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.

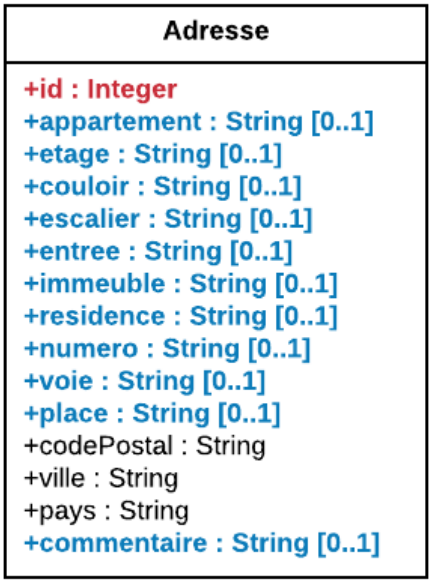
On utilise une approche orientée objet pour représenter les composants de notre domaine fonctionnel. Le diagramme de classes servira de base à la modélisation de celui-ci. Le diagramme de classes fait partie des diagrammes qui respectent la norme de modélisation graphique UML (Unified Modeling Language).

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.

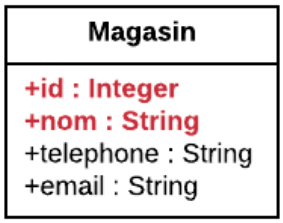
Adresse



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.

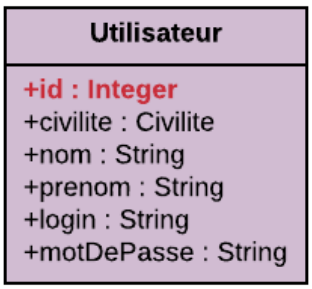


Les composants de la partie utilisateur.

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.

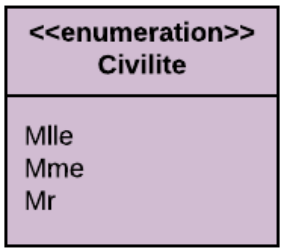
Pour les employés on a juste besoin de connaître leur rôle pour leur affecter des droits sur le système. On va créer donc un composant "Utilisateur" et deux composant qui vont en hérités "Client" et "Employe".

Utilisateur

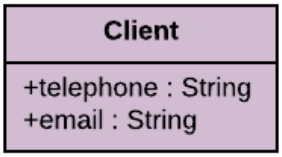


Un champ unique "id" permet d'identifier un utilisateur. Chaque "Utilisateur" a un "Magasin" attitré soit parce que c'est son magasin auprès duquel il effectue ses commandes ou soit parce qu'il y travaille. On fait référence à ce "Magasin" par son identifiant unique "id". On utilise une énumération "Civilite"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

Civilite <<enum>>

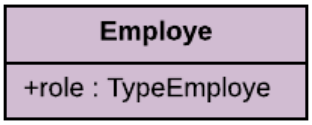


Client



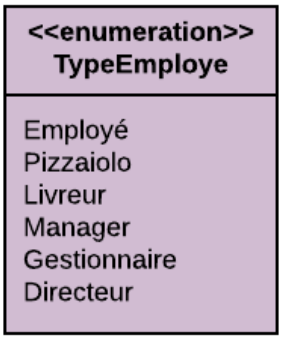
Contrairement à un employé, on doit connaître les coordonnées du "Client" pour pouvoir effectuer la vente et la livraison. L'adresse de livraison est référencée par un identifiant du composant "Adresse".

Employe

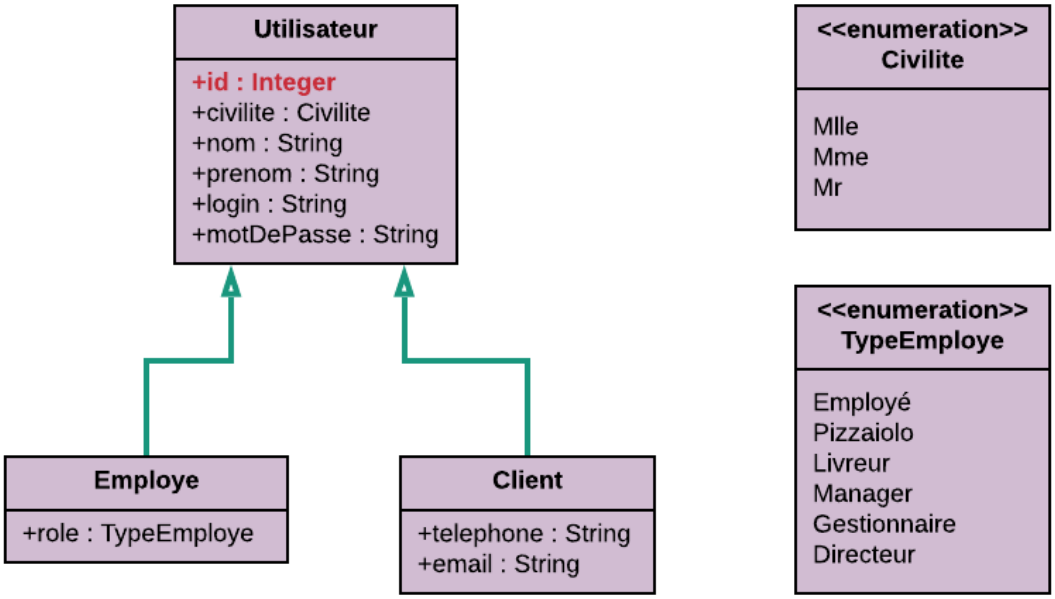


Pour définir le rôle de l'employé donc ses droits sur le système, on utilise une énumération "TypeEmploye".

