



Dossier Spécifications Techniques



Suivi des évolutions

Mises à jour			
Version	Date	Auteurs	Objet de la mise à jour
1.0	30 Avril 2021	Johann Ulma	Création du document



Sommaire

1	• Cadre du projet	p. 4
2	• Rappel des choix	p. 5
3	• Le domaine fonctionnel	p. 6
4	• Diagramme de classe	p. 7
5	• Détail des classes	p. 8 - 11
6	• Détail des énumérations	p. 12 - 14
7	• Relation de composants	p. 15 - 16
8	• Le modèle physique de données	p. 17 - 19
9	• Les données et les requêtes	p. 20 - 23
10	• Diagramme de composants	p. 24 - 25
11	• Diagramme de déploiement	p. 26

Cadre du projet

Contexte

Le groupe OC Pizza est une chaîne de pizzerias en plein essor. Comptant actuellement 5 points de ventes, le groupe prévoit d'en ouvrir au moins 3 nouveaux d'ici 6 mois. Le système informatique actuel n'est pas adapté et doit être repensé.

Besoin

Le client a spécifié ses besoins pour ce projet.

Le système doit permettre les actions suivantes :

- Gestion des commandes : de la réception à la livraison en passant par la préparation
- Suivi des commandes en temps réel : commandes passées, en préparation, en livraison
- Gestion du stock des pizzerias : stock des ingrédients restant pour connaître les pizzas pouvant être encore réalisées
- Mise en place d'un site web pour les clients permettant de :
 - Passer des commandes
 - Payer en ligne
 - Modifier, annuler des commandes
- Gestion des recettes : Mise en place d'un aide mémoire

Objectifs

- Mise en place d'un site web et d'une application
- Suivi du groupe
- Prise de commandes sur place, à emporter et en ligne
- Avoir une gestion centralisée

Direction de projet

Nom du projet : OC Pizza : Mise en place d'un nouveau système

Date de lancement : Mars 2021

Nom du client : OC Pizza

Référents client : Lola & Franck (Gérants)

Référents agence : Alexandra et Johann

Deadline du projet : Dans 6 mois (Septembre 2021)

Rappel des choix

Lors de la première phase du projet nous avons pu identifier les acteurs du projet qui seront :

- Les clients
- Les livreurs
- Les hôtesses
- Les pizzaiolos
- Les gérants

On a pu par la même occasion identifier les fonctionnalités que nous allons retrouver dans notre domaine fonctionnel.

Et nous avons effectué les choix techniques sur les technologies qui seront utilisées notamment la base de données qui sera du MySQL.

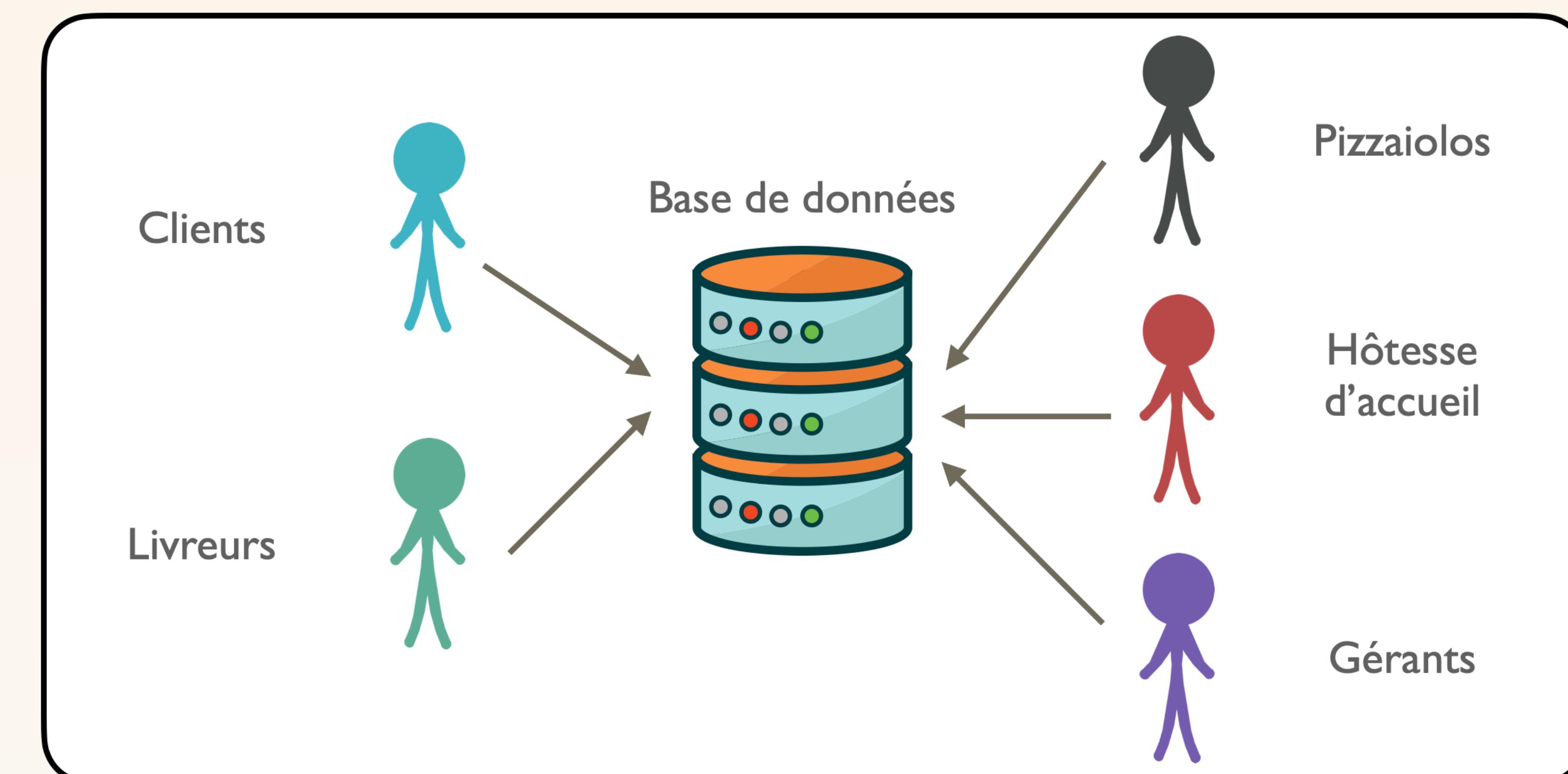


Pour rappel :

Le choix de MySQL est lié aux faits suivants : C'est une technologie Open Source et gratuite, et selon DB-Engines c'est la deuxième base de données la plus utilisée au monde.

Le développement portera donc sur la création d'un site Web, d'une application codée pour iOS et d'une base de données.

Le système de paiement Monext étant externalisé pour des soucis de certifications en vigueur.



Le domaine fonctionnel

Le domaine fonctionnel va nous permettre d'identifier les éléments et les informations que l'on veut dans notre base de données. C'est la première étape avant de passer à la modélisation de notre base de données.

Nous allons adopter l'approche orientée objet (AOO) afin de représenter les concepts.

Pour ce faire la modélisation de notre diagramme de classe utilisera la norme UML afin de représenter les aspects de la conception du système.

En corrélation avec le travail effectué auparavant on va définir les classes qui seront nos différentes tables pour la base de données.

Le lexique utilisé sera donc celui de UML dont la légende est fournie ci-dessous.

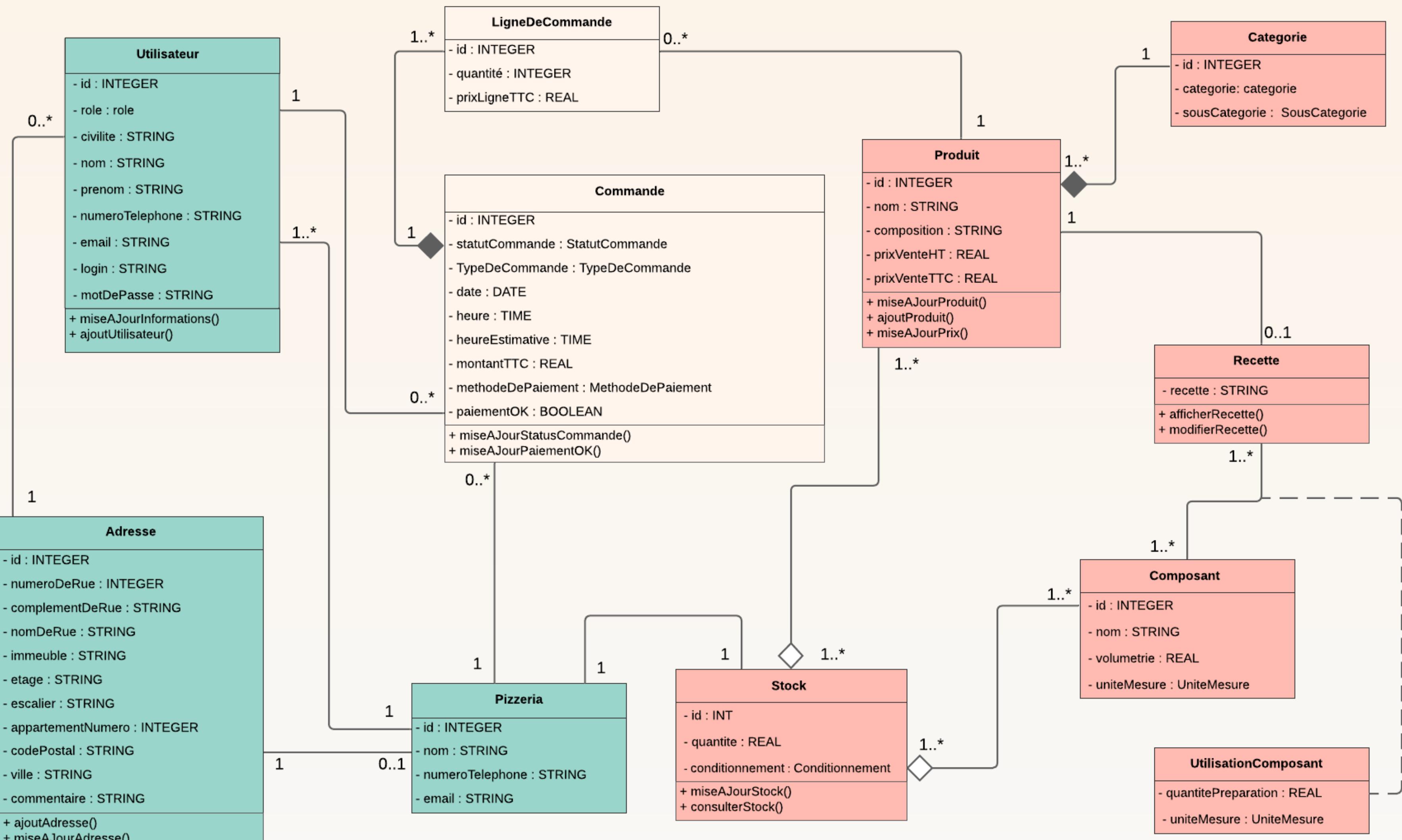
Les relations s'expriment sous forme de multiplicités (Combien d'instances d'une classe peuvent être liées à une instance de la classe de l'autre côté de l'association), légende ci-dessous.

