OC Pizza spécifications techniques

OC Pizza

# Introduction.

Ce document constitue les spécifications techniques du projet OC Pizza. Le commanditaire veut gagner en efficacité en optimisant le processus de gestion des commandes. Il souhaite un site Web pour pouvoir effectuer, modifier, payer et suivre une commande.

Il veut suivre en temps réel l'évolution des commandes et celui du stock. Les préparateurs doivent avoir à disposition les recettes des pizzas.

La première partie constitue la description du domaine fonctionnel et des différentes entités utilisées dans le projet.

La seconde partie détaille les relations entre les différents composants internes ou externes du système. En effet dans une base de données relationnelle, les données sont regroupées par concept dans des tables et les concepts sont liés les uns aux autres par des relations.

La dernière partie décrit le déploiement du système.

# Le domaine fonctionnel.

On doit identifier les éléments et les informations que l'on veut enregistrer dans notre base de données pour que cela forme un système cohérent, c'est le domaine fonctionnel. On utilisera dans notre étude une base de données relationnel car il y a des relations entre les différentes parties qui composent notre domaine fonctionnel.

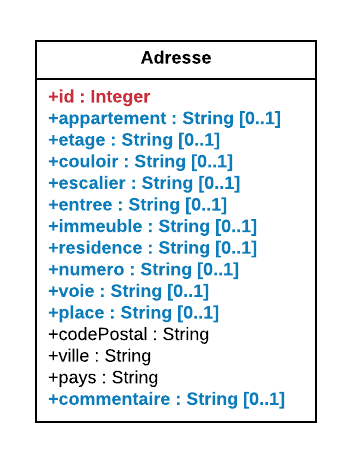
On utilise une approche orientée objet pour représenter les composants de notre domaine fonctionnel. Le diagramme de classes servira de base à la modélisation de celui-ci. Le diagramme de classes fait partie des diagrammes qui respectent la norme de modélisation graphique UML (Unified Modeling Language).

Chaque objet de notre domaine fonctionnel peut être soit réel (un client, un produit...) ou abstrait (commande, stock…). Il possède un identifiant unique et des attributs.

## Les composants généraux.

On utilise un composant spécifique pour gérer les adresses car des "Utilisateur" peuvent avoir la même adresse et cela évite les redondances de données enregistrées. Les champs sont basés sur la nomenclature de la Poste. Le composant "Magasin" permet d'identifier un magasin et ses coordonnées.

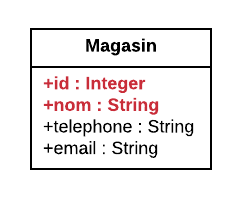
### Adresse



La plupart des champs ne sont pas obligatoire comme "voie" car une personne peut habiter sur une "place" et inversement. On ajoute une ligne de commentaire dans l'adresse pour ajouter des informations comme un code de digicode.

### Magasin

Un objet "Magasin" permet de classer les différents magasins. On utilise un attribut "id" pour avoir un identifiant unique qui sert de référence et on spécifie aussi que le "nom" est un identifiant pour faire des recherches plus rapidement.

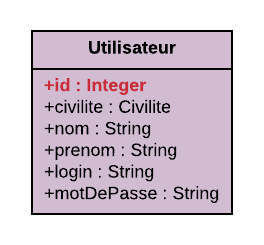


## Les composants de la partie utilisateur.

L'utilisateur principal du site web est le client. On doit avoir son identification (nom, prénom, login, mot de passe) et ses coordonnées (téléphone, adresse, email). On doit aussi identifier les employés : accueil, pizzaiolo, livreur, manager, gestionnaire, direction, comptable, direction.

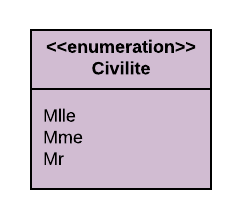
Pour les employés on a juste besoin de connaître leur rôle pour leur affecter des droits sur le système. On va créer donc un composant "Utilisateur" et deux composant qui vont en hérités "Client" et "Employe".

### Utilisateur

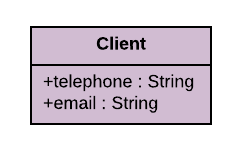


Un champ unique "id" permet d'identifier un utilisateur. Chaque "Utilisateur" a un "Magasin" attitré soit parce que c'est son magasin auprès duquel il effectue ses commandes ou soit parce qu'il y travaille. On fait référence à ce "Magasin" par son identifiant unique "id". On utilise une énumération "Civilite" pour la civilité afin de limiter les choix et d'éviter les erreurs.

### Civilite <<enum>>

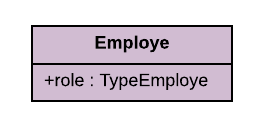


### Client



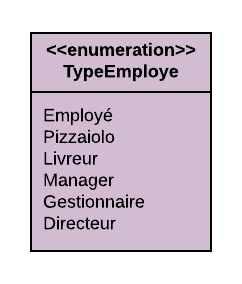
Contrairement à un employé, on doit connaître les coordonnées du "Client" pour pouvoir effectuer la vente et la livraison. L'adresse de livraison est référencée par un identifiant du composant "Adresse".

### Employe

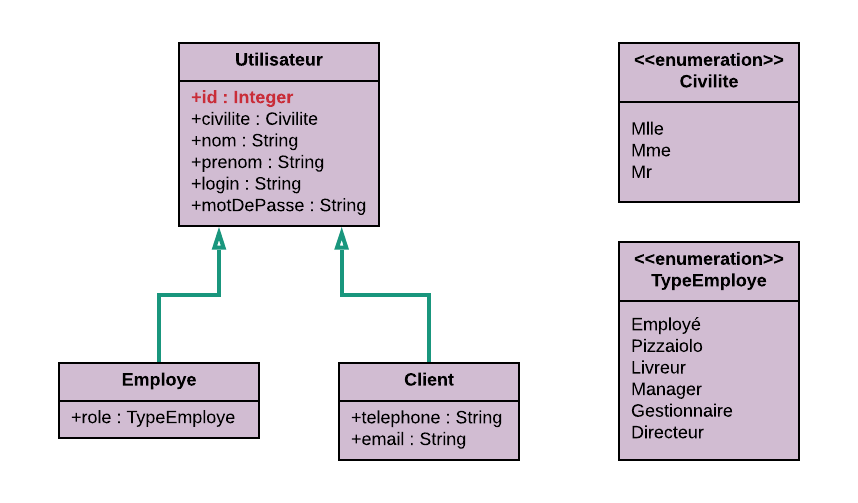


Pour définir le rôle de l'employé donc ses droits sur le système, on utilise une énumération "TypeEmploye".

### TypeEmploye <<enum>>



### Apperçu de la partie Utilisateur

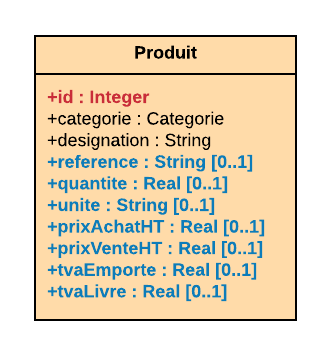


On voit bien l'héritage que font les objets "Employe" et "Client" de "Utilisateur"

## Les composants de la partie produit.

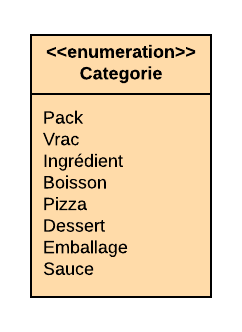
On doit gérer les ingrédients de base, les produits manufacturés et les produits additionnels comme les boissons. On doit enregistrer le prix d'achat et le prix de vente hors taxes pour chaque produit vendable. La TVA est gérée lors du passage de la commande suivant si elle est livrée ou emportée.

### Produit



Le fournisseur n'est pas obligatoire si le produit est une référence de pizza. En outre comme une pizza n'a pas de prix d'achat et un ingrédient n'a pas de prix de vente, ces attributs ne sont pas indispensables. On doit aussi avoir l'information sur le taux de tva de chaque produit vendu qui peut différer pour un produit emporté ou livré.

### Categorie <<enum>>

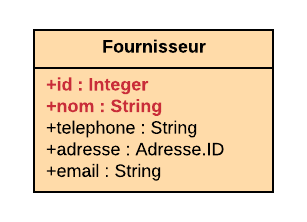


Un composant "Categorie" est une énumération qui regroupe les différentes "categorie" de "Produit". On utilisera :

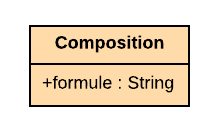
* Pack : pour les produits en groupes livrés par le fournisseur (caisse de 24 canettes),
* Boisson : pour les produits en stock qui sont à l'origine des pack mais vendu à l'unité.
* Ingrédient : pour les produits livrés par le fournisseur, au poids ou au volume et utilisés en vrac (farine, légumes, viande, poisson…)
* Vrac : pour les produits en stock qui sont à l'origine des ingrédients qui ont été livrés.

### Fournisseur

Un objet "Fournisseur" permet de classer les différents fournisseurs. On utilise un attribut "id" pour avoir un identifiant unique qui sert de référence et on spécifie aussi que le "nom" est un identifiant pour faire des recherches plus rapidement.



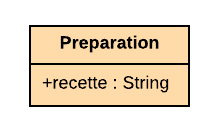
### Composition



C'est l'identifiant du produit qui sert de référence pour les "Composition".

On utilise un composant spécifique pour renseigner de la composition d'un produit vendu sur le site Web, qu'il soit manufacturé dans le magasin ou non. Un autre composant "Preparation" servira pour enregistrer les recettes pour les "Pizzaiolo".

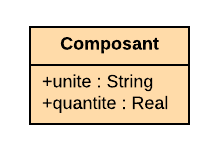
### Preparation



C'est l'identifiant du produit qui sert de référence pour les "Preparation".

On doit garder l'information des produits et la quantité nécessaire pour la fabrication de chaque pizza pour pouvoir modifier le stock des ingrédients en cas de vente d'une pizza. On utilise l'objet "Composant".

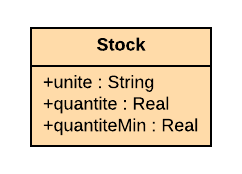
### Composant



Une pizza et un ingrédient sont identifiés par leur identifiant dans le composant "Produit". On ajoute un attribut "unite" pour savoir si la quantité est un poids, un volume ou une unité.

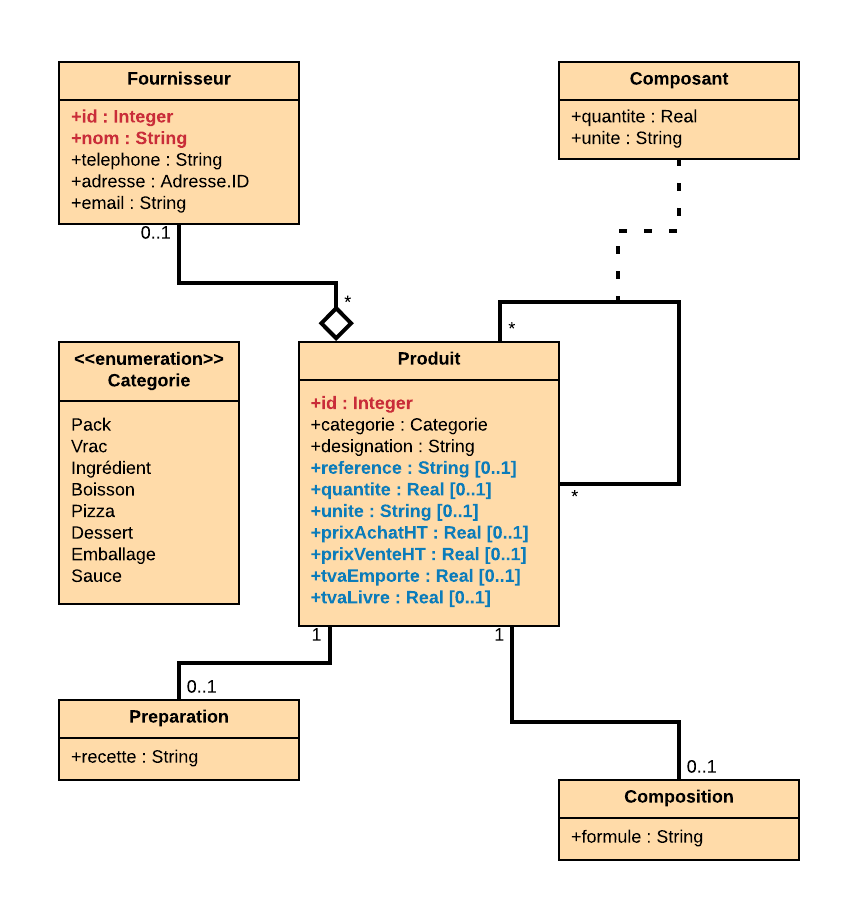
Un objet "Stock" permet de garder les informations sur les stocks de chaque produit dans chaque magasin.

### Stock



L'information dépend des identifiants du "Magasin" et du "Produit". On ajoute la notion de quantité minimale dans un attribut pour pouvoir gérer par la suite les commandes automatiques d'un produit.

### Apperçu de la partie Produit

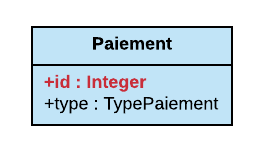


On remarque bien l'objet "Composant" qui comporte l'information entre les composés "Produit" et les composants "Produit".

## Les composants de la partie paiement.

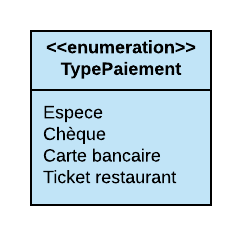
On utilise un objet "Paiement" dont héritera les différents types de paiement : "CarteBancaire", "TicketRestaurant" et "Cheque". Les paiements par espèce n'ont pas d'information spécifique donc ils n'ont pas besoin d'un composant qui hérite de "Paiement".

### Paiement

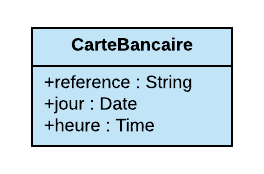


On utilise en attribut un identifiant commun à tous les paiement et un autre pour connaître le type de paiement qui fait référence à une énumération "TypePaiement".

### TypePaiement <<enum>>

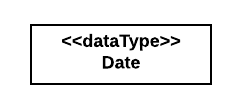


### CarteBancaire

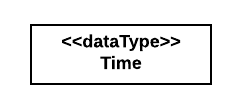


On utilise des <<dataType>> pour le jour et l'heure.

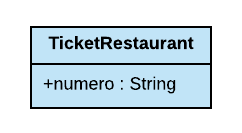
### Date <<dataType>>



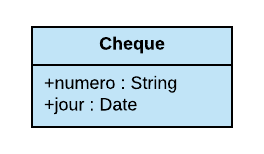
### Time <<dataType>>



### TicketRestaurant

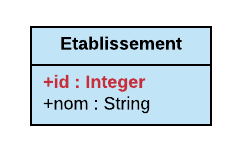


### Cheque



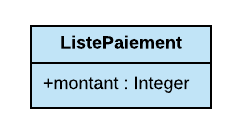
Pour éviter les mauvaises saisies et la redondance on utilise un composant "Etablissement" pour faire référence aux banques émettrices de "Cheque" et aux organismes de gestion des "TicketRestaurant".

### Etablissement



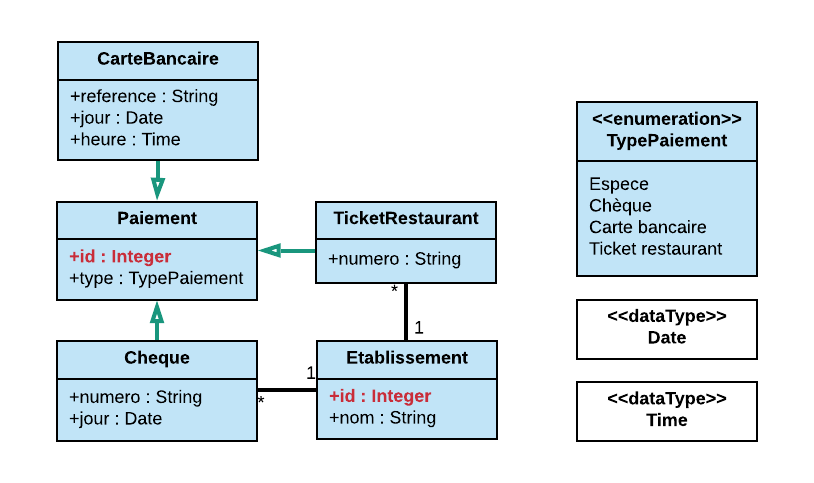
Cet objet contiendra les noms des établissements bancaires et des organismes de titre restaurant.

### ListePaiement



C'est lui qui fait la liaison entre une "Commande" et ses différents "Paiement" car un "Client" peut régler sa commande avec plusieurs modes de paiement.

### Apperçu de la partie Paiement

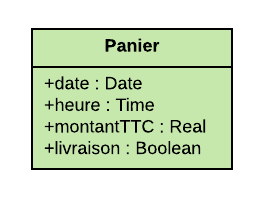


Les objets "CarteBancaire", "Cheque" et "TicketRestaurant" héritent de "Paiement". Le "Paiement" par espèce sera directement dans la classe mère car il n'a pas d'attribut en plus.

## Les composants de la partie commande.

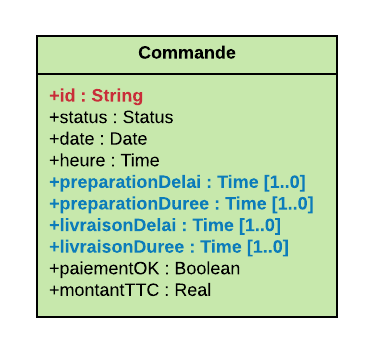
On distingue le panier que le client rempli de produits et la commande qui contient les paniers validés. Le composant "Panier" permet de garder les informations sur les produits que le Client choisi avant de valider sa commande. On doit enregistrer la date est l'heure pour pouvoir effacer les paniers non validés à partir d'un certain moment après la déconnexion du client.

### Panier



Chaque panier est lié à un seul "Utilisateur". On utilise aussi les <<dataType>> "Date" et "Time". L'attribut "livraison" permet de savoir si la commande sera livrée ou retirée sur place pour calculer le montant du panier en fonction des taux de TVA.

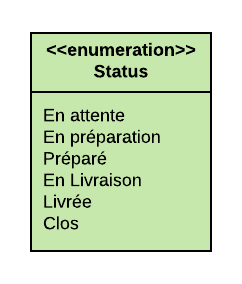
### Commande



Le composant "Commande" permet de garder les informations sur le déroulement de la commande comme le "Status" qui est une énumération et les différents champs permettant de suivre les temps de préparations, le montant et la validation du paiement.

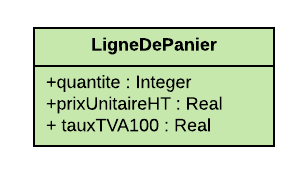
* preparationDelai : est le temps entre la validation de la commande par l'utilisateur et le début de préparation par le "Pizzaiolo".
* preparationDuree : est le temps de préparation de la commande par le "Pizzaiolo".
* livraisonDelai : est le temps mis pour finaliser la commande avant son départ en livraison.
* livraisonDurée : est le temps mis par le "Livreur" pour livrer les produits au client.

### Statut <<enum>>



Les informations sur le contenu des "Commande" et des "Panier" sera enregistré dans les objets "LigneDeCommande" et "LigneDePanier" respectivement. On garde les prix en Hors-Taxes car si la vente est à emporter le taux de TVA n'est pas le même.

### LigneDePanier



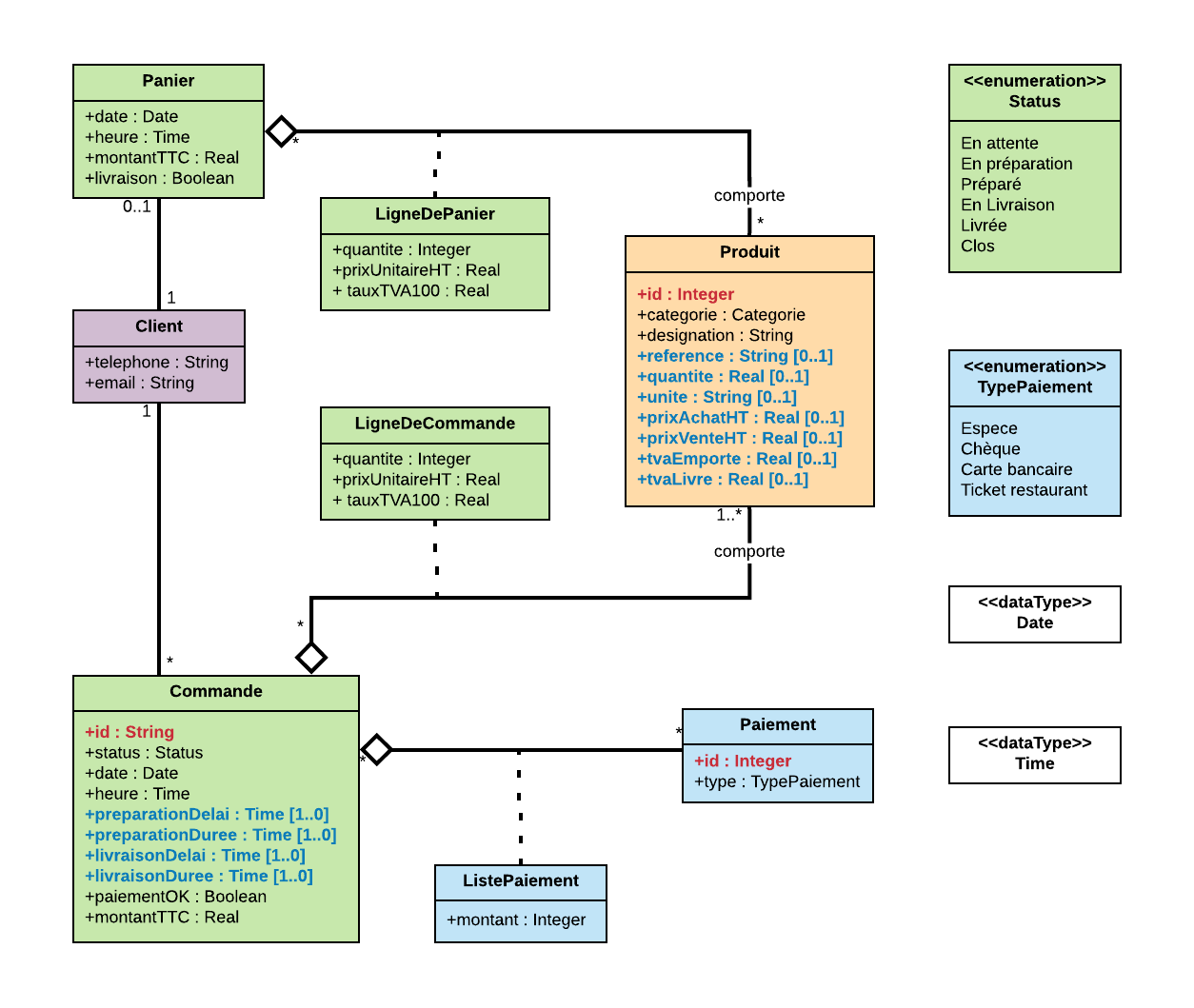
On identifie chaque tuple par l'identifiant du "Panier" donc celui "Utilisateur" et l'identifiant du "Produit".