TypeEmploye <<enum>>



Apperçu de la partie Utilisateur

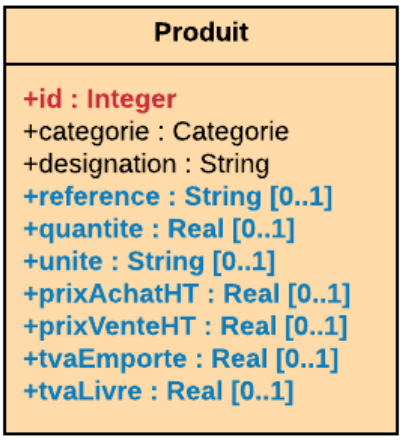


On voit bien l'héritage que font les objets "Employe" et "Client" de "Utilisateur"

Les composants de la partie produit.

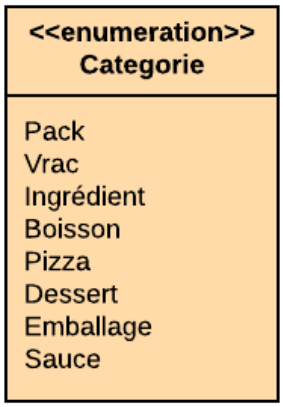
On doit gérer les ingrédients de base, les produits manufacturés et les produits additionnels comme les boissons. On doit enregistrer le prix d'achat et le prix de vente hors taxes pour chaque produit vendable. La TVA est gérée lors du passage de la commande suivant si elle est livrée ou emportée.

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>>

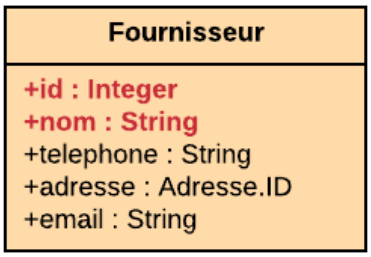


Un composant "Categorie" est une énumération qui regroupe les différentes "categorie" de "Produit". On utilisera :

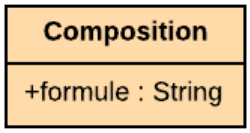
- Pack : pour les produits en groupes livrés par le fournisseur (caisse de 24 canettes),
- Boisson : pour les produits en stock qui sont à l'origine des pack mais vendu à l'unité.
- Ingrédient : pour les produits livrés par le fournisseur, au poids ou au volume et utilisés en vrac (farine, légumes, viande,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



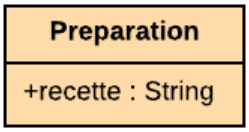
Composition



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".

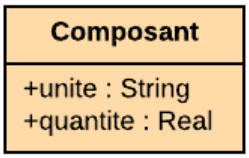
Preparation



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".

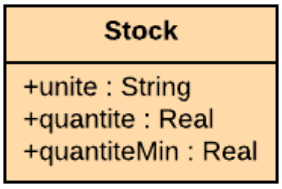
Composant



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.

