C Pizza

OC_PIZZA
Vente de pizzas en ligne

Dossier de conception technique

Version **1.0**

Auteur
DERISBOURG Arnaud

C Pizza

TABLE DES MATIÈRES

C Pizza

1 - Glossaire

SGBD	Système de gestion de Base de Données
HTTPS	Hypertext Transfer Protocol Secure
UI	Interface Utilisateur
FRONTEND	Partie visible
BACKEND	Partie Logique métier
FIFO	First In First Out

2 - Versions

Auteur	Date	Description	Version
DERISBOURG Arnaud	21/09/21	Création du document	V1.0

3 - Introduction

3.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application **OC PIZZA**. Il est destiné à l'attention des développeurs, mainteneurs et de l'équipe technique d'OC PIZZA.

L'objectif de l'analyse technique étant de lister les contraintes spécifiques dont les développeurs vont devoir tenir compte pour coder l'application, le présent document présentera les langages et les conventions de développement, les conventions de développement, l'architecture logicielle et de déploiement ainsi que les logs, le monitoring propres à l'application.

- de notre premier entretien datant du 01//04//2020.
- du document de spécifications techniques réalisé par notre entreprise JAVAC++Mieux.

3.2 - Références

Pour de plus amples informations, se référer également aux éléments suivants:

1. **Projet 8 - Dossier de conception fonctionnelle.pdf – 1.0** : Dossier de conception fonctionnelle de l'application
2. **Projet 8 - Dossier d_exploitation.pdf – 1.0** : Dossier d'exploitation de l'application.
3. **Projet 8 - PV Livraison.pdf – 1.0 : Procès-verbal de livraison finale.**

4 - Architecture Technique

4.1 - Application Web

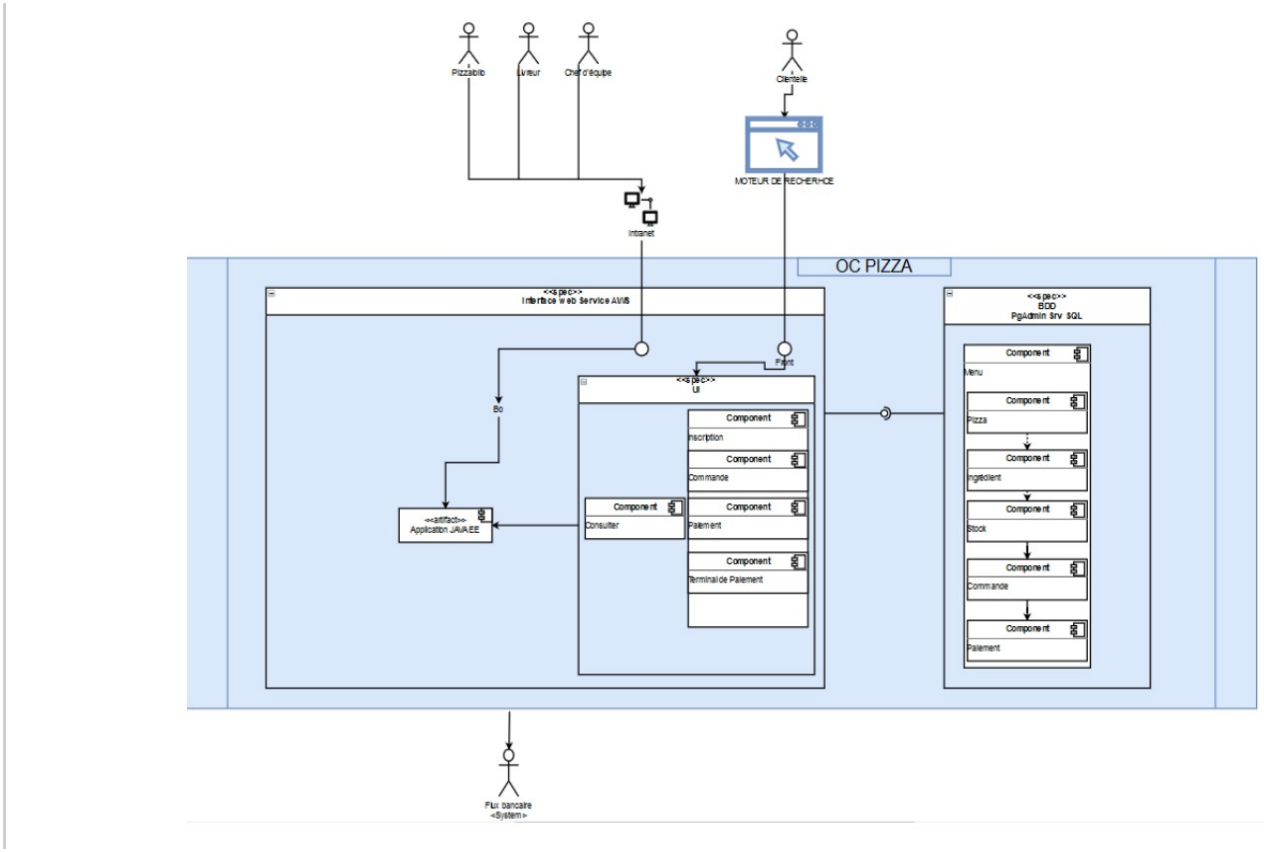
La pile logicielle est la suivante :