Le critère NULL signifie que l'attribut ne contient aucune valeur, à l'inverse le NOT NULL spécifie que l'attribut aura obligatoirement une valeur.

Type de données utilisées		
Type	Définition	Exemple
BOOLEAN	Pour les valeurs Vrai/ Faux	True / false
INTEGER	Pour les nombres entiers	100
REAL	Pour les nombres décimaux	123.56
STRING	Pour les chaînes de caractères (texte)	Pizza Calzone
Enum	Pour les références aux énumérations	role -> client, livreur, ...

Synthèse des multiplicités		
Multiplicité	Abréviation	Cardinalités
0..0	0	Aucune instance
0..1		Aucune ou une seule instance
1..1	1	Exactement une instance
0..*	*	Aucune, une ou plusieurs instances (aucune limite)
1..*		Au moins une instance (aucune limite maximum)
x..x	x	Exactement x instance(s)
m..n		Au moins m et au plus n instances

Diagramme de classe



Enum Role
client
pizzaiolo
livreur
hotesse
gérant
administrateur

Enum Categorie
Entree
Pizza
Salade
Bruschetta
Dessert
Boisson
Sauce

Enum StatutCommande
En attente
En preparation
Prete
A retirer
En Livraison
Livree
Close
Annulée

Enum SousCategorie
Base crème
Base tomate
Vegan
Spéciale
Sucrée
Glacé
Patisserie
Gazeuse
Plate

Enum TypeDeCommande
retrait
livraison
uber eat
deliveroo

Enum UniteMesure
Gramme
Milititre
Centilitre
Piece
Tranche

Enum MethodeDePaiement
Especie
Carte Bancaire
Titre Restaurant
Paypal

Enum Conditionnement
boite
bouteille
pot
sachet
canette
piece

Détail des classes

Nous allons détailler les différentes classes, leurs attributs, fournir un exemple du type de données et enfin préciser si un attribut sera dit nullable (NULL) ou non.

Adresse
- id : INTEGER
- numeroDeRue : INTEGER
- complementDeRue : STRING
- nomDeRue : STRING
- immeuble : STRING
- etage : STRING
- escalier : STRING
- appartementNumero : INTEGER
- codePostal : STRING
- ville : STRING
- commentaire : STRING
+ ajoutAdresse()
+ miseAJourAdresse()

Cette classe va contenir les adresses des utilisateurs et des pizzerias. Elle sera reliée aux classes Utilisateur et Pizzeria.				
id	Identifiant unique attribué lors de la création d'une adresse	56	NOT NULL	
numeroDeRue	Numéro lié à la rue de l'utilisateur ou de la pizzeria	124	NULL	
complementDeRue	Type spécifique de rue (bis, ter, lieu dit, ...)	Bis	NULL	
nomDeRue	Nom de la rue de l'utilisateur ou de la pizzeria	Rue Francis Bacon	NOT NULL	
immeuble	Nom de l'immeuble	Immeuble Lumière	NULL	
etage	Numéro de l'étage	3	NULL	
escalier	Numéro d'escalier	3	NULL	
appartementNumero	Numéro de l'appartement	45	NULL	
codePostal	Code Postal lié à l'adresse	78200	NOT NULL	
ville	Nom de la ville	Mantes-la-Jolie	NOT NULL	
commentaire	Permet de renseigner des éléments spécifiques (digicode, ...)	Digicode 1212, Sonner en bas	NULL	
ajoutAdresse	Permet la création d'une adresse dans la base de données			
miseAJourAdresse	Permet la mise à jour de l'adresse en cas de changement			

Numéro de Rue est possiblement NULL car la personne peut-être à une voie sans numéro

Pizzeria
- id : INTEGER
- nom : STRING
- numeroTelephone : STRING
- email : STRING

Cette classe va contenir la liste des Pizzerias du groupe. Elle sera reliée aux classes Adresse, Utilisateur, Commande et Stock.				
id	Identifiant unique attribué lors de la création d'une pizzeria	9	NOT NULL	
nom	Nom de la Pizzeria	OC Pizza Mantes-la-jolie	NOT NULL	
numeroTelephone	Numéro de téléphone de la Pizzeria	0199000278	NOT NULL	
email	Adresse mail de contact de la Pizzeria	mantes@ocpizza.fr	NOT NULL	

Projet OC Pizza : Spécifications Technique

Utilisateur	Cette classe va contenir les informations, coordonnées des utilisateurs (clients et employés). Elle sera reliée aux classes Adresse, Pizzeria et Commande.			
- id : INTEGER			99	NOT NULL
- role : role		Pizzaiolo		NOT NULL
- civilité : STRING		Mr		NULL
- nom : STRING		Bachelard		NOT NULL
- prénom : STRING		Morgan		NOT NULL
- numéroTelephone : STRING		0199000277		NOT NULL
- email : STRING		morgan.bachelard@ocpizza.fr		NOT NULL
- login : STRING		morgan.bachelard@ocpizza.fr		NOT NULL
- motDePasse : STRING		9kDcYh2Q		NOT NULL
+ miseAJourInformations()	Changer les informations de l'utilisateur			
+ ajoutUtilisateur()	ajoutUtilisateur	Ajout d'un utilisateur		

Commande	Cette classe va contenir l'ensemble des commandes. Elle sera reliée aux classes Utilisateur, Pizzeria et LigneDeCommande.			
- id : INTEGER		173		NOT NULL
- statutCommande : StatutCommande		En préparation		NOT NULL
- TypeDeCommande : TypeDeCommande		Livraison		NOT NULL
- date : DATE		14-04-2021		NOT NULL
- heure : TIME		12:36:23		NOT NULL
- heureEstimative : TIME		13:03:00		NOT NULL
- montantTTC : REAL		35.6		NOT NULL
- méthodeDePaiement : MéthodeDePaiement		Carte bancaire		NULL
- paiementOK : BOOLEAN		true		NOT NULL
+ miseAJourStatusCommande()	Changer le statut d'une commande			
+ miseAJourPaiementOK()	miseAJourPaiementOK	Permet la mise à jour du paiement, une fois effectué		

PaiementOK est un Booléen permettant rapidement de changer le statut de celui-ci

true	Le paiement est validé
false	Le paiement n'est pas encore validé

Projet OC Pizza : Spécifications Technique

LigneDeCommande
- id : INTEGER
- quantité : INTEGER
- prixLigneTTC : REAL

Cette classe va contenir le détail d'une ligne de commande.
Elle sera reliée aux classes Commande et Produit

id	Identifiant unique pour chaque ligne	1	NOT NULL
quantité	Défini le nombre d'éléments	2	NOT NULL
prixLigneTTC	Prix de la Ligne TTC	30.0	NOT NULL

Produit
- id : INTEGER
- nom : STRING
- composition : STRING
- prixVenteHT : REAL
- prixVenteTTC : REAL
+ miseAJourProduit()
+ ajoutProduit()
+ miseAJourPrix()

Cette classe va contenir les informations des produits vendus dans l'enseigne.
Elle sera reliée aux classes LigneDeCommande, Categorie, Recette et Stock.

id	Identifiant unique attribué lors de la création d'un produit	53	NOT NULL
nom	Nom du produit	Onion Rings	NOT NULL
composition	Détail des éléments qui font le produit	6 anneaux d'oignons panés	NULL
prixVenteHT	Prix de vente du produit HT	3.33	NULL
prixVenteTTC	Prix de vente du produit TTC	3.99	NOT NULL
miseAJourProduit	Mettre à jour la liste des produits disponibles		
ajoutProduit	Ajouter de nouveaux produits		
miseAJourPrix	Mettre à jour les prix		

Categorie
- id : INTEGER
- categorie: categorie
- sousCategorie : SousCategorie

Cette classe va contenir les catégories et sous catégories de produits.
Elle sera reliée à la classe Produit.

id	Identifiant unique attribué à une catégorie	3	NOT NULL
categorie	Type de produit	pizza	NOT NULL
sousCategorie	Sous Catégorie pour un type de produit	Base crème	NULL

Recette
- recette : STRING
+ afficherRecette()
+ modifierRecette()

Cette classe va contenir le détail des étapes d'une recette.
Elle sera reliée aux classes Produit et UtilisationComposant.

recette	Affiche le détail de la recette de la pizza	1 - Préchauffer ...	NOT NULL
afficherRecette	Affiche le détail de la recette pour le cuisinier		
modifierRecette	Modifier une recette		

Composant
- id : INTEGER
- nom : STRING
- volumetrie : REAL
- uniteMesure : UniteMesure