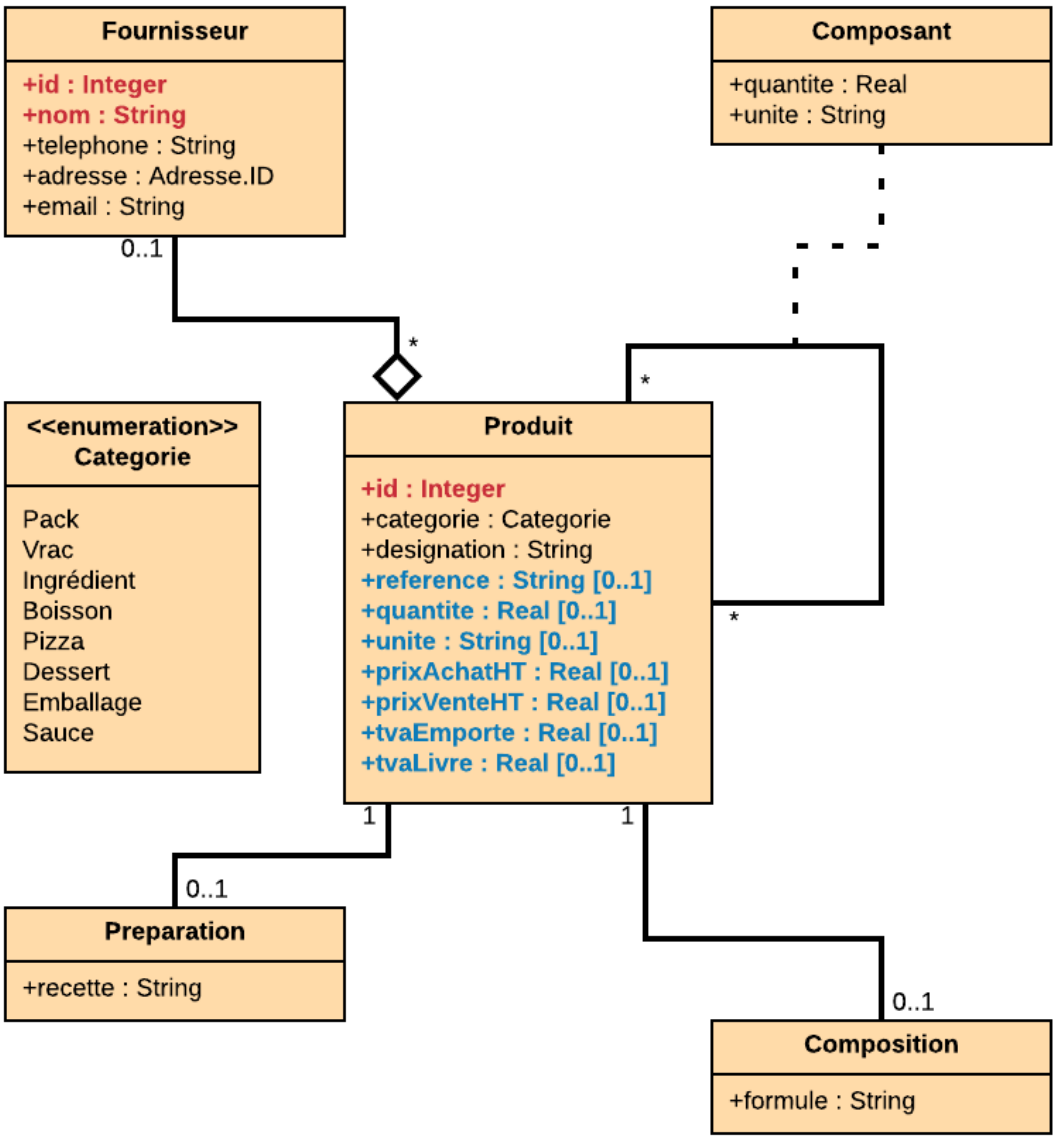
Un objet "Stock" permet de garder les informations sur les stocks de chaque produit dans chaque magasin.

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

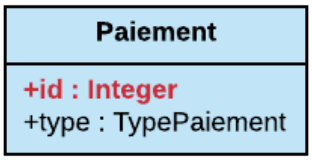


On remarque bien l'objet "Composant" qui comporte l'information entre les composés "Produit" et les composants "Produit".

Les composants de la partie paiement.

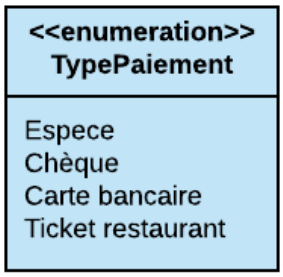
On utilise un objet "Paiement" dont héritera les différents types de paiement : "CarteBancaire", "TicketRestaurant" et "Cheque". Les paiements par espèce n'ont pas d'information spécifique donc ils n'ont pas besoin d'un composant qui hérite de "Paiement".

Paiement

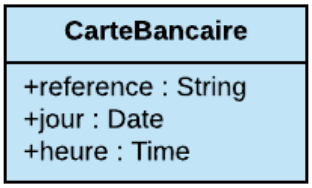


On utilise en attribut un identifiant commun à tous les paiement et un autre pour connaître le type de paiement qui fait référence à une énumération "TypePaiement".

TypePaiement <<enum>>

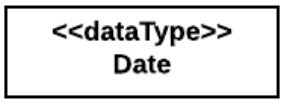


CarteBancaire

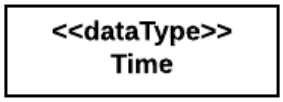


On utilise des <<dataType>> pour le jour et l'heure.

Date <<dataType>>

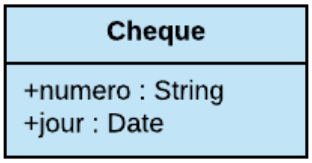


Time <<dataType>>



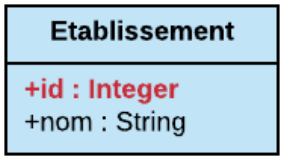
TicketRestaurant

Cheque



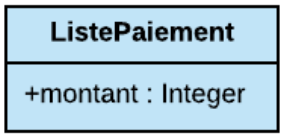
Pour éviter les mauvaises saisies et la redondance on utilise un composant "Etablissement" pour faire référence aux banques émettrices de "Cheque" et aux organismes de gestion des "TicketRestaurant".