### LigneDeCommande



On identifie chaque tuple par l'identifiant de la "Commande" et celui du "Produit". On ajoute ici l'information sur la TVA du produit qui dépend de l'adresse de livraison. Lors de la validation du "Panier", il suffit de transférer les tuples de "LigneDePanier" à "LigneDeCommande" et de créer un nouvel objet "Commande".

### Apperçu de la partie Commande



On remarque la similitude entre "LigneDePanier" et "LigneDeCommande" pour pouvoir déplacer les données de l'un à l'autre en cas de validation du "Panier".

# Les relations entre les composants.

## Les associations avec Utilisateur.

### Utilisateur – Magasin



Un "Utilisateur" est lié à un magasin soit parce qu'il y travaille ou soit parce qu'il est client. Chaque "Utilisateur" est lié à un seul "Magasin". Un "Magasin" a au moins un "Utilisateur", c'est le "Magager".

## Les associations avec Client.

### Client – Adresse



Un "Client" a une seule "Adresse" et une "Adresse" peut être commune à plusieurs "Client" ou aucun car elle peut être "Adresse" d'un "Magasin".

### Client – Panier

Confère l'association Panier-Client.

### Client – Commande

Confère l'association Panier-Client.

## Les associations avec Panier.

### Panier – Client



Chaque "Panier" est lié à un seul "Client" mais si un "Client" est enregistré mais n'a pas encore commandé, il n'a pas de "Panier".

### Panier – Produit



Un "Panier" peut contenir aucun ou plusieurs "Produit", c'est une agrégation de "Produit". Si on supprime un "Panier", les produits ne sont pas détruits. Un "Produit" peut être dans aucun ou plusieurs "Panier".

## Les associations avec Commande.

### Commande-Client



Chaque "Commande" est liée forcément à un seul "Client". Un "Client" peut avoir fait aucune "Commande" ou plusieurs.

### Commande – Produit



Une "Commande" doit contenir au moins un ou plusieurs "Produit", c'est une agrégation de "Produit". Si on supprime une "Commande", les produits ne sont pas détruits. Les "Produit" peuvent être dans plusieurs "Commande".

### Commande – Adresse



Une "Commande" a obligatoirement une et une seule "Adresse". Un "Client" pouvant commander plusieurs "Commande", on aura plusieurs "Commande" avec la même adresse. C'est aussi le cas ou plusieurs "Client" comme des colocataires habitent à la même "Adresse".

### Commande – Paiement



Une "Commande" peut être une agrégation de "Paiement" si le "Client" paie avec différents moyens (ticket restaurant et espèce). Pour des raisons comptables on ne doit pas supprimer les "Paiement" si l'on supprime la "Commande". On devra faire un remboursement. Les "Paiement" ne resteront pas longtemps sans "Commande" associé mais il est clair que des "Paiement" seront dans plusieurs "Commande".

## Les associations avec Produit.

### Produit – Panier

Confère l'association Panier-Produit.

### Produit – Commande

Confère l'association Commande-Produit.

### Produit – Magasin



Chaque "Magasin" a un stock de plusieurs "Produit", c'est une agrégation. On ne considère pas comme une composition pour ne pas détruire un stock de "Produit" et éviter le gaspillage. Un "Magasin" pourra ne pas avoir de "Produit" avant son ouverture s'il n'a pas encore reçu sa marchandise. Pour rentabiliser les pizzérias on va utiliser les mêmes "Produits" dans toutes.

### Produit - Fournisseur



Un "Produit" manufacturé n'a pas de "Fournisseur" sinon les autres en ont un seul. Un "Fournisseur" peut avoir aucun ou plusieurs "Produit". On retrouve ici une agrégation où l'on ne supprimera pas les "Produit" à la disparition du "Fournisseur".

### Produit – Composition



On pourra indiquer pour certain "Produit" leur "Composition" qui sera unique. En effet c'est difficile d'avoir deux sodas avec la même composition et vendre deux pizzas avec les mêmes ingrédients sous des noms différents. On a la possibilité de ne pas indiquer la composition mais c'est mieux envers la clientèle pour se prémunir des allergies ou des restrictions diététiques.

### Produit – Preparation



Pour le "Pizzaiolo" on enregistre les recettes des "Produit" manufacturé comme les pizzas dans "Preparation". Chaque "Preparation est liée à un seul "Produit" et un "Produit" n'est pas obligé d'avoir une recette.

### Produit - Produit



Cette association démontre le lien entre les "Produit" qui sont des ingrédients et les "Produit" qui sont des produits manufacturés comme les pizzas. Chaque pizza à plusieurs ingrédients et un ingrédient peut être dans plusieurs pizzas comme le fromage.

## Les autres associations.

### Magasin – Adresse



Un "Magasin" a forcément une seule "Adresse" et une "Adresse" n'est pas forcément celle d'un "Magasin".

### Fournisseur - Adresse



Comme pour les "Magasin" un "Fournisseur" a forcément une seule "Adresse" et une "Adresse" n'est pas forcément celle d'un " Fournisseur ".

### Cheque – Etablissement



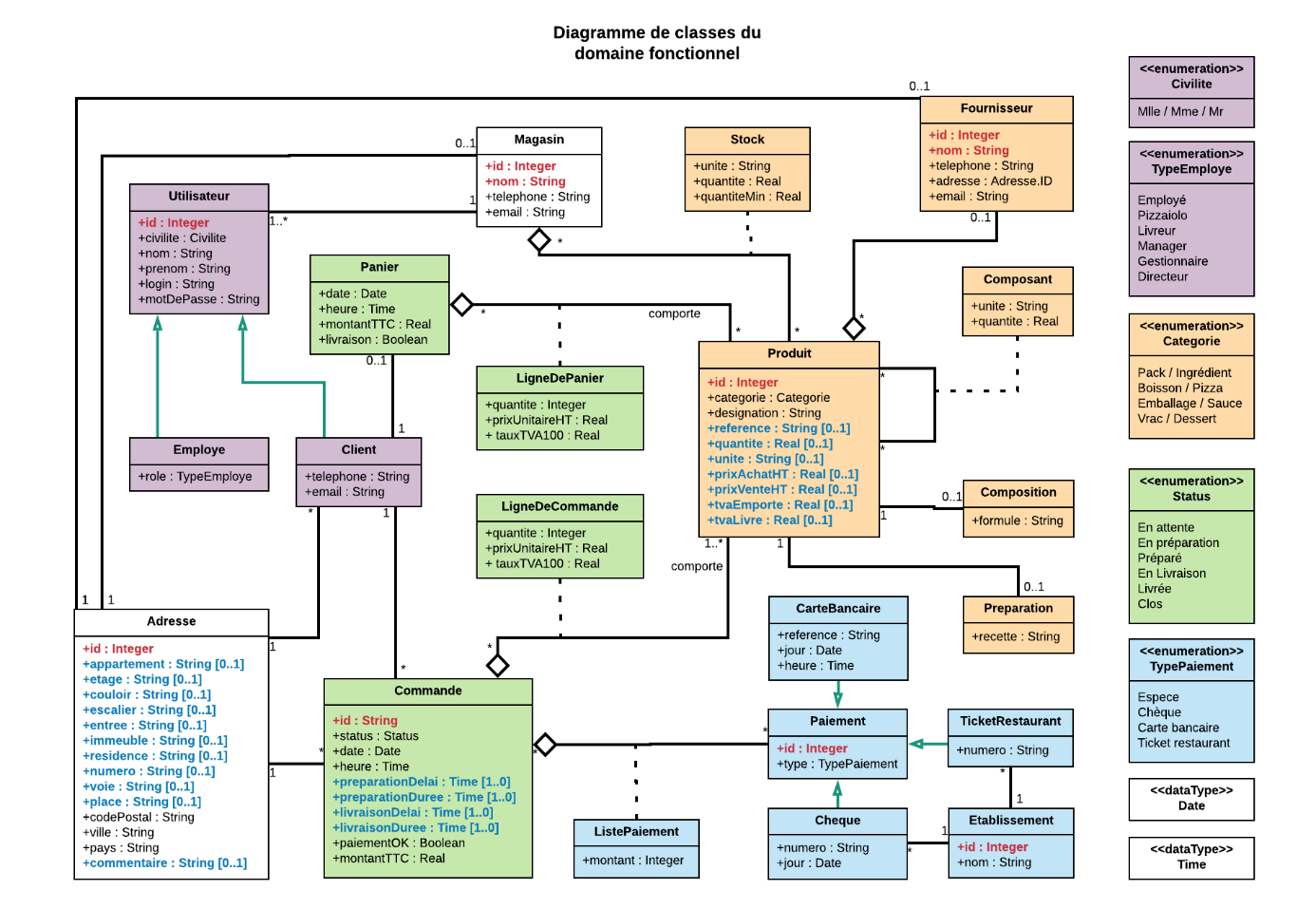
Un "Cheque" provient d'un seul "Etablissement" et ce dernier peut émettre aucun ou plusieurs "Cheque".

### TicketRestaurant - Etablissement



Un "TicketRestaurant" provient d'un seul "Etablissement" et ce dernier peut émettre aucun ou plusieurs " TicketRestaurant ".

## Le diagramme de classes du domaine fonctionnel



Les héritages sont indiqués en vert et les agrégations comportent un losange blanc sur la classe est une agrégation de l'autre classe. Les attributs en rouge sont les identifiants qui doivent être uniques pour l'objet. Les attribut qui ne sont pas obligatoire avec une multiplicité [0..1] sont en bleu et les autres en noir. Sur le côté on a la liste des **<<enumeration>>** et des **<<dataType>>**. On va se baser sur le diagramme de classes du domaine fonctionnel pour élaborer le modèle physique de données MDP. On utilisera **SQL Power Architect** pour élaborer le MDP et générer un script pour générer notre base de données sous **MySQL**.

# Le modèle physique de données MDP.

## Les types de données.

On transcrit chaque classe et énumération en table de base de données. Les types Integer seront transcrit en :

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Bytes** | **Valeurs** | **Description** |
| TINYINT UNSIGNED | 1 | 0 à 254 | Pour les identifiants avec peu de valeurs |
| INT UNSIGNED | 4 | 0 à 4294967294 | Pour les identifiants avec beaucoup de valeurs |
| DECIMAL (5,2) | 4 | -999,99 à 999,99 | Pour les attributs monétaires et les quantités avec une précision de 5 chiffres avant la virgule et 2 après. |

Les String en :

* VARCHAR(N) pour les chaînes de faible longueur ou N est la taille maximale.
* TEXT pour les grandes chaînes comme les commentaire, recette et formule.

Les <<dataType>> :

* Date en DATE.
* Time en TIME

### Dénomination.

* AI (AUTO INCREMENT) : La valeur est automatiquement incrémentée si elle n'est pas spécifiée.
* PK (PRIMARY KEY) : Clé primaire servant d'identifiant aux tuples de la table.
* FK (FOREIGN KEY) : Clé étrangère servant à identifier des tuples d'une autre table.
* NN (NOT NULL) : La valeur ne peut pas être nulle comme pour les clés primaires.

## La partie Utilisateur.

### Utilisateur

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **id** | **INT UNSIGNED** | **10** | **X** | **X** |  | **X** |
| civilite | ENUM ('Mlle','Mme','M') |  |  |  |  | X |
| nom | VARCHAR | 50 |  |  |  | X |
| prenom | VARCHAR | 50 |  |  |  | X |
| login | VARCHAR | 50 |  |  |  | X |
| mot\_de\_passe | VARCHAR | 255 |  |  |  | X |
| magasin\_id | INT UNSIGNED | 10 |  |  | X | X |

#### Contrainte d'unicité.

On ajoute une contrainte d'unicité "UNIQUE" sur le "login" pour ne pas avoir de problème de connexion.

#### Clé étrangère sur magasin\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| civilite | id | NO ACTION | NO ACTION |

### Client

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **utilisateur\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| telephone | VARCHAR | 10 |  |  |  | X |
| adresse\_id | INT UNSIGNED | 10 |  |  | X | X |
| email | VARCHAR | 255 |  |  |  | X |

#### Contrainte d'unicité.

On ajoute une contrainte d'unicité "UNIQUE" sur "email" pour permettre la récupération de mot de passe par email en.

#### Clé étrangère sur utilisateur\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| utilisateur | id | CASCADE | CASCADE |

Si on met à jour ou supprime un "utilisateur", "client" sera modifié ou supprimé.

#### Clé étrangère sur adresse\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| adresse | id | NO ACTION | NO ACTION |

### Employé

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **utilisateur\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| role | ENUM ('Accueil', 'Pizzaiolo', 'Livreur', 'Manager', 'Gestionnaire', 'Comptable', 'Direction') | | | | | X |

#### Clé étrangère sur utilisateur\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| utilisateur | id | CASCADE | CASCADE |

Si on met à jour ou supprime un "utilisateur", "employe" sera modifié ou supprimé.

## La partie Produit.

### Produit

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **id** | **INT UNSIGNED** | **10** | **X** | **X** |  | **X** |
| designation | VARCHAR | 50 |  |  |  | X |
| categorie | ENUM ('pack', 'vrac', 'ingrédient', 'pizza', 'boisson', 'dessert', 'emballage', 'sauce') | | | | | X |
| fournisseur\_id | INT UNSIGNED | 10 |  |  | X |  |
| reference | VARCHAR | 20 |  |  |  |  |
| quantite | DECIMAL | (5,2) |  |  |  |  |
| unite | VARCHAR | 3 |  |  |  |  |
| prix\_achat\_ht | DECIMAL | (5,2) |  |  |  |  |
| prix\_vente\_ht | DECIMAL | (5,2) |  |  |  |  |
| tva\_emporte | DECIMAL | (3,1) |  |  |  |  |
| tva\_livre | DECIMAL | (3,1) |  |  |  |  |

#### Contrainte d'unicité.

On ajoute une contrainte d'unicité "UNIQUE" sur le "designation" pour ne pas avoir deux fois le même produit.

#### Clé étrangère sur fournisseur\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| fournisseur | id | NO ACTION | NO ACTION |

### Composition

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **produit\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| formule | TEXT |  |  |  |  | X |

#### Clé étrangère sur produit\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| produit | id | CASCADE | CASCADE |

Si on met à jour ou supprime un "produit", "composition" sera modifié ou supprimé.

### Préparation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **produit\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| recette | TEXT |  |  |  |  | X |

#### Clé étrangère sur produit\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| produit | id | CASCADE | CASCADE |

Si on met à jour ou supprime un "produit", "preparation" sera modifié ou supprimé.

### Stock

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| magasin\_id | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| produit\_id | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| quantite | DECIMAL | (5,2) |  |  |  | X |
| quantite\_min | DECIMAL | (5,2) |  |  |  | X |
| unite | VARCHAR | 3 |  |  |  | X |

#### Clé étrangère sur produit\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| produit | id | NO ACTION | NO ACTION |

#### Clé étrangère sur magasin\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| magasin | id | NO ACTION | NO ACTION |

### Composant

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **X** |
| produit\_id | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| ingredient\_id | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| quantite | DECIMAL | (5,2) |  |  |  | X |
| unite | VARCHAR | 3 |  |  |  | X |

#### Clé étrangère sur produit\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| produit | id | NO ACTION | NO ACTION |

#### Clé étrangère sur ingredient\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| produit | id | NO ACTION | NO ACTION |

## La partie Paiement.

### Paiement

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **paiement\_id** | **INT UNSIGNED** | **10** | **X** | **X** |  | **X** |
| type | ENUM ('espèce', 'carte bancaire', 'ticket restaurant',  'chèque bancaire', 'sans') | | | | | X |

### Carte bancaire

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **paiement\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| reference | VARCHAR | 100 |  |  |  | X |
| jour | DATE |  |  |  |  | X |
| heure | TIME |  |  |  |  | X |

#### Clé étrangère sur paiement\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| paiement | id | CASCADE | CASCADE |

Si on met à jour ou supprime un "paiement", "carte\_bancaire" sera modifié ou supprimé.

### Chèque

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **paiement\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| banque | TINYINT UNSIGNED | 3 |  |  | X | X |
| numero | VARCHAR | 100 |  |  |  | X |
| jour | DATE |  |  |  |  | X |

#### Clé étrangère sur paiement\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| paiement | id | CASCADE | CASCADE |

Si on met à jour ou supprime un "paiement", "cheque" sera modifié ou supprimé.

### Ticket restaurant

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **paiement\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| numero | VARCHAR | 50 |  |  |  | X |
| etablissement\_id | TINYINT UNSIGNED | 3 |  |  | X | X |

#### Clé étrangère sur paiement\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| paiement | id | CASCADE | CASCADE |

Si on met à jour ou supprime un "paiement", "ticket\_restaurant" sera modifié ou supprimé.

### Établissement

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **id** | **TINYINT UNSIGNED** | **3** |  | **X** |  | **X** |
| nom | VARCHAR | 20 |  |  |  | **X** |

### Liste paiement

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **commande\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| **paiement\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| montant | DECIMAL | (5,2) |  |  |  | **X** |

On doit garder une trace des paiements donc on ne peut pas les modifier ou supprimer en cas de modification ou suppression dans les tables "paiement" et "commande".

#### Clé étrangère sur paiement\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| paiement | id | NO ACTION | NO ACTION |

#### Clé étrangère sur commande\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| commande | id | NO ACTION | NO ACTION |

## La partie Commande.

### Panier

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **utilisateur\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| jour | DATE |  |  |  |  | X |
| heure | TIME |  |  |  |  | X |
| montant\_ttc | DECIMAL | (5,2) |  |  |  | X |
| livraison | TINYINT UNSIGNED |  |  |  |  | X |

#### Clé étrangère sur utilisateur\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| client | id | NO ACTION | NO ACTION |

### Commande

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **id** | **INT UNSIGNED** | **10** | **X** | **X** |  | **X** |
| **utilisateur\_id** | **INT UNSIGNED** | **10** |  |  | **X** | **X** |
| **adresse\_id** | **INT UNSIGNED** | **10** |  |  |  | **X** |
| status | ENUM ('En attente', 'En préparation', 'Préparée', 'En livraison', 'Livrée', 'Clos') | | | | | X |
| jour | DATE |  |  |  |  | X |
| heure | TIME |  |  |  |  | X |
| montant\_ttc | DECIMAL | (5,2) |  |  |  | X |
| preparation\_delai | TIME |  |  |  |  |  |
| preparation\_duree | TIME |  |  |  |  |  |
| livraison\_delai | TIME |  |  |  |  |  |
| livraison\_duree | TIME |  |  |  |  |  |
| paiement\_ok | TINYINT | 1 |  |  |  | X |

#### Clé étrangère sur adresse\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| adresse | id | NO ACTION | NO ACTION |

#### Clé étrangère sur utilisateur\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| utilisateur | id | NO ACTION | NO ACTION |

### Ligne de panier

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **utilisateur\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| **produit\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| quantite | DECIMAL | (2,0) |  |  |  | X |
| prix\_unitaire\_ht | DECIMAL | (5,2) |  |  |  | X |
| taux\_tva | DECIMAL | (3,1) |  |  |  | X |

#### Clé étrangère sur utilisateur\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| panier | id | CASCADE | CASCADE |

Si le "panier" est supprimé on supprime les tuples correspondant dans la table "ligne\_de\_panier".

#### Clé étrangère sur produit\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| produit | id | NO ACTION | NO ACTION |

### Ligne de commande

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **commande\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| **produit\_id** | **INT UNSIGNED** | **10** |  | **X** | **X** | **X** |
| quantite | DECIMAL | (2,0) |  |  |  | X |
| prix\_unitaire\_ht | DECIMAL | (5,2) |  |  |  | X |
| taux\_tva | DECIMAL | (3,1) |  |  |  | X |

#### Clé étrangère sur commande\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| commande | id | CASCADE | CASCADE |

Si la "commande" est supprimée on supprime les tuples correspondant dans la table "ligne\_de\_commande".

#### Clé étrangère sur produit\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| produit | id | NO ACTION | NO ACTION |

## Les autres tables.

### Adresse

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **id** | **INT UNSIGNED** | **10** | **X** | **X** |  | **X** |
| appartement | VARCHAR | 4 |  |  |  |  |
| etage | VARCHAR | 3 |  |  |  |  |
| couloir | VARCHAR | 3 |  |  |  |  |
| escalier | VARCHAR | 3 |  |  |  |  |
| entree | VARCHAR | 3 |  |  |  |  |
| immeuble | VARCHAR | 10 |  |  |  |  |
| residence | VARCHAR | 20 |  |  |  |  |
| numero | VARCHAR | 5 |  |  |  |  |
| voie | VARCHAR | 50 |  |  |  |  |
| place | VARCHAR | 50 |  |  |  |  |
| code | VARCHAR | 5 |  |  |  | X |
| ville | VARCHAR | 20 |  |  |  | X |
| pays | VARCHAR | 20 |  |  |  | X |
| commentaire | TEXT |  |  |  |  |  |

#### Contrainte d'unicité.

Pour ne pas avoir deux fois la même adresse on implémentera des TRIGGER BEFORE INSERT et BEFORE UPDATE.