Cette classe va contenir les composants lors de la préparation d'un produit.
Elle sera reliée aux classes Recette, Stock et UtilisationComposant.

id	Identifiant unique attribué lors de la création d'un composant	28	NOT NULL
nom	Nom de l'ingrédient	Roquette	NOT NULL
volumetrie	Défini la quantité numéraire	200.00	NOT NULL
uniteMesure	Défini l'unité de mesure (gramme, pièce, ...)	grammes(s)	NOT NULL

UtilisationComposant
- quantitePreparation : REAL
- uniteMesure : UniteMesure

Cette classe va afficher des ingrédients de la pizza pour le client.
Elle sera reliée aux classes Composant et Recette.

quantitePreparation	Défini la quantité numéraire d'un composant	7	NOT NULL
uniteMesure	Défini l'unité de mesure (gramme, pièce, ...)	tranche(s)	NOT NULL

Stock
- id : INT
- quantite : REAL
- conditionnement : Conditionnement
+ miseAJourStock()
+ consulterStock()

Cette classe va contenir le détail du Stock disponible d'une pizzeria.
Elle sera reliée aux classes Pizzeria, Composant et Produit.

id	Nom de l'élément du stock	29	NOT NULL
quantite	Défini la quantité numéraire	6.00	NOT NULL
conditionnement	Défini l'unité de stockage de l'élément (boîte, bouteille, ...)	sachet(s)	NOT NULL
miseAJourStock			Mise à jour du stock de la pizzeria
consulterStock			Consulter le stock de la pizzeria

Détail des énumérations

Comme vu précédemment, nous avons recours à des énumérations pour différentes classes nous allons détailler les paramètres possibles.

Enum Role
client
pizzaiolo
livreur
hotesse
gérant
administrateur

Détail des différents types de rôles possibles pour l'utilisateur. Il s'agit d'une énumération liée à la classe Utilisateur.	
Client	La personne n'est pas un membre de l'équipe OC Pizza, les droits sont limités
Pizzaiolo	Définit les droits pour le Pizzaiolo
Livreur	Définit les droits pour le Livreur
Hôtesse	Définit les droits pour l'hôtesse
Gérant	Définit les droits pour le Gérant
Administrateur	Définit les droits pour l'Administrateur

Le gérant dispose de l'accès le plus haut incluant la gestion des comptes, hors aspects techniques.

L'administrateur dispose d'un accès plus poussé que le gérant celui-ci qui va inclure la gestion technique au site Web et à l'application (accès au code source).

Enum StatutCommande
En attente
En préparation
Prête
A retirer
En livraison
Livrée
Close
Annulée

Détail des différents statuts possibles pour la commande. Il s'agit d'une énumération liée à la classe Commande.	
En attente	La commande est passée mais pas encore lancée en cuisine
En préparation	La commande est en cuisine
Prête	La commande est finie côté cuisine
A retirer	La commande est disponible en retrait
En livraison	La commande est en cours de livraison
Livrée	La commande est confirmée comme livrée
Close	La commande est clôturée
Annulée	La commande est annulée

Le suivi temps réel va permettre de modifier le statut de la commande.

Enum MethodeDePaiement
Espèce
Carte Bancaire
Titre Restaurant
Paypal

Cette Enumération va contenir les méthodes de Paiement proposées. Il s'agit d'une énumération liée à la classe Commande.	
Espèce	Le client paye en espèce
Carte Bancaire	Le client paye par carte ou sans contact
Titre Restaurant	Le client utilise un Titre Restaurant (Ticket Restaurant, Chèque Déjeuner, ...)
Paypal	Le client paye via Paypal

Enum TypeDeCommande
retrait
livraison
<i>uber eat</i>
<i>deliveroo</i>

Définit le type de commande effectuée par le client.
 Il s'agit d'une énumération liée à la classe Commande
 Le fait de choisir un Enum permet d'implémenter de futures méthodes de commande (**exemple en rouge**)

- | | |
|----------|------------------|
| 1 | Retrait |
| 2 | Livraison |
| 3 | Uber eat |
| 4 | Deliveroo |

Enum Categorie
Entree
Pizza
Salade
Bruschetta
Dessert
Boisson
Sauce

Détail des différents types de catégories pour l'appartenance d'un produit.
 Il s'agit d'une énumération liée à la classe Categorie.

- | | |
|------------|--|
| Entree | Le produit est du type Entree (Produits panés) |
| Pizza | Le produit est du type Pizza |
| Salade | Le produit est du type Salade |
| Bruschetta | Le produit est du type Bruschetta |
| Dessert | Le produit est du type Dessert |
| Boisson | Le produit est du type Boisson |
| Sauce | Le produit est du type Sauce |

Enum SousCategorie
Base crème
Base tomate
Vegan
Spéciale
Sucrée
Glacé
Patisserie
Gazeuse
Plate

Détail des différents types de catégories pour l'appartenance d'un produit.
 Il s'agit d'une énumération liée à la classe Categorie.

- | | |
|-------------|------------------------|
| Base crème | Sous-classe de pizza |
| Base tomate | Sous-classe de pizza |
| Vegan | Sous-classe de pizza |
| Spéciale | Sous-classe de pizza |
| Sucrée | Sous-classe de pizza |
| Glacé | Sous-classe de dessert |
| Patisserie | Sous-classe de dessert |
| Gazeuse | Sous-classe de boisson |
| Plate | Sous-classe de boisson |

Enum UniteMesure
Gramme
Millilitre
Centilitre
Pièce
Tranche

Détail des différentes unités de mesure pour les composants.
Il s'agit d'une énumération liée aux classes Composant et UtilisationComposant.

Enum Conditionnement
boîte
bouteille
pot
sachet
canette
pièce

Détail des différents conditionnements pour les composants.
Il s'agit d'une énumération liée à la classe Stock.

Boîte	Le contenant du composant est une boîte
Bouteille	Le contenant du composant est une bouteille
Pot	Le contenant du composant est un pot
Sachet	Le contenant du composant est un sachet
Canette	Le contenant du composant est une canette
Pièce	Le composant est une pièce sans contenant spécifique

On a recours à plusieurs énumérations car de nombreuses données ont besoin de paramètres spécifiques et qui vont se répéter, de plus il sera facile d'ajouter de nouveaux paramètres comme on peut le voir pour l'enum TypeDeCommande.

7

Relations de composants

Comme nous l'avons vu dans le diagramme de classe, nous avons des multiplicités qui existent. La manière de les interpréter est détaillée ci-dessous.



Un « Utilisateur » est lié à une seule « Adresse ». Mais une « Adresse » peut être liée à aucun ou plusieurs « Utilisateur » par exemple dans le cas d'une collocation.



Un « Utilisateur » est lié à une seule « Pizzeria », que ce soit un employé ou un client. Une « Pizzeria » comportera au moins à un « Utilisateur » car nous aurons à minima les employés.



Une « Adresse » peut-être liée à aucune ou une seule « Pizzeria ». Cependant une « Pizzeria » sera liée à une seule « Adresse ».



Une « Commande » est attribuée à seule « Pizzeria ». Mais une « Pizzeria » peut-être liée à aucune ou une multitude de « Commande ».



Une « Commande » est obligatoirement liée à un seul « Utilisateur ». Tandis qu'un « Utilisateur » peut-être lié à aucune « Commande » ou plusieurs.



Une « Commande » va contenir une ou plusieurs « LigneDeCommande ». Une « LigneDeCommande » appartient à une seule « Commande ».



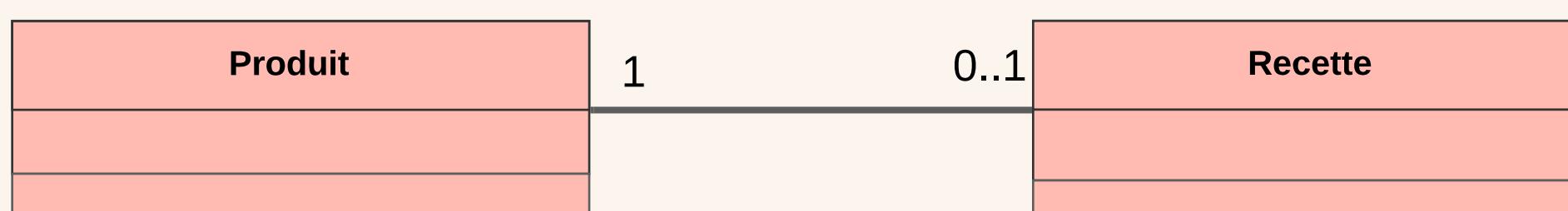
Une « LigneDeCommande » est contient un seul « Produit ». Mais un « Produit » peut-être attribué à aucune ou une multitude de « LigneDeCommande ».

Synthèse des multiplicités

Multiplicité	Abréviation	Cardinalités
0..0	0	Aucune instance
0..1		Aucune ou une seule instance
1..1	1	Exactement une instance
0..*	*	Aucune, une ou plusieurs instances (aucune limite)
1..*		Au moins une instance (aucune limite maximum)
x..x	x	Exactement x instance(s)
m..n		Au moins m et au plus n instances



Un « Produit » appartient à une seule « Categorie ».
Mais un « Categorie » peut-être attribuée à un ou une multitude de « Produit ».



Un « Produit » peut comprendre une seule « Recette » voir aucune.
Cependant une « Recette » est sera liée à un seul « Produit ».