Etablissement



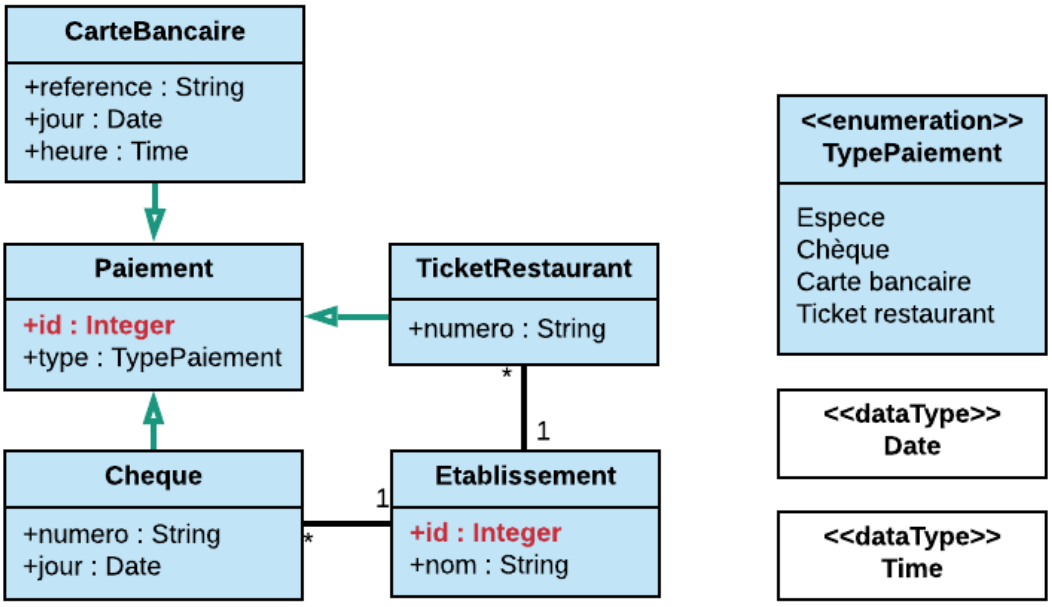
Cet objet contiendra les noms des établissements bancaires et des organismes de titre restaurant.

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

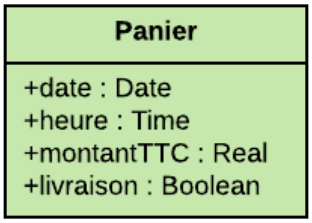


Les objets "CarteBancaire", "Cheque" et "TicketRestaurant" héritent de "Paiement". Le "Paiement" par espèce sera directement dans la classe mère car il n'a pas d'attribut en plus.

Les composants de la partie commande.

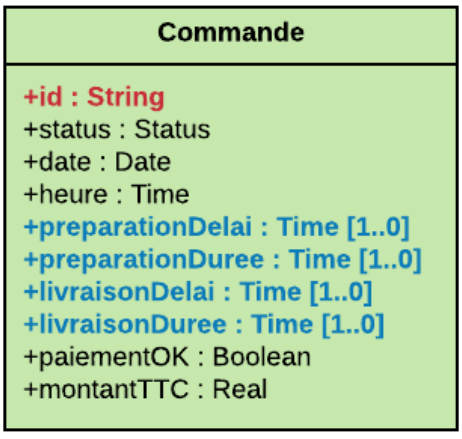
On distingue le panier que le client rempli de produits et la commande qui contient les paniers validés. Le composant "Panier" permet de garder les informations sur les produits que le Client choisi avant de valider sa commande. On doit enregistrer la date est l'heure pour pouvoir effacer les paniers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

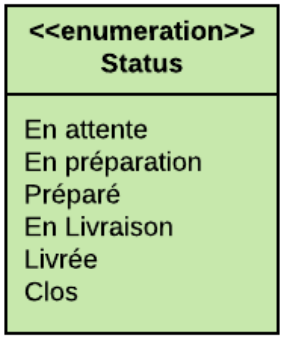
Commande



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.

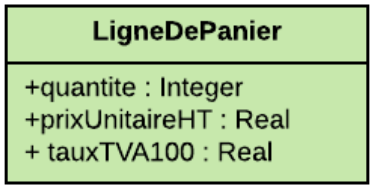
- preparationDelai : est le temps entre la validation de la commande par l'utilisateur et le début de préparation par le "Pizzaiolo".
- preparationDuree : est le temps de préparation de la commande par le "Pizzaiolo".
- livraisonDelai : est le temps mis pour finaliser la commande avant son départ en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

Statut <<enum>>



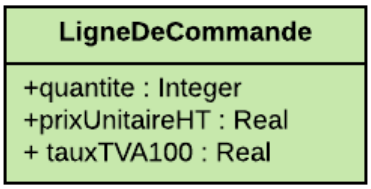
Les informations sur le contenu des "Commande" et des "Panier" sera enregistré dans les objets "LigneDeCommande" et "LigneDePanier" respectivement. On garde les prix en Hors-Taxes car si la vente est à emporter le taux de TVA n'est pas le même.

