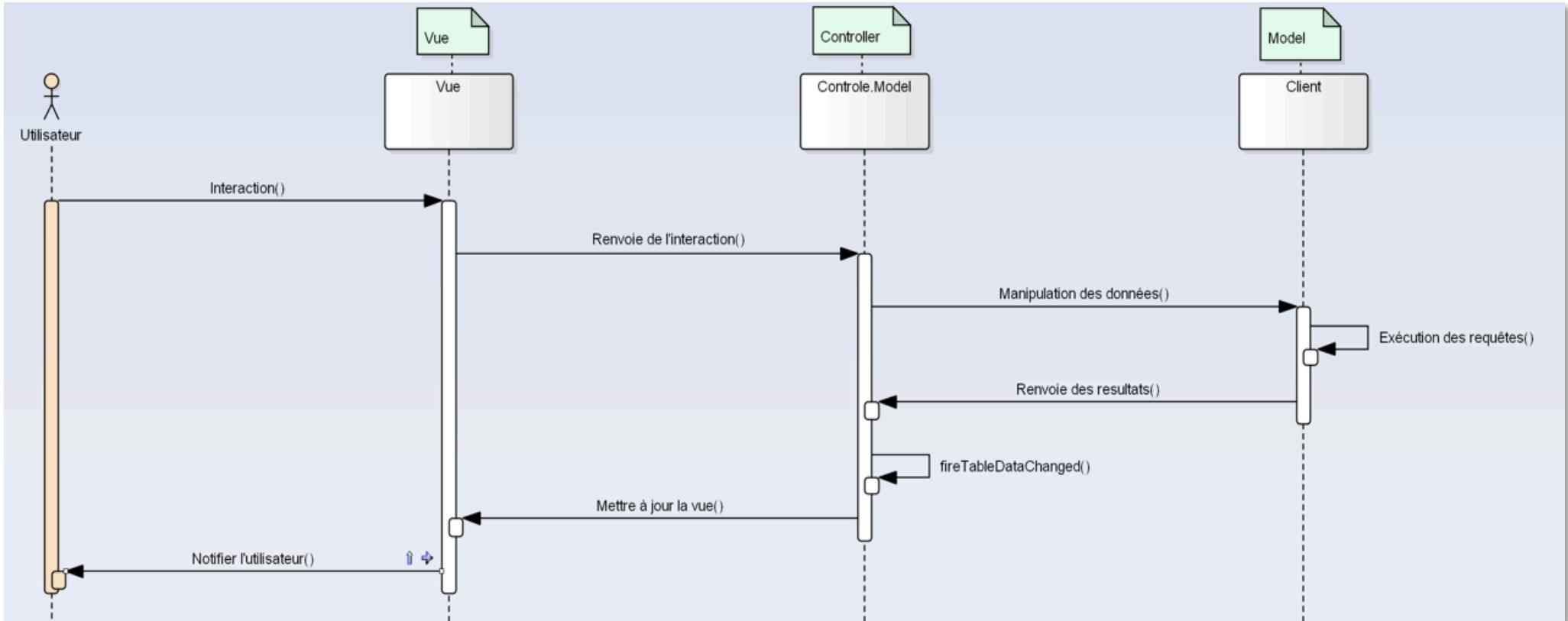


# DIAGRAMME DE SEQUENCES : AUTHENTIFICATION



# DIAGRAMME DE SEQUENCES : MVC

*Exemple Pattern MVC*



# APPLICATION

Pour les besoins du programme, j'affiche les identifiants de connexion permettant de se connecter au programme de gestion. Ceux-ci sont déjà présents dans la base de données.

Utilisateur	Mot De Passe	Droits
admin	admin	2
util1	util1	1
util2	util2	1
util3	util3	0
util4	util4	0

Au lancement du programme, une tentative de connexion vers la base de données est effectuée grâce à la classe **ControleConnexionSingleton**.

Cette classe ira chercher les paramètres de connexion stockés dans un fichier crypté appelé '**parametres.enc**' au sein du dossier '**files**'.

Ce fichier peut être généré par un administrateur en se rendant dans la fenêtre de paramètres, qu'il pourra apercevoir en appuyant sur le bouton '**Paramètres**' dans le programme de gestion.



Ceci générera un fichier crypté qui permettra la connexion vers la base de données.  
Le procédé de cryptage est expliqué plus bas.