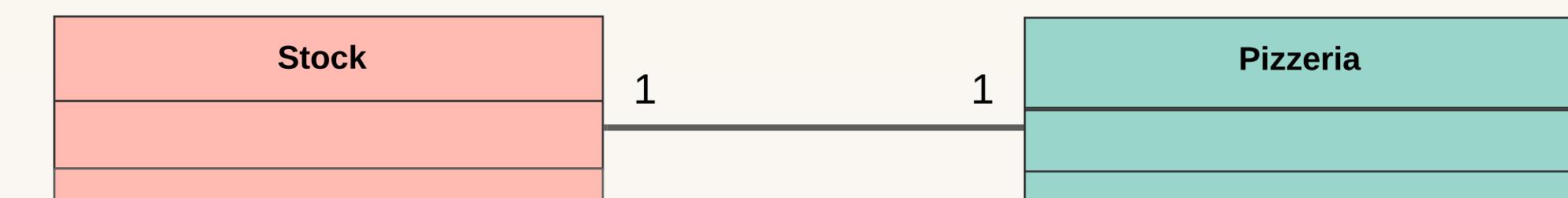
Un « Recette » est comprendra un ou plusieurs « Composant ».
Mais un « Composant » peut appartenir à une ou plusieurs « Recette ».



Un « Composant » peut appartenir à plusieurs « Stock » de Pizzeria.
Un « Stock » comprendra à minima un ou plusieurs « Composant ».



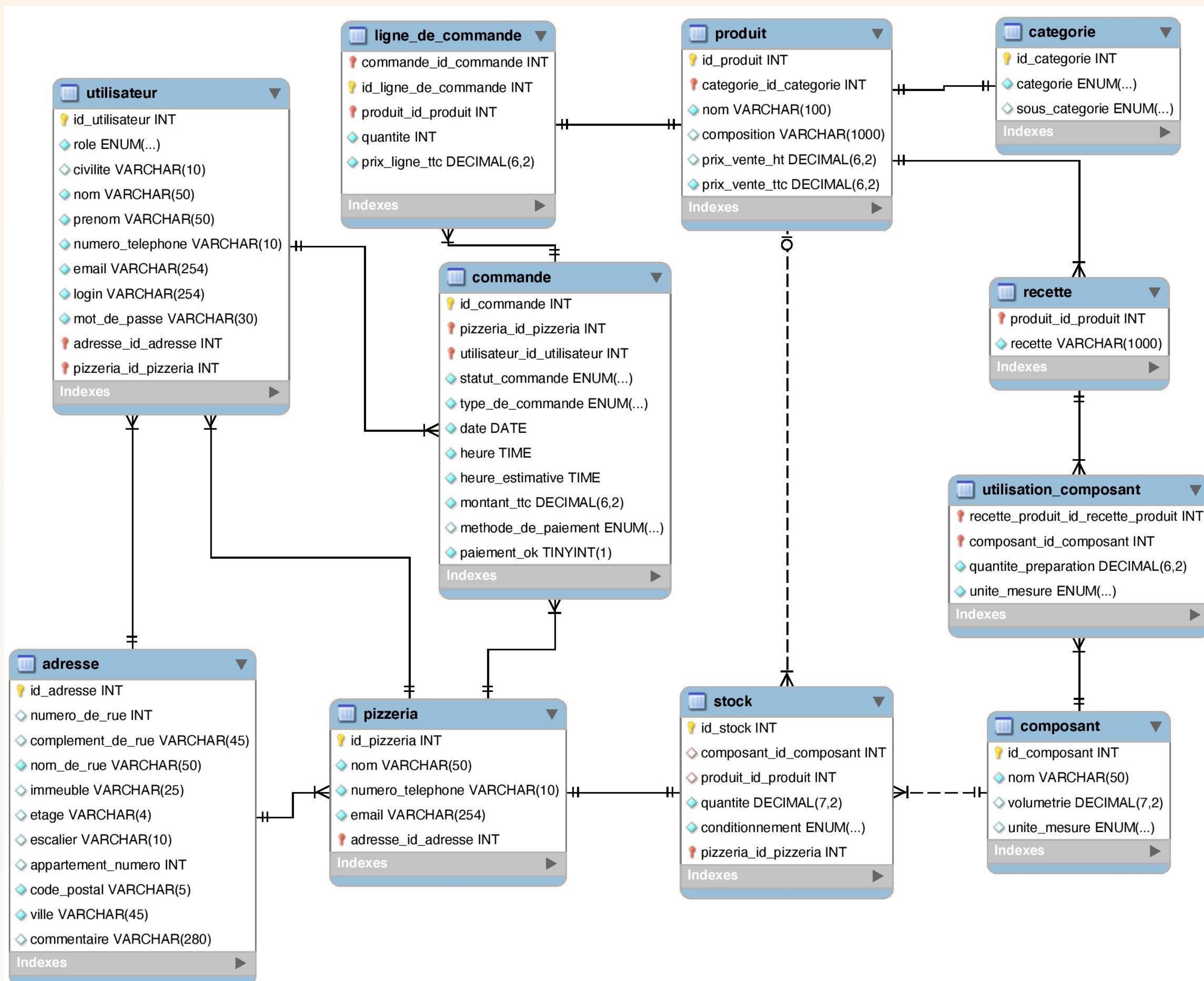
Un « Stock » contiendra à minima un ou plusieurs « Produit ».
Ainsi un « Produit » sera lié à plusieurs « Stock » de Pizzeria.



Un « Stock » appartient à une « Pizzeria ».
Donc une « Pizzeria » à un seul « Stock ».

Synthèse des multiplicités		
Multiplicité	Abréviation	Cardinalités
0..0	0	Aucune instance
0..1		Aucune ou une seule instance
1..1	1	Exactement une instance
0..*	*	Aucune, une ou plusieurs instances (aucune limite)
1..*		Au moins une instance (aucune limite maximum)
x..x	x	Exactement x instance(s)
m..n		Au moins m et au plus n instances

Le modèle physique de données



Nous avons choisi le SGBD (Système de Gestion de Base de Données) MySQL pour notre base de données, afin de la créer nous avons utilisé MySQL Workbench qui est un logiciel de modélisation de base de données.

On a donc le modèle physique de données qui est représenté à côté qui va nous permettre de générer le script de création de la base de données.

Légende diagramme

Yellow key symbol	Clé primaire (PK)
Red key symbol	Clé primaire étrangère (PKF)
Diamond symbol	Clé étrangère (FK)
Cyan diamond symbol	Clé NOT NULL (une valeur est obligatoire)
Light blue diamond symbol	Clé NULL (une valeur est optionnelle)
INT	Nombre entier
VARCHAR	Chaine de caractères
DECIMAL	Nombre non entier
ENUM	Enumération de choix
TINYINT	Booléen (vrai / faux)
DATE	Donnée de type date au format AAAA-MM-JJ
TIME	Heure au format HH:MM:SS

Projet OC Pizza : Spécifications Technique

Comme on peut le voir certaines colonnes de nos tables ont des nombres entre parenthèses, on définit la précision voulue pour nos données.

Exemple :

Ainsi un nom de ville est limité à 45 caractères.

Un numéro de téléphone sera délimité à 10 caractères.

ville VARCHAR(45)
numero_telephone VARCHAR(10)

nom	prenom	numero_telephone
Rambeau	Lola	0639983750

Pour les décimales, les nombres entre parenthèses ont une virgule.

Le premier chiffre définit le nombre total de chiffres dans un nombre et le second va définir la précision après la virgule.

Exemple :

volumétrie(7,2) va donner 10000,45

quantité(6,2) va donner 5000,20

id_stock	composant_id_compos...	produit_id_produ...	qu...	conditionneme...	pizzeria_id_pizzeria
1	1	HULL	330.00	pièce(s)	1
213	1	HULL	300.00	pièce(s)	3
2	2	HULL	267.00	pièce(s)	1

On a recours aux clés primaires, afin de définir des unicités. Plus précisément on veut que nos données n'ait pas de doublon tel la table utilisateurs.

Pour cela on va utiliser à plusieurs reprises des id qui seront auto-incrémentés afin de jouer ce rôle d'unicité.

id_utilisateur INT
id_adresse INT

Les clés étrangères quant à elles sont là pour vérifier la véracité de nos données.

Dans MySQL, elles vont agir comme des contraintes pour nous éviter de renseigner des choses incorrectes.

Exemple :

Je veux ajouter une commande à un client qui n'existe pas j'aurai alors une erreur.

```
INSERT INTO oc_pizza.commande (
    id_commande, pizzeria_id_pizzeria, utilisateur_id_utilisateur, statut_commande, type_de_commande, date, heure
) VALUES
(31, 1, 104, 'en attente', 'livraison', '2021-04-30', '18:03:27', '18:51:00', 39.6, 'carte bancaire', true);
```

```
x 2 10:21:31 INSERT INTO oc_pizza.commande ( id_commande, pizzeria_id_pizzeria, utilisateur_id_utilisateur, statut_commande, type_de... Error Code: 1452.
```

```
Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('oc_pizza`.`commande`, CONSTRAINT `fk_commande_utilisateur`
```