### Magasin

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **id** | **INT UNSIGNED** | **10** | **X** | **X** |  | **X** |
| nom | VARCHAR | 50 |  | X |  | X |
| telephone | VARCHAR | 10 |  |  |  | X |
| email | VARCHAR | 255 |  |  |  | X |
| adresse\_id | INT UNSIGNED | 10 |  |  | X | X |

#### Clé primaire sur "nom"

Définir une clé primaire sur le "nom" permet d'avoir des noms uniques pour les magasins et d'accélérer les recherches sur le "nom".

#### Clé étrangère sur "adresse\_id"

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| adresse | id | NO ACTION | NO ACTION |

### Fournisseur

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Precision** | **AI** | **PK** | **FK** | **NN** |
| **id** | **INT UNSIGNED** | **10** | **X** | **X** |  | **X** |
| nom | VARCHAR | 50 |  | X |  | X |
| telephone | VARCHAR | 10 |  |  |  | X |
| email | VARCHAR | 255 |  |  |  | X |
| adresse\_id | INT UNSIGNED | 10 |  |  | X | X |

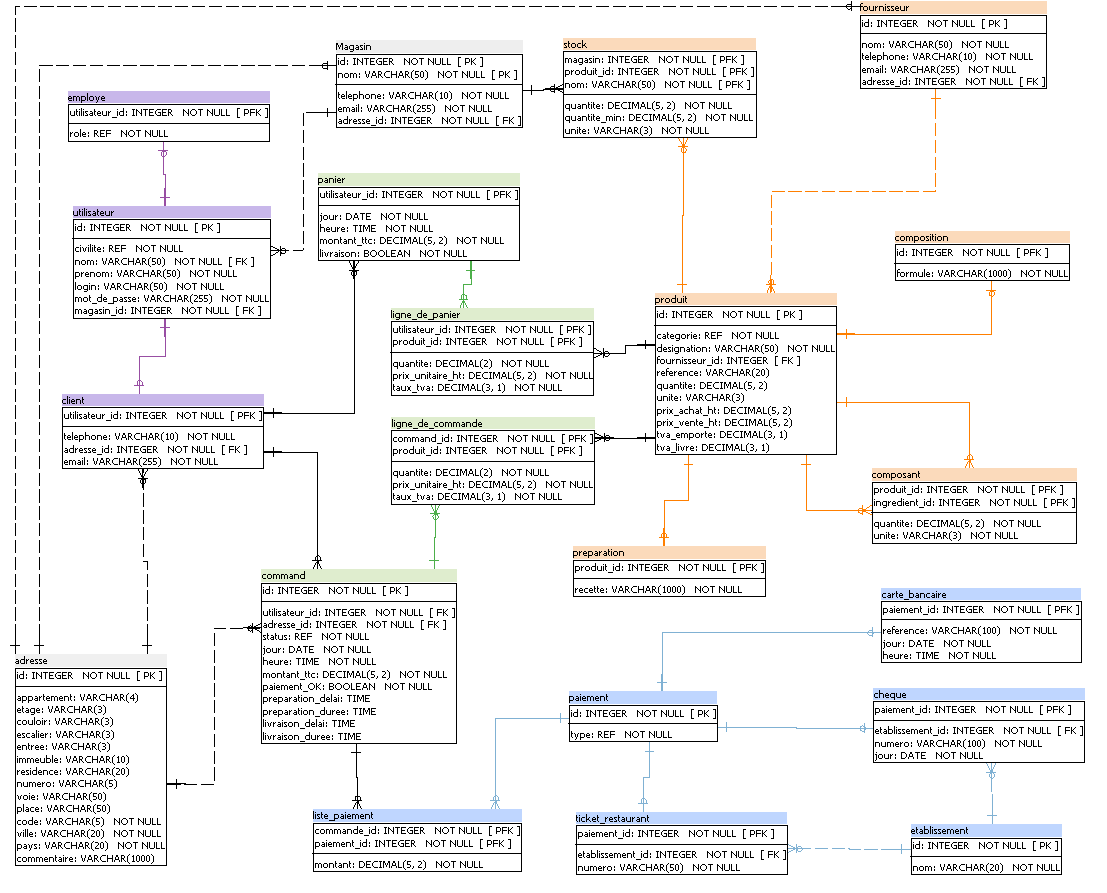
#### Clé primaire sur "nom"

Définir une clé primaire sur le "nom" permet d'avoir des noms uniques pour les fournisseurs et d'accélérer les recherches sur le "nom".

#### Clé étrangère sur "adresse\_id"

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Colonne** | **ON DELETE** | **ON UPDATE** |
| adresse | id | NO ACTION | NO ACTION |

## Diagramme du Modèle Physique de Données.



L'outil de création du modèle physique de données ne contient pas tous les types de MySQL.

* Les VACHAR(1000) sont remplacés par des type TEXT.
* Les BOOLEAN sont remplacés par des TINYINT.

Une table a été ajoutée dans la base pour afficher les messages d'erreur. Elle n'apparait pas dans le diagramme.

# Scripts de la base de données MySQL.

## Création de la base de données oc\_pizza.

|  |
| --- |
| CREATE DATABASE oc\_pizza CHARACTER SET 'utf8';  USE oc\_pizza;  ################################################################################  # GLOBAL #  ################################################################################  ####################################################################### ERREUR #  DROP TABLE IF EXISTS erreur;  CREATE TABLE erreur (  id TINYINT UNSIGNED AUTO\_INCREMENT NOT NULL, -- identifiant de l'erreur  message VARCHAR(100) UNIQUE, -- message d'erreur  PRIMARY KEY (id)  )ENGINE=InnoDB;  DESCRIBE erreur;  -- remplissage de la table des erreurs pour créer une erreur lors d'une ré-insertion  INSERT INTO erreur (message) VALUES ('ERREUR : l''adresse doit être unique!');  INSERT INTO erreur (message) VALUES ('ERREUR : le login doit être unique!');  INSERT INTO erreur (message) VALUES ('ERREUR : le nom du magasin doit être unique!');  INSERT INTO erreur (message) VALUES ('ERREUR : le nom du fournisseur doit être unique!');  INSERT INTO erreur (message) VALUES ('ERREUR : un produit n''est plus disponible!');  INSERT INTO erreur (message) VALUES ('ERREUR : la commande n''est pas en attente!');  INSERT INTO erreur (message) VALUES ('ERREUR : la commande n''est pas en préparation!');  INSERT INTO erreur (message) VALUES ('ERREUR : la commande n''est pas préparée!');  SELECT \* FROM erreur;  ###################################################################### ADRESSE #  DROP TABLE IF EXISTS adresse;  CREATE TABLE adresse (  id INT UNSIGNED AUTO\_INCREMENT NOT NULL,  appartement VARCHAR(4),  etage VARCHAR(3),  couloir VARCHAR(3),  escalier VARCHAR(3),  entree VARCHAR(3),  immeuble VARCHAR(10),  residence VARCHAR(20),  numero VARCHAR(5),  voie VARCHAR(50),  place VARCHAR(50),  code VARCHAR(5) NOT NULL,  ville VARCHAR(20) NOT NULL,  pays VARCHAR(20) DEFAULT 'FRANCE' NOT NULL,  commentaire TEXT, -- pour un code de digicode ou informations complémentaires  PRIMARY KEY (id)  )ENGINE=InnoDB;  DESCRIBE adresse;  -- trigger pour vérifier l'insertion d'adresse unique  DROP TRIGGER IF EXISTS before\_insert\_adresse;  DELIMITER |  CREATE TRIGGER before\_insert\_adresse  BEFORE INSERT ON adresse FOR EACH ROW  BEGIN  DECLARE existedeja INTEGER;  SELECT COUNT(\*) INTO existedeja  FROM adresse  WHERE IFNULL(appartement,'NULL') = IFNULL(NEW.appartement,'NULL')  AND IFNULL(etage,'NULL') = IFNULL(NEW.etage,'NULL')  AND IFNULL(couloir,'NULL') = IFNULL(NEW.couloir,'NULL')  AND IFNULL(escalier,'NULL') = IFNULL(NEW.escalier,'NULL')  AND IFNULL(entree,'NULL') = IFNULL(NEW.entree,'NULL')  AND IFNULL(immeuble,'NULL') = IFNULL(NEW.immeuble,'NULL')  AND IFNULL(residence,'NULL') = IFNULL(NEW.residence,'NULL')  AND IFNULL(numero,'NULL') = IFNULL(NEW.numero,'NULL')  AND IFNULL(voie,'NULL') = IFNULL(NEW.voie,'NULL')  AND IFNULL(place,'NULL') = IFNULL(NEW.place,'NULL')  AND IFNULL(code,'NULL') = IFNULL(NEW.code,'NULL')  AND IFNULL(ville,'NULL') = IFNULL(NEW.ville,'NULL')  AND IFNULL(pays,'NULL') = IFNULL(NEW.pays,'NULL');    IF existedeja > 0 THEN  INSERT INTO erreur (message) VALUES ('ERREUR : l''adresse doit être unique!');  END IF;  END |  DELIMITER ;  DROP TRIGGER IF EXISTS before\_update\_adresse;  DELIMITER |  CREATE TRIGGER before\_update\_adresse  BEFORE UPDATE ON adresse FOR EACH ROW  BEGIN  DECLARE existedeja INTEGER;  SELECT COUNT(\*) INTO existedeja  FROM adresse  WHERE IFNULL(appartement,'NULL') = IFNULL(NEW.appartement,'NULL')  AND IFNULL(etage,'NULL') = IFNULL(NEW.etage,'NULL')  AND IFNULL(couloir,'NULL') = IFNULL(NEW.couloir,'NULL')  AND IFNULL(escalier,'NULL') = IFNULL(NEW.escalier,'NULL')  AND IFNULL(entree,'NULL') = IFNULL(NEW.entree,'NULL')  AND IFNULL(immeuble,'NULL') = IFNULL(NEW.immeuble,'NULL')  AND IFNULL(residence,'NULL') = IFNULL(NEW.residence,'NULL')  AND IFNULL(numero,'NULL') = IFNULL(NEW.numero,'NULL')  AND IFNULL(voie,'NULL') = IFNULL(NEW.voie,'NULL')  AND IFNULL(place,'NULL') = IFNULL(NEW.place,'NULL')  AND IFNULL(code,'NULL') = IFNULL(NEW.code,'NULL')  AND IFNULL(ville,'NULL') = IFNULL(NEW.ville,'NULL')  AND IFNULL(pays,'NULL') = IFNULL(NEW.pays,'NULL');    IF existedeja > 0 THEN  INSERT INTO erreur (message) VALUES ('ERREUR : l''adresse doit être unique!');  END IF;  END |  DELIMITER ;  ################################################################## FOURNISSEUR #  DROP TABLE IF EXISTS fournisseur;  CREATE TABLE fournisseur (  id INT UNSIGNED AUTO\_INCREMENT NOT NULL,  nom VARCHAR(50) NOT NULL,  telephone VARCHAR(10) NOT NULL,  email VARCHAR(255) NOT NULL,  adresse\_id INT UNSIGNED NOT NULL,  PRIMARY KEY (id,nom)  )ENGINE=InnoDB;  -- l'identifiant de l'adresse et la la table adresse sont liés  ALTER TABLE fournisseur ADD CONSTRAINT adresse\_fournisseur\_fk  FOREIGN KEY (adresse\_id)  REFERENCES adresse (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  DESCRIBE fournisseur;  ###################################################################### MAGASIN #  DROP TABLE IF EXISTS magasin;  CREATE TABLE magasin (  id INT UNSIGNED AUTO\_INCREMENT NOT NULL,  nom VARCHAR(50) NOT NULL,  telephone VARCHAR(10) NOT NULL,  email VARCHAR(255) NOT NULL,  adresse\_id INT UNSIGNED NOT NULL,  PRIMARY KEY (id,nom)  )ENGINE=InnoDB;  -- l'identifiant de l'adresse et la la table adresse sont liés  ALTER TABLE magasin ADD CONSTRAINT adresse\_magasin\_fk  FOREIGN KEY (adresse\_id)  REFERENCES adresse (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  ALTER TABLE magasin  ADD INDEX magasin\_nom\_idx (nom);  DESCRIBE magasin;  ################################################################################  # UTILISATEUR #  ################################################################################  ################################################################## UTILISATEUR #  DROP TABLE IF EXISTS utilisateur;  CREATE TABLE utilisateur (  id INT UNSIGNED AUTO\_INCREMENT NOT NULL,  civilite ENUM('Mlle','Mme','M') NOT NULL,  nom VARCHAR(50) NOT NULL,  prenom VARCHAR(50) NOT NULL,  login VARCHAR(50) NOT NULL UNIQUE,  mot\_de\_passe VARCHAR(255) NOT NULL,  magasin\_id INT UNSIGNED NOT NULL,  PRIMARY KEY (id)  )ENGINE=InnoDB;  -- un utilisateur est lié à un magasin par son identifiant  ALTER TABLE utilisateur ADD CONSTRAINT magasin\_utilisateur\_fk  FOREIGN KEY (magasin\_id)  REFERENCES magasin (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  -- ajout d'un index pour une recherche sur les noms  ALTER TABLE utilisateur  ADD INDEX utilisateur\_nom\_idx (nom);  DESCRIBE utilisateur;  ###################################################################### EMPLOYE #  DROP TABLE IF EXISTS employe;  CREATE TABLE employe (  utilisateur\_id INT UNSIGNED NOT NULL,  role ENUM ('Accueil','Pizzaiolo','Livreur','Manager',  'Gestionnaire','Comptable','Direction') NOT NULL,  PRIMARY KEY (utilisateur\_id)  )ENGINE=InnoDB;  --un employé est un utilisateur  ALTER TABLE employe ADD CONSTRAINT utilisateur\_employe\_fk  FOREIGN KEY (utilisateur\_id)  REFERENCES utilisateur (id)  ON DELETE CASCADE  ON UPDATE CASCADE;  DESCRIBE employe;  ####################################################################### CLIENT #  DROP TABLE IF EXISTS client;  CREATE TABLE client (  utilisateur\_id INT UNSIGNED NOT NULL,  telephone VARCHAR(10) NOT NULL,  adresse\_id INT UNSIGNED NOT NULL,  email VARCHAR(255) NOT NULL UNIQUE,  PRIMARY KEY (utilisateur\_id)  )ENGINE=InnoDB;  --un client est un utilisateur  ALTER TABLE client ADD CONSTRAINT utilisateur\_client\_fk  FOREIGN KEY (utilisateur\_id)  REFERENCES utilisateur (id)  ON DELETE CASCADE  ON UPDATE CASCADE;  -- l'identifiant de l'adresse et la la table adresse sont liés  ALTER TABLE client ADD CONSTRAINT adresse\_client\_fk  FOREIGN KEY (adresse\_id)  REFERENCES adresse (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  DESCRIBE client;  ################################################################################  # PRODUIT #  ################################################################################  ###################################################################### PRODUIT #  DROP TABLE IF EXISTS produit;  CREATE TABLE produit (  id INT UNSIGNED AUTO\_INCREMENT NOT NULL,  designation VARCHAR(100) UNIQUE NOT NULL,  categorie ENUM ('pack','vrac','ingrédient','pizza','boisson','dessert','emballage','sauce') NOT NULL,  fournisseur\_id INT UNSIGNED,  reference VARCHAR(20),  quantite DECIMAL(5,2) DEFAULT (0.0),  unite VARCHAR(3),  prix\_achat\_ht DECIMAL(5,2) DEFAULT (0.0),  prix\_vente\_ht DECIMAL(5,2) DEFAULT (0.0),  tva\_emporte DECIMAL(3,1) DEFAULT (10.0) NOT NULL,  tva\_livre DECIMAL(3,1) DEFAULT (10.0) NOT NULL,  PRIMARY KEY (id)  )ENGINE=InnoDB;  -- un produit peut être lié à un fournisseur  ALTER TABLE produit ADD CONSTRAINT fournisseur\_produit\_fk  FOREIGN KEY (fournisseur\_id)  REFERENCES fournisseur (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  DESCRIBE produit;  ################################################################## PREPARATION #  DROP TABLE IF EXISTS preparation;  CREATE TABLE preparation (  produit\_id INT UNSIGNED NOT NULL,  recette TEXT NOT NULL,  PRIMARY KEY (produit\_id)  )ENGINE=InnoDB;  -- un produit comme la pizza peut avoir une recette donc l'identifiant de la  -- préparation est le même que celui du produit  ALTER TABLE preparation ADD CONSTRAINT produit\_preparation\_fk  FOREIGN KEY (produit\_id)  REFERENCES produit (id)  ON DELETE CASCADE  ON UPDATE CASCADE;  DESCRIBE preparation;  ################################################################## COMPOSITION #  DROP TABLE IF EXISTS composition;  CREATE TABLE composition (  produit\_id INT UNSIGNED NOT NULL,  formule TEXT NOT NULL,  PRIMARY KEY (produit\_id)  )ENGINE=InnoDB;  -- un produit comme peut avoir une composition donc l'identifiant de la  -- composition est le même que celui du produit  ALTER TABLE composition ADD CONSTRAINT produit\_composition\_fk  FOREIGN KEY (produit\_id)  REFERENCES produit (id)  ON DELETE CASCADE  ON UPDATE CASCADE;  DESCRIBE composition;  #################################################################### COMPOSANT #  DROP TABLE IF EXISTS composant;  CREATE TABLE composant (  produit\_id INT UNSIGNED NOT NULL,  ingredient\_id INT UNSIGNED NOT NULL,  quantite DECIMAL(5,2) DEFAULT (0) NOT NULL,  unite VARCHAR(3) NOT NULL,  PRIMARY KEY (produit\_id, ingredient\_id)  )ENGINE=InnoDB;  -- le produit composant est un produit identifié par son id  ALTER TABLE composant ADD CONSTRAINT produit\_composition\_produit\_fk  FOREIGN KEY (produit\_id)  REFERENCES produit (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  -- un ingrédient est un produit identifié par son id  ALTER TABLE composant ADD CONSTRAINT produit\_composition\_ingredient\_fk  FOREIGN KEY (ingredient\_id)  REFERENCES produit (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  DESCRIBE composant;  ######################################################################## STOCK #  DROP TABLE IF EXISTS stock;  CREATE TABLE stock (  magasin\_id INT UNSIGNED NOT NULL,  produit\_id INT UNSIGNED NOT NULL,  quantite DECIMAL(5,2) DEFAULT (0) NOT NULL,  quantite\_min DECIMAL(5,2) DEFAULT (0.0) NOT NULL,  unite VARCHAR(3) DEFAULT 'KG' NOT NULL,  PRIMARY KEY (magasin\_id, produit\_id)  )ENGINE=InnoDB;  -- liaison entre le stock et les magasins  ALTER TABLE stock ADD CONSTRAINT magasin\_stock\_fk  FOREIGN KEY (magasin\_id)  REFERENCES magasin (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  -- liaison entre le stock et les produits  ALTER TABLE stock ADD CONSTRAINT produit\_stock\_fk  FOREIGN KEY (produit\_id)  REFERENCES produit (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  DESCRIBE stock;  ################################################################################  # COMMANDE #  ################################################################################  ####################################################################### PANIER #  DROP TABLE IF EXISTS panier;  CREATE TABLE panier (  utilisateur\_id INT UNSIGNED NOT NULL,  jour DATE DEFAULT (CURRENT\_DATE()) NOT NULL,  heure TIME DEFAULT (CURRENT\_TIME()) NOT NULL,  montant\_ttc DECIMAL(5,2) DEFAULT (0.0) NOT NULL,  livraison TINYINT UNSIGNED DEFAULT TRUE NOT NULL,  PRIMARY KEY (utilisateur\_id)  )ENGINE=InnoDB;  -- un panier est lié à un client  ALTER TABLE panier ADD CONSTRAINT client\_panier\_fk  FOREIGN KEY (utilisateur\_id)  REFERENCES client (utilisateur\_id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  DESCRIBE panier;  ############################################################## LIGNE\_DE\_PANIER #  DROP TABLE IF EXISTS ligne\_de\_panier;  CREATE TABLE ligne\_de\_panier (  utilisateur\_id INT UNSIGNED NOT NULL,  produit\_id INT UNSIGNED NOT NULL,  quantite DECIMAL(2) DEFAULT (0) NOT NULL,  prix\_unitaire\_ht DECIMAL(5,2) DEFAULT (0.0) NOT NULL,  taux\_tva DECIMAL(3,1) DEFAULT (0.0) NOT NULL,  PRIMARY KEY (utilisateur\_id, produit\_id)  )ENGINE=InnoDB;  -- une ligne de panier est liée à un client donc un utilisateur  ALTER TABLE ligne\_de\_panier ADD CONSTRAINT utilisateur\_ligne\_de\_panier\_fk  FOREIGN KEY (utilisateur\_id)  REFERENCES utilisateur (id)  ON DELETE CASCADE  ON UPDATE CASCADE;  -- une ligne de panier est liée à un produit  ALTER TABLE ligne\_de\_panier ADD CONSTRAINT produit\_ligne\_de\_panier\_fk  FOREIGN KEY (produit\_id)  REFERENCES produit (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  DESCRIBE ligne\_de\_panier;  ##################################################################### COMMANDE #  DROP TABLE IF EXISTS commande;  CREATE TABLE commande (  id INT UNSIGNED AUTO\_INCREMENT NOT NULL,  utilisateur\_id INT UNSIGNED NOT NULL,  adresse\_id INT UNSIGNED NOT NULL,  statut ENUM ('En attente', 'En préparation', 'Préparée', 'En livraison', 'Livrée', 'Clos') NOT NULL,  jour DATE DEFAULT (CURRENT\_DATE()) NOT NULL,  heure TIME DEFAULT (CURRENT\_TIME()) NOT NULL,  preparation\_delai TIME,  preparation\_duree TIME,  livraison\_delai TIME,  livraison\_duree TIME,  paiement\_OK BOOLEAN DEFAULT false NOT NULL,  montant\_ttc DECIMAL(5,2) DEFAULT (0.0) NOT NULL,  PRIMARY KEY (id)  )ENGINE=InnoDB;  -- une commande à une adresse de livraison unique  ALTER TABLE commande ADD CONSTRAINT adresse\_commande\_fk  FOREIGN KEY (adresse\_id)  REFERENCES adresse (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  -- une commande est liée à un utilisateur  ALTER TABLE commande ADD CONSTRAINT client\_commande\_fk  FOREIGN KEY (utilisateur\_id)  REFERENCES client (utilisateur\_id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  DESCRIBE commande;  ############################################################ LIGNE\_DE\_COMMANDE #  DROP TABLE IF EXISTS ligne\_de\_commande;  CREATE TABLE ligne\_de\_commande (  commande\_id INT UNSIGNED NOT NULL,  produit\_id INT UNSIGNED NOT NULL,  quantite DECIMAL(2) DEFAULT 1 NOT NULL,  prix\_unitaire\_ht DECIMAL(5,2) NOT NULL,  taux\_tva DECIMAL(3,1) DEFAULT (0.0) NOT NULL,  PRIMARY KEY (commande\_id, produit\_id)  )ENGINE=InnoDB;  -- chaque ligne de commande est liée à une commande  ALTER TABLE ligne\_de\_commande ADD CONSTRAINT commande\_ligne\_de\_commande\_fk  FOREIGN KEY (commande\_id)  REFERENCES commande (id)  ON DELETE CASCADE  ON UPDATE CASCADE;  -- chaque ligne de commande est liée à un produit  ALTER TABLE ligne\_de\_commande ADD CONSTRAINT produit\_ligne\_de\_commande\_fk  FOREIGN KEY (produit\_id)  REFERENCES produit (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  DESCRIBE ligne\_de\_commande;  ################################################################################  # PAIEMENT #  ################################################################################  ################################################################ ETABLISSEMENT #  DROP TABLE IF EXISTS etablissement;  CREATE TABLE etablissement (  id TINYINT UNSIGNED AUTO\_INCREMENT NOT NULL,  nom VARCHAR(20) NOT NULL,  PRIMARY KEY (id)  )ENGINE=InnoDB;  DESCRIBE etablissement;  ##################################################################### PAIEMENT #  DROP TABLE IF EXISTS paiement;  CREATE TABLE paiement (  id INT UNSIGNED AUTO\_INCREMENT NOT NULL,  type ENUM ('espèce','carte bancaire','ticket restaurant','chèque bancaire','sans') NOT NULL,  PRIMARY KEY (id)  )ENGINE=InnoDB;  DESCRIBE paiement;  ############################################################ TICKET\_RESTAURANT #  DROP TABLE IF EXISTS ticket\_restaurant;  CREATE TABLE ticket\_restaurant (  paiement\_id INT UNSIGNED NOT NULL,  numero VARCHAR(50) NOT NULL,  etablissement\_id TINYINT UNSIGNED NOT NULL,  PRIMARY KEY (paiement\_id)  )ENGINE=InnoDB;  -- ticket\_restaurant est un paiement  ALTER TABLE ticket\_restaurant ADD CONSTRAINT paiement\_ticket\_restaurant\_fk  FOREIGN KEY (paiement\_id)  REFERENCES paiement (id)  ON DELETE CASCADE  ON UPDATE CASCADE;  DESCRIBE ticket\_restaurant;  ############################################################### CARTE\_BANCAIRE #  DROP TABLE IF EXISTS carte\_bancaire;  CREATE TABLE carte\_bancaire (  paiement\_id INT UNSIGNED NOT NULL,  reference VARCHAR(100) NOT NULL,  jour DATE DEFAULT (CURRENT\_DATE()),  heure TIME DEFAULT (CURRENT\_TIME()),  PRIMARY KEY (paiement\_id)  )ENGINE=InnoDB;  -- carte\_bancaire est un paiement  ALTER TABLE carte\_bancaire ADD CONSTRAINT paiement\_carte\_bancaire\_fk  FOREIGN KEY (paiement\_id)  REFERENCES paiement (id)  ON DELETE CASCADE  ON UPDATE CASCADE;  DESCRIBE carte\_bancaire;  ####################################################################### CHEQUE #  DROP TABLE IF EXISTS cheque;  CREATE TABLE cheque (  paiement\_id INT UNSIGNED NOT NULL,  banque TINYINT UNSIGNED NOT NULL,  numero VARCHAR(100) NOT NULL,  jour DATE DEFAULT (CURRENT\_DATE()),  PRIMARY KEY (paiement\_id)  )ENGINE=InnoDB;  -- cheque est un paiement  ALTER TABLE cheque ADD CONSTRAINT paiement\_cheque\_fk  FOREIGN KEY (paiement\_id)  REFERENCES paiement (id)  ON DELETE CASCADE  ON UPDATE CASCADE;  DESCRIBE cheque;  ############################################################### LISTE PAIEMENT #  DROP TABLE IF EXISTS liste\_paiement;  CREATE TABLE liste\_paiement (  commande\_id INT UNSIGNED NOT NULL,  paiement\_id INT UNSIGNED NOT NULL,  montant DECIMAL(5,2) DEFAULT (0.0)NOT NULL,  PRIMARY KEY (commande\_id,paiement\_id )  )ENGINE=InnoDB;  -- une liste de paiement est liée à un paiement  ALTER TABLE liste\_paiement ADD CONSTRAINT paiement\_liste\_paiement\_fk  FOREIGN KEY (paiement\_id)  REFERENCES paiement (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  -- une liste de paiement est liée à une commande  ALTER TABLE liste\_paiement ADD CONSTRAINT commande\_liste\_paiement\_fk  FOREIGN KEY (commande\_id)  REFERENCES commande (id)  ON DELETE NO ACTION  ON UPDATE NO ACTION;  DESCRIBE liste\_paiement;  SHOW TABLES; |