LigneDePanier



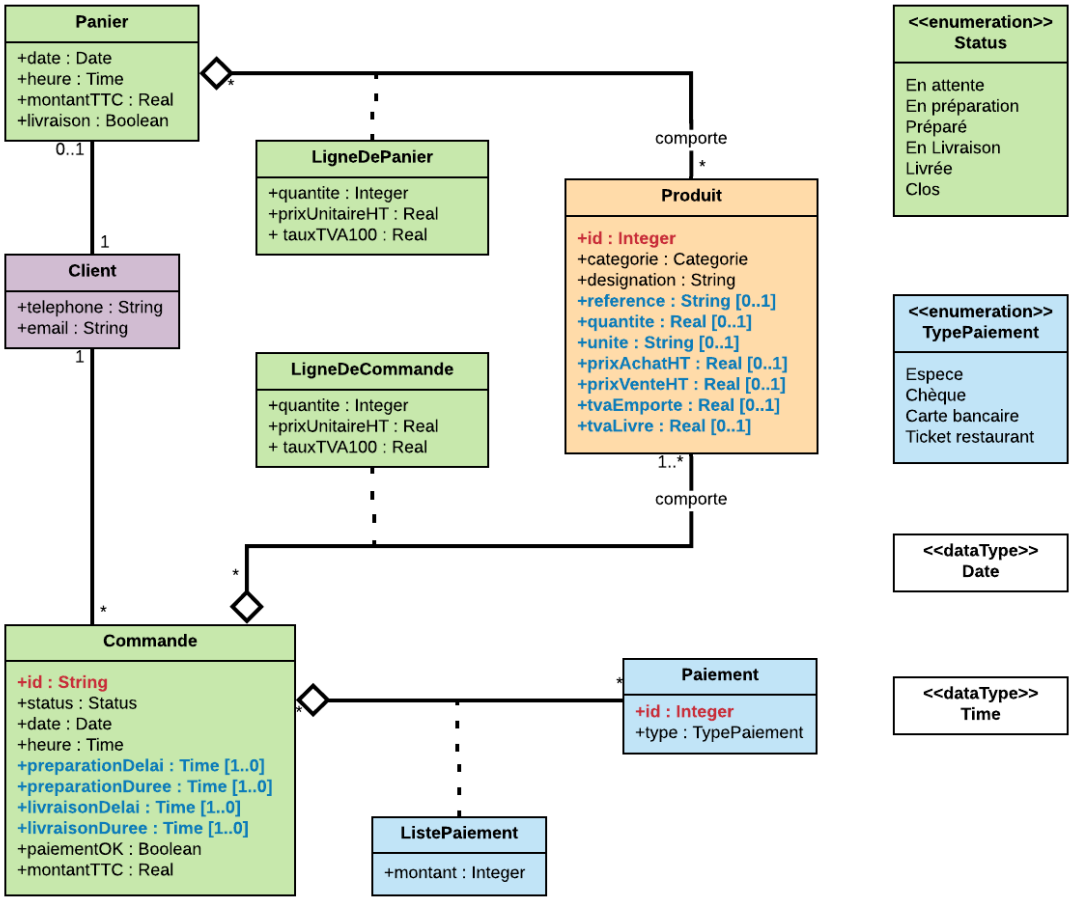
On identifie chaque tuple par l'identifiant du "Panier" donc celui "Utilisateur" et l'identifiant du "Produit".

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 ".