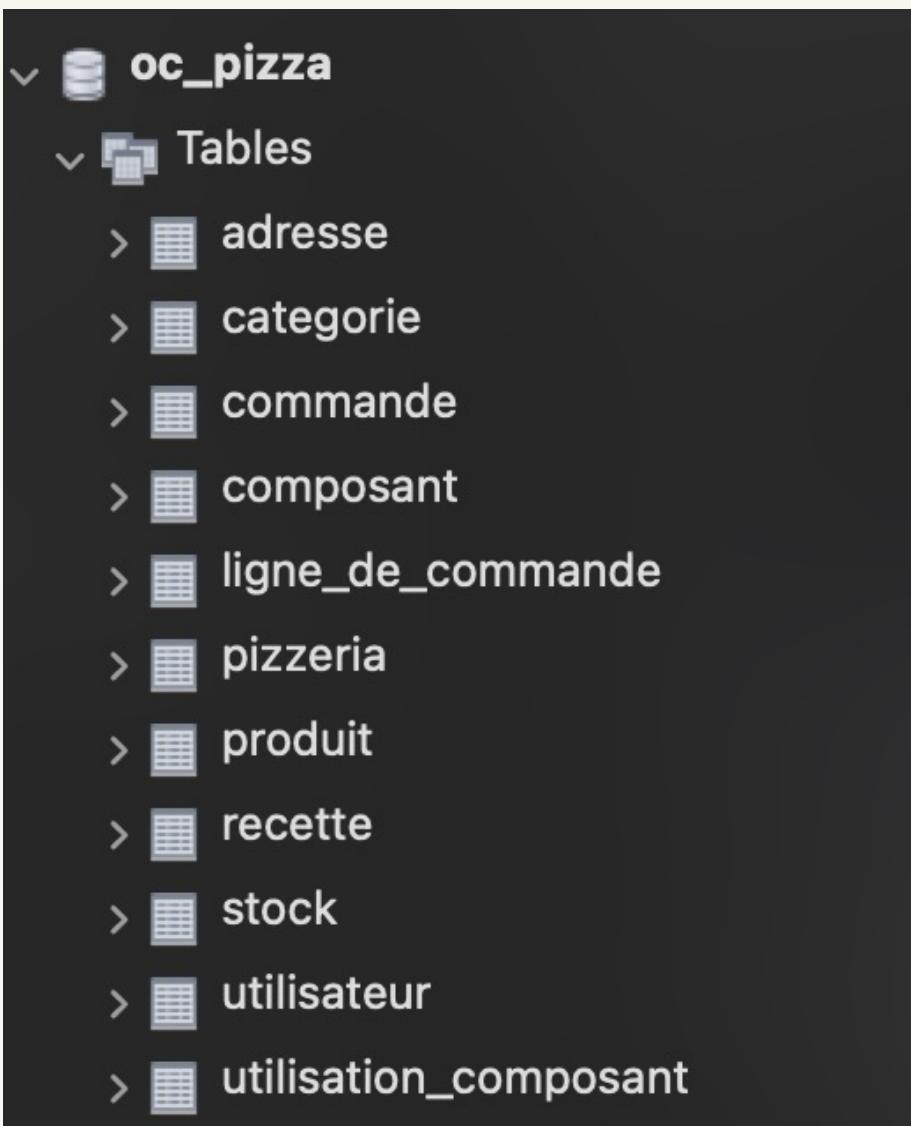
L'erreur est bien liée à l'utilisateur ici le 104 qui n'existe pas.

Projet OC Pizza : Spécifications Technique

Notre script a ensuite été testé sur MySQL Workbench mais il est compatible avec tout logiciel permettant de déployer un script MySQL sur un serveur conforme à ce SGBD tel phpMyAdmin ou DBeaver pour ne citer qu'eux.

```
1 -- MySQL Script generated by MySQL Workbench
2 -- Fri May 7 16:43:47 2021
3 -- Model: New Model Version: 1.0
4 -- MySQL Workbench Forward Engineering
5
6 • SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
7 • SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
8 • SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_
9
10 -- -----
11 -- Schema oc_pizza
12 --
13 --
14 --
15 -- Schema oc_pizza
16 --
17 • CREATE SCHEMA IF NOT EXISTS `oc_pizza` DEFAULT CHARACTER SET utf8mb4 ;
18 • USE `oc_pizza` ;
19
20 -- -----
```

Soit le résultat suivant, qui va contenir l'ensemble des tables de notre MPD.



Le logiciel n'a pas rencontré d'erreur ou alerte lors de son exécution, le script est donc parfaitement fonctionnel.

Action	Time	Output	Response	Duration / Fetch Time
1 16:44:19 DROP DATABASE `oc_pizza`			11 row(s) affected	0.204 sec
2 16:44:40 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0			0 row(s) affected	0.0019 sec
3 16:44:40 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0			0 row(s) affected	0.00032 sec
4 16:44:40 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_			0 row(s) affected	0.00025 sec
5 16:44:40 CREATE SCHEMA IF NOT EXISTS `oc_pizza` DEFAULT CHARACTER SET utf8mb4 ;			1 row(s) affected	0.0025 sec
6 16:44:40 USE `oc_pizza`			0 row(s) affected	0.0010 sec
7 16:44:40 CREATE TABLE IF NOT EXISTS `oc_pizza`.`adresse` (`id_adresse` INT ...) ENGINE=InnoDB			0 row(s) affected	0.034 sec
8 16:44:40 CREATE TABLE IF NOT EXISTS `oc_pizza`.`utilisateur` (`id_utilisateur` INT ...) ENGINE=InnoDB			0 row(s) affected	0.021 sec
9 16:44:40 CREATE TABLE IF NOT EXISTS `oc_pizza`.`pizzeria` (`id_pizzeria` INT ...) ENGINE=InnoDB			0 row(s) affected	0.0100 sec
10 16:44:40 CREATE TABLE IF NOT EXISTS `oc_pizza`.`commande` (`id_commande` INT ...) ENGINE=InnoDB			0 row(s) affected	0.014 sec
11 16:44:40 CREATE TABLE IF NOT EXISTS `oc_pizza`.`categorie` (`id_categorie` INT ...) ENGINE=InnoDB			0 row(s) affected	0.0044 sec
12 16:44:40 CREATE TABLE IF NOT EXISTS `oc_pizza`.`produit` (`id_produit` INT NOT NULL) ENGINE=InnoDB			0 row(s) affected	0.0095 sec
13 16:44:40 CREATE TABLE IF NOT EXISTS `oc_pizza`.`ligne_de_commande` (`com...` INT NOT NULL) ENGINE=InnoDB			0 row(s) affected	0.011 sec
14 16:44:40 CREATE TABLE IF NOT EXISTS `oc_pizza`.`composant` (`id_composant` INT NOT NULL) ENGINE=InnoDB			0 row(s) affected	0.0098 sec
15 16:44:40 CREATE TABLE IF NOT EXISTS `oc_pizza`.`stock` (`id_stock` INT NOT NULL) ENGINE=InnoDB			0 row(s) affected	0.011 sec
16 16:44:40 CREATE TABLE IF NOT EXISTS `oc_pizza`.`recette` (`produit_id_produi...` INT NOT NULL) ENGINE=InnoDB			0 row(s) affected	0.0058 sec
17 16:44:40 CREATE TABLE IF NOT EXISTS `oc_pizza`.`utilisation_composant` (`rec...` INT NOT NULL) ENGINE=InnoDB			0 row(s) affected	0.0077 sec
18 16:44:40 SET SQL_MODE=@OLD_SQL_MODE			0 row(s) affected	0.00014 sec
19 16:44:40 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS			0 row(s) affected	0.00024 sec
20 16:44:40 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS			0 row(s) affected	0.00061 sec
21 16:45:30 INSERT INTO `oc_pizza`.`adresse` (`id_adresse`, numero_de_rue, compleme...) VALUES (1, '123 Rue du Soleil', 'Paris')	16:45:30	73 row(s) affected Records: 73 Duplicates: 0 Warnings: 0	0.043 sec	
22 16:45:30 INSERT INTO `oc_pizza`.`pizzeria` (`id_pizzeria`, nom, numero_telephone, e...) VALUES (1, 'Pizzeria A', '01 23 45 67 89', 'pizza@example.com')	16:45:30	8 row(s) affected Records: 8 Duplicates: 0 Warnings: 0	0.0040 sec	
23 16:45:30 INSERT INTO `oc_pizza`.`utilisateur` (`id_utilisateur`, role, civilite, nom, pren...) VALUES (1, 'ROLE_ADMIN', 'M', 'John', 'Doe')	16:45:30	98 row(s) affected Records: 98 Duplicates: 0 Warnings: 0	0.0092 sec	
24 16:45:30 INSERT INTO `oc_pizza`.`categorie` (`id_categorie`, categorie, sous_categor...) VALUES (1, 'Categorie 1', 'Sous-Categorie 1')	16:45:30	14 row(s) affected Records: 14 Duplicates: 0 Warnings: 0	0.0022 sec	
25 16:45:30 INSERT INTO `oc_pizza`.`produit` (`id_produit`, categorie_id_categorie, nom...) VALUES (1, 1, 'Margherita')	16:45:30	55 row(s) affected Records: 55 Duplicates: 0 Warnings: 0	0.0069 sec	
26 16:45:30 INSERT INTO `oc_pizza`.`composant` (`id_composant`, nom, volumetrie, unit...) VALUES (1, 'Tomate', '100g', 'g')	16:45:30	67 row(s) affected Records: 67 Duplicates: 0 Warnings: 0	0.0058 sec	
27 16:45:30 INSERT INTO `oc_pizza`.`recette` (`produit_id_produit`, recette) VALUES (1, 1)	16:45:30	21 row(s) affected Records: 21 Duplicates: 0 Warnings: 0	0.0024 sec	
28 16:45:30 INSERT INTO `oc_pizza`.`utilisation_composant` (`recette_produit_id_recett...`) VALUES (1, 1)	16:45:30	130 row(s) affected Records: 130 Duplicates: 0 Warnings: 0	0.0035 sec	
29 16:45:30 INSERT INTO `oc_pizza`.`stock` (`composant_id_composant`, produit_id_produ...) VALUES (1, 1, 1)	16:45:30	848 row(s) affected Records: 848 Duplicates: 0 Warnings: 0	0.023 sec	
30 16:45:30 INSERT INTO `oc_pizza`.`commande` (`id_commande`, pizzeria_id_pizzeria,...) VALUES (1, 1)	16:45:30	30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0	0.0018 sec	
31 16:45:30 INSERT INTO `oc_pizza`.`ligne_de_commande` (`commande_id_commande`, ...) VALUES (1, 1)	16:45:30	65 row(s) affected Records: 65 Duplicates: 0 Warnings: 0	0.0021 sec	

Le script est fournit en pièce jointe pour pouvoir recréer la base de données, il suffit de l'exécuter.

script_oc_pizza.sql qui est disponible dans le repo GitHub

Les données et les requêtes

Nous aurons ici trois liens:

Celui du dump de la base avec ses données

Celui des données seule

Enfin un exemple de requêtes avec des jointures

Afin de présenter la base de données en fonctionnement, un jeu de données a été créé pour illustrer chacune des tables.

dump_oc_pizza.sql

donnees_demo_oc_pizza.sql

requetes.txt

La suite de cette partie va revenir sur le fonctionnement des requêtes et servira de base pour les livrables qui seront fournis au client.