- Application **J2EE** (JDK version 15) / Spring
- Serveur d'application **Tomcat / aws**
-

4.2 - Application OC PIZZA

Cette application a pour but d'autonomisé la gestion interne et externe des l'entreprises notamment dans la gestion des stocks, gestion des matières première FIFO, dans le cadre de l'expansion de la pizzeria elle devra aussi permettre de listé toute les pizzerias et de les référencés, le client quant à lui auras la possibilité de choisir son mode de paiement (CB, espèce) ainsi que de choisir son type de livraison (à emporter, sur place)

5 - Architecture de Déploiement



Nous avons identifié quatre environnement spécifiques pour les utilisateurs

Artifact

Il s'agit d'ordinateurs, tablettes ou smartphones que les clients et l'équipes utilisent pour se connecter à l' application (site web).
Le personnel et les clients s'y rendent pour passer des commandes. Les l'utilisent pour passer commande sans intermédiaire. L'équipe peut passer une commande pour le client lorsque celui-ci souhaite récupérer sa commande sur place, ou lorsque celui-ci téléphone.

Les cadres utilisent l'application principalement pour gérer les pizzerias d'un point de vue financier. Les clients et l'équipe utilisent un navigateur web qui, va lancer des requêtes HTTPS, et permettre de se connecter au deuxième nœud, le serveur d'application (Application Server).

Front UI

Il s'agit du site web divisé en une partie front-end (partie visible) et une partie back-end (partie logique) celle de l'application le front-end utilise comme ressources HTML le framework bootstrap ainsi que thymeleaf, quant au back-end ce sera un projet Maven construit en langage JAVA suivant le principe MVC couplé avec les framework Spring

Le serveur de base de données

Un protocole HTTP permet au nœud serveur d'application de communiquer avec le nœud serveur de base de données qui est considéré comme un service externe.
Ce serveur héberge la base de données PgAdmin V4 dans lequel il se trouve les composants suivants **"Menu,Pizza,Commande,Ingrédiant,Stock,Paiement"**

Le serveur bancaire

Un protocole API permet au nœud serveur d'application de communiquer avec le nœud serveur bancaire qui est considéré comme un service externe. Son rôle est gérer les opérations de paiement.