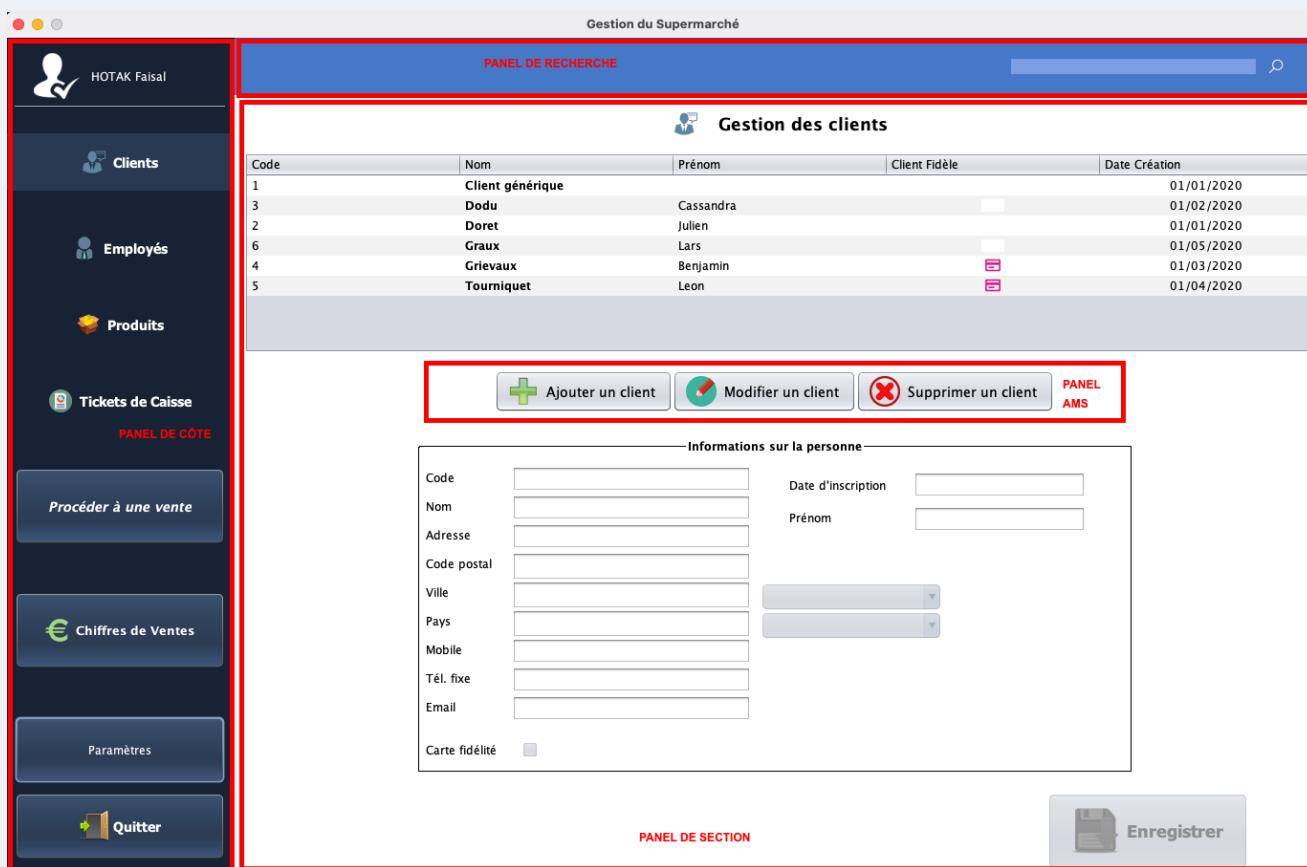
Une fois la tentative de connexion réussie, la fenêtre qui sera affichée est la suivante.



Ici, nous devrons remplir les champs avec les identifiants de connexion présents dans la base de données. Il faut savoir que les mots de passe stockés dans la base de données sont aussi cryptés.

Une fois les identifiants rentrés et la connexion réussie, une fenêtre de gestion apparaîtra. La fenêtre peut varier selon les droits de l'utilisateur connecté.

Niveau de droits	Que peut-on faire ?
0	Consultation des données
1	Ajout et modification de données
2	Ajout, modification et suppression de données Gestion des paramètres de connexion Gestion des utilisateurs



La fenêtre est divisée en plusieurs panels.

En haut nous avons un panel de recherche, permettant de filtrer les données selon la recherche entrée.

Sur la gauche nous avons le panel regroupant toutes les sections pouvant être accédées par l'utilisateur.

Au milieu nous retrouvons le contenu de chaque section.

## Contenu des sections

Tout en haut nous retrouvons une jTable contenant les données de ladite section.

Code	Nom	Prénom	Client Fidèle	Date Crédit
1	Client générique			01/01/2020
3	Dodu	Cassandra		01/02/2020
2	Doret	Julien		01/01/2020
6	Graux	Lars		01/05/2020
4	Grievaux	Benjamin	✉	01/03/2020
5	Tourniquet	Leon	✉	01/04/2020

En sélectionnant un élément de cette table, le formulaire se remplira automatiquement avec les données de l'élément. C'est grâce aux méthodes CRUD contenues dans les classes dites « Model ». Elles effectuent des requêtes vers la base de données et le contrôleur s'occupera de la mise en forme, l'affichage, le contrôle de ces informations vers les fenêtres affichées à l'utilisateur, que l'on appelle la « Vue ». Ce pattern s'appelle le « MVC ».

**Informations sur la personne**

Code	5	Date d'inscription	2020-04-01
Nom	Tourniquet	Prénom	Leon
Adresse	Avenue médiocre 15		
Code postal	7500		
Ville	Tournai		
Pays	Belgique		
Mobile			
Tél. fixe	069.45.58.47		
Email	l.tourniquet@gmail.com		
Carte fidélité	<input checked="" type="checkbox"/>		

Selon les droits que possède l'utilisateur, il lui sera possible d'ajouter, modifier ou supprimer des éléments de la table.



*Remarque :*

Lors d'un ajout ou d'une modification, il ne sera pas possible d'effectuer d'autres manipulations telles que changer de section, manipuler des données, ...



## Procéder à une vente

En appuyant sur le bouton ‘Procéder à une vente’, cette fenêtre s’affichera.