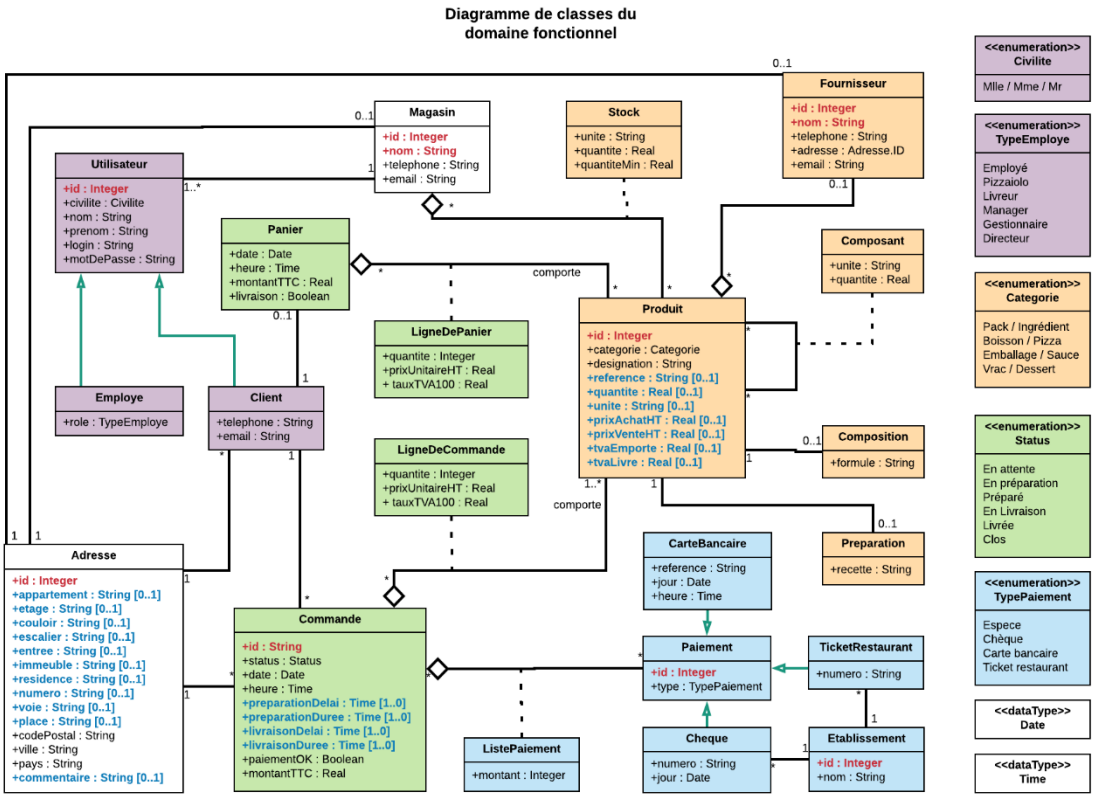
Cheque – Etablissement

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

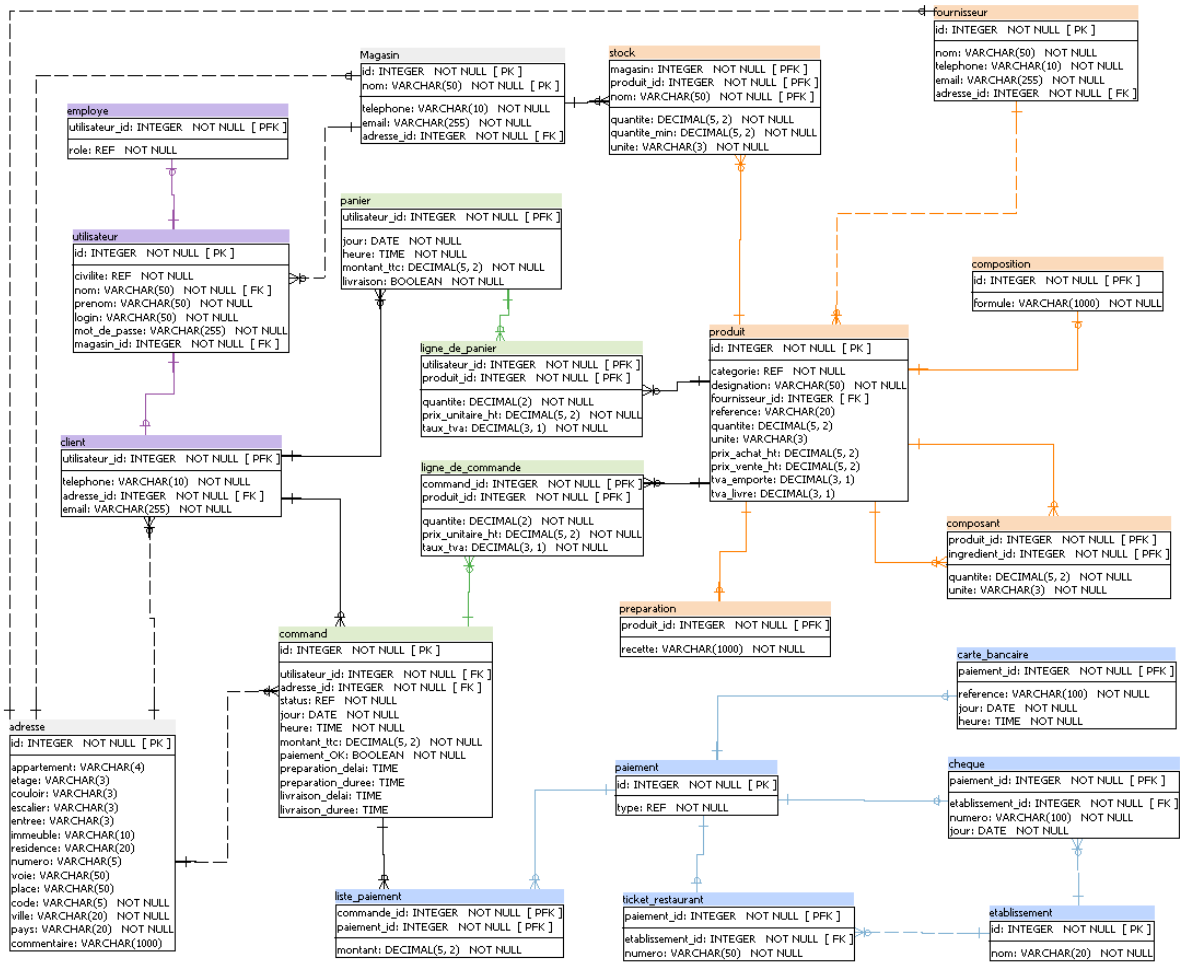
Clé primaire sur "nom"