5.1 - Serveur de Base de données

Description

Le SGBD est hébergé sur un serveur aws, est le SGBD est gèré par PGAdmin4 (PGSQL)

6 - Architecture logicielle

6.1 - Principes généraux

Les sources et versions du projet sont gérées par **Git**, les dépendances et le packaging par **Apache Maven**

6.1.1 - Les couches

L'architecture applicative est la suivante :

- une couche **model** : implémentation du modèle des objets métiers
- une couche **service**: responsable de la logique métier du composant
- une couche **Controller**: qui va est crée pour mettre en place l'affichage des méthode lié a la partie de la couche service
- une couche **Vue** qui va contenir toute les pages HTLM

6.1.2 - Structure des sources

La structuration des répertoires du projet suit la logique suivante :

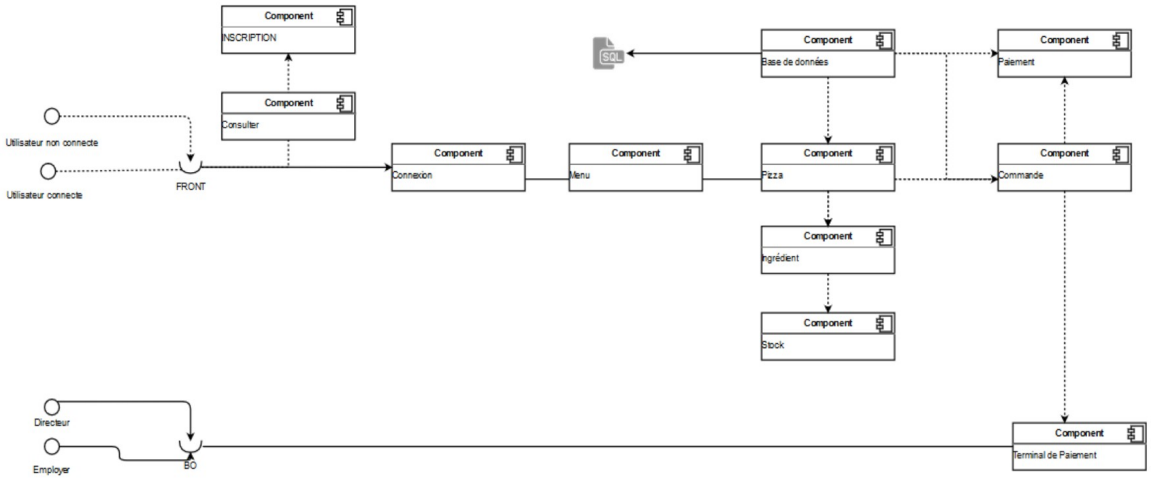
- les répertoires sources sont crées de façon à respecter la philosophie Maven (à savoir : « convention plutôt que configuration »)

```
│   .gitignore
│   com.OcPizzaTree.iml
│   HELP.md
│   mvnw
│   mvnw.cmd
│   pom.xml
├───.mvn
│   └───wrapper
│           maven-wrapper.jar
│           maven-wrapper.properties
│           MavenWrapperDownloader.java
└───src
    ├───main
    │   ├───java
    │   │   └───OcPizzaTree
    │   │       └───com
    │   │           └───OcPizzaTree
    │   │                   Application.java
    │   │                   ├───controller
    │   │                   ├───model
    │   │                   │       Adresse.java
    │   │                   │       AideMemoire.java
    │   │                   │       Client.java
    │   │                   │       Commande.java
    │   │                   │       Employee.java
    │   │                   │       Facturation.java
    │   │                   │       Ingrediant.java
    │   │                   │       LigneAdresse.java
    │   │                   │       LigneCommande.java
    │   │                   │       LigneRecettes.java
    │   │                   │       Pizza.java
    │   │                   │       PointDeVentes.java
    │   │                   │       Stock.java
    │   │                   └───service
    │   │                           AdresseService.java
    │   │                           AideMemoireService.java
    │   │                           ClientService.java
    │   │                           CommandeService.java
    │   │                           EmployeeService.java
    │   │                           FacturationService.java
    │   │                           IngrediantService.java
    │   │                           LigneAdresseService.java
    │   │                           LigneCommandeService.java
    │   │                           LigneRecettesService.java
    │   │                           PizzaService.java
    │   │                           PointDeVentesService.java
    │   │                           StockService.java
```