Date générée automatiquement lors de l'ouverture de cette fenêtre

# Vente

Date : 02/02/2021 (01:05:54)

Client : Client générique

Récupération des clients grâce à la liste des clients

Récupération du tableau de produits

Code	Nom	Code Barres	Prix	Taux TVA	Stock
9	Aiguillettes poulet ...	05400141051858	12.99 €	6.0 %	20
1	Barre chocolat noir	05400141051858	0.8 €	6.0 %	20
2	Cookies biscuit cho...	05400141307030	0.65 €	6.0 %	20
3	Essuie-tout	05400141051394	2.69 €	6.0 %	20
5	Frites surgelées	05400141237702	0.99 €	6.0 %	20
6	L&M volume tabac	05400141051411	5.0 €	21.0 %	20
10	Mélange de légume...	05400141054545	8.99 €	6.0 %	20
7	Pain spécial de cam...	3564700756218	1.99 €	6.0 %	20
4	Salami sans ail	05400141242973	2.29 €	6.0 %	20

Ajouter

Quantité Achetée	Nom du Produit	Prix Unitaire	Total
2	Aiguillettes poulet panées Sweet...	13.77 €	27.54 €
5	Cookies biscuit chocolat	0.69 €	3.45 €
10	Saumon	1.05 €	10.50 €

Création d'un tableau qui contiendra les produits ajoutés

PANIER DE PRODUITS

Retirer

TOTAL : 41.49 €

Un champ calculé permettant d'avoir un suivi du montant total

Le ticket sera créé et placé dans la section des tickets pour pouvoir le solder

Passage à la caisse

Annuler

### Remarques :

Le prix unitaire dans le panier est un champ automatiquement calculé : (prix \* (1 + taux tva)) du produit.

Le stock des différents produits n'apparaîtra pas modifié après l'ajout dans le panier. Cependant, le programme vérifiera la quantité dans le panier et le stock initial disponible. Il ne pourra pas être possible de dépasser cette limite.

De plus, il n'apparaîtra jamais des mêmes lignes de produit dans le panier. Si le produit que l'on désire ajouter est déjà présent dans le panier, on augmentera simplement la quantité du produit qui s'y trouve déjà. Cela nous permettra d'éviter d'avoir des lignes de même produit dans un ticket de caisse.

Une fois le ticket accepté, le programme nous redirigera automatiquement vers la section des tickets et nous sélectionnera le ticket en question fraîchement ajouté.

**Gestion des tickets**

La liste des tickets enregistrés

Numéro du Ticket	Date du Ticket	Client Associé	Employé Associé	Payé En Totalité ?	Montant Total	Montant Payé	Montant Restant
1	02/02/2021 (01:0...)	Client générique	Hotak Faisal		41.49 €	0.00 €	41.49 €

Supprimer un ticket

Les lignes de produit associés au ticket sélectionné. Elle s'adaptera au ticket sélectionné et se mettra à jour.

**Lignes du ticket de caisse**

Quantité Achetée	Nom du Produit	Prix Unitaire	Total
2	Aiguillettes poulet panées Sweet&Chili	13.77 €	27.54 €
5	Cookies biscuit chocolat	0.69 €	3.45 €
10	Saumon	1.05 €	10.50 €

Permet l'impression au format PDF du ticket de caisse

Générer le ticket

Solder

Paiement en une ou plusieurs fois du ticket.

Lors de la sélection d'un ticket, le tableau en dessous affichera les lignes de produit contenues au sein du ticket.

Il sera possible de payer un ticket en plusieurs fois. Lors du paiement, vous aurez cette fenêtre de dialogue.



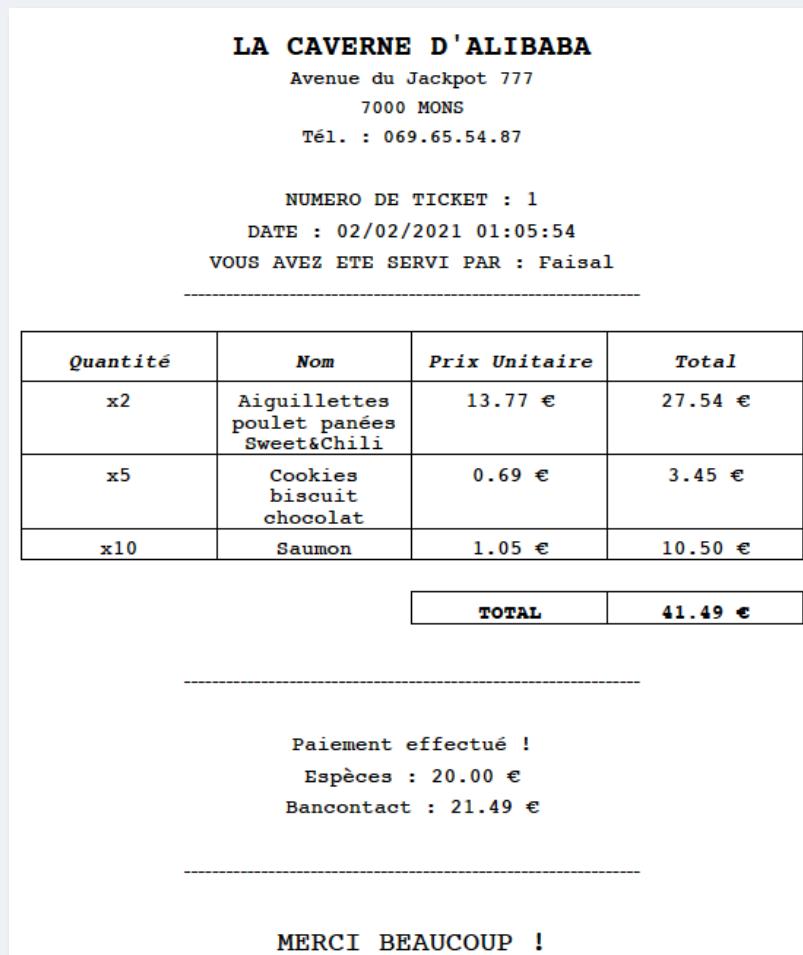
#### Remarques :

Il sera possible de payer en plusieurs fois à l'aide de plusieurs modes de paiement.  
On ne peut dépasser le montant restant à payer et l'on ne peut entrer un montant négatif ou nul.