Nous avons voir plusieurs types de requêtes :

Celles que nous appellerons simple : La table peut être lue sans avoir recours à une autre.

La table **Utilisateur** peut-être lue sans jointure

```
SELECT civilite, nom, prenom, numero_telephone, email
FROM oc_pizza.utilisateur
WHERE role = 'client';
```

De même pour notre table Pizzeria

```
SELECT * FROM pizzeria;
```

Dans nos deux exemples, nous avons des tables parfaitement lisibles.

Avec deux types de requêtes:

La première nous utilisons **SELECT** pour interroger uniquement les colonnes voulues, la seconde avec ***** pour l'ensemble des colonnes de la table.

Le **FROM** spécifie la table cible de notre requête.

Enfin le **WHERE** permet de définir un critère dans le premier cas on sélectionne uniquement les clients via la colonne **role**.

	civilite	nom	prenom	numero_telephone	email
1	NULL	Riviere	Edouard	0639988622	edouard37@example.fr
2	NULL	Bernard	Sarah	0639982745	sarah.bernard@example.fr
3	NULL	Boyer	Henry	0261912612	henry.boyer435@example.fr
4	NULL	Jacob	Nora	0639989467	norah@aomail.fr
5	Pr	Baresi	Judit	0261912634	jbaresi@aomail.fr
6	NULL	Amsel	Sophie	0639980323	amselsophie@example.fr
7	NULL	Sullivan	Jack	0261911532	sullivan.jack@example.fr

	id_pizzeria	nom	numero_telephone	email	adresse_id_adresse
1	1	OC PIZZA Tours Grammont	0261913714	tours.grammont@ocpizza.fr	1
2	2	OC PIZZA Angers	0261914910	angers@ocpizza.fr	2
3	3	OC PIZZA Nantes	0261914416	nantes@ocpizza.fr	3
4	4	OC PIZZA Rennes	0261913504	rennes@ocpizza.fr	4
5	5	OC PIZZA Tours Maginot	0261913715	tours.matinot@ocpizza.fr	5
6	6	OC PIZZA Paris 14	0199000114	paris14@ocpizza.fr	6
7	7	OC PIZZA Le Mans	0261917230	lemans@ocpizza.fr	7
8	8	OC PIZZA La Rochelle	0536491704	larochelle@ocpizza.fr	8

Cependant si nous voulons des informations plus complètes dans les 2 cas nous auront besoin des jointures **JOIN**.

Ici nous avons besoin de connaitre les adresses des pizzerias et des utilisateurs.

Les jointures nous permettent des résultats plus complets dans notre base de données relationnelle par une fusion de contenu de plusieurs tables.

Si nous reprenons nos deux exemples précédents nous voulons fusionner les tables :

Utilisateur et **Adresse** mais aussi les tables **Pizzeria** et **Adresse**.

Les requêtes seront donc similaires. Nous allons détailler uniquement la jointure de table **Utilisateur** et **Adresse** mais le principe est le même pour toutes les tables.

```
SELECT utilisateur.civilité AS 'Civilité',
utilisateur.nom AS 'Nom',
utilisateur.prenom AS 'Prénom',
utilisateur.numero_telephone 'Numéro de téléphone',
utilisateur.email AS 'Mail',
adresse.numero_de_rue AS 'Numéro de rue',
adresse.nom_de_rue AS 'Nom de rue',
adresse.complement_de_rue AS 'Complément de rue',
adresse.immeuble AS 'Nom immeuble',
adresse.etage AS 'Etage',
adresse.appartement_numero AS 'Numéro appartement',
adresse.commentaire AS 'Commentaire',
adresse.code_postal AS 'Code Postal',
adresse.ville AS 'Ville',
pizzeria.nom AS 'Pizzeria du client'
FROM oc_pizza.utilisateur
INNER JOIN adresse ON adresse.id_adresse = utilisateur.adresse_id_adresse
INNER JOIN pizzeria ON pizzeria.id_pizzeria = utilisateur.pizzeria_id_pizzeria
WHERE role = 'client';
```

Dans notre **SELECT** on spécifie les colonnes des tables dont on va avoir besoin, celles à afficher.

Le **AS** permet de définir un alias c'est à dire de renommer la colonne dans un nom plus adapté.

Le **FROM** réfère toujours à la table qui contient les données.

Enfin le **INNER JOIN** qui peut-être abrégé en **JOIN** nous permet la fusion des tables avec notre table **Utilisateur**.

On sélectionne ici les colonnes voulues des 3 tables **Utilisateur**, **Adresse** et **Pizzeria**.

Projet OC Pizza : Spécifications Technique

Soit le résultat suivant qui est donc la présentation complète des données issues de nos 3 tables.

On a donc les informations sur le client, son adresse et sa pizzeria de rattachement.

	Civilité	Nom	Prénom	Numéro de téléphone	Mail	Numéro de rue	Nom de rue	Complément de rue	Nom immeuble	Etage	Numéro appartement	Commentaire	Code Postal	Ville	Pizzeria du client
1	NULL	Riviere	Edouard	0639988622	edouard37@example.fr	3	Rue Delahaye	Résidence Soleil	C	3	34	Digicode 8104	37000	Tours	OC PIZZA Tours Grammont
2	NULL	Bernard	Sarah	0639982745	sarah.bernard@example.fr	3	Rue Delahaye	Résidence Soleil	B	4	32	Digicode 8104	37000	Tours	OC PIZZA Tours Grammont
3	NULL	Boyer	Henry	0261912612	henry.boyer435@example.fr	57	Place du Grand Marché	NULL	NULL	NULL	NULL	NULL	37000	Tours	OC PIZZA Tours Grammont
4	NULL	Jacob	Nora	0639989467	norah@aomail.fr	43	Avenue de Pont-Cher	NULL	NULL	NULL	NULL	NULL	37200	Tours	OC PIZZA Tours Grammont
5	Pr	Baresi	Judit	0261912634	jbaresi@aomail.fr	16	Rue Emile Aron	NULL	NULL	NULL	NULL	NULL	37200	Tours	OC PIZZA Tours Grammont
6	NULL	Amself	Sophie	0639980323	amselsophie@example.fr	3	Allée Guy Charff	Résidence Technopole A	2	NULL	Digicode 1222	37200	Tours	OC PIZZA Tours Grammont	
7	NULL	Sullivan	Jack	0261911532	sullivan.jack@example.fr	31	Allée Ferdinand de Lesseps	NULL	NULL	NULL	NULL	37200	Tours	OC PIZZA Tours Grammont	
8	NULL	Gary	Tony	0639982467	tony22@yaxemple.com	36	Rue Descartes	NULL	NULL	NULL	NULL	37000	Tours	OC PIZZA Tours Grammont	
9	Dr	Castillo	Adrian	0639981745	adrian765@example.fr	17	Rue Colbert	NULL	NULL	NULL	NULL	37000	Tours	OC PIZZA Tours Grammont	
10	NULL	Lacroix	Kevin	0639989273	kevin.lacroix@aomail.fr	8	Rue de la lamproie	NULL	NULL	NULL	NULL	37000	Tours	OC PIZZA Tours Grammont	
11	Mme	Chiang	Assia	0261910909	assiachi@yaexemple.com	67	Rue de l ancienne école normale	NULL	NULL	NULL	NULL	37000	Tours	OC PIZZA Tours Grammont	
12	NULL	Ghadir	Leila	0639982882	ileo@dotmail.com	2	Allée de Beauséjour	bis	NULL	NULL	NULL	37000	Tours	OC PIZZA Tours Grammont	

Certaines requêtes sont plus spécifiques comme pour la jointure des tables Stock + Produit + Composant + Pizzeria où nous allons avoir besoin d'un OUTER JOIN.

En effet nous avons des tables qui peuvent être NULL et une requête JOIN ne donnerait alors aucun résultat.

Ici nous feront 2 LEFT OUTER JOIN, il permet de retourner tous les enregistrements de la table de gauche même s'il n'y a pas de correspondance avec la table de droite.

```
SELECT stock.pizzeria_id_pizzeria AS 'Numéro Pizzeria',
pizzeria.nom AS 'Nom de la pizzeria',
produit.nom AS 'Nom du produit',
composant.nom AS 'Nom du composant',
stock.quantite AS 'Quantité',
stock.conditionnement AS 'Conditionnement'
FROM oc_pizza.stock
JOIN pizzeria ON pizzeria.id_pizzeria = stock.pizzeria_id_pizzeria
LEFT OUTER JOIN composant ON composant.id_composant = stock.composant_id_composant
LEFT OUTER JOIN produit ON produit.id_produit = stock.produit_id_produit;
```

	Numéro Pizzeria	Nom de la pizzeria	Nom du produit	Nom du composant	Quantité	Conditionnement
62	1	OC PIZZA Tours Grammont	NULL	emmental	5	pièce(s)
63	1	OC PIZZA Tours Grammont	NULL	demi boite à pizza	230	pièce(s)
64	1	OC PIZZA Tours Grammont	NULL	récipient carton friture	230	pièce(s)
65	1	OC PIZZA Tours Grammont	NULL	filet saumon fumé	6	pièce(s)
66	1	OC PIZZA Tours Grammont	NULL	farine	3	sachet(s)
67	1	OC PIZZA Tours Grammont	NULL	levure	10	pot(s)
68	1	OC PIZZA Tours Grammont	Mars glacé	NULL	30	pièce(s)
69	1	OC PIZZA Tours Grammont	Snickers glacé	NULL	30	pièce(s)
70	1	OC PIZZA Tours Grammont	Bounty glacé	NULL	30	pièce(s)

Une manière plus commune de mémoriser ces requêtes serait de passer par des Vues VIEW, pour cela il suffit de rajouter une ligne de commande avant notre requête SELECT comme celle-ci :

```
CREATE OR REPLACE VIEW stock_global
```

Et ajouter le AS devant notre requête

AS SELECT

Il nous devient alors possible d'interroger notre VIEW comme pour une requête normale.

```
SELECT * FROM stock_global WHERE `Numéro Pizzeria` = 1;
```

Projet OC Pizza : Spécifications Technique

Dans le document en annexe nous donnons les requêtes les différentes tables de jointures que nous avons pu identifier pour le client.