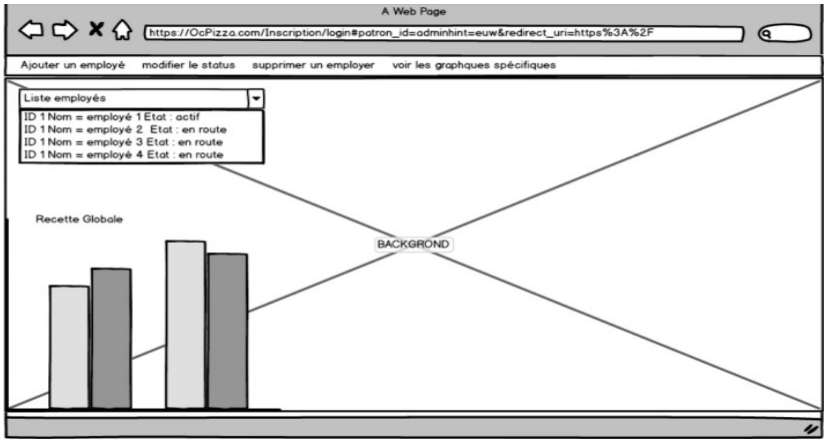
```
│   │                       └──impl
│   │                              AdresseImpl.java
│   │                              AideMemoireImpl.java
│   │                              ClientImpl.java
│   │                              CommandeImpl.java
│   │                              EmployeeImpl.java
│   │                              FacturationImpl.java
│   │                              IngrediantImpl.java
│   │                              LigneAdresseImpl.java
│   │                              LigneCommandeImpl.java
│   │                              LigneRecettesImpl.java
│   │                              PizzaImpl.java
│   │                              PointDeVentesImpl.java
│   │                              StockImpl.java
│   └──resources
│           application.properties
└──test
    └──java
        └──OcPizzaTree
            └──com
                └──OcPizzaTree
                        ApplicationTests.java
```

6.2 - Application Web

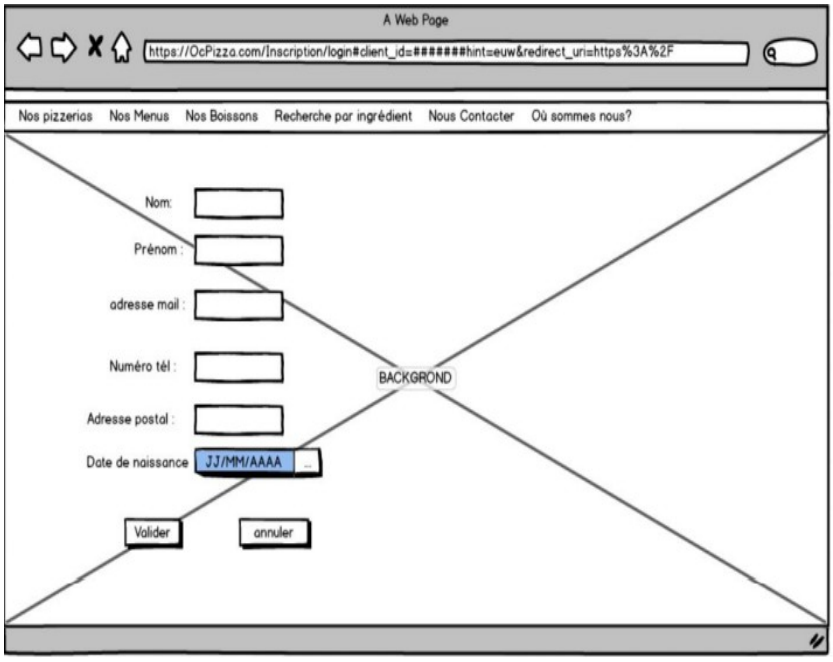
Si besoin, diagramme UML de composants pour monter les différents modules et leur inter-dépendances