Une fois le ticket payé dans son entièreté, les champs du ticket seront mis à jour comme ci-dessous, nous montrant que le ticket a bien été payé en totalité et qu'il ne reste aucun montant à payer.

Payé En Totalité ?	Montant Total	Montant Payé	Montant Restant
	41.49 €	41.49 €	0.00 €

Il sera aussi possible de générer un document PDF contenant toutes les informations du ticket en question. Le document généré ressemblera à cela.



Le fichier PDF généré aura comme nom ‘Ticket\_XXX’ -> XXX sera le numéro du ticket. Il sera placé dans le fichier ‘**temp**’ à la racine. A la génération, le programme par défaut pour les PDF sera lancé.

Nous avons tout d’abord les informations propres à l’établissement en en-tête.

Ensuite nous y retrouvons le numéro de ticket, la date à laquelle le ticket a été créé et l’employé ayant encodé ce ticket.

La liste des produits achetés sous forme d’un tableau pour plus de clarté.

Ce ticket est donc assez détaillé dans l’ensemble puisqu’on y trouve aussi les différents modes de paiement et leurs montants respectifs.

*Remarques :*

Si le ticket n’est pas payé dans son entièreté, le ticket qui en sort mentionnera le message ci-dessous.

En attente de paiement ...

## Chiffres de ventes

Une fenêtre apparaîtra lorsque l'on appuiera sur le bouton 'Chiffres de ventes'.

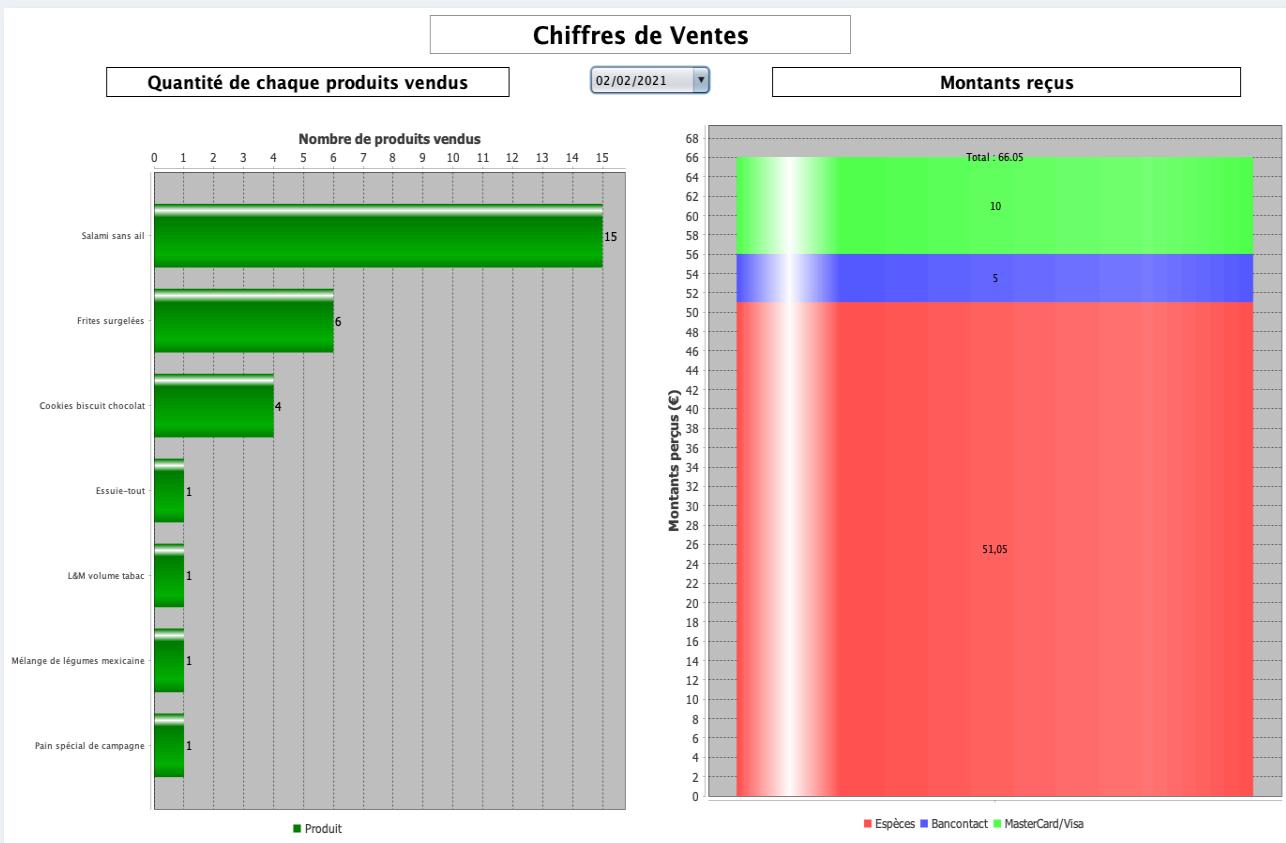
### Chiffres de Ventes

Quantité de chaque produits vendus

02/02/2021

Montants reçus

En déroulant la Combo Box, on pourra sélectionner une date, un jour précisément et en afficher les statistiques de ventes.



#### Remarques :

Les produits qui sont contenues dans des tickets non payées en entier s'affichent aussi ici dans ce diagramme. Je pars du principe que les produits ont été vendus mais les paiements n'ont pas été reçus dans leur entier.

Le diagramme des montants reçus reprendra donc tous les paiements effectués aux dates du ticket correspondant à notre choix.