## Création des procédures.

|  |
| --- |
| ################################################################################  # OC PIZZA CREATE PROCEDURES #  ################################################################################  ############################################################### CREATE ADRESSE #  DROP PROCEDURE IF EXISTS create\_adresse;  DELIMITER |  CREATE PROCEDURE create\_adresse(  IN p\_numero VARCHAR(5), -- numéro de la rue  IN p\_voie VARCHAR(50), -- nom de la voie  IN p\_ville VARCHAR(20), -- nom de la ville  IN p\_code VARCHAR(5), -- code postal  OUT p\_id INT(10) UNSIGNED) -- retourne identifiant de l'adresse correspondante  BEGIN  SET p\_id = 0;  -- on cherche si l'adresse existe déjà  SELECT DISTINCT id INTO p\_id FROM adresse  WHERE numero = p\_numero AND voie = p\_voie AND ville = p\_ville AND code = p\_code  ORDER BY id DESC LIMIT 1;  -- si l'adresse n'existe pas on la créé sinon on renvoie l'identifiant trouvé  IF p\_id = 0 THEN  INSERT INTO adresse(numero, voie, ville, code)  VALUES (p\_numero,p\_voie, p\_ville, p\_code);  #on recherche ID de l'adresse créée  SELECT DISTINCT id INTO p\_id  FROM adresse  WHERE numero=p\_numero AND voie=p\_voie AND ville = p\_ville AND code = p\_code  ORDER BY id DESC LIMIT 1;  END IF;  END |  DELIMITER ;  ############################################################### CREATE MAGASIN #  DROP PROCEDURE IF EXISTS create\_magasin;  DELIMITER |  CREATE PROCEDURE create\_magasin(  IN p\_nom VARCHAR(50), -- nom du magasin  IN p\_telephone VARCHAR(10), -- téléphone du magasin  IN p\_email VARCHAR(255), -- email du magasin  IN p\_numero VARCHAR(5), -- numéro de la rue  IN p\_voie VARCHAR(50), -- nom de la voie  IN p\_ville VARCHAR(20), -- nom de la ville  IN p\_code VARCHAR(5), -- code postal  OUT p\_id INT(10) UNSIGNED) -- retourne l'identifiant du magasin correspondant  BEGIN  DECLARE v\_adresse\_id INT(10) UNSIGNED;    SET p\_id = 0;  SET v\_adresse\_id = 0;  -- on effectue la création de l'adresse pour avoir l'identifiant  CALL create\_adresse(p\_numero, p\_voie, p\_ville, p\_code, v\_adresse\_id);  -- on cherche si un magasin existe déjà  SELECT DISTINCT id INTO p\_id FROM magasin  WHERE nom = p\_nom AND telephone = p\_telephone AND email = p\_email AND adresse\_id = v\_adresse\_id  ORDER BY id DESC LIMIT 1;  IF p\_id > 0 THEN  INSERT INTO erreur (message) VALUES ('ERREUR : le nom du magasin doit être unique!');  ELSE  INSERT INTO magasin (nom, telephone,email,adresse\_id)  VALUES (p\_nom, p\_telephone,p\_email,v\_adresse\_id);  #on recherche ID du magasin créée  SELECT id INTO p\_id  FROM magasin  WHERE nom=p\_nom;  END IF;  END |  DELIMITER ;  ########################################################### CREATE FOURNISSEUR #  DROP PROCEDURE IF EXISTS create\_fournisseur;  DELIMITER |  CREATE PROCEDURE create\_fournisseur(  IN p\_nom VARCHAR(50), -- nom du fournisseur  IN p\_telephone VARCHAR(10), -- téléphone du fournisseur  IN p\_email VARCHAR(255), -- email du fournisseur  IN p\_numero VARCHAR(5), -- numéro du fournisseur  IN p\_voie VARCHAR(50), -- nom de la voie  IN p\_ville VARCHAR(20), -- nom de la ville  IN p\_code VARCHAR(5), -- code postal  OUT p\_id INT(10) UNSIGNED) -- retourne l'identifiant du fournisseur créé  BEGIN  DECLARE v\_adresse\_id INT(10) UNSIGNED;    SET p\_id = 0;  SET v\_adresse\_id = 0;  -- création de l'adresse  CALL create\_adresse(p\_numero, p\_voie, p\_ville, p\_code, v\_adresse\_id);  -- recherche si le fournisseur existe  SELECT id INTO p\_id FROM fournisseur  WHERE nom = p\_nom AND telephone = p\_telephone AND email = p\_email AND adresse\_id = v\_adresse\_id  ORDER BY id DESC LIMIT 1;  -- si il existe il y a une erreur de duplication sinon on créé le fournisseur  IF p\_id > 0 THEN  INSERT INTO erreur (message) VALUES ('ERREUR : le nom du fournisseur doit être unique!');  ELSE  INSERT INTO fournisseur (nom, telephone,email,adresse\_id)  VALUES (p\_nom, p\_telephone,p\_email,v\_adresse\_id);    -- on recherche ID du fournisseur créée  SELECT id INTO p\_id  FROM fournisseur  WHERE nom=p\_nom;  END IF;  END |  DELIMITER ;  ################################################################################  # UTILISATEURS #  ################################################################################  ########################################################### CREATE UTILISATEUR #  DROP PROCEDURE IF EXISTS create\_utilisateur;  DELIMITER |  CREATE PROCEDURE create\_utilisateur(  IN p\_civilite ENUM ('Mlle','Mme','M'), -- civilité  IN p\_nom VARCHAR(50), -- nom de l'utilisateur  IN p\_prenom VARCHAR(50), -- prénom de l'utilisateur  IN p\_login VARCHAR(50), -- login  IN p\_mot\_de\_passe VARCHAR(255), -- mot de passe  IN p\_magasin\_id INT UNSIGNED, -- son magasin attitré  OUT p\_id INT(10) UNSIGNED) -- retourne l'identifiant  BEGIN  SET p\_id = 0;  SELECT id INTO p\_id FROM utilisateur  WHERE login = p\_login;  IF p\_id > 0 THEN  INSERT INTO erreur (message) VALUES ('ERREUR : le login doit être unique!');  ELSE  INSERT INTO utilisateur (civilite, nom, prenom, login, mot\_de\_passe, magasin\_id)  VALUES (p\_civilite, p\_nom, p\_prenom, p\_login, p\_mot\_de\_passe, p\_magasin\_id);  #on recherche ID de l'utilisateur créée  SELECT id INTO p\_id  FROM utilisateur  WHERE login = p\_login;  END IF;  END |  DELIMITER ;  ############################################################### CREATE EMPLOYE #  DROP PROCEDURE IF EXISTS create\_employe;  DELIMITER |  CREATE PROCEDURE create\_employe(  IN p\_civilite ENUM ('Mlle','Mme','M'), -- civilité  IN p\_nom VARCHAR(50), -- nom de l'employé  IN p\_prenom VARCHAR(50), -- prénom de l'employé  IN p\_login VARCHAR(50), -- login de l'employé  IN p\_mot\_de\_passe VARCHAR(255), -- mot de passe de l'employé  IN p\_magasin\_id INT(10) UNSIGNED, -- magasin ou il travaille  IN p\_role ENUM ('Accueil','Pizzaiolo','Livreur','Manager',  'Gestionnaire','Comptable','Direction'), -- son poste  OUT p\_id INT(10) UNSIGNED) -- retourne l'identifiant  BEGIN  CALL create\_utilisateur(p\_civilite, p\_nom, p\_prenom, p\_login, p\_mot\_de\_passe, p\_magasin\_id, p\_id);  INSERT INTO employe (utilisateur\_id, role)  VALUES (p\_id, p\_role);  END |  DELIMITER ;  ################################################################ CREATE CLIENT #  DROP PROCEDURE IF EXISTS create\_client;  DELIMITER |  CREATE PROCEDURE create\_client(  IN p\_civilite ENUM ('Mlle','Mme','M'), -- civilité  IN p\_nom VARCHAR(50), -- nom du client  IN p\_prenom VARCHAR(50), -- prénom du client  IN p\_login VARCHAR(50), -- login du client  IN p\_mot\_de\_passe VARCHAR(255), -- mot de passe du client  IN p\_magasin\_id INT(10) UNSIGNED, -- magasin préféré  IN p\_telephone VARCHAR(10), -- téléphone du client  IN p\_email VARCHAR(255), -- email du client  IN p\_numero VARCHAR(5), -- addresse simplifiée  IN p\_voie VARCHAR(50),  IN p\_ville VARCHAR(20),  IN p\_code VARCHAR(5),  OUT p\_id INT(10) UNSIGNED) -- retourne l'identifiant  BEGIN  DECLARE v\_adresse\_id INT(10) UNSIGNED;    CALL create\_adresse(p\_numero, p\_voie, p\_ville, p\_code, v\_adresse\_id);  CALL create\_utilisateur(p\_civilite, p\_nom, p\_prenom, p\_login, p\_mot\_de\_passe, p\_magasin\_id, p\_id);  INSERT INTO client (utilisateur\_id, telephone, adresse\_id, email)  VALUES (p\_id, p\_telephone, v\_adresse\_id, p\_email);  END |  DELIMITER ;  ################################################################################  # PAIEMENT #  ################################################################################  ############################################################## CREATE PAIEMENT #  DROP PROCEDURE IF EXISTS create\_paiement;  DELIMITER |  CREATE PROCEDURE create\_paiement(  IN p\_type ENUM ('espèce','carte bancaire','ticket restaurant','chèque bancaire','sans'),  OUT p\_id INT(10) UNSIGNED)  BEGIN  INSERT INTO paiement (type) VALUES (p\_type);  SELECT id INTO p\_id FROM paiement  WHERE type = p\_type  ORDER BY id DESC LIMIT 1;  END |  DELIMITER ;  ############################################### ADD PAIEMENT TICKET RESTAURANT #  DROP PROCEDURE IF EXISTS add\_paiement\_ticket\_restaurant;  DELIMITER |  CREATE PROCEDURE add\_paiement\_ticket\_restaurant(  IN p\_commande\_id INT(10) UNSIGNED,  IN p\_montant DECIMAL(5,2),  IN p\_numero VARCHAR(50),  IN p\_etablissement\_id TINYINT UNSIGNED,  OUT p\_paiement\_id INT(10) UNSIGNED)  BEGIN  CALL create\_paiement('ticket restaurant',p\_paiement\_id);  INSERT INTO ticket\_restaurant (paiement\_id,numero,etablissement\_id)  VALUES (p\_paiement\_id, p\_numero, p\_etablissement\_id);  INSERT INTO liste\_paiement (commande\_id,paiement\_id,montant)  VALUES (p\_commande\_id, p\_paiement\_id, p\_montant);  IF reste\_du(p\_commande\_id) = 0 THEN  UPDATE commande SET paiement\_ok = TRUE WHERE id = p\_commande\_id;  END IF;  END |  DELIMITER ;  ################################################## ADD PAIEMENT CARTE BANCAIRE #  DROP PROCEDURE IF EXISTS add\_paiement\_carte\_bancaire;  DELIMITER |  CREATE PROCEDURE add\_paiement\_carte\_bancaire(  IN p\_commande\_id INT(10) UNSIGNED,  IN p\_montant DECIMAL(5,2),  IN p\_reference VARCHAR(100),  OUT p\_paiement\_id INT(10) UNSIGNED)  BEGIN  CALL create\_paiement('carte bancaire',p\_paiement\_id);  INSERT INTO carte\_bancaire (paiement\_id,reference,jour,heure)  VALUES (p\_paiement\_id, p\_reference,CURRENT\_DATE(), CURRENT\_TIME());  INSERT INTO liste\_paiement (commande\_id,paiement\_id,montant)  VALUES (p\_commande\_id, p\_paiement\_id, p\_montant);  IF reste\_du(p\_commande\_id) = 0 THEN  UPDATE commande SET paiement\_ok = TRUE WHERE id = p\_commande\_id;  END IF;  END |  DELIMITER ;  ########################################################## ADD PAIEMENT CHEQUE #  DROP PROCEDURE IF EXISTS add\_paiement\_cheque;  DELIMITER |  CREATE PROCEDURE add\_paiement\_cheque(  IN p\_commande\_id INT(10) UNSIGNED,  IN p\_montant DECIMAL(5,2),  IN p\_banque TINYINT UNSIGNED,  IN p\_numero VARCHAR(100),  OUT p\_paiement\_id INT(10) UNSIGNED)  BEGIN  CALL create\_paiement('chèque bancaire',p\_paiement\_id);  INSERT INTO cheque (paiement\_id,banque,numero,jour)  VALUES (p\_paiement\_id, p\_banque,p\_numero,CURRENT\_DATE);  INSERT INTO liste\_paiement (commande\_id,paiement\_id,montant)  VALUES (p\_commande\_id, p\_paiement\_id, p\_montant);  IF reste\_du(p\_commande\_id) = 0 THEN  UPDATE commande SET paiement\_ok = TRUE WHERE id = p\_commande\_id;  END IF;  END |  DELIMITER ;  ########################################################## ADD PAIEMENT ESPECE #  DROP PROCEDURE IF EXISTS add\_paiement\_espece;  DELIMITER |  CREATE PROCEDURE add\_paiement\_espece(  IN p\_commande\_id INT(10) UNSIGNED,  IN p\_montant DECIMAL(5,2),  OUT p\_paiement\_id INT(10) UNSIGNED)  BEGIN  CALL create\_paiement('espèce',p\_paiement\_id);  INSERT INTO liste\_paiement (commande\_id,paiement\_id,montant)  VALUES (p\_commande\_id, p\_paiement\_id, p\_montant);  IF reste\_du(p\_commande\_id) = 0 THEN  UPDATE commande SET paiement\_ok = TRUE WHERE id = p\_commande\_id;  END IF;  END |  DELIMITER ;  ##################################################################### RESTE DU #  DROP FUNCTION IF EXISTS reste\_du;  DELIMITER |  CREATE FUNCTION reste\_du(  p\_commande\_id INT(10) UNSIGNED)  RETURNS DECIMAL(5,2)  DETERMINISTIC  BEGIN  DECLARE v\_total\_paiement DECIMAL(5,2);  DECLARE v\_montant DECIMAL(5,2);  SELECT montant\_TTC INTO v\_montant FROM commande WHERE id = p\_commande\_id;  SELECT SUM(montant) INTO v\_total\_paiement FROM liste\_paiement WHERE commande\_id = p\_commande\_id;  IF v\_total\_paiement IS NULL THEN  RETURN (v\_montant);  ELSE  RETURN (v\_montant - v\_total\_paiement);  END IF;  END |  DELIMITER ;  ################################################################################  # PRODUIT #  ################################################################################  ############################################################### CREATE PRODUIT #  DROP PROCEDURE IF EXISTS create\_produit;  DELIMITER |  CREATE PROCEDURE create\_produit(  IN p\_designation VARCHAR(100),  IN p\_categorie ENUM ('pack','vrac','ingrédient','pizza','boisson','dessert','emballage','sauce'),  IN p\_fournisseur\_id INT UNSIGNED,  IN p\_reference VARCHAR(20),  IN p\_quantite DECIMAL(5,2),  IN p\_unite VARCHAR(3),  IN p\_prix\_achat\_ht DECIMAL(5,2),  IN p\_prix\_vente\_ht DECIMAL(5,2),  IN p\_tva\_emporte DECIMAL(3,1),  IN p\_tva\_livre DECIMAL(3,1),  IN p\_formule TEXT,  IN p\_recette TEXT,  OUT p\_id INT(10) UNSIGNED)  BEGIN  INSERT INTO produit (designation,categorie,fournisseur\_id,reference,quantite,unite,prix\_achat\_ht,prix\_vente\_ht,tva\_emporte,tva\_livre)  VALUES (p\_designation,p\_categorie,p\_fournisseur\_id,p\_reference,p\_quantite,p\_unite,p\_prix\_achat\_ht,p\_prix\_vente\_ht,p\_tva\_emporte,p\_tva\_livre);  -- on récupère l'identifiant du produit créé  SELECT id INTO p\_id FROM produit WHERE designation = p\_designation;  -- création de la composition  IF p\_formule IS NOT NULL THEN  INSERT INTO composition (produit\_id,formule) VALUES (p\_id,p\_formule);  END IF;  -- création de la recette  IF p\_recette IS NOT NULL THEN  INSERT INTO preparation (produit\_id,recette) VALUES (p\_id,p\_recette);  END IF;  END |  DELIMITER ;  ########################################################## CHERCHE PRODUIT ID #  DROP PROCEDURE IF EXISTS cherche\_produit\_id;  DELIMITER |  CREATE PROCEDURE cherche\_produit\_id(  IN p\_designation VARCHAR(100),  IN p\_categorie ENUM ('pack','vrac','ingrédient','pizza','boisson','dessert','emballage','sauce'),  OUT p\_id INT(10) UNSIGNED)  BEGIN    SELECT DISTINCT id INTO p\_id FROM produit  WHERE categorie = p\_categorie AND designation LIKE CONCAT("%", p\_designation, "%");  END |  DELIMITER ;  ################################################################ ADD COMPOSANT #  DROP PROCEDURE IF EXISTS add\_composant;  DELIMITER |  CREATE PROCEDURE add\_composant(  IN p\_produit\_id INT(10) UNSIGNED,  IN p\_ingredient\_id INT(10) UNSIGNED,  IN p\_quantite DECIMAL(5,2),  IN p\_unite VARCHAR(3))  BEGIN  INSERT INTO composant (produit\_id,ingredient\_id,quantite,unite)  VALUES (p\_produit\_id,p\_ingredient\_id,p\_quantite,p\_unite);  END |  DELIMITER ;  #################################################################### LIVRAISON #  DROP PROCEDURE IF EXISTS livraison;  DELIMITER |  CREATE PROCEDURE livraison(  IN p\_magasin\_id INT(10) UNSIGNED,  IN p\_produit\_id INT(10) UNSIGNED,  IN p\_quantite DECIMAL(5,2))  BEGIN  DECLARE v\_unite\_par\_produit DECIMAL(5,2) DEFAULT (1.0);  DECLARE v\_vrac\_id INT(10) UNSIGNED;  DECLARE v\_exist INT DEFAULT 0 ;  DECLARE v\_stock\_initial DECIMAL(5,2) DEFAULT (0.0);  SELECT quantite,ingredient\_id INTO v\_unite\_par\_produit,v\_vrac\_id FROM composant WHERE produit\_id = p\_produit\_id;  SELECT quantite, COUNT(\*) INTO v\_stock\_initial,v\_exist FROM stock  WHERE magasin\_id = p\_magasin\_id AND produit\_id = v\_vrac\_id;  IF v\_exist = 0 THEN  INSERT INTO stock (magasin\_id,produit\_id,quantite)  VALUES (p\_magasin\_id, v\_vrac\_id,p\_quantite\*v\_unite\_par\_produit);  ELSE  UPDATE stock  SET quantite = p\_quantite\*v\_unite\_par\_produit + v\_stock\_initial  WHERE magasin\_id = p\_magasin\_id AND produit\_id = v\_vrac\_id;  END IF;  END |  DELIMITER ;  ############################################################ LIVRE MAGASIN #  DROP PROCEDURE IF EXISTS livre\_magasin;  DELIMITER |  CREATE PROCEDURE livre\_magasin()  BEGIN  DECLARE v\_magasin\_id INT(10) DEFAULT 1;  DECLARE v\_produit\_id INT(10) DEFAULT 1;  DECLARE done INT DEFAULT FALSE;  DECLARE curs\_produit CURSOR FOR SELECT id FROM produit WHERE categorie IN ("ingrédient","pack");  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  OPEN curs\_produit;  loop\_curseur: LOOP  FETCH curs\_produit INTO v\_produit\_id;  IF done THEN  LEAVE loop\_curseur;  END IF;  REPEAT  CALL livraison(v\_magasin\_id,v\_produit\_id,1);  SET v\_magasin\_id = v\_magasin\_id + 1;  UNTIL v\_magasin\_id > 4  END REPEAT;  SET v\_magasin\_id = 1;  END LOOP loop\_curseur;  CLOSE curs\_produit;  END |  DELIMITER ;  ################################################################################  # COMMANDE #  ################################################################################  ######################################################## UPDATE MONTANT PANIER #  DROP PROCEDURE IF EXISTS update\_panier;  DELIMITER |  CREATE PROCEDURE update\_panier(  IN p\_utilisateur\_id INT(10),  IN p\_livraison TINYINT UNSIGNED)  BEGIN  DECLARE v\_montant DECIMAL(5,2) DEFAULT (0.0);  DECLARE v\_livraison\_modifie TINYINT DEFAULT FALSE;  -- on cherche si on a modifier le type de livraison  SELECT (livraison <> p\_livraison) INTO v\_livraison\_modifie FROM panier WHERE utilisateur\_id = p\_utilisateur\_id;  IF v\_livraison\_modifie THEN  IF p\_livraison THEN  -- livraison  UPDATE ligne\_de\_panier  SET taux\_tva = (SELECT tva\_livre FROM produit WHERE produit.id = ligne\_de\_panier.produit\_id)  WHERE ligne\_de\_panier.utilisateur\_id = p\_utilisateur\_id;  ELSE  -- take away  UPDATE ligne\_de\_panier  SET taux\_tva = (SELECT tva\_emporte FROM produit WHERE produit.id = ligne\_de\_panier.produit\_id)  WHERE ligne\_de\_panier.utilisateur\_id = p\_utilisateur\_id;  END IF;  END IF;  SELECT SUM(quantite\*prix\_unitaire\_ht\*(1+taux\_tva/100)) INTO v\_montant FROM ligne\_de\_panier WHERE utilisateur\_id = p\_utilisateur\_id;  INSERT INTO panier (utilisateur\_id,jour,heure,montant\_ttc,livraison)  VALUES ( p\_utilisateur\_id,CURRENT\_DATE(), CURRENT\_TIME(), v\_montant, p\_livraison)  ON DUPLICATE KEY UPDATE jour = CURRENT\_DATE(), heure = CURRENT\_TIME(), montant\_ttc = v\_montant, livraison = p\_livraison;  END |  DELIMITER ;  ################################################################# AJOUT PANIER #  DROP PROCEDURE IF EXISTS ajoute\_panier;  DELIMITER |  CREATE PROCEDURE ajoute\_panier(  IN p\_utilisateur\_id INT(10),  IN p\_produit\_id INT(10),  IN p\_quantite DECIMAL(2,0),  IN p\_livraison TINYINT UNSIGNED)  BEGIN  DECLARE v\_quantite\_old DECIMAL(2,0) DEFAULT (0.0);  DECLARE v\_prix\_unitaire\_ht DECIMAL(5,2) DEFAULT (0.0);  DECLARE v\_taux\_tva DECIMAL(5,2) DEFAULT (0.0);    -- on récupère l'ancienne quantité  SELECT quantite INTO v\_quantite\_old FROM ligne\_de\_panier  WHERE utilisateur\_id = p\_utilisateur\_id AND produit\_id = p\_produit\_id;  IF v\_quantite\_old IS NULL THEN  SET v\_quantite\_old = 0;  END IF;  IF p\_livraison THEN  -- livraison à domicile  SELECT prix\_vente\_ht, tva\_livre INTO v\_prix\_unitaire\_ht,v\_taux\_tva FROM produit WHERE id = p\_produit\_id;  ELSE  -- vente à emporter  SELECT prix\_vente\_ht, tva\_emporte INTO v\_prix\_unitaire\_ht,v\_taux\_tva FROM produit WHERE id = p\_produit\_id;  END IF;    INSERT INTO ligne\_de\_panier (utilisateur\_id,produit\_id,quantite,prix\_unitaire\_ht,taux\_tva)  VALUES (p\_utilisateur\_id,p\_produit\_id,p\_quantite+v\_quantite\_old,v\_prix\_unitaire\_ht,v\_taux\_tva)  ON DUPLICATE KEY UPDATE quantite = p\_quantite + v\_quantite\_old, prix\_unitaire\_ht = v\_prix\_unitaire\_ht, taux\_tva = v\_taux\_tva;  CALL update\_panier(p\_utilisateur\_id,p\_livraison);  END |  DELIMITER ;  ################################################################ ENLEVE PANIER #  DROP PROCEDURE IF EXISTS enleve\_panier;  DELIMITER |  CREATE PROCEDURE enleve\_panier(  IN p\_utilisateur\_id INT(10),  IN p\_produit\_id INT(10),  IN p\_quantite DECIMAL(2,0),  IN p\_livraison TINYINT UNSIGNED)  BEGIN  DECLARE v\_quantite\_old DECIMAL(2,0) DEFAULT (0.0);  DECLARE v\_prix\_unitaire\_ht DECIMAL(5,2) DEFAULT (0.0);  DECLARE v\_taux\_tva DECIMAL(5,2) DEFAULT (0.0);    -- on récupère l'ancienne quantité  SELECT quantite INTO v\_quantite\_old FROM ligne\_de\_panier  WHERE utilisateur\_id = p\_utilisateur\_id AND produit\_id = p\_produit\_id;  -- on sélectionne le prix et la tva suivant la livraison  IF p\_livraison THEN  -- livraison à domicile  SELECT prix\_vente\_ht, tva\_livre INTO v\_prix\_unitaire\_ht,v\_taux\_tva FROM produit WHERE id = p\_produit\_id;  ELSE  -- vente à emporter  SELECT prix\_vente\_ht, tva\_emporte INTO v\_prix\_unitaire\_ht,v\_taux\_tva FROM produit WHERE id = p\_produit\_id;  END IF;  -- on diminue la quantité  IF v\_quantite\_old IS NULL OR v\_quantite\_old <= p\_quantite THEN  DELETE FROM ligne\_de\_panier WHERE utilisateur\_id = p\_utilisateur\_id AND produit\_id = p\_produit\_id;  ELSE  INSERT INTO ligne\_de\_panier (utilisateur\_id,produit\_id,quantite,prix\_unitaire\_ht,taux\_tva)  VALUES (p\_utilisateur\_id,p\_produit\_id,v\_quantite\_old - p\_quantite,v\_prix\_unitaire\_ht,v\_taux\_tva)  ON DUPLICATE KEY UPDATE