A titre d'informations les tables de jointures seront les suivantes :

Consultation des informations Clients	Utilisateur + Adresse + Pizzeria
Consultation des informations Employés	Utilisateur + Adresse + Pizzeria
Consultation des informations pizzerias	Pizzeria + Adresse
Consultation du stock d'une pizzeria	Stock + Produit + Composant + Pizzeria
Listing des commandes	Commande + Utilisateur + Pizzeria
Détail d'une commande	LigneDeCommande + Commande + Produit + Categorie + Utilisateur
Recette et ses ingrédients	Recette + Produit + UtilisationComposant + Composant
Produit affiché comme un Menu	Produit + Categorie

Enfin nous avons des exemples de changements ou suppressions de valeurs pour montrer que la base est opérationnelle.

```
UPDATE utilisateur
SET mot_de_passe = 'i67xMbF6P'
WHERE id_utilisateur = 3;
```

Mise à jour du mot de passe

```
UPDATE liste_clients
SET Mail = 'kevin.lacroix@example.fr'
WHERE Mail = 'kevin.lacroix@aomail.fr';
```

Mise à jour d'une adresse mail

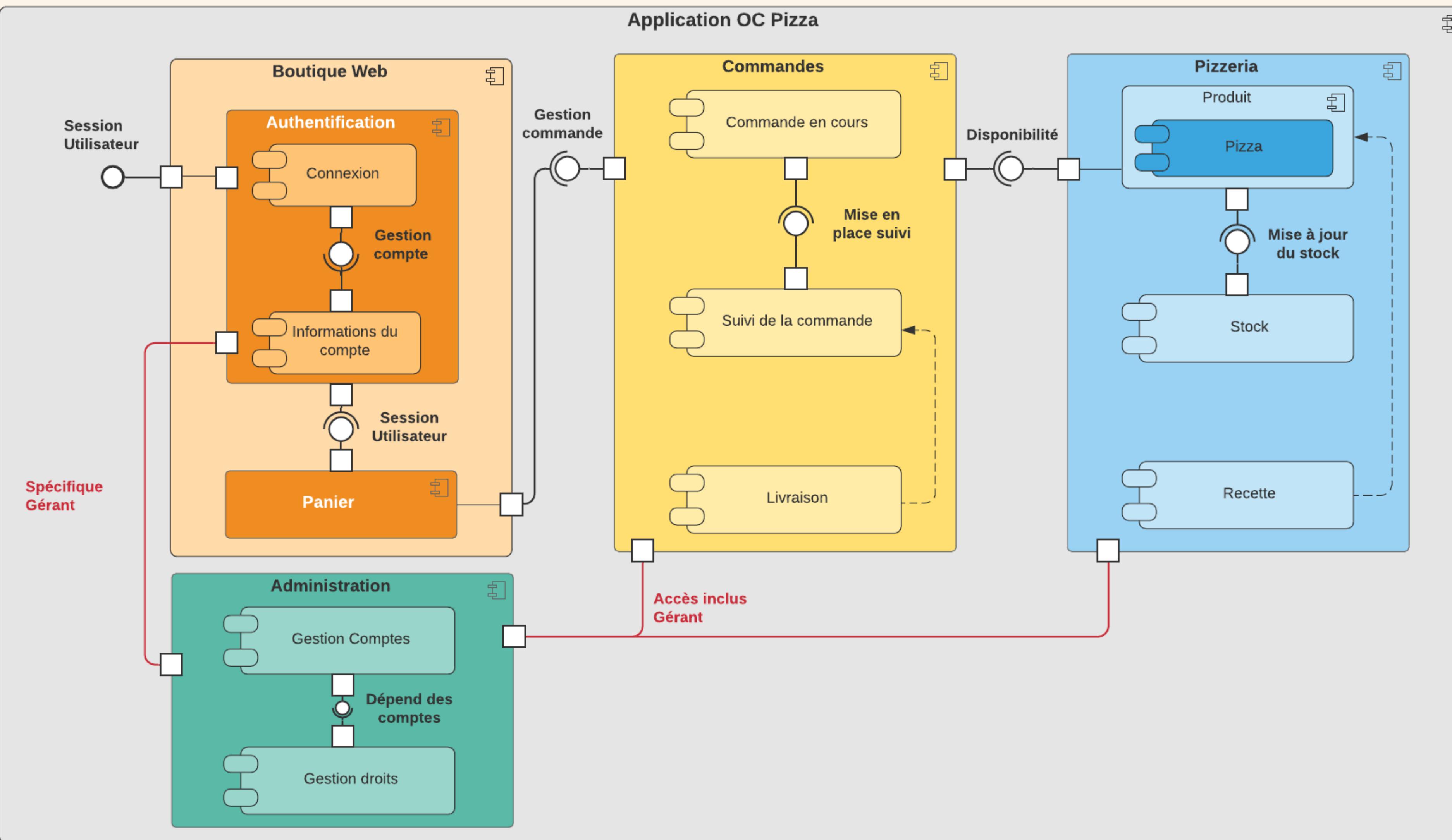
```
UPDATE adresse
SET commentaire = NULL
WHERE id_adresse =
SELECT adresse_id_adresse
FROM utilisateur
WHERE utilisateur.email = 'amselsophie@example.fr'
);
```

Mise à jour d'une adresse en se référant au mail de l'utilisateur

```
DELETE FROM utilisateur
WHERE id_utilisateur = 5;
```

Suppression d'un utilisateur

A titre d'informations, les mises à jour sont uniquement possibles sur les tables dites UPDATABLE cela exclut les tables avec des OUTER JOIN ou des DISTINCT.

Diagramme de composants

Le diagramme de composant nous permet d'illustrer la relation entre les différents composants du système.

Dans un premier lieu on va illustrer les composants internes avant de modéliser les composants externes comme la base de données.

Nous retrouvons donc les composants **Boutique Web**, **Commandes** et **Pizzeria**, à cela s'ajoute le composant **Administration** qui est spécifique à l'Administrateur ou au Gérant.

On a deux types d'accès celui du client et celui du personnel de la Pizzeria.

Avant de pouvoir commander le client doit se connecter.

Son **Panier** va être lié automatiquement au composant **Commandes** qui lui-même va nécessiter une interface de **Pizzeria** afin de savoir ce qui est disponible.

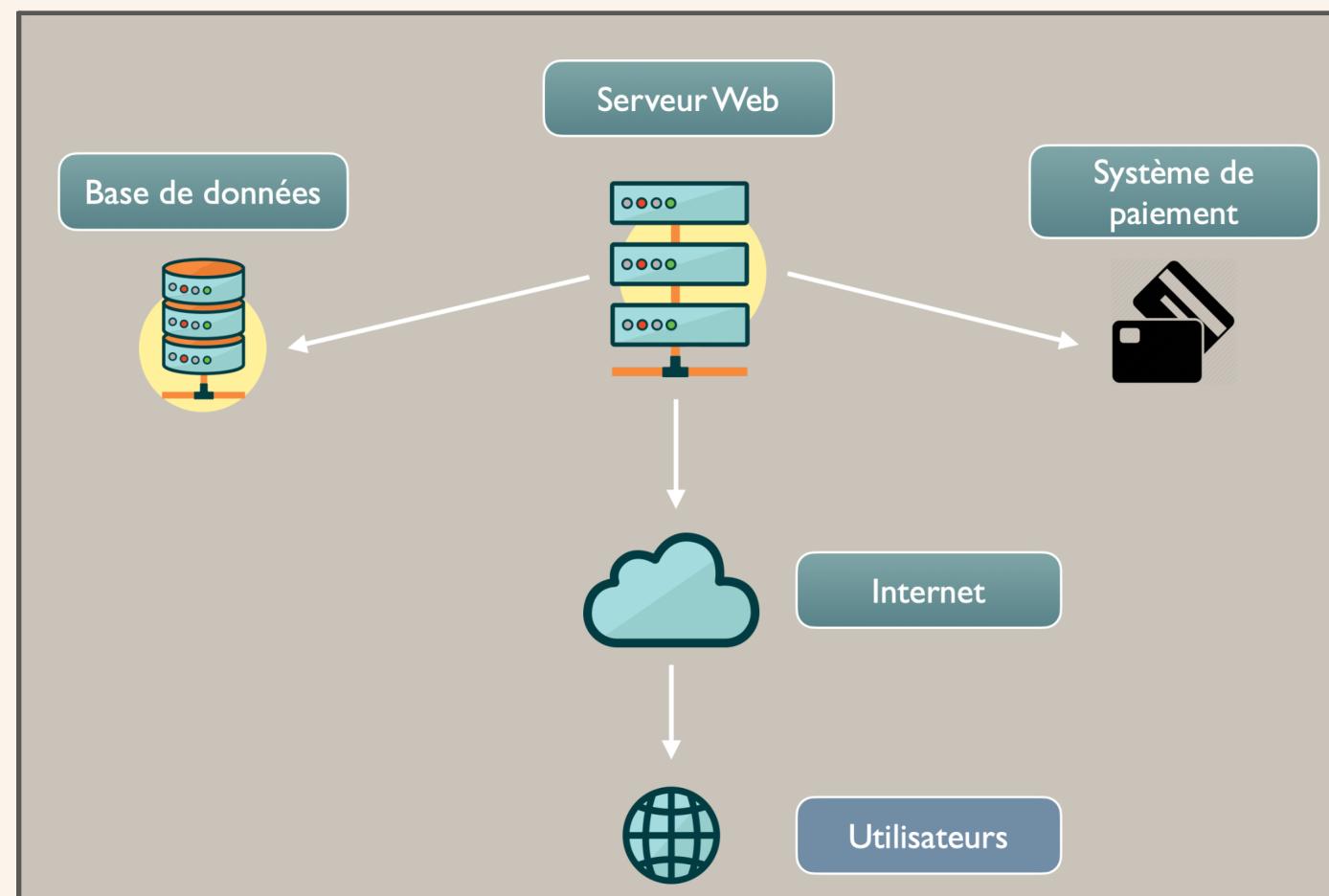
Dès qu'un **Panier** est validé il devient une **Commande en cours** et le processus de **Suivi** s'active.