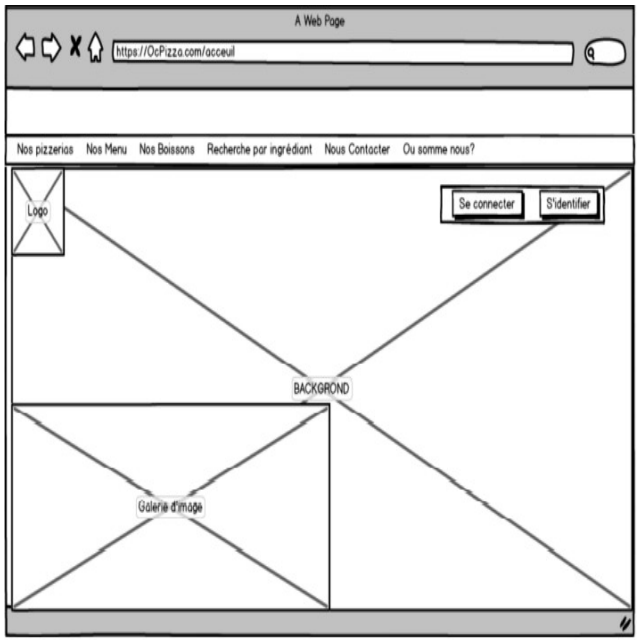
le patron de la pizzéria devra pouvoir ajouter un employé via le site web consulter la recette des différents points de vente



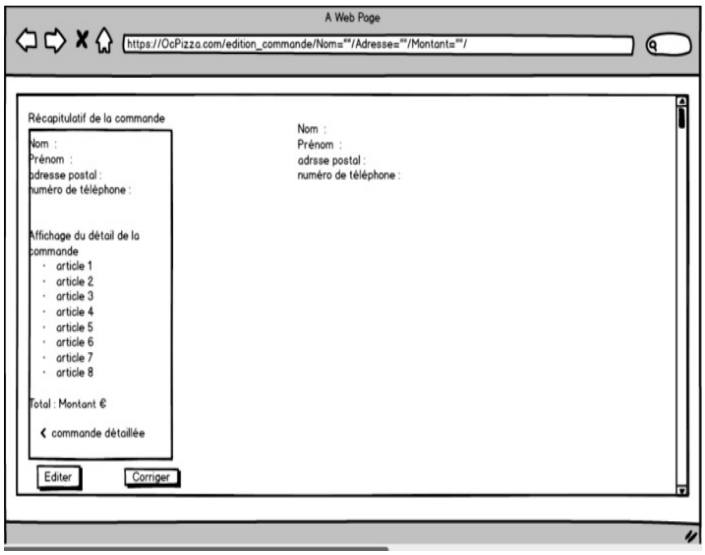
le client devra pouvoir s'inscrire sur le site via un formulaire d'inscription



En cliquant sur « s'identifier » le système prendra en compte le statut et les droits de chacun selon le préfixe de connexion par exemple pour les employés l'identifiant de connexion sera emp_Nom_ID

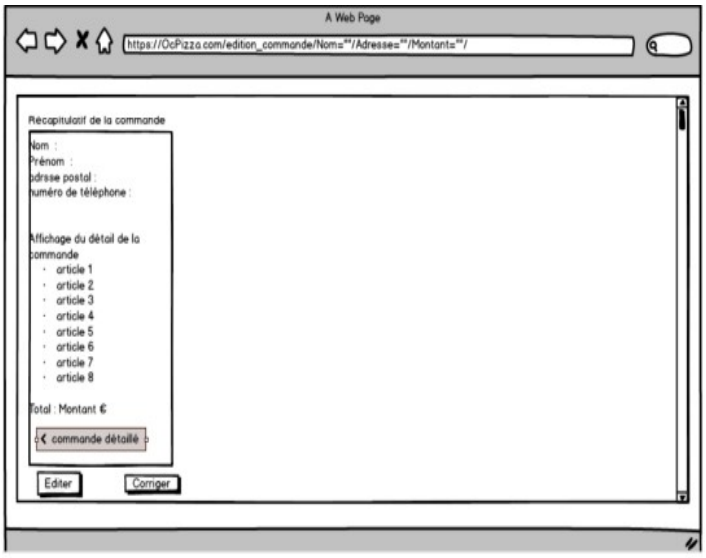


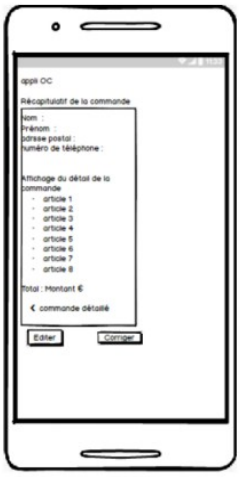
L'employé devra, quand le client passe une nouvelle commande, générer puis éditer la facture du client



Si le client choisit une livraison a emporter, l'employé disposera d'une application GPS(gratuite) pour ce rendre au domicile du client.