quantite = v\_quantite\_old - p\_quantite, prix\_unitaire\_ht = v\_prix\_unitaire\_ht, taux\_tva = v\_taux\_tva;  END IF;  -- on met à jour le panier  CALL update\_panier(p\_utilisateur\_id,p\_livraison);  END |  DELIMITER ;  ################################################################ DIMINUE STOCK #  DROP PROCEDURE IF EXISTS diminue\_stock;  DELIMITER |  CREATE PROCEDURE diminue\_stock(  IN p\_magasin\_id INT(10),  IN p\_produit\_id INT(10),  IN p\_quantite DECIMAL(5,2))  BEGIN  DECLARE v\_compose INT DEFAULT FALSE;  DECLARE done INT DEFAULT FALSE;  DECLARE v\_quantite\_old DECIMAL(5,2);  DECLARE v\_produit\_id INT(10) UNSIGNED;  DECLARE v\_quantite DECIMAL(5,2);  DECLARE curs\_produit\_panier CURSOR FOR  SELECT ingredient\_id, quantite FROM composant WHERE produit\_id = p\_produit\_id;  DECLARE CONTINUE HANDLER FOR  NOT FOUND SET done = TRUE;    -- on vérifie si le produit est composé  SELECT (COUNT(\*)>0) INTO v\_compose FROM composant WHERE produit\_id = p\_produit\_id;  IF v\_compose THEN  -- le produit est composé on utilise le cursor  OPEN curs\_produit\_panier;  loop\_curseur: LOOP  -- On récupère les valeurs du curseur dans deux variables  FETCH curs\_produit\_panier INTO v\_produit\_id,v\_quantite;  IF done THEN  LEAVE loop\_curseur;  END IF;  SELECT quantite INTO v\_quantite\_old FROM stock WHERE magasin\_id = p\_magasin\_id AND produit\_id = v\_produit\_id;  IF v\_quantite\_old <= v\_quantite THEN  UPDATE stock SET quantite = 0 WHERE magasin\_id = p\_magasin\_id AND produit\_id = v\_produit\_id;  ELSE  UPDATE stock SET quantite = v\_quantite\_old - v\_quantite WHERE magasin\_id = p\_magasin\_id AND produit\_id = v\_produit\_id;  END IF;  END LOOP loop\_curseur;  CLOSE curs\_produit\_panier;  ELSE  -- produit simple  SELECT quantite INTO v\_quantite\_old FROM stock WHERE magasin\_id = p\_magasin\_id AND produit\_id = p\_produit\_id;  IF v\_quantite\_old <= p\_quantite THEN  UPDATE stock SET quantite = 0 WHERE magasin\_id = p\_magasin\_id AND produit\_id = v\_produit\_id;  ELSE  UPDATE stock SET quantite = v\_quantite\_old - p\_quantite WHERE magasin\_id = p\_magasin\_id AND produit\_id = v\_produit\_id;  END IF;  END IF;    END |  DELIMITER ;  ############################################################## VALIDE COMMANDE #  DROP PROCEDURE IF EXISTS valide\_commande;  DELIMITER |  CREATE PROCEDURE valide\_commande(  IN p\_utilisateur\_id INT(10) UNSIGNED,  OUT p\_commande\_id INT(10) UNSIGNED)  BEGIN  DECLARE v\_jour DATE DEFAULT CURRENT\_DATE();  DECLARE v\_heure TIME DEFAULT CURRENT\_TIME();  DECLARE v\_montant\_ttc DECIMAL(5,2) DEFAULT (0.0);  DECLARE v\_adresse\_id INT(10) UNSIGNED;  DECLARE v\_livraison TINYINT UNSIGNED DEFAULT TRUE;  DECLARE v\_magasin\_id INT(10) UNSIGNED;  DECLARE v\_produit\_id INT(10) UNSIGNED;  DECLARE v\_quantite DECIMAL(5,2) DEFAULT (0.0);  DECLARE done INT DEFAULT FALSE;  DECLARE curseur\_verification CURSOR FOR  SELECT produit\_id,quantite FROM ligne\_de\_panier WHERE utilisateur\_id = p\_utilisateur\_id;  DECLARE curseur\_modification CURSOR FOR  SELECT produit\_id,quantite FROM ligne\_de\_panier WHERE utilisateur\_id = p\_utilisateur\_id;  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  -- on sélectionne le magasin du client pour chercher les stock  SELECT magasin\_id INTO v\_magasin\_id FROM utilisateur WHERE id = p\_utilisateur\_id;  -- vérification de la présence de tous les produits en stock  OPEN curseur\_verification;  loop\_vérification: LOOP  FETCH curseur\_verification INTO v\_produit\_id,v\_quantite;  IF done THEN  LEAVE loop\_vérification;  END IF;  IF produit\_est\_disponible(v\_magasin\_id,v\_produit\_id,v\_quantite) = FALSE THEN  -- on enlève le produit correspondant et on provoque une erreur  CALL enleve\_panier(p\_utilisateur\_id,v\_produit\_id,v\_quantite);  INSERT INTO erreur (message) VALUES ('ERREUR : un produit n''est plus disponible!');  END IF;  END LOOP loop\_vérification;  CLOSE curseur\_verification;  -- on récupère le montant du panier et savoir si c'est une livraison  SELECT montant\_ttc,livraison INTO v\_montant\_ttc,v\_livraison from panier WHERE utilisateur\_id = p\_utilisateur\_id;  IF v\_livraison THEN  -- on prend l'adresse du client  SELECT adresse.id INTO v\_adresse\_id FROM adresse  JOIN client ON adresse.id = client.adresse\_id  WHERE client.utilisateur\_id = p\_utilisateur\_id;  ELSE  -- on prend l'adresse du magasin  SELECT adresse.id INTO v\_adresse\_id FROM adresse  JOIN magasin ON adresse.id = magasin.adresse\_id  WHERE magasin.id = v\_magasin\_id;  END IF;  -- on créée une nouvelle commande a  INSERT INTO commande (utilisateur\_id, adresse\_id, statut, jour, heure, montant\_ttc)  VALUES (p\_utilisateur\_id, v\_adresse\_id, 'En attente', v\_jour, v\_heure, v\_montant\_ttc);  -- on récupère l'identifiant de la commande  SELECT DISTINCT id INTO p\_commande\_id FROM commande  WHERE utilisateur\_id = p\_utilisateur\_id AND adresse\_id = v\_adresse\_id AND statut = 'En attente' AND jour = v\_jour AND heure = v\_heure;  -- diminution des stocks  SET done = FALSE;    OPEN curseur\_modification;  loop\_modification: LOOP  FETCH curseur\_modification INTO v\_produit\_id,v\_quantite;  IF done THEN  LEAVE loop\_modification;  END IF;  CALL diminue\_stock(v\_magasin\_id,v\_produit\_id,v\_produit\_id);  END LOOP loop\_modification;  CLOSE curseur\_modification;  -- on copie les lignes de panier dans les lignes de commande  INSERT INTO ligne\_de\_commande (commande\_id,produit\_id,quantite,prix\_unitaire\_ht, taux\_tva)  SELECT p\_commande\_id,produit\_id,quantite,prix\_unitaire\_ht, taux\_tva FROM ligne\_de\_panier  WHERE utilisateur\_id = p\_utilisateur\_id;      -- on vide le panier  DELETE FROM ligne\_de\_panier WHERE utilisateur\_id = p\_utilisateur\_id;  DELETE FROM panier WHERE utilisateur\_id = p\_utilisateur\_id;  END |  DELIMITER ;  ##################################################### PIZZAIOLO PREND COMMANDE #  DROP PROCEDURE IF EXISTS pizzaiolo\_prend\_commande;  DELIMITER |  CREATE PROCEDURE pizzaiolo\_prend\_commande(  IN p\_commande\_id INT(10) UNSIGNED)  BEGIN  DECLARE v\_start\_date DATE;  DECLARE v\_start\_time TIME;  DECLARE v\_preparation\_delai TIME;  DECLARE v\_statut ENUM ('En attente', 'En préparation', 'Préparée', 'En livraison', 'Livrée', 'Clos');  SELECT statut, jour, heure INTO v\_statut,v\_start\_date,v\_start\_time FROM commande WHERE id = p\_commande\_id;  -- le pizzaiolo ne peut prendre que les statut en attente  IF v\_statut = 'En attente' THEN  SET v\_preparation\_delai = TIMEDIFF(NOW(),TIMESTAMP(v\_start\_date,v\_start\_time));  UPDATE commande SET statut = 'En préparation', preparation\_delai = v\_preparation\_delai WHERE id = p\_commande\_id;  ELSE  INSERT INTO erreur (message) VALUES ('ERREUR : la commande n''est pas en attente!');  END IF;  END |  DELIMITER ;  ################################################### PIZZAIOLO TERMINE COMMANDE #  DROP PROCEDURE IF EXISTS pizzaiolo\_termine\_commande;  DELIMITER |  CREATE PROCEDURE pizzaiolo\_termine\_commande(  IN p\_commande\_id INT(10) UNSIGNED)  BEGIN  DECLARE v\_start\_date DATE;  DECLARE v\_start\_time TIME;  DECLARE v\_preparation\_delai TIME;  DECLARE v\_preparation\_duree TIME;  DECLARE v\_statut ENUM ('En attente', 'En préparation', 'Préparée', 'En livraison', 'Livrée', 'Clos');  SELECT statut, jour, heure, preparation\_delai INTO v\_statut, v\_start\_date,v\_start\_time,v\_preparation\_delai FROM commande WHERE id = p\_commande\_id;  -- le pizzaiolo ne peut terminer une commande qui n'est pas en préparation  IF v\_statut = 'En préparation' THEN  SET v\_preparation\_duree = TIMEDIFF(NOW(),TIMESTAMP(v\_start\_date,v\_start\_time)) - v\_preparation\_delai;  UPDATE commande SET statut = 'Préparée', preparation\_duree = v\_preparation\_duree WHERE id = p\_commande\_id;  ELSE  INSERT INTO erreur (message) VALUES ('ERREUR : la commande n''est pas en préparation!');  END IF;  END |  DELIMITER ;  ####################################################### LIVREUR PREND COMMANDE #  DROP PROCEDURE IF EXISTS livreur\_prend\_commande;  DELIMITER |  CREATE PROCEDURE livreur\_prend\_commande(  IN p\_commande\_id INT(10) UNSIGNED)  BEGIN  DECLARE v\_start\_date DATE;  DECLARE v\_start\_time TIME;  DECLARE v\_preparation\_delai TIME;  DECLARE v\_preparation\_duree TIME;  DECLARE v\_livraison\_delai TIME;  DECLARE v\_statut ENUM ('En attente', 'En préparation', 'Préparée', 'En livraison', 'Livrée', 'Clos');  SELECT statut, jour, heure, preparation\_delai,preparation\_duree  INTO v\_statut,v\_start\_date,v\_start\_time, v\_preparation\_delai, v\_preparation\_duree  FROM commande WHERE id = p\_commande\_id;  IF v\_statut = 'Préparée' THEN  SET v\_livraison\_delai = TIMEDIFF(NOW(),TIMESTAMP(v\_start\_date,v\_start\_time)) - v\_preparation\_delai - v\_preparation\_duree;  UPDATE commande SET statut = 'En livraison', livraison\_delai = v\_livraison\_delai WHERE id = p\_commande\_id;  ELSE  INSERT INTO erreur (message) VALUES ('ERREUR : la commande n''est pas préparée!');  END IF;  END |  DELIMITER ;  ######################################################## CLIENT PREND COMMANDE #  DROP PROCEDURE IF EXISTS client\_prend\_commande;  DELIMITER |  CREATE PROCEDURE client\_prend\_commande(  IN p\_commande\_id INT(10) UNSIGNED)  BEGIN  DECLARE v\_start\_date DATE;  DECLARE v\_start\_time TIME;  DECLARE v\_preparation\_delai TIME;  DECLARE v\_preparation\_duree TIME;  DECLARE v\_livraison\_delai TIME;  DECLARE v\_livraison\_duree TIME;  DECLARE v\_statut ENUM ('En attente', 'En préparation', 'Préparée', 'En livraison', 'Livrée', 'Clos');  SELECT statut, jour, heure, preparation\_delai,preparation\_duree,livraison\_delai  INTO v\_statut, v\_start\_date,v\_start\_time, v\_preparation\_delai, v\_preparation\_duree, v\_livraison\_delai  FROM commande WHERE id = p\_commande\_id;  -- la commande doit être préparée pour pouvoir passer en status livrée  IF v\_statut <> 'Préparée' AND v\_statut <> 'En livraison' THEN  INSERT INTO erreur (message) VALUES ('ERREUR : la commande n''est pas préparée!');  END IF;  -- un client peut prendre la commande directement en magasin donc elle ne passe pas par la livraison  IF v\_livraison\_delai IS NULL THEN  SET v\_livraison\_duree = TIMEDIFF(NOW(),TIMESTAMP(v\_start\_date,v\_start\_time)) - v\_preparation\_delai - v\_preparation\_duree;  UPDATE commande SET statut = 'Livrée', livraison\_delai = 0, livraison\_duree = v\_livraison\_duree WHERE id = p\_commande\_id;  ELSE  SET v\_livraison\_duree = TIMEDIFF(NOW(),TIMESTAMP(v\_start\_date,v\_start\_time)) - v\_preparation\_delai - v\_preparation\_duree - v\_livraison\_delai;  UPDATE commande SET statut = 'Livrée', livraison\_duree = v\_preparation\_duree WHERE id = p\_commande\_id;  END IF;  END |  DELIMITER ;  #################################################### LISTE MAGASIN SUIVANT NOM #  DROP PROCEDURE IF EXISTS liste\_magasin\_suivant\_nom;  DELIMITER |  CREATE PROCEDURE liste\_magasin\_suivant\_nom(  IN p\_nom VARCHAR(50))  BEGIN  IF p\_nom IS NULL OR p\_nom = '\*'THEN  SELECT magasin.id,  magasin.nom,  magasin.telephone,  magasin.email,  CONCAT\_WS("",adresse.appartement,  adresse.etage,  adresse.couloir,  adresse.escalier,  adresse.entree,  adresse.immeuble,  adresse.residence,  adresse.numero,  adresse.voie,  adresse.place,  adresse.code,  adresse.ville,  adresse.pays) AS adresse  FROM magasin  INNER JOIN adresse  ON magasin.adresse\_id = adresse.id;  ELSE  SELECT magasin.id,  magasin.nom,  magasin.telephone,  magasin.email,  CONCAT\_WS("",adresse.appartement,  adresse.etage,  adresse.couloir,  adresse.escalier,  adresse.entree,  adresse.immeuble,  adresse.residence,  adresse.numero,  adresse.voie,  adresse.place,  adresse.code,  adresse.ville,  adresse.pays) AS adresse  FROM magasin  INNER JOIN adresse  ON magasin.adresse\_id = adresse.id  WHERE nom LIKE CONCAT('%',p\_nom,'%');  END IF;  END |  DELIMITER ;  ################################################ LISTE FOURNISSEUR SUIVANT NOM #  DROP PROCEDURE IF EXISTS liste\_fournisseur\_suivant\_nom;  DELIMITER |  CREATE PROCEDURE liste\_fournisseur\_suivant\_nom(  IN p\_nom VARCHAR(50))  BEGIN  IF p\_nom IS NULL OR p\_nom = '\*'THEN  SELECT fournisseur.id,  fournisseur.nom,  fournisseur.telephone,  fournisseur.email,  CONCAT\_WS("",adresse.appartement,  adresse.etage,  adresse.couloir,  adresse.escalier,  adresse.entree,  adresse.immeuble,  adresse.residence,  adresse.numero,  adresse.voie,  adresse.place,  adresse.code,  adresse.ville,  adresse.pays) AS adresse  FROM fournisseur  INNER JOIN adresse  ON fournisseur.adresse\_id = adresse.id;  ELSE  SELECT fournisseur.id,  fournisseur.nom,  fournisseur.telephone,  fournisseur.email,  CONCAT\_WS("",adresse.appartement,  adresse.etage,  adresse.couloir,  adresse.escalier,  adresse.entree,  adresse.immeuble,  adresse.residence,  adresse.numero,  adresse.voie,  adresse.place,  adresse.code,  adresse.ville,  adresse.pays) AS adresse  FROM fournisseur  INNER JOIN adresse  ON fournisseur.adresse\_id = adresse.id  WHERE nom LIKE CONCAT('%',p\_nom,'%');  END IF;  END |  DELIMITER ;  ################################################################################  # REQUETE SELECT UTILISATEUR #  ################################################################################  #################################################### LISTE EMPLOYE SUIVANT NOM #  DROP PROCEDURE IF EXISTS liste\_employe\_suivant\_nom;  DELIMITER |  CREATE PROCEDURE liste\_employe\_suivant\_nom(  IN p\_nom VARCHAR(50))  BEGIN  IF p\_nom IS NULL OR p\_nom = '\*'THEN  SELECT utilisateur.id,  utilisateur.civilite,  utilisateur.prenom,  utilisateur.nom,  employe.role FROM utilisateur  INNER JOIN employe  ON employe.utilisateur\_id = utilisateur.id;  ELSE  SELECT utilisateur.id,  utilisateur.civilite,  utilisateur.prenom,  utilisateur.nom,  employe.role FROM utilisateur  INNER JOIN employe  ON employe.utilisateur\_id = utilisateur.id  WHERE nom LIKE CONCAT('%',p\_nom,'%');  END IF;  END |  DELIMITER ;  ################################################# LISTE EMPLOYE SUIVANT PRENOM #  DROP PROCEDURE IF EXISTS liste\_employe\_suivant\_prenom;  DELIMITER |  CREATE PROCEDURE liste\_employe\_suivant\_prenom(  IN p\_prenom VARCHAR(50))  BEGIN  IF p\_prenom IS NULL OR p\_prenom = '\*'THEN  SELECT utilisateur.id,  utilisateur.civilite,  utilisateur.prenom,  utilisateur.nom,  employe.role FROM utilisateur  INNER JOIN employe  ON employe.utilisateur\_id = utilisateur.id;  ELSE  SELECT utilisateur.id,  utilisateur.civilite,  utilisateur.prenom,  utilisateur.nom,  employe.role FROM utilisateur  INNER JOIN employe  ON employe.utilisateur\_id = utilisateur.id  WHERE prenom LIKE CONCAT('%',p\_prenom,'%');  END IF;  END |  DELIMITER ;  ################################################### LISTE EMPLOYE SUIVANT ROLE #  DROP PROCEDURE IF EXISTS liste\_employe\_suivant\_role;  DELIMITER |  CREATE PROCEDURE liste\_employe\_suivant\_role(  IN p\_role ENUM('Accueil','Pizzaiolo','Livreur','Manager','Gestionnaire','Comptable','Direction') )  BEGIN  IF p\_role IS NULL OR p\_role = '\*'THEN  SELECT utilisateur.id,  utilisateur.civilite,  utilisateur.prenom,  utilisateur.nom,  employe.role FROM utilisateur  INNER JOIN employe  ON utilisateur.id = employe.utilisateur\_id;  ELSE  SELECT utilisateur.id,  utilisateur.civilite,  utilisateur.prenom,  utilisateur.nom,  employe.role FROM utilisateur  INNER JOIN employe  ON employe.utilisateur\_id = utilisateur.id  WHERE role LIKE CONCAT('%',p\_role,'%');  END IF;  END |  DELIMITER ;  ##################################################### LISTE CLIENT SUIVANT NOM #  DROP PROCEDURE IF EXISTS liste\_client\_suivant\_nom;  DELIMITER |  CREATE PROCEDURE liste\_client\_suivant\_nom(  IN p\_nom VARCHAR(50))  BEGIN  IF p\_nom IS NULL OR p\_nom = '\*'THEN  SELECT utilisateur.id,  utilisateur.civilite,  utilisateur.prenom,  utilisateur.nom,  client.telephone,  client.email,  CONCAT\_WS("",adresse.appartement,  adresse.etage,  adresse.couloir,  adresse.escalier,  adresse.entree,  adresse.immeuble,  adresse.residence,  adresse.numero,  adresse.voie,  adresse.place,  adresse.code,  adresse.ville,  adresse.pays) AS adresse  FROM utilisateur  INNER JOIN client ON client.utilisateur\_id = utilisateur.id  INNER JOIN adresse ON adresse.id = client.adresse\_id;  ELSE  SELECT utilisateur.id,  utilisateur.civilite,  utilisateur.prenom,  utilisateur.nom,  client.telephone,  client.email,  CONCAT\_WS("",adresse.appartement,  adresse.etage,  adresse.couloir,  adresse.escalier,  adresse.entree,  adresse.immeuble,  adresse.residence,  adresse.numero,  adresse.voie,  adresse.place,  adresse.code,  adresse.ville,  adresse.pays) AS adresse  FROM utilisateur  INNER JOIN client ON client.utilisateur\_id = utilisateur.id  INNER JOIN adresse ON adresse.id = client.adresse\_id  WHERE utilisateur.nom LIKE CONCAT('%',p\_nom,'%');  END IF;  END |  DELIMITER ;  ################################################## LISTE CLIENT SUIVANT PRENOM #  DROP PROCEDURE IF EXISTS liste\_client\_suivant\_prenom;  DELIMITER |  CREATE PROCEDURE liste\_client\_suivant\_prenom(  IN p\_prenom VARCHAR(50))  BEGIN  IF p\_prenom IS NULL OR p\_prenom = '\*'THEN  SELECT utilisateur.id,  utilisateur.civilite,  utilisateur.prenom,  utilisateur.nom,  client.telephone,  client.email,  CONCAT\_WS("",adresse.appartement,  adresse.etage,  adresse.couloir,  adresse.escalier,  adresse.entree,  adresse.immeuble,  adresse.residence,  adresse.numero,  adresse.voie,  adresse.place,  adresse.code,  adresse.ville,  adresse.pays) AS adresse  FROM utilisateur  INNER JOIN client ON client.utilisateur\_id = utilisateur.id  INNER JOIN adresse ON adresse.id = client.adresse\_id;  ELSE  SELECT utilisateur.id,  utilisateur.civilite,  utilisateur.prenom,  utilisateur.nom,  client.telephone,  client.email,  CONCAT\_WS("",adresse.appartement,  adresse.etage,  adresse.couloir,  adresse.escalier,  adresse.entree,  adresse.immeuble,  adresse.residence,  adresse.numero,  adresse.voie,  adresse.place,  adresse.code,  adresse.ville,  adresse.pays) AS adresse  FROM utilisateur  INNER JOIN client ON client.utilisateur\_id = utilisateur.id  INNER JOIN adresse ON adresse.id = client.adresse\_id  WHERE utilisateur.prenom LIKE CONCAT('%',p\_prenom,'%');  END IF;  END |  DELIMITER ;  ################################################################################  # REQUETE SELECT PRODUIT #  ################################################################################  ############################################################### GET PRODUIT ID #  DROP FUNCTION IF EXISTS get\_produit\_id;  DELIMITER |  CREATE FUNCTION get\_produit\_id(  p\_designation VARCHAR(100))  RETURNS INT(10) UNSIGNED  DETERMINISTIC  BEGIN  DECLARE v\_id INT(10) UNSIGNED;  SELECT id INTO v\_id  FROM produit  WHERE designation LIKE CONCAT('%',p\_designation,'%')  ORDER BY id ASC LIMIT 1;  RETURN (v\_id);  END |  DELIMITER ;  ############################################## LISTE INGREDIENT PRODUIT PAR ID #  DROP PROCEDURE IF EXISTS liste\_ingredient\_produit\_par\_id;  DELIMITER |  CREATE PROCEDURE liste\_ingredient\_produit\_par\_id(  IN p\_id INT(10) UNSIGNED)  BEGIN  SELECT produit.id AS ID,  produit.designation AS Désignation,  composant.quantite AS Quantité,  composant.unite AS Unité FROM composant  INNER JOIN produit  ON composant.ingredient\_id = produit.id  WHERE composant.produit\_id = p\_id;  END|  DELIMITER ;  ##################################### LISTE INGREDIENT PRODUIT PAR DESIGNATION #  DROP PROCEDURE IF EXISTS liste\_ingredient\_produit\_par\_designation;  DELIMITER |  CREATE PROCEDURE liste\_ingredient\_produit\_par\_designation(  IN p\_designation VARCHAR(100))  BEGIN  DECLARE v\_id INT(10) UNSIGNED;  SET v\_id = get\_produit\_id(p\_designation);  SELECT produit.id AS ID,  produit.designation AS Désignation,  composant.quantite AS Quantité,  composant.unite AS Unité FROM composant  INNER JOIN produit  ON composant.ingredient\_id = produit.id  WHERE composant.produit\_id = v\_id;  END|  DELIMITER ;  ####################################################### LISTE PRODUIT VENDABLE #  DROP PROCEDURE IF EXISTS liste\_produit\_vendable;  DELIMITER |  CREATE PROCEDURE liste\_produit\_vendable(  IN p\_magasin\_id INT(10) UNSIGNED)  BEGIN  SELECT produit.id, produit.designation, produit.prix\_vente\_ht FROM produit  LEFT JOIN stock ON produit.id = stock.produit\_id  WHERE (magasin\_id = p\_magasin\_id OR magasin\_id IS NULL)  AND (categorie NOT IN ('pack','vrac','ingrédient'))  AND (stock.quantite > 0 OR stock.quantite IS NULL);  END |  DELIMITER ;  SHOW PROCEDURE STATUS WHERE Db ='oc\_pizza'; |