## Gestion des paramètres

Voici la fenêtre de paramètres, accessible que par un utilisateur avec un niveau de droits équivalent à ‘2’.

 **Paramètres**

Permet l'affichage des mots de passe en décryptés

**Gestion des utilisateurs**

Liste des utilisateurs

Id	Utilisateur	Mot De P...	Droits	Nom	Prenom
1	admin	py2eKTz...	2	Hotak	Faisal
2	util1	1K60Sb5...	1	Durif	Sylvain
3	util2	o28rnAy...	1	Chmonfils	Thierry
4	util3	PIEB8WO...	0	Adi	Jacques
5	util4	8mM+8K...	0	Non	Mehdi

 **Ajouter**    **Modifier**    **Retirer**

 Nom d'utilisateur:

 Mdp utilisateur:

 Priviléges:

 Employé:

Paramètres de connexion à la base de données

**Paramètres de connexion à la base de données**

 Nom admin:   
 Mdp admin:

 Driver SGBD:   
 Serveur BD:

Génère un fichier 'paramètres.enc' avec les paramètres rentrés ci-dessus.

 **Générer**

 **Quitter**

Mentionné plus haut dans le rapport, il est possible de générer un fichier crypté avec les paramètres renseignés dans la partie de droite.

L’utilisateur privilégié avec ce niveau de droits aura la possibilité d’ajouter, modifier ou supprimer des utilisateurs. Un petit bouton a été mis à disposition pour pouvoir afficher les mots de passe en décryptés.

### Remarques :

Le nom d’utilisateur est unique donc il ne peut pas y avoir des noms d’utilisateur identiques.

Un employé ne peut avoir qu’un seul nom d’utilisateur. Lors de la création d’un utilisateur, il nous sera obligatoirement demandé de choisir un employé auquel il sera relié.

# LIBRAIRIE JFREECHART



Lien : <https://www.jfree.org/jfreechart/>

C'est une librairie java totalement gratuite qui permet de faire des graphiques.

*Exemples :*



```
// Pour créer un tableau de données sur lequel le graphique se basera
DefaultCategoryDataset dcd = new DefaultCategoryDataset();

// On passe sur les éléments du HashMap et on les ajoute aux données du graphique
for (String i : listeProduitsVendus.keySet()) {
    dcd.setValue(listeProduitsVendus.get(i), "Produit", i);
}

// Nous créons un type de graphique, ici un « BarChart » qui est un diagramme en bâtons
JFreeChart jchart = ChartFactory.createBarChart("", "", "Nombre de produits vendus", dcd, PlotOrientation.HORIZONTAL, true, true, false);

// Ici le PlotOrientation.HORIZONTAL
// signifie que le diagramme sera en horizontal

// Pour les catégories
CategoryPlot plot = jchart getCategoryPlot();
plot.setRangeGridlinePaint(Color.black);
plot.getRenderer().setSeriesPaint(0, new Color(0, 125, 0));
```

```

// On ajoute les catégories aux axes
CategoryAxis domainAxis = plot.getDomainAxis();
domainAxis.setTickLabelFont(new Font("Verdana Helvetica Arial", Font.PLAIN,
9));
domainAxis.setCategoryLabelPositions(CategoryLabelPositions.STANDARD);

// Pour générer les labels de catégorie
StandardCategoryItemLabelGenerator generator = new
StandardCategoryItemLabelGenerator();

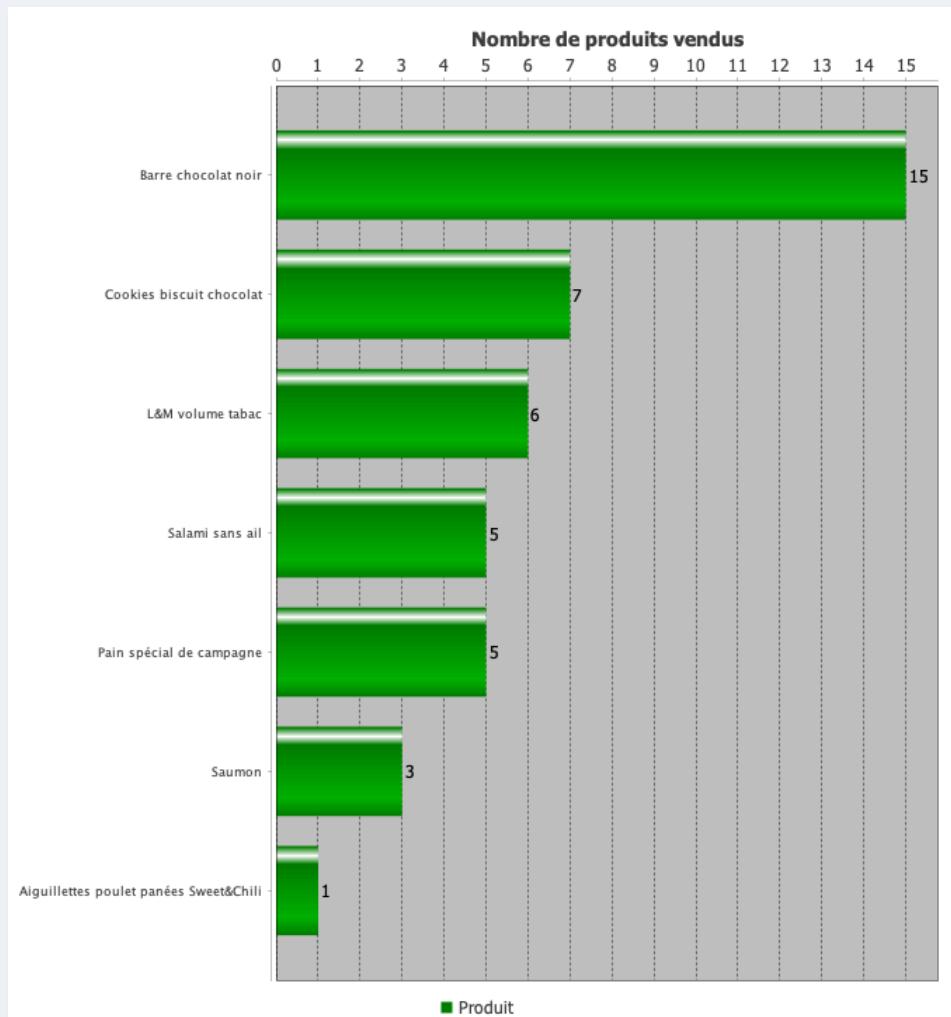
// Crédit d'un panel pour graphiques qui contiendra le graphique
ChartPanel chartPanel = new ChartPanel(jchart);
chartPanel.setPreferredSize(new
java.awt.Dimension(jPanel_Produits.getWidth(), jPanel_Produits.getHeight()));
chartPanel.setSize(new
java.awt.Dimension(jPanel_Produits.getWidth(), jPanel_Produits.getHeight()));

// On ajoute le panel pour graphiques au panel de la JFrame
jPanel_Produits.add(chartPanel);

// On met à jour la vue du panel
jPanel_Produits.updateUI();