Une fois que le client a passé une commande et qu'elle lui a été livrée ou qu'il est venu la chercher sur place, l'employé pourra éditer la facture du client depuis le site web avec les retours d'informations liés à cette commande.






C Pizza

7 - Points particuliers

7.1 - Gestion des logs

Le DataDog AWS offre une plateforme complète de gestion et d'analyse de logs pour l'entreprise du client. Cela offre une vue sur le comportement des serveurs physiques et virtuels, et permet de prévoir d'éventuels incidents ou dysfonctionnements.

7.2 - Fichiers de configuration

7.2.1.1 - Fichier application.properites

C'est un fichier dans lequel il y a les inforamtion relative au serveur ainsi que celle de la BDD

7.3 - Ressources

application.properties

commander.html

connect.html

inscription.html

7.4 - Environnement de développement

L'utilisation d'un IDE est recommandée tel que IntelliJ, il est possible d'utiliser également l'IDE Eclipse

7.5 - Procédure de packaging / livraison

L'application fera l'objet d'un déploiement sur le serveur AWS au moment de la livraison finale. À cette occasion le dossier d'exploitation **Projet 8 - Dossier d_exploitation.pdf – 1.0** permettant de lister tout ce dont le client a besoin pour assurer la continuité de l'utilisation de l'application lui sera remis.