## Création des fonctions.

|  |
| --- |
| ################################################################################  # OC PIZZA CREATE FUNCTION #  ################################################################################  ####################################################### PRODUIT EST DISPONIBLE #  DROP FUNCTION IF EXISTS produit\_est\_disponible;  DELIMITER |  CREATE FUNCTION produit\_est\_disponible(  p\_magasin\_id INT(10) UNSIGNED,  p\_produit\_id INT(10) UNSIGNED,  p\_quantite DECIMAL(2))  RETURNS INT  DETERMINISTIC  BEGIN  DECLARE v\_disponible INT DEFAULT FALSE;  SELECT quantite>=p\_quantite INTO v\_disponible FROM stock  WHERE magasin\_id = p\_magasin\_id AND produit\_id = p\_produit\_id;  IF v\_disponible = 1 THEN  RETURN (v\_disponible);  ELSE  SELECT SUM(stock.quantite>= composant.quantite\*p\_quantite)=COUNT(\*) INTO v\_disponible FROM composant  JOIN stock ON composant.ingredient\_id = stock.produit\_id  WHERE magasin\_id = p\_magasin\_id AND composant.produit\_id = p\_produit\_id;  END IF;  RETURN (v\_disponible);  END |  DELIMITER ;  ############################################################## GET PRODUIT ID #  DROP FUNCTION IF EXISTS get\_produit\_id;  DELIMITER |  CREATE FUNCTION get\_produit\_id(  p\_designation VARCHAR(100),  p\_categorie ENUM ('pack','vrac','ingrédient','pizza','boisson','dessert','emballage','sauce'))  RETURNS INT  DETERMINISTIC  BEGIN  DECLARE v\_id INT(10) UNSIGNED;  SELECT DISTINCT id INTO v\_id FROM produit  WHERE categorie = p\_categorie AND designation LIKE CONCAT("%", p\_designation, "%")  ORDER BY id LIMIT 1;  RETURN (v\_id);  END |  DELIMITER ;  ############################################################## GET VRAC ID #  DROP FUNCTION IF EXISTS get\_vrac\_id;  DELIMITER |  CREATE FUNCTION get\_vrac\_id(  p\_designation VARCHAR(100))  RETURNS INT  DETERMINISTIC  BEGIN  DECLARE v\_id INT(10) UNSIGNED;  SELECT DISTINCT id INTO v\_id FROM produit  WHERE categorie = 'vrac' AND designation LIKE CONCAT("%", p\_designation, "%")  ORDER BY id LIMIT 1;  RETURN (v\_id);  END |  DELIMITER ;  ######################################################## GET CLIENT ADRESSE ID #  DROP FUNCTION IF EXISTS get\_client\_adresse\_id;  DELIMITER |  CREATE FUNCTION get\_client\_adresse\_id(  p\_utilisateur\_id INT(10) UNSIGNED)  RETURNS INT(10) UNSIGNED  DETERMINISTIC  BEGIN  DECLARE v\_adresse\_id INT(10) UNSIGNED DEFAULT 0;  SELECT adresse\_id INTO v\_adresse\_id FROM client WHERE utilisateur\_id = p\_utilisateur\_id;  RETURN (v\_adresse\_id);  END |  DELIMITER ;  ######################################################### GET ETABLISSEMENT ID #  DROP FUNCTION IF EXISTS get\_etablissement\_id;  DELIMITER |  CREATE FUNCTION get\_etablissement\_id(  p\_nom VARCHAR(20))  RETURNS INT(10) UNSIGNED  DETERMINISTIC  BEGIN  DECLARE v\_id INT(10) UNSIGNED DEFAULT 0;  SELECT DISTINCT id INTO v\_id FROM etablissement WHERE nom LIKE CONCAT('%',p\_nom,'%')  ORDER BY id ASC LIMIT 1;  RETURN (v\_id);  END |  DELIMITER ;  ################################################################## EST MAGASIN #  DROP FUNCTION IF EXISTS est\_adresse\_de\_magasin;  DELIMITER |  CREATE FUNCTION est\_adresse\_de\_magasin(  p\_adresse\_id INT(10) UNSIGNED)  RETURNS TINYINT  DETERMINISTIC  BEGIN  DECLARE v\_reponse TINYINT DEFAULT FALSE;  SELECT COUNT(\*) INTO v\_reponse FROM magasin INNER JOIN adresse ON magasin.adresse\_id = adresse.id WHERE adresse.id = p\_adresse\_id;  RETURN (v\_reponse);  END |  DELIMITER ;  SHOW PROCEDURE STATUS WHERE Db ='oc\_pizza'; |

## Création des requêtes.

|  |
| --- |
| ################################################################################  # OC PIZZA CREATE REQUETE #  ################################################################################  ################################################################################  # UTILISATEUR #  ################################################################################  PREPARE l\_employe FROM  'SELECT utilisateur.id AS id, CONCAT(civilite," ", prenom," ", utilisateur.nom) AS nom, login, role, magasin.nom FROM employe  INNER JOIN utilisateur ON employe.utilisateur\_id = utilisateur.id  INNER JOIN magasin ON utilisateur.magasin\_id = magasin.id';  PREPARE l\_employe\_dans\_magasin FROM  'SELECT utilisateur.id, utilisateur.prenom, utilisateur.nom, utilisateur.login, employe.role FROM utilisateur  JOIN employe ON employe.utilisateur\_id = utilisateur.id  WHERE utilisateur.magasin\_id = ?';  PREPARE l\_client FROM  'SELECT utilisateur.id AS id, CONCAT(civilite," ", prenom," ", utilisateur.nom) AS nom, login,client.telephone,client.email,CONCAT(numero,",",voie,",",code,",",ville) AS adresse, magasin.nom AS magasin FROM client  INNER JOIN utilisateur ON client.utilisateur\_id = utilisateur.id  INNER JOIN adresse ON client.adresse\_id = adresse.id  INNER JOIN magasin ON utilisateur.magasin\_id = magasin.id';  PREPARE l\_client\_dans\_magasin FROM  'SELECT utilisateur.id,utilisateur.civilite,utilisateur.prenom,utilisateur.nom,utilisateur.login,client.telephone,client.email FROM utilisateur  JOIN client ON client.utilisateur\_id = utilisateur.id  WHERE utilisateur.magasin\_id = ?';  ################################################################################  # PAIEMENT #  ################################################################################  PREPARE l\_paiement\_TR FROM  'SELECT paiement.id, paiement.type, ticket\_restaurant.numero, etablissement.nom FROM paiement  JOIN ticket\_restaurant ON ticket\_restaurant.paiement\_id = paiement.id  JOIN etablissement ON etablissement.id = ticket\_restaurant.etablissement\_id';  PREPARE l\_paiement\_CB FROM  'SELECT paiement.id, paiement.type, carte\_bancaire.reference, carte\_bancaire.jour, carte\_bancaire.heure FROM paiement  JOIN carte\_bancaire ON carte\_bancaire.paiement\_id = paiement.id';  PREPARE l\_paiement\_cheque FROM  'SELECT paiement.id, paiement.type, cheque.banque, cheque.numero, cheque.jour FROM paiement  JOIN cheque ON cheque.paiement\_id = paiement.id';  ################################################################################  # COMMANDE #  ################################################################################  PREPARE l\_commande\_en\_attente FROM  'SELECT id,utilisateur\_id, statut, jour, heure, paiement\_OK FROM commande WHERE statut = "En attente"';  PREPARE l\_commande\_en\_preparation FROM  'SELECT id,utilisateur\_id, statut, jour, heure, paiement\_OK FROM commande WHERE statut = "En préparation"';  PREPARE l\_commande\_préparee FROM  'SELECT id,utilisateur\_id, statut, jour, heure, paiement\_OK FROM commande WHERE statut = "Préparée"';  PREPARE l\_commande\_en\_livraison FROM  'SELECT id,utilisateur\_id, statut, jour, heure, paiement\_OK FROM commande WHERE statut = "En livraison"';  PREPARE l\_commande\_livree FROM  'SELECT id,utilisateur\_id, statut, jour, heure, paiement\_OK FROM commande WHERE statut = "Livrée"';  SHOW PROCEDURE STATUS WHERE Db ='oc\_pizza'; |