```

Résultat :



Pour le graphique à barres empilées, le principe reste le même.

# METHODE DE CRYPTAGE/DECRYPTAGE

En faisant mes recherches, j'avais au début pensé à utiliser le **DES** (Data Encryption Standard), mais il est aujourd'hui considéré comme un algorithme très peu sécurisé. J'ai alors décidé d'utiliser l'**AES** (Advanced Encryption Standard) qui est un algorithme de cryptage aussi utilisé par les États-Unis pour crypter des documents. Il est considéré plus sécurisé que le DES.

## *Méthode setKey()*

```
public static void setKey(String myKey){  
    MessageDigest sha = null;  
  
    try {  
        key = myKey.getBytes("UTF-8");  
        sha = MessageDigest.getInstance("SHA-1");  
        key = sha.digest(key);  
        key = Arrays.copyOf(key, 16);  
        secretKey = new SecretKeySpec(key, "AES");  
    }  
    catch (NoSuchAlgorithmException e){  
        System.out.println("Algorithme du message digest incorrect !");  
        e.printStackTrace();  
    }  
    catch (UnsupportedEncodingException e){  
        System.out.println("Format d'encodage incorrect !");  
        e.printStackTrace();  
    }  
}
```

La classe **MessageDigest** nous permettra de hasher la clef passée en paramètre. Nous allons utiliser l'algorithme « SHA-1 ».

Cette clef se trouve en chaîne de caractères, à l'état brut, dans le fichier ‘clef.txt’ au sein du dossier ‘files’.

## *Pourquoi « SHA-1 » et pas « SHA-2 » ou « SHA-256 » ?*

Tout simplement car elle nous fournira un hashage moins complexe avec un code plus petit. Nous n'avons pas besoin de plus gros pour les besoins de notre TP.

Pour pouvoir faire appel à la méthode **digest()** du **MessageDigest**, il nous faut un tableau de byte. C'est donc pour ça que l'on va convertir la clef en un tableau de byte grâce à la méthode **getBytes()**. On passera le format unicode « UTF-8 » pour la formalisation.

Nous allons donc faire un **Arrays.copyOf** en lui donnant la clef (qui est un tableau de byte) et la taille que l'on veut, 16 dans notre cas. Cette méthode va donc retourner ce même tableau mais avec la taille modifiée. Si la taille donnée est plus petite que la taille d'origine, elle sera tronquée, si par contre la taille donnée est plus grande, nous aurons des **zéros** dans les nouveaux indices.

Donc maintenant nous allons finalement crypter notre clef hashée en « AES », nous allons utiliser la classe **SecretKeySpec** et lui faire passer la clef de taille 16 (128 bits) que l'on a récupéré avec la méthode au-dessus et notre algorithme de cryptage qui est « AES ».

*Nous voilà avec notre clef cryptée !*

## Méthode encrypt()

```
public static String encrypt (String strToEncrypt, String secret){  
    try{  
        setKey(secret);  
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");  
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);  
        return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));  
    }  
    catch (Exception e){  
        System.out.println("Erreur lors du cryptage : ");  
        e.printStackTrace();  
    }  
    return null;  
}
```

Pour crypter une chaîne de caractères, nous allons appeler la méthode **encrypt()** en lui passant comme arguments la chaîne voulue et la clef qui permettra de crypter/décrypter cette chaîne. La clef va en premier lieu se faire crypter grâce à la méthode **setKey()**.

La classe **Cipher** sera utilisée. Cette classe a une fonctionnalité de cryptographie qui permet de crypter/décrypter une chaîne (sous forme de tableau de byte).

La méthode **getInstance()** sur le **Cipher** permettra de lui faire passer la **transformation** voulue.

### Qu'est-ce que la transformation ?

La transformation est une chaîne lui décrivant l'opération à exécuter, cette chaîne est le nom d'un algorithme de cryptographie.

Ex. : Cipher c = Cipher.getInstance(« **AES/ECB/PKCS5Padding** ») ;

Nous avons donc notre objet cipher, maintenant nous devons passer cet objet en mode cryptage. Le cipher a plusieurs modes possibles, nous aurons besoin du « **ENCRYPT\_MODE** » pour crypter. On va le passer dans ce mode et il aura aussi besoin de la clef cryptée.