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,pri
x_vente_ht,tva_emporte,tva_livre)
   VALUES
(p_designation,p_categorie,p_fournisseur_id,p_reference,p_quantite,p_unite,p_pri
x_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_ta
ux_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','Direc
tion') )
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,utilisate
ur.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,'Accue
il',@ID);
CALL
create_employe('M','TINTIN','Milou','Milou',SHA1('TINTIN'),1,'Livreur',@ID);
CALL
create_employe('M','HADDOCK','Capitaine','Capitaine',SHA1('HADDOCK'),1,'Directio
n',@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.rack
ham@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,@COM
P);
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,@CO
MP);
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.

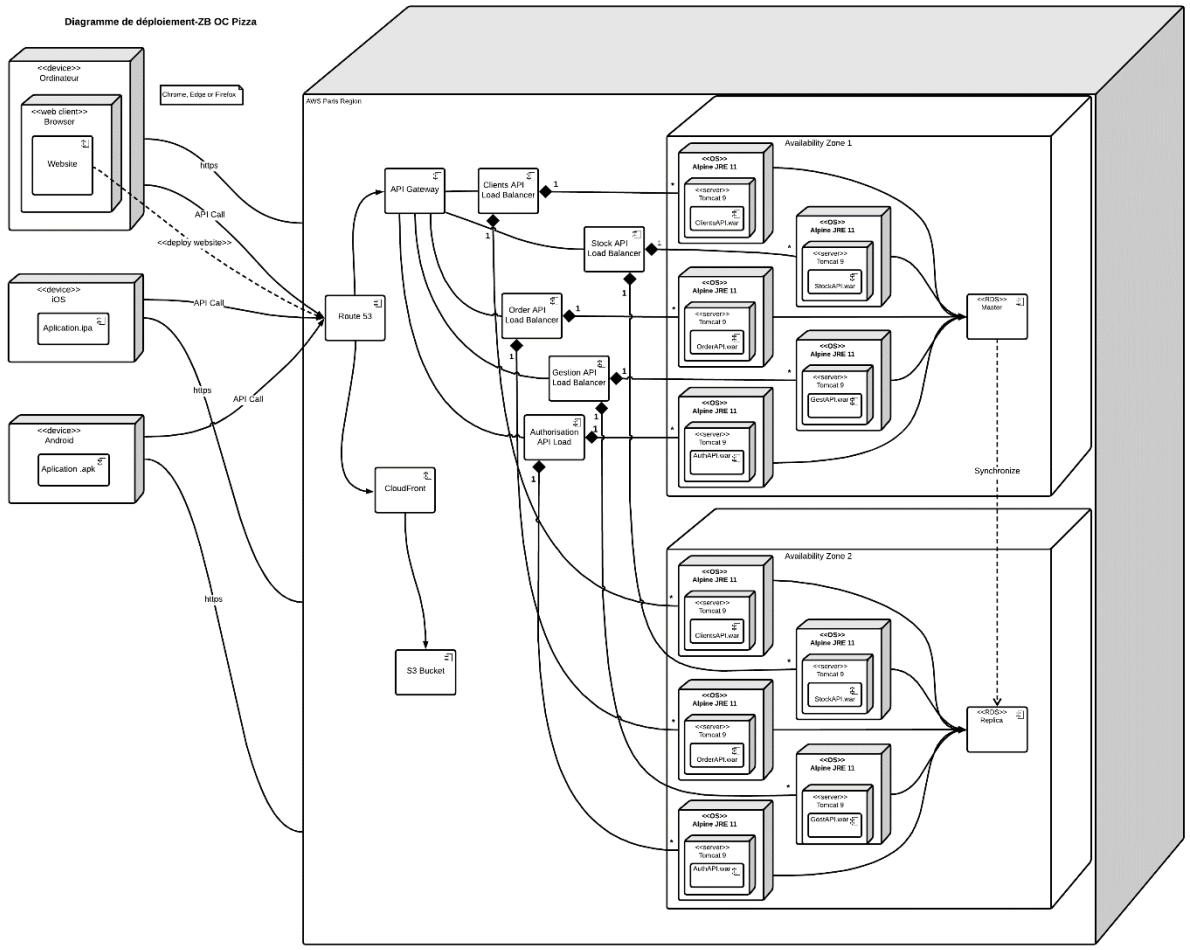
Les microservices.