## Remplissage de la base.

|  |
| --- |
| ################################################################################  # OC PIZZA FILL DATABASE #  ################################################################################  ################################################################################  # GLOBAL DATA #  ################################################################################  ################################################################ ETABLISSEMENT #  LOCK TABLE etablissement WRITE;  INSERT INTO etablissement VALUES  (1,'La Banque Poste'),  (2,'BNP Paribas'),  (3,'Société Générale'),  (4,'Crédit Agricole'),  (5,'LCL'),  (6,'Banque Populaire'),  (7,'Caisse d\'Epargne'),  (8,'HSBC'),  (9,'Crédit Mutuel'),  (10,'Chèque Déjeuner'),  (11,'Pass Restaurant'),  (12,'Ticket Restaurant'),  (13,'Chèque Apetiz'),  (14,'Ticket Restaurant');  UNLOCK TABLE;  SELECT \* FROM etablissement;  ###################################################################### MAGASIN #  CALL create\_magasin('Pizza Napoli', '0381501111','napoli@ocpizza.com', '1', 'rue des Teinturiers', 'Lyon','69003', @ID);  CALL create\_magasin('Pizza Firenze','0381501654','firenze@ocpizza.com', '45', 'rue des Bleuet', 'Dole','39000',@ID);  CALL create\_magasin('Pizza Roma', '0381501112','roma@ocpizza.com', '4', 'rue des Acacias', 'Vesoul','70000',@ID);  CALL create\_magasin('Pizza Torino', '0381501113','torino@ocpizza.com', '12', 'rue des Chataigniers', 'Belfort','90000',@ID);  SELECT magasin.id,nom,telephone,email,CONCAT(numero,',',voie,',',code,',',ville) AS adresse FROM magasin JOIN adresse ON adresse\_id = adresse.id;  ################################################################## FOURNISSEUR #  CALL create\_fournisseur('Global Food', '0145686811','client@globalfood.com', '14', 'rue de Paris', 'RUNGIS','94150', @ID);  CALL create\_fournisseur('Italia Food', '0181654654','commande@italiafood.com', '15', 'rue de Paris', 'RUNGIS','94150', @ID);  SELECT fournisseur.id,nom,telephone,email,CONCAT(numero,',',voie,',',code,',',ville) AS adresse FROM fournisseur JOIN adresse ON adresse\_id = adresse.id;  ################################################################################  # UTILISATEUR DATA #  ################################################################################  ###################################################################### EMPLOYE #  CALL create\_employe('Mlle','CASTAFIORE','Bianca','Bianca',SHA1('CASTAFIORE'),1,'Accueil',@ID);  CALL create\_employe('M','TINTIN','Milou','Milou',SHA1('TINTIN'),1,'Livreur',@ID);  CALL create\_employe('M','HADDOCK','Capitaine','Capitaine',SHA1('HADDOCK'),1,'Direction',@ID);  CALL create\_employe('M','ALCAZAR','Général','Général',SHA1('ALCAZAR'),1,'Pizzaiolo',@ID);  CALL create\_employe('M','MULLER','Docteur','Docteur',SHA1('MULLER'),1,'Comptable',@ID);  CALL create\_employe('M','LAMPION','Séraphin','Séraphin',SHA1('LAMPION'),1,'Manager',@ID);  CALL create\_employe('M','SPONSZ','Colonel','Colonel',SHA1('SPONSZ'),1,'Gestionnaire',@ID);  EXECUTE l\_employe;  ####################################################################### CLIENT #  CALL create\_client('M','RACKHAM','Red','Red',SHA1('RACKHAM'),1,'0381565422','red.rackham@gmail.com','1', 'Rue de Naple','Besançon','25000',@ID);  CALL create\_client('M','DA FIGUEIRA','Oliveira','Oliveira',SHA1('DA FIGUEIRA'),1,'0381565422','oliveira.dafigueira@hotmail.com','10', 'Rue des Frères Mercier','Besançon','25000',@ID);  CALL create\_client('M','WOLF','Frank','Frank',SHA1('WOLF'),2,'0381565422','frank.wolf@orange.fr','7', 'Rue des Grand Bas','Besançon','25000',@ID);  CALL create\_client('M','THOMPSON','Alan','Alan',SHA1('THOMPSON'),2,'0381565422','alan.thompson@red.com','4', 'Rue de la Paix','Besançon','25000',@ID);  CALL create\_client('M','DUPON','ThierryAlan','Thierry',SHA1('DUPON'),1,'0381565422','thierry.dupon@belga.be','34', 'Rue de la Résistance','Besançon','25000',@ID);  CALL create\_client('M','DUPON','Daniel','Daniel',SHA1('DUPON'),2,'0381565422','daniel.dupon@belga.be','9', 'Rue Battant','Besançon','25000',@ID);  EXECUTE l\_client;  ################################################################################  # PRODUIT DATA #  ################################################################################  ################################################################# LES PRODUITS #  # create\_produit( IN p\_designation,p\_categorie,p\_fournisseur\_id,p\_reference, #  # p\_quantite,p\_unite, p\_prix\_achat\_ht, p\_prix\_vente\_ht, #  # p\_tva\_emporte, p\_tva\_livre, p\_formule, p\_recette, #  # OUT p\_id INT(10)) #  ################################################################################  ######################################################################## SECHE #  CALL create\_produit('farine de blé T55 - 25Kg','ingrédient',1,'fkjh6546',25.00,'KG',33.40,NULL,0.0,0.0,'farine',NULL,@COMP);  CALL create\_produit('farine de blé T55 - Vrac','vrac',1,'fkjh6546',1.0,'KG',NULL,NULL,0.0,0.0,'farine',NULL,@ING);  CALL add\_composant(@COMP,@ING,25.0,'KG');  ###################################################################### PRIMEUR #  CALL create\_produit('Champignon pied coupé moyen catégorie 1 - 3Kg','ingrédient',1,'31347',3.00,'KG',14.5,NULL,0.0,0.0,'champignon',NULL,@COMP);  CALL create\_produit('Champignon pied coupé moyen catégorie 1 - Vrac','vrac',1,'31347',1.00,'KG',NULL,NULL,0.0,0.0,'champignon',NULL,@ING);  CALL add\_composant(@COMP,@ING,3.0,'KG');  CALL create\_produit('Poivron mixte calibre 80/100 catégorie 1 - 4Kg','ingrédient',1,'82657',4.00,'KG',15.00,NULL,0.0,0.0,'poivron',NULL,@COMP);  CALL create\_produit('Poivron mixte calibre 80/100 catégorie 1 - Vrac','vrac',1,'82657',1.00,'KG',NULL,NULL,0.0,0.0,'poivron',NULL,@ING);  CALL add\_composant(@COMP,@ING,4.0,'KG');  CALL create\_produit('Tomate ronde calibre 57/67 océanecatégorie extra - 6Kg','ingrédient',1,'93945',6.00,'KG',15.00,NULL,0.0,0.0,'tomate',NULL,@COMP);  CALL create\_produit('Tomate ronde calibre 57/67 océanecatégorie extra - Vrac','vrac',1,'93945',1.0,'KG',NULL,NULL,0.0,0.0,'tomate',NULL,@ING);  CALL add\_composant(@COMP,@ING,6.0,'KG');  CALL create\_produit('Aubergine calibre 300/400 catégorie 1 - 6Kg','ingrédient',1,'59884',6.00,'KG',20.00,NULL,0.0,0.0,'aubergine',NULL,@COMP);  CALL create\_produit('Aubergine calibre 300/400 catégorie 1 - Vrac','vrac',1,'59884',1.00,'KG',NULL,NULL,0.0,0.0,'aubergine',NULL,@ING);  CALL add\_composant(@COMP,@ING,6.0,'KG');  CALL create\_produit('Oignon charcutier calibre 70/100 catégorie 1 - 10Kg','ingrédient',1,'702815',10.00,'KG',20.00,NULL,0.0,0.0,'oignon',NULL,@COMP);  CALL create\_produit('Oignon charcutier calibre 70/100 catégorie 1 - Vrac','vrac',1,'702815',1.00,'KG',NULL,NULL,0.0,0.0,'oignon',NULL,@ING);  CALL add\_composant(@COMP,@ING,10.0,'KG');  #################################################################### BOUCHERIE #  CALL create\_produit('Bacon standard sous vide fumé - 1.5Kg','ingrédient',2,'236179',1.50,'KG',55.00,NULL,0.0,0.0,'porc',NULL,@COMP);  CALL create\_produit('Bacon standard sous vide fumé - Vrac','vrac',2,'236179',1.0,'KG',NULL,NULL,0.0,0.0,'porc',NULL,@ING);  CALL add\_composant(@COMP,@ING,1.5,'KG');  CALL create\_produit('Jambon de Vendée à l''ancienne - 3Kg','ingrédient',2,'235440',3.00,'KG',35.00,NULL,0.0,0.0,'porc',NULL,@COMP);  CALL create\_produit('Jambon de Vendée à l''ancienne - Vrac','vrac',2,'235440',1.0,'KG',NULL,NULL,0.0,0.0,'porc',NULL,@ING);  CALL add\_composant(@COMP,@ING,3.0,'KG');  CALL create\_produit('Pepperoni - 1.8Kg','ingrédient',2,'157623',1.8,'KG',35.00,NULL,0.0,0.0,'porc',NULL,@COMP);  CALL create\_produit('Pepperoni - Vrac','vrac',2,'157623',1.0,'KG',NULL,NULL,0.0,0.0,'porc',NULL,@ING);  CALL add\_composant(@COMP,@ING,1.8,'KG');  CALL create\_produit('Chorizo fort - 1.8Kg','ingrédient',2,'157623',1.8,'KG',35.00,NULL,0.0,0.0,'porc',NULL,@COMP);  CALL create\_produit('Chorizo fort - Vrac','vrac',2,'157623',1.0,'KG',NULL,NULL,0.0,0.0,'porc',NULL,@ING);  CALL add\_composant(@COMP,@ING,1.8,'KG');  ######################################################################## MAREE #  CALL create\_produit('Saumon sauvage filet sous vide - 5Kg','ingrédient',1,'706424',5.0,'KG',111.10,NULL,0.0,0.0,'poisson',NULL,@COMP);  CALL create\_produit('Saumon sauvage filet sous vide - Vrac','vrac',1,'706424',1.0,'KG',NULL,NULL,0.0,0.0,'poisson',NULL,@ING);  CALL add\_composant(@COMP,@ING,5.0,'KG');  ##################################################################### CREMERIE #  CALL create\_produit('Oeuf calibre gros fermier - 30U','ingrédient',1,'159123',30,'U',10.1,NULL,0.0,0.0,'oeuf',NULL,@COMP);  CALL create\_produit('Oeuf calibre gros fermier - Vrac','vrac',1,'159123',1,'U',NULL,NULL,0.0,0.0,'oeuf',NULL,@ING);  CALL add\_composant(@COMP,@ING,30,'U');  CALL create\_produit('Lait 1/2 écrémé UHT 1.6% MG brique - 6L','ingrédient',1,'247890',6,'L',0.89,NULL,0.0,0.0,'lait',NULL,@COMP);  CALL create\_produit('Lait 1/2 écrémé UHT 1.6% MG brique - Vrac','vrac',1,'247890',1,'L',NULL,NULL,0.0,0.0,'lait',NULL,@ING);  CALL add\_composant(@COMP,@ING,6.0,'L');  CALL create\_produit('Crème liquide UHT 30% MG - 6L','ingrédient',1,'246755',6,'L',1.10,NULL,0.0,0.0,'lait',NULL,@COMP);  CALL create\_produit('Crème liquide UHT 30% MG - Vrac','vrac',1,'246755',1,'L',NULL,NULL,0.0,0.0,'lait',NULL,@ING);  CALL add\_composant(@COMP,@ING,6.0,'L');  CALL create\_produit('Sauce tomate - 6L','ingrédient',1,'247890',6,'L',2.59,NULL,0.0,0.0,'tomate,poivre,sel',NULL,@COMP);  CALL create\_produit('Sauce tomate - Vrac','vrac',1,'247890',1,'L',NULL,NULL,0.0,0.0,'tomate,poivre,sel',NULL,@ING);  CALL add\_composant(@COMP,@ING,6.0,'L');  CALL create\_produit('Fromage rapé - 5Kg','ingrédient',2,'654654',5.00,'KG',50.00,NULL,0.0,0.0,'fromage',NULL,@COMP);  CALL create\_produit('Fromage rapé - Vrac','vrac',2,'654654',1.0,'KG',NULL,NULL,0.0,0.0,'fromage',NULL,@ING);  CALL add\_composant(@COMP,@ING,5.0,'KG');  CALL create\_produit('Mozzarella - 5Kg','ingrédient',2,'78989',5.00,'KG',80.00,NULL,0.0,0.0,'fromage',NULL,@COMP);  CALL create\_produit('Mozzarella - Vrac','vrac',2,'78989',1.0,'KG',NULL,NULL,0.0,0.0,'fromage',NULL,@ING);  CALL add\_composant(@COMP,@ING,5.0,'KG');  CALL create\_produit('Chèvre - 2Kg','ingrédient',2,'78989',2.00,'KG',40.00,NULL,0.0,0.0,'fromage',NULL,@COMP);  CALL create\_produit('Chèvre - Vrac','vrac',2,'78989',1.0,'KG',NULL,NULL,0.0,0.0,'fromage',NULL,@ING);  CALL add\_composant(@COMP,@ING,2.0,'KG');  CALL create\_produit('Fourme d''Amber AOP - 2Kg','ingrédient',2,'78989',2.00,'KG',40.00,NULL,0.0,0.0,'fromage',NULL,@COMP);  CALL create\_produit('Fourme d''Amber AOP - Vrac','vrac',2,'78989',1.0,'KG',NULL,NULL,0.0,0.0,'fromage',NULL,@ING);  CALL add\_composant(@COMP,@ING,2.0,'KG');  ################################################################# LES BOISSONS #  CALL create\_produit('Eau plate - 50cl/24','pack',1,'65465',24,'U',NULL,NULL,0.0,0.0,'eau',NULL,@COMP);  CALL create\_produit('Eau plate - 50cl/1','boisson',1,'65465',1,'U',NULL,1.8,5.5,10.0,'eau',NULL,@ING);  CALL add\_composant(@COMP,@ING,24,'U');  CALL create\_produit('Eau gazeuze - 50cl/24','pack',1,'654898',1,'U',NULL,NULL,0.0,0.0,'eau',NULL,@COMP);  CALL create\_produit('Eau gazeuze - 50cl/1','boisson',1,'654898',1,'U',NULL,1.8,5.5,10.0,'eau',NULL,@ING);  CALL add\_composant(@COMP,@ING,24,'U');  CALL create\_produit('Cola - 33cl/24','pack',1,'654898',1,'U',NULL,NULL,0.0,0.0,NULL,NULL,@COMP);  CALL create\_produit('Cola - 33cl/1','boisson',1,'654898',1,'U',NULL,1.8,5.5,10.0,NULL,NULL,@ING);  CALL add\_composant(@COMP,@ING,24,'U');  CALL create\_produit('Jus d''orange - 33cl/24','pack',1,'888554',1,'U',NULL,NULL,0.0,0.0,NULL,NULL,@COMP);  CALL create\_produit('Jus d''orange - 33cl/1','boisson',1,'888554',1,'U',NULL,1.8,5.5,10.0,NULL,NULL,@ING);  CALL add\_composant(@COMP,@ING,24,'U');  CALL create\_produit('Jus de pomme - 33cl/24','pack',1,'644898',1,'U',NULL,NULL,0.0,0.0,NULL,NULL,@COMP);  CALL create\_produit('Jus de pomme - 33cl/1','boisson',1,'644898',1,'U',NULL,2.1,5.5,10.0,NULL,NULL,@ING);  CALL add\_composant(@COMP,@ING,24,'U');  ################################################################### LES PIZZAS #  CALL create\_produit('Pizza margarita','pizza',1,'Pmargarita',1,'U',NULL,15.3,10.0,10.0,NULL,NULL,@COMP);  CALL add\_composant(@COMP,get\_vrac\_id('farine'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('jambon'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Sauce tomate'),0.05,'L');  CALL add\_composant(@COMP,get\_vrac\_id('Fromage rapé'),0.10,'KG');  CALL create\_produit('Pizza 4 fromages','pizza',1,'P4fromages',1,'U',NULL,13.3,10.0,10.0,NULL,NULL,@COMP);  CALL add\_composant(@COMP,get\_vrac\_id('farine'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Mozzarella'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Fromage rapé'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Fourme'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Chevre'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Sauce tomate'),0.05,'L');  CALL create\_produit('Pizza extravaganzza','pizza',1,'Pextravag',1,'U',NULL,16.5,10.0,10.0,NULL,NULL,@COMP);  CALL add\_composant(@COMP,get\_vrac\_id('farine'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Champignon'),0.05,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Poivron'),0.05,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Mozzarella'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Fromage rapé'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Pepperoni'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Jambon'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Bacon'),0.10,'KG');  CALL add\_composant(@COMP,get\_vrac\_id('Sauce tomate'),0.05,'L');  SELECT \* FROM produit;  SELECT \* FROM composition;  SELECT \* FROM preparation;  SELECT \* FROM composant;  SELECT "Livraison des magasins";  CALL livre\_magasin();  SELECT \* FROM stock; |

## Vie d'une commande.

|  |
| --- |
| ################################################################################  # OC PIZZA DEROULEMENT COMMANDE #  ################################################################################  SELECT "Le client 8 rempli son panier avec une livraison";  CALL ajoute\_panier(8,get\_produit\_id("Pizza margarita","pizza"),1,TRUE);  CALL ajoute\_panier(8,get\_produit\_id("Pizza margarita","pizza"),1,TRUE);  CALL ajoute\_panier(8,get\_produit\_id("Cola - 33cl/1","boisson"),1,TRUE);  CALL ajoute\_panier(8,get\_produit\_id("Eau gazeuze - 50cl/1","boisson"),1,TRUE);  SELECT \* FROM panier WHERE utilisateur\_id = 8;  SELECT \* FROM ligne\_de\_panier WHERE utilisateur\_id = 8;  SELECT "Le client 8 enleve un produit et passe en take away";  CALL enleve\_panier(8,get\_produit\_id("margarita","pizza"),1,FALSE);  SELECT \* FROM panier ;  SELECT \* FROM ligne\_de\_panier;  SELECT "Le client 8 ajoute un produit et repasse en livraison";  CALL ajoute\_panier(8,get\_produit\_id("4 fromages","pizza"),1,TRUE);  SELECT \* FROM panier ;  SELECT \* FROM ligne\_de\_panier;  SELECT "Le client 9 rempli son panier en take away";  CALL ajoute\_panier(9,get\_produit\_id("Pizza margarita","pizza"),1,FALSE);  CALL ajoute\_panier(9,get\_produit\_id("Cola - 33cl/1","boisson"),1,FALSE);  SELECT \* FROM panier;  SELECT \* FROM ligne\_de\_panier ;  SELECT "Le client 8 valide son panier";  CALL valide\_commande(8,@ID);  SELECT "Le panier est vide";  SELECT \* FROM panier WHERE utilisateur\_id = 8;  SELECT \* FROM ligne\_de\_panier WHERE utilisateur\_id = 8;  SELECT "La commande est validée non payée et avec le statut En attente";  SELECT id,utilisateur\_id, statut, jour, heure, paiement\_OK FROM commande WHERE utilisateur\_id = 8;  SELECT commande\_id AS IDC, produit\_id AS IDP, quantite AS Quantité, prix\_unitaire\_ht\*(1+taux\_tva/100) AS Prix FROM ligne\_de\_commande WHERE commande\_id = @ID;  SELECT reste\_du(@ID);  SELECT "Le client 8 effectue le paiement par CB";  CALL add\_paiement\_carte\_bancaire(@ID,35.42,"ERGQGQD6546",@IDPAIEMENT);  SELECT "La commande est payée";  SELECT id,utilisateur\_id, statut, jour, heure, paiement\_OK FROM commande WHERE utilisateur\_id = 8;  SELECT commande\_id, paiement\_id, montant, type FROM liste\_paiement JOIN paiement ON paiement.id = liste\_paiement.paiement\_id WHERE commande\_id = @ID;  SELECT reste\_du(@ID);  SELECT "Le client 9 valide son panier";  CALL valide\_commande(9, @commande);  SELECT "Le panier est vide";  SELECT \* FROM panier WHERE utilisateur\_id = 9;  SELECT \* FROM ligne\_de\_panier WHERE utilisateur\_id = 9;  SELECT "La commande est validée non payée et avec le statut En attente";  SELECT id,utilisateur\_id, statut, jour, heure, paiement\_OK FROM commande WHERE utilisateur\_id = 9;  SELECT commande\_id AS IDC, produit\_id AS IDP, quantite AS Quantité, prix\_unitaire\_ht\*(1+taux\_tva/100) AS Prix FROM ligne\_de\_commande WHERE commande\_id = @commande;  SELECT reste\_du(@commande);  SELECT "Le client 9 effectue le paiement par Ticket Restaurant";  CALL add\_paiement\_ticket\_restaurant(@commande,8.20,"ERGQGQD6546",get\_etablissement\_id("Pass"),@IDPAIEMENT);  SELECT id,utilisateur\_id, statut, jour, heure, paiement\_OK FROM commande WHERE utilisateur\_id = 9;  SELECT commande\_id, paiement\_id, montant, type FROM liste\_paiement JOIN paiement ON paiement.id = liste\_paiement.paiement\_id WHERE commande\_id = @commande;  SELECT reste\_du(@commande);  SELECT "Le client 9 effectue le reste du paiement en espèce";  CALL add\_paiement\_espece(@commande,29,26,@IDPAIEMENT);  SELECT "La commande est payée";  SELECT id,utilisateur\_id, statut, jour, heure, paiement\_OK FROM commande WHERE utilisateur\_id = 9;  SELECT commande\_id, paiement\_id, montant, type FROM liste\_paiement JOIN paiement ON paiement.id = liste\_paiement.paiement\_id WHERE commande\_id = @commande;  SELECT reste\_du(@commande);  SELECT id,utilisateur\_id, statut, jour, heure, preparation\_delai AS Attente, preparation\_duree AS Préparation, livraison\_delai AS Finalisation, livraison\_duree AS Livraison,paiement\_OK FROM commande;  CALL pizzaiolo\_prend\_commande(1);  SELECT id,utilisateur\_id, statut, jour, heure, preparation\_delai AS Attente, preparation\_duree AS Préparation, livraison\_delai AS Finalisation, livraison\_duree AS Livraison,paiement\_OK FROM commande;  CALL pizzaiolo\_termine\_commande(1);  SELECT id,utilisateur\_id, statut, jour, heure, preparation\_delai AS Attente, preparation\_duree AS Préparation, livraison\_delai AS Finalisation, livraison\_duree AS Livraison,paiement\_OK FROM commande;  CALL pizzaiolo\_prend\_commande(2);  SELECT id,utilisateur\_id, statut, jour, heure, preparation\_delai AS Attente, preparation\_duree AS Préparation, livraison\_delai AS Finalisation, livraison\_duree AS Livraison,paiement\_OK FROM commande;  CALL livreur\_prend\_commande(1);  SELECT id,utilisateur\_id, statut, jour, heure, preparation\_delai AS Attente, preparation\_duree AS Préparation, livraison\_delai AS Finalisation, livraison\_duree AS Livraison,paiement\_OK FROM commande;  CALL pizzaiolo\_termine\_commande(2);  SELECT id,utilisateur\_id, statut, jour, heure, preparation\_delai AS Attente, preparation\_duree AS Préparation, livraison\_delai AS Finalisation, livraison\_duree AS Livraison,paiement\_OK FROM commande;  CALL client\_prend\_commande(1);  SELECT id,utilisateur\_id, statut, jour, heure, preparation\_delai AS Attente, preparation\_duree AS Préparation, livraison\_delai AS Finalisation, livraison\_duree AS Livraison,paiement\_OK FROM commande;  CALL client\_prend\_commande(2);  SELECT id,utilisateur\_id, statut, jour, heure, preparation\_delai AS Attente, preparation\_duree AS Préparation, livraison\_delai AS Finalisation, livraison\_duree AS Livraison,paiement\_OK FROM commande; |

# Étude de déploiement.

L'étude du déploiement de l'application **OC Pizza** est basée sur **AWS** **(Amazon Web Services)**.