La dernière étape consiste à crypter la chaîne voulue grâce au **cipher**. Pour cela, la méthode **doFinal()** sur le **cipher** procédera au cryptage. On doit lui passer comme arguments, la chaîne désirée au format d'un tableau de byte, ce que l'on fera avec la méthode **getBytes()**.

On se retrouvera avec un code crypté contenant des symboles non lisibles par les éditeurs. Pour cela, on peut forcer le cipher à nous remettre un code avec un pattern du type « A-Za-z9+ ». Nous allons donc utiliser la classe **Base64**, avec la méthode **getEncoder().encodeToString()** pour retrouver le code crypté sous forme de chaîne avec le pattern du Base64.

*Nous voilà avec une chaîne cryptée !*

## Méthode decrypt()

```
public static String decrypt (String strToDecrypt, String secret){
    try{
        setKey(secret);
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
    }
    catch (Exception e){
        System.out.println("Erreur lors du décryptage");
        e.printStackTrace();
    }
    return null;
}
```

Nous allons procéder comme la méthode **encrypt()**, sauf que l'on passera le **cipher** en mode « **DECRYPT\_MODE** » pour déchiffrer.

Pour retourner le code déchiffré en chaîne, on va utiliser donc la méthode **doFinal()** cette fois en décodant avec la méthode **getDecoder().decode()** de la classe Base64. Le tout converti en chaîne avec « `new String()` ».

**Nous voilà avec une chaîne déchiffrée !**

### EXAMPLE

```
public class TestAES {
    public static void main(String[] args) {
        String encryptedString, decryptedString;
        final String clefSecrete = lectureClefSecrete();

        String chaine = "Ceci est un test";

        encryptedString = AES.encrypt(chaine, clefSecrete);
        decryptedString = AES.decrypt(encryptedString, clefSecrete);

        System.out.println("Voici la clef : " + clefSecrete);
        System.out.println("Voici la chaîne de base : " + chaine);
        System.out.println("Voici la chaîne cryptée : " + encryptedString);
        System.out.println("Voici la chaîne déchiffrée : " + decryptedString);
    }

    private static String lectureClefSecrete(){
        String clef;

        clef = Fichier.lire("files/clef.txt");

        return clef;
    }
}
```

### RESULTAT

```
Voici la chaîne de base : Ceci est un test
Voici la chaîne cryptée : x0rXY+Rn5swKN4lFm4hHrRN3gY0SGJaBXElsri36AC0=
Voici la chaîne déchiffrée : Ceci est un test
BUILD SUCCESSFUL (total time: 0 seconds)
```

# QUELQUES EXEMPLES DU CODE SOURCE JAVA

## Ajout d'un keyListener personnalisé

```
private void ajoutListenerJTextFields(){
    // On va redéfinir des méthodes du KeyListener de l'objet keyListener
    // Cet objet sera par la suite relié aux jTextFields des formulaires
    KeyListener keyListener = new KeyListener(){
        @Override
        public void keyTyped(KeyEvent e) {
            enableSave();
        }

        @Override
        public void keyPressed(KeyEvent e) {...}

        @Override
        public void keyReleased(KeyEvent e) {...}
    };

    this.jTXT_Adresse.addKeyListener(keyListener);
    this.jTXT_CP.addKeyListener(keyListener);
    this.jTXT_CodeBarres.addKeyListener(keyListener);
    this.jTXT_Email.addKeyListener(keyListener);
    this.jTXT_Mobile.addKeyListener(keyListener);
    this.jTXT_Pays.addKeyListener(keyListener);
    this.jTXT_Prix.addKeyListener(keyListener);
    this.jTXT_Taux.addKeyListener(keyListener);
    this.jTXT_Telephone.addKeyListener(keyListener);
    this.jTXT_Ville.addKeyListener(keyListener);
    this.jTextArea_Description.addKeyListener(keyListener);
    this.jTxtNom.addKeyListener(keyListener);
    this.jTxt_NomProduit.addKeyListener(keyListener);
    this.jTxt_DateCreation.addKeyListener(keyListener);
    this.jTxt_Prenom.addKeyListener(keyListener);
    this.jTxt_Quantite.addKeyListener(keyListener);
    this.jTxt_Salaire.addKeyListener(keyListener);
}

private void enableSave(){
    if (ajout || modif) jBtn_Enregistrer.setEnabled(true);
}
```

Au lieu d'avoir une quantité astronomique de méthodes keyTyped sur les différents jTextField, je crée un objet de type KeyListener et j'y redéfinis les méthodes.

Dans notre cas, on aura besoin du keyTyped, ainsi à chaque touche entrée dans un jTextField, on dégrisera le bouton de sauvegarde, signifiant qu'un élément a été modifié et donc la sauvegarde peut s'effectuer.

J'ajoute donc ce keyListener à tous mes champs jTextField.

Ce procédé a aussi été mis en place pour les jLabels ainsi que les jTables.

## Vérifier si une fenêtre est déjà lancé

```
private void jButton_ChiffresVentesMouseReleased(java.awt.event.MouseEvent evt) {
    if (fenetre_ChiffresVentes != null) fenetre_ChiffresVentes.dispose();
    fenetre_ChiffresVentes = new Fenetre_ChiffresVentes(leModeleTicketCaisse);
    fenetre_ChiffresVentes.setVisible(true);
}
```

Ici, nous vérifierons si la fenêtre a déjà été initialisé. Si c'est le cas, on l'éteint et on en démarre une autre. Cela permet d'éviter que plusieurs fenêtres du même type ne soient présentes.