On peut aussi constater des dépendances comme la **Livraison** qui va dépendre du **Suivi** ou la **Recette** qui dépend du **Produit**.

Projet OC Pizza : Spécifications Technique

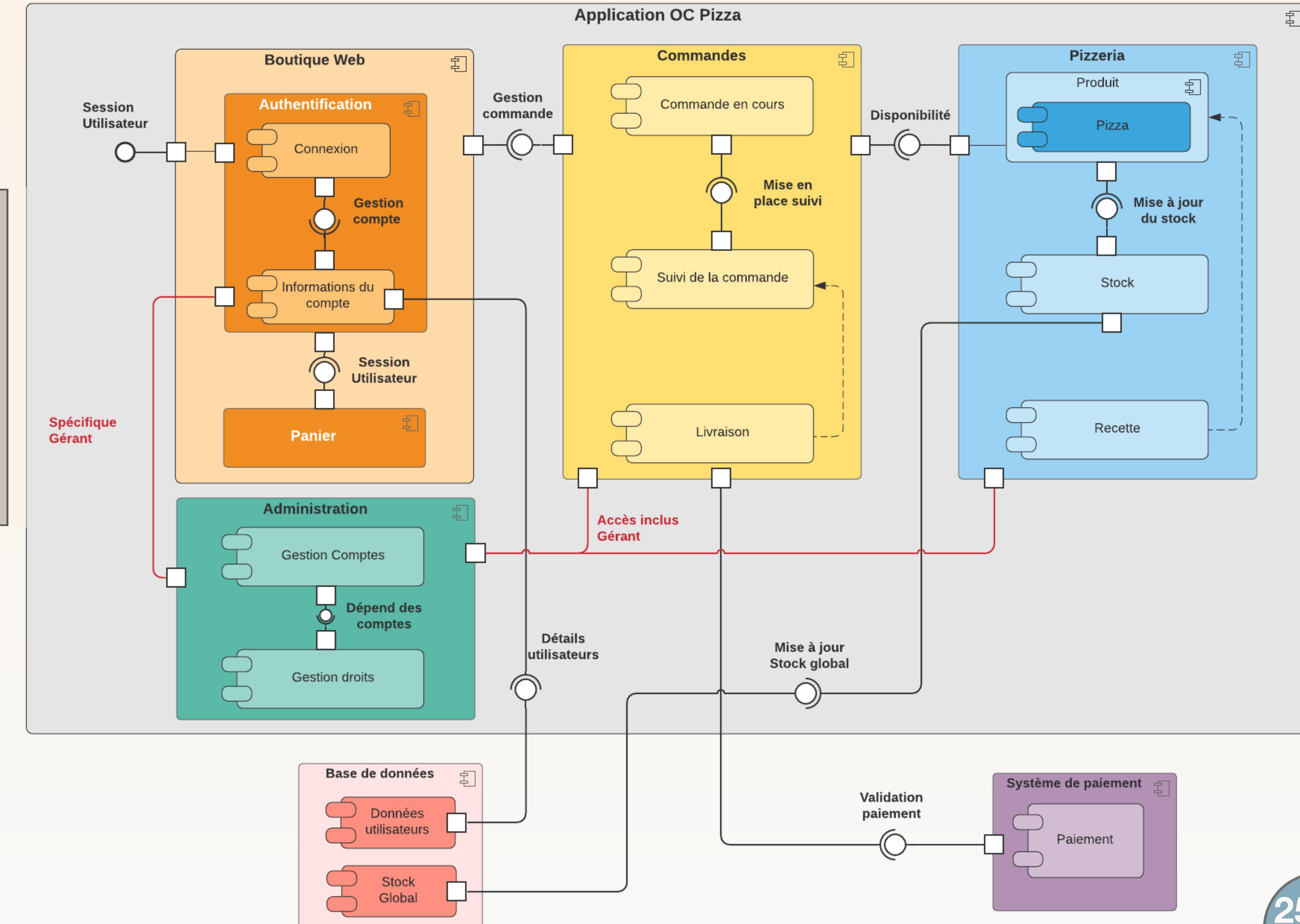
La représentation avec les composants externes sera donc la suivante.

Le schéma ci-dessous représente la vue simplifiée évoquée lors de la représentation du domaine fonctionnel .



L'accès à la base de données sera primordial car elle va contenir notamment les données utilisateurs et le stock de l'ensemble des magasins.

Le système de paiement lui est lié à la commande et va valider son paiement.



11

Diagramme de déploiement

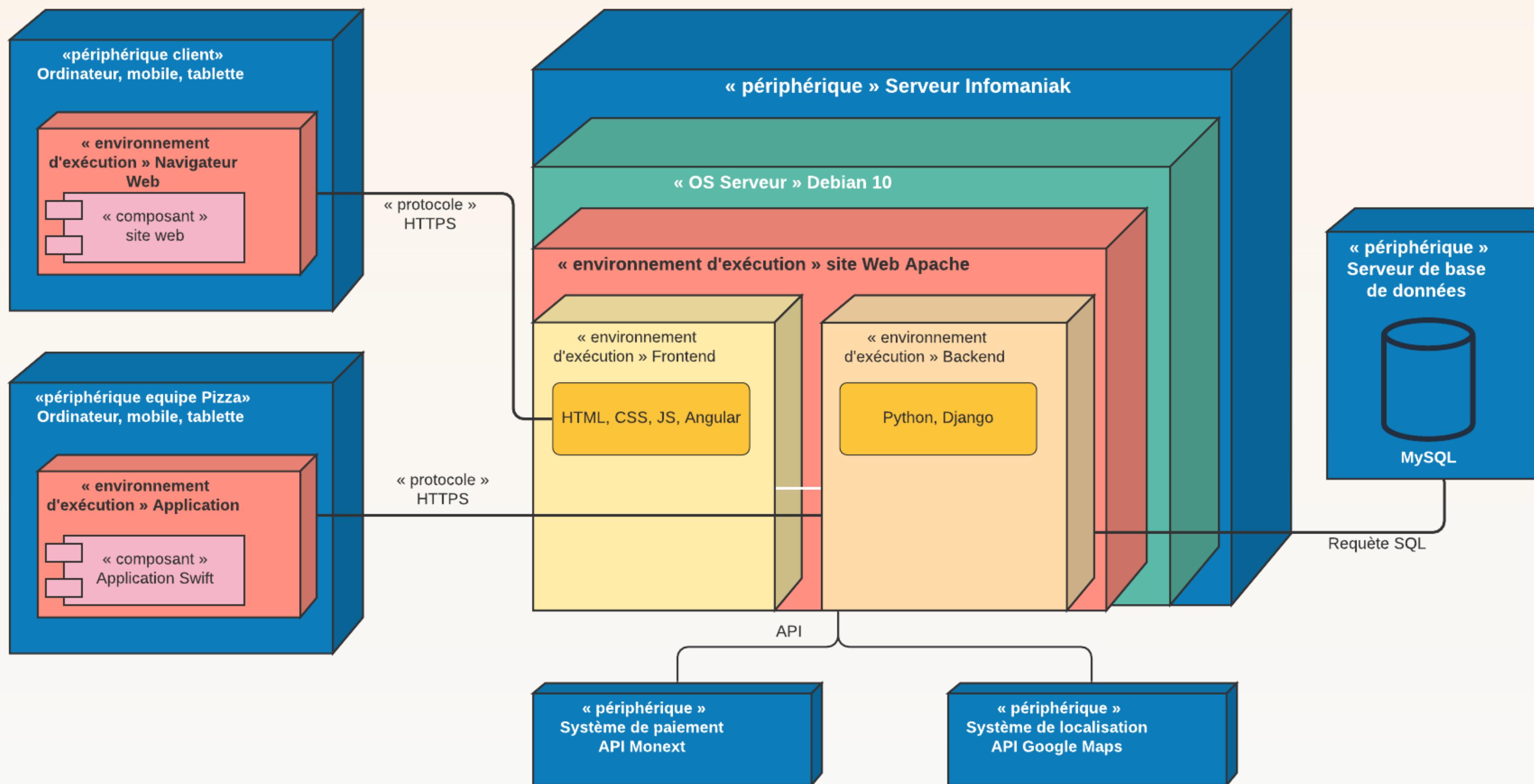
Le diagramme de déploiement décrit les infrastructures physiques, la manière dont les composants sont répartis. Il représente aussi les relations des différents composants.

Notre système est déployé sur un serveur Infomaniak dont l'OS sera un Debian 10.

Le site web sera propulsé par un serveur Apache dans sa dernière version stable.

Le Frontend du serveur utilise du HTML, du CSS du Javascript et sera encadré par un framework Angular.

Pour le Backend nous serons sur du Python et le framework Django. La communication entre le Frontend et Backend est représenté par le lien blanc.



Les utilisateurs accèdent à la partie Frontend depuis leur navigateur Web via le protocole HTTPS.

L'application mobile sera codée en Swift et sera connectée au Backend.

La base de données communique avec le serveur d'application côté Backend uniquement.

Enfin notre système va avoir besoin de l'API Monext pour la connexion au système de paiement et de l'API Google Maps pour la partie Guidage de nos livreurs.