## La partie utilisateur.

Les utilisateurs doivent pouvoir se connecter au site web **OC Pizza** à partir de n'importe quel navigateur web sur un ordinateur, une tablette ou un smartphone. On doit avoir un site responsive pour s'adapter aux trois résolutions différentes de ces médias.

Pour une meilleure intégration avec les smartphones on développera des applicatifs spécifiques. Surtout pour le livreur qui devra valider la livraison lors de son déplacement chez le client.

* **APK** pour **Android**,
* **APP** pour **Apple**.

## La partie base de données.

La base de données sera hébergée par **AWS** avec **Amazon RDS (Relational Database Service)**. Cela permet de gérer facilement la base de données relationnelle dans le cloud avec une grande souplesse d'évolution. On prendra le mode **Multi-AZ** pour avoir une copie de notre base de données, dans une autre région, synchronisée qui pourra prendre le relai en cas de maintenance ou de défaillance de la base principale.

## Les services annexes.

### Le DNS.

Pour permettre une meilleure connexion des utilisateurs à notre application, on utilise **Amazon Route 53** qui sert de **Domain Name Server**. Il converti les adresses nominatives en adresse **IP**. En plus de rendre le site web toujours visible par tous le monde, on a la possibilité de modifier à la volée l'adresse **IP** de nos services en toute transparence pour l'utilisateur.

### La zone de mémoire cache.

**Amazon CloudFront** est un service web qui accélère la distribution de vos contenus web statiques et dynamiques, tels que des fichiers **.html**, **.css**, **.js**, **multimédias** et **image**, à vos utilisateurs. **CloudFront** diffuse votre contenu à travers un réseau mondial de centres de données appelés emplacements périphériques. Lorsqu'un utilisateur demande le contenu que vous proposez avec **CloudFront**, il est dirigé vers l'emplacement périphérique qui fournit la latence la plus faible et, par conséquent, le contenu est remis avec les meilleures performances possibles.

Si le contenu se trouve déjà dans l'emplacement périphérique avec la plus faible latence, **CloudFront** le remet immédiatement.

Si le contenu ne se trouve pas à cet emplacement périphérique, **CloudFront** va le chercher dans un **Bucket** comme **Amazon S3** ou le demande à un serveur **HTTPS** que l'on a identifié comme étant la source originale du contenu de notre application.

### Le stockage des données publiques.

On utilise un simple **Bucket** comme **Amazon Simple Storage** **Service** pour stocker tous les fichiers de notre site Web qui peuvent être publiques. **Amazon S3** offre une interface simple de services Web qui permet de stocker et de récupérer n'importe quelle quantité de données, à tout moment, de n'importe où sur le Web. Il permet aux développeurs d'accéder à la même infrastructure de stockage de données hautement évolutive, fiable, rapide, peu coûteuse qu'**Amazon** utilise pour faire fonctionner son propre réseau mondial de sites. Ce service vise à maximiser les avantages d'échelle et à en faire bénéficier les développeurs.

### Gestion de la sécurité.

Pour exposer les **APIs** **REST** des **microservices** on utilise **Amazon** **API Gateway**. **Amazon API Gateway** est un service qui permet de créer, de publier, de maintenir, de surveiller et de sécuriser les **API REST** à n'importe quelle échelle. Les développeurs **d'API** peuvent créer des **API** qui accèdent à **AWS** ou à d'autres services web, ainsi qu'aux données stockées dans le cloud **AWS**. En tant que développeur **d'API API Gateway**, on peut créer des **API** en vue de les utiliser dans nos applications client ou celles de développeurs d'applications tiers.

On couple **l'API** **Gateway** avec le service **Amazon** **Identity** **Access** **Management** pour garantir la sécurité de notre site web. **IAM** contrôle l'accès aux ressources **AWS**. On utilise **IAM** pour contrôler les personnes qui s'authentifient et sont autorisées à utiliser les ressources.

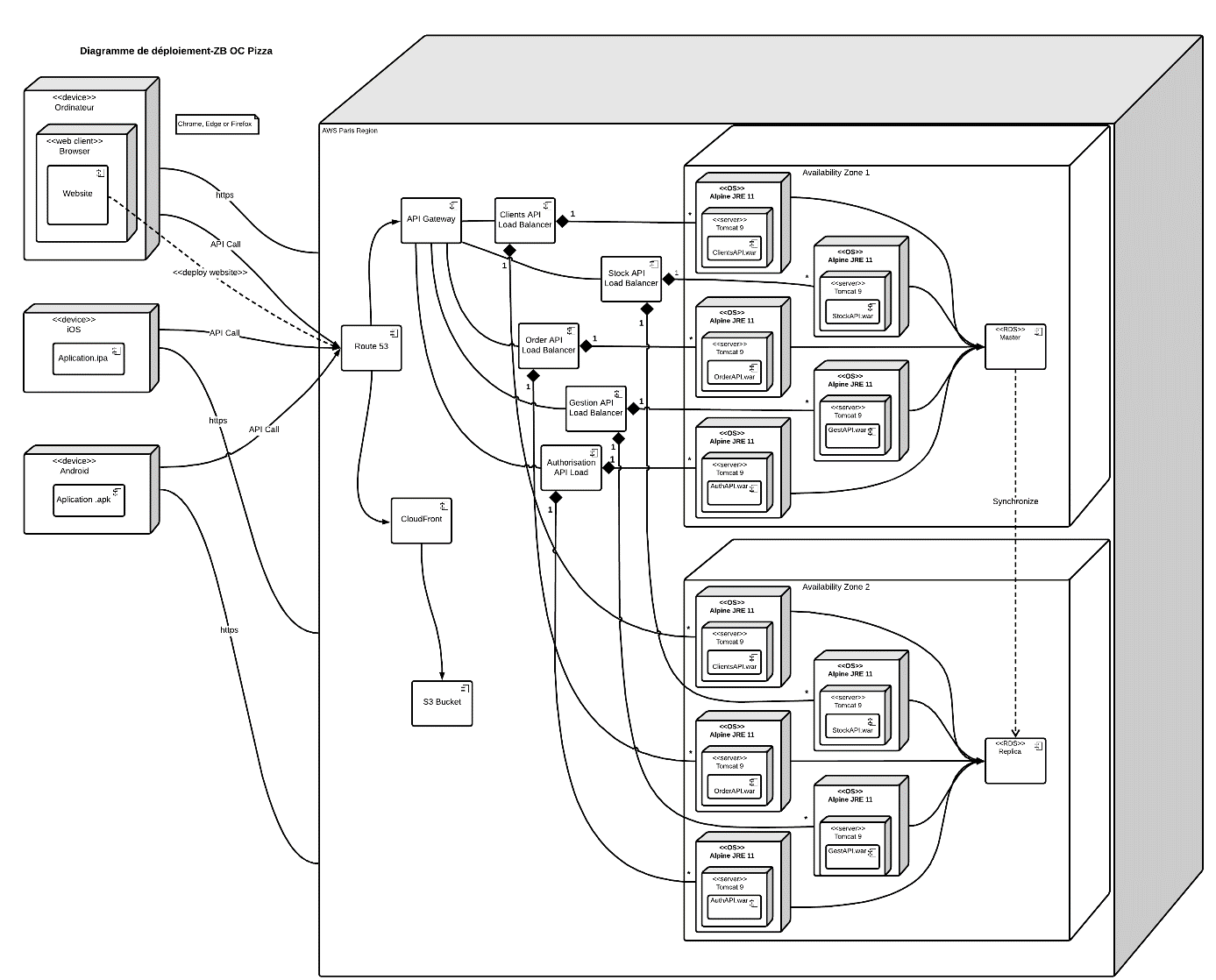
## Les microservices.

On a découpé l'application en **Microservice** pour une plus grande simplicité et une facilité à scalabilité pour répondre à l'accroissement de la demande des utilisateurs. Les **Microservices** sont des **API REST** qui sont exposée par **l'API Gateway**. Les **Microservices** sont gérés chacun par un **Load Balancer** qui permet de dupliquer l'**API REST**.

Les APIs sont installées sur des instances **Amazon EC2 (Elastic Compute Cloud)**. **Amazon EC2** offre une capacité de calcul évolutive dans le cloud **AWS**. L'utilisation d'**EC2** dispense d'investir à l'avance dans du matériel et, par conséquent, on peut développer et déployer les applications plus rapidement. On peut utiliser **Amazon EC2** pour lancer autant de serveurs virtuels que nécessaire, configurer la sécurité et la mise en réseau, et gérer le stockage. Il permet également de monter ou descendre en puissance rapidement, avec les **Load Balancer**, afin de gérer l'évolution des exigences ou des pics de popularité, ce qui permet de réduire la nécessité de prévoir le trafic du serveur.

On utilise **UNIX** comme système d'exploitation des **EC2** pour faire fonctionner nos **API**.

## Diagramme de déploiement.



# Les composants.

## La partie client.

### Navigateur web.

Le client doit accéder au site web avec la plupart des navigateurs depuis :

* Un ordinateur sous **Microsoft** **Windows 10**
* Un ordinateur sous **Apple OS X 10.14.6**
* Une tablette sous **Apple iOS 11**
* Une tablette sous **Android Nougat 7.1**
* Un smartphone sous **Apple iOS 11**
* Un smartphone sous **Android Nougat 7.1**

L'application doit être responsive pour s'adapter aux différents écrans. La communication se fait en utilisant le protocole sécurisé **HTTPS** sur internet.

### Applicatif APP/APK.

Une application simplifiée sur les deux principaux **OS** des smartphones permet de mieux répondre aux attentes des clients.

* Une **APP** pour les smartphones sous **Apple iOS 11**
* Une **APK** pour les smartphones sous **Android Nougat 7.1**

Il faut faire aussi une application **APK** dédié pour les livreurs sur **Android Nougat 7.1** pour valider la réception des commandes et le paiement.

## La partie publique.

Le site web sera implémenté en **Java** avec **SPRING** et packagé en **WAR** avec **MAVEN** pour faciliter la mise à jour et le déploiement. Il contiendra tous les fichiers publics du site web comme les images, les fichiers **.html**, **.css**, **.js**. L'archive **WAR** sera déployée sur un serveur **Tomcat 9.0.24** qui tourne sur une instance **AWS S3** sous **UNIX**.

L'accès se fera par le port **8080** du serveur en **HTTPS**.

## La partie sécurisée.

### La base de données.

Le Système de Gestion de Base de Données **MySQL 8.0.16** sera déployé sur une instance de serveur **AWS RDS** sous **UNIX**. La communication se fait en **HTTPS** via le port **3306**.

### Les Microservices

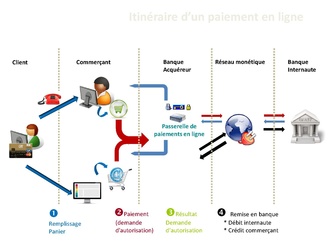
Ils seront implémentés en **Java** avec **SPRING** et packagé en **JAR** avec **MAVEN** pour faciliter la mise à jour et le déploiement. L'archive **JAR** sera déployée sur un serveur **Tomcat 9.0.24** qui tourne sur une instance **AWS EC2** sous **UNIX**. L'accès se fait par requête **REST** via **HTTPS** via le port **8080**. Les **Microservices** sont des **API REST** exposées par l'**API Gateway**.

Les **Microservices** sont les seuls à pouvoir accéder à la base de données.

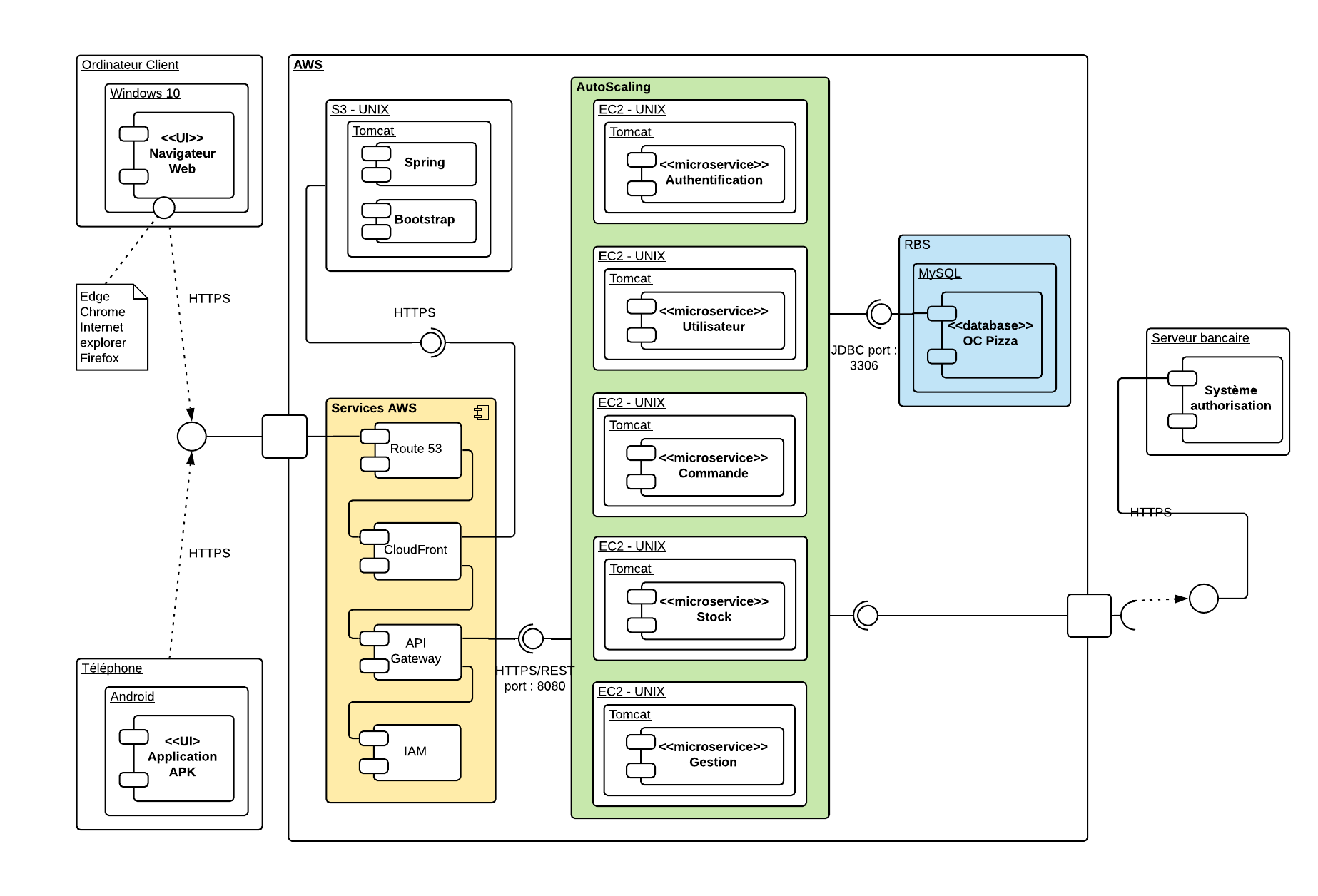
## La partie externe.

### Le système bancaire.

La banque doit nous fournir l'interface avec son système pour pouvoir effectuer les paiements via notre application. La communication se fera par le protocole sécurisé **HTTPS** et en utilisant un cryptage des données par clé publique et privée.

[](https://commons.wikimedia.org/wiki/File:Itin%C3%A9raire_d%27un_paiement_avec_E-transactions.pdf?uselang=fr)

## Le diagramme de composant.



# Table des matières.

[Introduction. 2](#_Toc22056087)

[Le domaine fonctionnel. 3](#_Toc22056088)

[Les composants généraux. 3](#_Toc22056089)

[Adresse 3](#_Toc22056090)

[Magasin 4](#_Toc22056091)

[Les composants de la partie utilisateur. 5](#_Toc22056092)

[Utilisateur 5](#_Toc22056093)

[Civilite <<enum>> 5](#_Toc22056094)

[Client 6](#_Toc22056095)

[Employe 6](#_Toc22056096)

[TypeEmploye <<enum>> 6](#_Toc22056097)

[Apperçu de la partie Utilisateur 7](#_Toc22056098)

[Les composants de la partie produit. 8](#_Toc22056099)

[Produit 8](#_Toc22056100)

[Categorie <<enum>> 8](#_Toc22056101)

[Fournisseur 9](#_Toc22056102)

[Composition 9](#_Toc22056103)

[Preparation 10](#_Toc22056104)

[Composant 10](#_Toc22056105)

[Stock 10](#_Toc22056106)

[Apperçu de la partie Produit 11](#_Toc22056107)

[Les composants de la partie paiement. 12](#_Toc22056108)

[Paiement 12](#_Toc22056109)

[TypePaiement <<enum>> 12](#_Toc22056110)

[CarteBancaire 12](#_Toc22056111)

[Date <<dataType>> 13](#_Toc22056112)

[Time <<dataType>> 13](#_Toc22056113)

[TicketRestaurant 13](#_Toc22056114)

[Cheque 13](#_Toc22056115)

[Etablissement 13](#_Toc22056116)

[ListePaiement 14](#_Toc22056117)

[Apperçu de la partie Paiement 14](#_Toc22056118)

[Les composants de la partie commande. 15](#_Toc22056119)

[Panier 15](#_Toc22056120)

[Commande 15](#_Toc22056121)

[Statut <<enum>> 16](#_Toc22056122)

[LigneDePanier 16](#_Toc22056123)

[LigneDeCommande 17](#_Toc22056124)

[Apperçu de la partie Commande 17](#_Toc22056125)

[Les relations entre les composants. 19](#_Toc22056126)

[Les associations avec Utilisateur. 19](#_Toc22056127)

[Utilisateur – Magasin 19](#_Toc22056128)

[Les associations avec Client. 19](#_Toc22056129)

[Client – Adresse 19](#_Toc22056130)

[Client – Panier 19](#_Toc22056131)

[Client – Commande 19](#_Toc22056132)

[Les associations avec Panier. 20](#_Toc22056133)

[Panier – Client 20](#_Toc22056134)

[Panier – Produit 20](#_Toc22056135)

[Les associations avec Commande. 21](#_Toc22056136)

[Commande-Client 21](#_Toc22056137)

[Commande – Produit 21](#_Toc22056138)

[Commande – Adresse 21](#_Toc22056139)

[Commande – Paiement 21](#_Toc22056140)

[Les associations avec Produit. 23](#_Toc22056141)

[Produit – Panier 23](#_Toc22056142)

[Produit – Commande 23](#_Toc22056143)

[Produit – Magasin 23](#_Toc22056144)

[Produit - Fournisseur 23](#_Toc22056145)

[Produit – Composition 23](#_Toc22056146)

[Produit – Preparation 24](#_Toc22056147)

[Produit - Produit 24](#_Toc22056148)

[Les autres associations. 25](#_Toc22056149)

[Magasin – Adresse 25](#_Toc22056150)

[Fournisseur - Adresse 25](#_Toc22056151)

[Cheque – Etablissement 25](#_Toc22056152)

[TicketRestaurant - Etablissement 25](#_Toc22056153)

[Le diagramme de classes du domaine fonctionnel 26](#_Toc22056154)

[Le modèle physique de données MDP. 27](#_Toc22056155)

[Les types de données. 27](#_Toc22056156)

[Dénomination. 27](#_Toc22056157)

[La partie Utilisateur. 28](#_Toc22056158)

[Utilisateur 28](#_Toc22056159)

[Client 28](#_Toc22056160)

[Employé 29](#_Toc22056161)

[La partie Produit. 30](#_Toc22056162)

[Produit 30](#_Toc22056163)

[Composition 30](#_Toc22056164)

[Préparation 30](#_Toc22056165)

[Stock 31](#_Toc22056166)

[Composant 31](#_Toc22056167)

[La partie Paiement. 33](#_Toc22056168)

[Paiement 33](#_Toc22056169)

[Carte bancaire 33](#_Toc22056170)

[Chèque 33](#_Toc22056171)

[Ticket restaurant 33](#_Toc22056172)

[Établissement 34](#_Toc22056173)

[Liste paiement 34](#_Toc22056174)

[La partie Commande. 35](#_Toc22056175)

[Panier 35](#_Toc22056176)

[Commande 35](#_Toc22056177)

[Ligne de panier 35](#_Toc22056178)

[Ligne de commande 36](#_Toc22056179)

[Les autres tables. 37](#_Toc22056180)

[Adresse 37](#_Toc22056181)

[Magasin 37](#_Toc22056182)

[Fournisseur 38](#_Toc22056183)

[Diagramme du Modèle Physique de Données. 39](#_Toc22056184)

[Script de création de la base de données MySQL. 40](#_Toc22056185)

[Étude de déploiement. 89](#_Toc22056186)

[La partie utilisateur. 89](#_Toc22056187)

[La partie base de données. 89](#_Toc22056188)

[Les services annexes. 90](#_Toc22056189)

[Le DNS. 90](#_Toc22056190)

[La zone de mémoire cache. 90](#_Toc22056191)

[Le stockage des données publiques. 90](#_Toc22056192)

[Gestion de la sécurité. 90](#_Toc22056193)

[Les microservices. 92](#_Toc22056194)

[Diagramme de déploiement. 93](#_Toc22056195)

[Les composants. 94](#_Toc22056196)

[La partie client. 94](#_Toc22056197)

[Navigateur web. 94](#_Toc22056198)

[Applicatif APP/APK. 94](#_Toc22056199)

[La partie publique. 95](#_Toc22056200)

[La partie sécurisée. 95](#_Toc22056201)

[La base de données. 95](#_Toc22056202)

[Les Microservices 95](#_Toc22056203)

[La partie externe. 96](#_Toc22056204)

[Le système bancaire. 96](#_Toc22056205)

[Le diagramme de composant. 97](#_Toc22056206)

[Table des matières. 98](#_Toc22056207)