## La recherche

```
// On appuie sur ENTER pour faire une recherche selon la section
private void jTXT_RechercheKeyPressed(java.awt.event.KeyEvent evt) {
    if (evt.getKeyCode() == evt.VK_ENTER){
        String recherche = this.jTXT_Recherche.getText();

        switch (mode){
            case 'C' : leModeleClient.rechercherMOD(recherche);
                         break;
            case 'E' : leModeleEmploye.rechercherMOD(recherche);
                         break;
            case 'P' : leModeleProduit.rechercherMOD(recherche);
                         break;
            case 'T' : leModeleTicketCaisse.rechercherMOD(recherche);
                         leModeleLigneTicketCaisse.getLesDonnees().clear();
                         leModeleLigneTicketCaisse.fireTableDataChanged();
                         break;
        }
    }
}
```

La recherche est lancée une fois que l'on aura appuyé sur la touche « ENTER ». Les différents ‘mode’ ici représentent les sections dans lesquelles on pourrait se trouver. C pour Clients, E pour Employés, P pour Produits et enfin T pour Tickets de caisse.

## Éléments à cacher selon les droits de l'utilisateur connecté

```
private void initInterfaceUser(){
    switch (utilisateur.getDroits())
    {
        case 0 : jBtn_Ajouter.setVisible(false);
                   jBtn_Modifier.setVisible(false);
                   jBtn_Supprimer.setVisible(false);
                   jBtn_Parametres.setVisible(false);
                   jPanel_AMS.setBorder(null);
                   jBtn_Enregistrer.setVisible(false);
                   jBtn_Vendre.setVisible(false);
                   jButton_Solder.setVisible(false);
                   break;

        case 1 : jBtn_Supprimer.setVisible(false);
                   jBtn_Parametres.setVisible(false);
                   break;
    }

    jLabel_Profil.setText(" " +
                        utilisateur.getEmploye().getNom().toUpperCase() + " " +
                        utilisateur.getEmploye().getPrenom());
}
```

Je fais appel à cette méthode au début du programme pour pouvoir cacher certains éléments de la fenêtre aux utilisateurs ayant des droits inférieurs à ‘2’.

## Réinitialisation des champs des formulaires

```
// Réinitialise tous les champs du formulaire
private void reinitFormulaire(){
    switch (mode)
    {
        case 'C' : jTxtCode.setText("");
        jTxt_DateCreation.setText("");
        jTxtNom.setText("");
        jTxt_Prenom.setText("");
        jTxt_Adresse.setText("");
        jTxt_CP.setText("");
        jCombo_Localite.setSelectedIndex(-1);
        jCombo_Pays.setSelectedIndex(-1);
        jTxt_Ville.setText("");
        jTxt_Pays.setText("");
        jTxt_Mobile.setText("");
        jTxt_Telephone.setText("");
        checkBox_Carte.setSelected(false);
        jTxt_Email.setText("");
        break;

        case 'E' : jTxtCode.setText("");
        jTxt_DateCreation.setText("");
        jTxtNom.setText("");
        jTxt_Prenom.setText("");
        jTxt_Adresse.setText("");
        jTxt_CP.setText("");
        jCombo_Localite.setSelectedIndex(-1);
        jCombo_Pays.setSelectedIndex(-1);
        jTxt_Ville.setText("");
        jTxt_Pays.setText("");
        jTxt_Mobile.setText("");
        jTxt_Telephone.setText("");
        jTxt_Salaire.setText("");
        jTxt_Email.setText("");
        jComboBox_Poste.setSelectedIndex(-1);
        break;

        case 'P' : jTxt_CodeProduit.setText("");
        jTxt_NomProduit.setText("");
        jTextArea_Description.setText("");
        jTxt_CodeBarres.setText("");
        jTxt_Prix.setText("");
        jTxt_Taux.setText("");
        jTxt_Quantite.setText("");
        break;

        case 'T' : break;
    }
}
```

Une méthode permettant de réinitialiser les champs du formulaire. Pratique après un ajout d'un client par exemple pour libérer les champs.

## Création d'un fichier de paramètres par défaut si inexistant

```
private static String initDefaultParameters(){
    String root = "root";
    String mdp = "test";
    String driver = "com.mysql.cj.jdbc.Driver";
    String db = "jdbc:mysql://localhost:3306/Supermarche";

    String parametresDefaut = AES.encrypt(root + "\n" + mdp + "\n"
        + driver + "\n" + db);

    ecrire(parametresDefaut, "files/parametres.enc");
    return parametresDefaut;
}
```

Si le fichier de paramètres n'existe pas, on en crée un par défaut.

## **CONCLUSION**

Au cours de ce projet nous avons pu mettre en œuvre les connaissances acquises durant les séances de cours ainsi que des travaux pratiques en vue de la création d'une base de données se voulant aussi proche de la réalité que possible. Ce sujet semblait à première vue simple mais il s'est avéré être plus difficile à mettre en place en tant qu'application.

Ce projet m'aura permis de découvrir la manière dont pouvait se faire le lien entre le SGBD (MySQL) et le langage de programmation (JDBC et Java) ainsi que de nous familiariser avec ces langages.

Au final, ce projet m'aura permis de comprendre la complexité qu'engendre la gestion des bases de données : les buts recherchés, les méthodes utilisées, les problèmes, difficultés rencontrées et finalement les solutions mises en place.

## **BIBLIOGRAPHIE**

- Cours de « Projet de développement SGBD » 2020-2021 de D. PASSELECQ
- [www.jfree.org](http://www.jfree.org) – Librairie JFreeChart
- <https://howtodoinjava.com/java/java-security/java-aes-encryption-example/>  
Cryptage/Décryptage AES