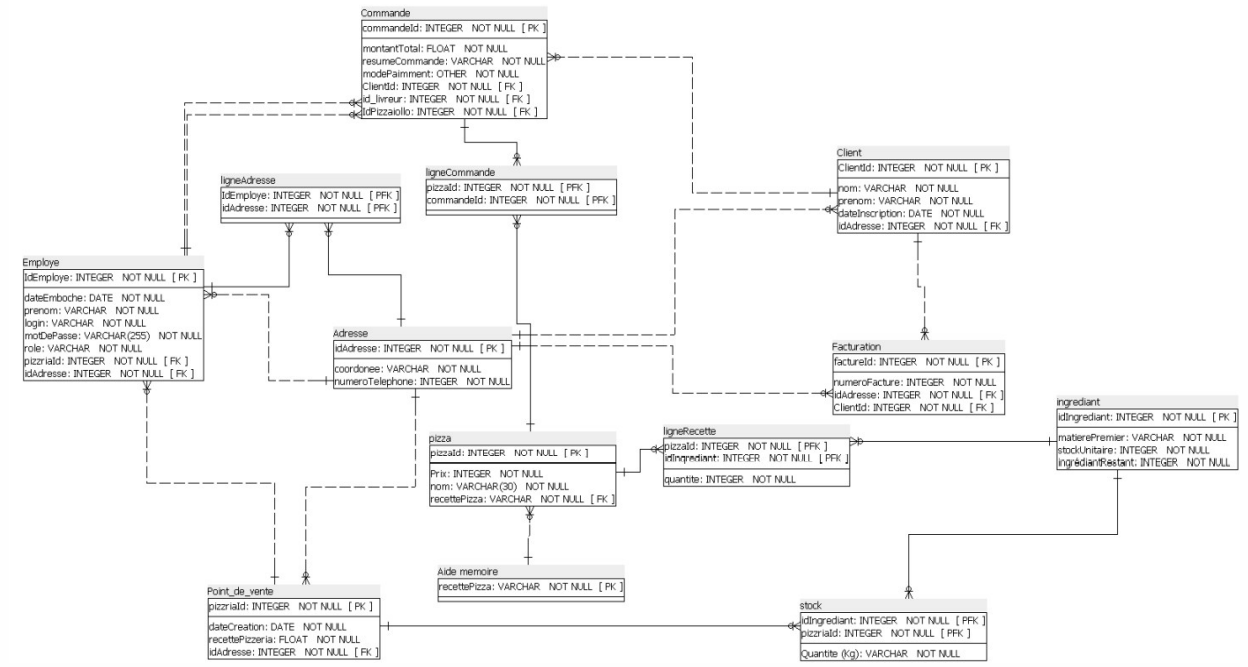
7.6 - Monitoring

Le service de monitoring d'AWS permet de suivre l'état de la machine et de déclencher automatiquement l'intervention d'un technicien dans le datacentre.

Tous les serveurs de nos clients ainsi que l'ensemble du réseau sont surveillés 24h/24 et 7j/7 par aws

AWS intervient dès le déclenchement d'une alerte (non réponse aux pings) afin de limiter au maximum le temps d'indisponibilité des serveurs et du réseau.

7.7 - MPD



La table « Employe »

La table et composé d'une PK « IdEmploye » en AUTO_INCREMENT elle est de type INT ce qui permet de différencier tout les employer par leur identifiant, a chaque fois qu'un nouvelle employer et crée « IdEmploye » augmente de 1, elle possède également 2 FK qui font références au table suivante la Table **Adresse** pour que chaque employer puisse avoir accès a l'adresse du client ainsi que la table **Commande**

pour les autres attributs le type est :

← Date date d'embauche

← VARCHAR(20) pour le prénom ainsi que pour le rôle

← VARCHAR(255) pour le mot de passe car encrypté

La table «**Point_de_vente**»

La table **Point de vente**, est composer d'une PK « pizzariaId » en AUTO_INCREMENT elle est de type INT, ainsi que 2 attributs

← dateCreation de type Date

← recettePizzaria de type FLOAT

← idAdresse FK en référence a la table **Adresse**

La table «**Commande**»

la table **Commande** est composer d'une PK commandeId en AUTO_INCREMENT elle est de type INT ainsi que 6 attribus, dont mondePaiment de type Enum qui sera composer des deux choix possible « carte bencaire » et « espèce »

← montantTotal de type Float

← resumeCommande de type VARCHAR

← modePaiment de type Enum

← (FK) ClientId de type Int

←(FK) id_livreur de type Int

←(FK) IdPizzaiollo de type Int

La table « **LigneAdresse** »

La table **LigneAdresse** est composer de 2 PFK en références a la table **employer** et **adresse**

La table « **LigneCommande** »

La table LigneCommande est composer de 2 PFK en références a la table **Pizza** et **Commande**

La table « **Pizza** »

La table **Pizza** est composé d'une PK en AUTO_INCREMENT NOT NULL ainsi que 2 attribut et une FK en référence a la table aideMemoire

C Pizza

La table « aide memoire »

est une table standalone qui a une PK recettePizza de type VARCHAR et qui est NOTNULL

La table « LigneRecette »

est une table qui contient 2 PFK

- ← PizzaId de type INT NOT NULL en références a la table Pizza
- ← idIngredient de type INT NOT NULL en références a la table Ingrédiant

La table « ingrédient »

est composé d'une PK et de 3 attributs

- ← idIngredient de type INTEGER en AUTO_INCREMENT NOT NULL
- ← matierePremier de type VARCHAR NOT NULL
- ← stockUnitaire de type INTEGER NOT NULL
- ← ingrédiantRestant de type INTEGER NOT NULL

La table « stock »

est composer de PFK et d'un attribut

- ← idIngrédient INTEGER NOT NULL en références a la table Ingrédient
- ← pizzariaId INTEGER NOT NULL en références a la table Pizzeria
- ← Quantité VARCHAR NOT NULL

La table « facturation »

est composer d'une PK, d'un attribut et de deux FK

- ← factureId INTEGER NOT NULL AUTO_INCREMENT
- ← numéroFacture INTEFER NOT NULL