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

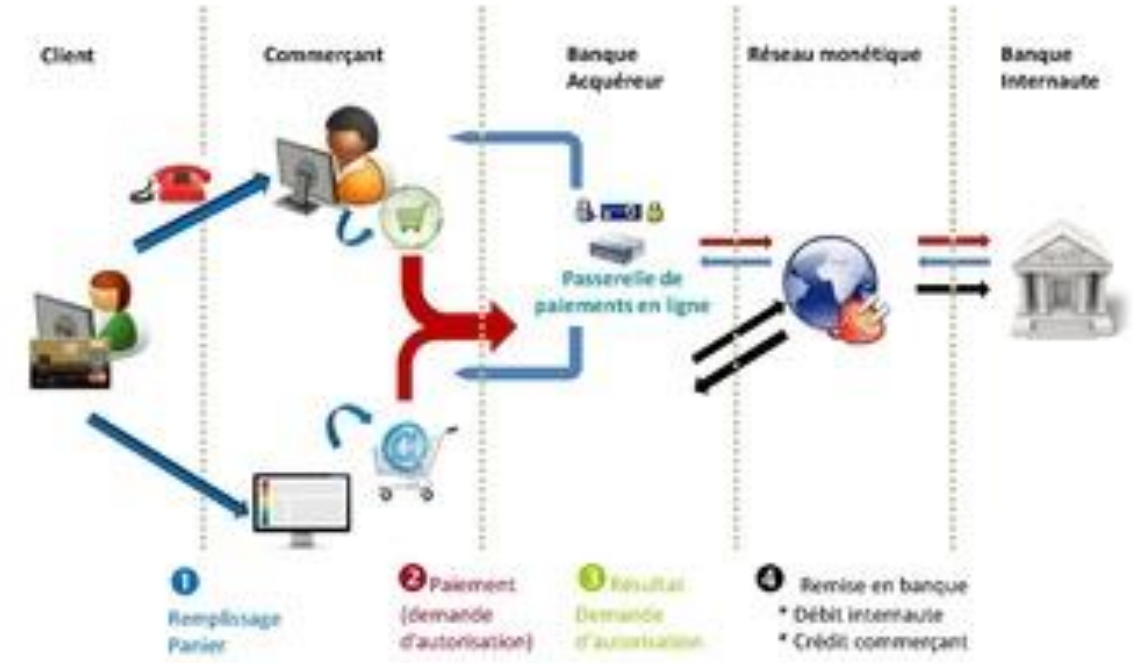
Ils seront implémentés en **Java** avec **SPRING** et packagé en **JAR** avec **MAVEN** pour faciliter la mise à jour et le déploiement. L'archive **JAR** sera déployée sur un serveur **Tomcat 9.0.24** qui tourne sur une instance **AWS EC2** sous **UNIX**. L'accès se fait par requête **REST** via **HTTPS**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



Le diagramme de composant.

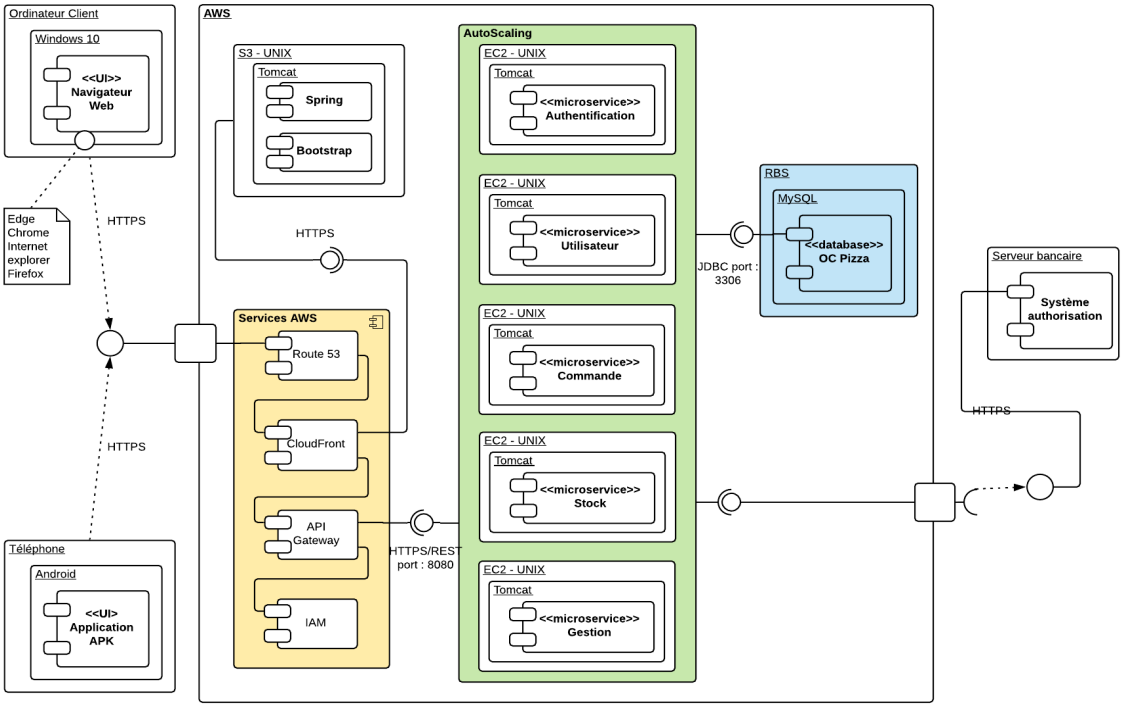


Table des matières.