|  |
| --- |
| OC PIZZA  NOUVEAU SYSTEME INFORMATIQUE  Dossier de conception technique  Version 1.0 |
| Auteur  Anthony TUFFERY  Analyste Programmeur |

Table des matières

1 - Versions 3

2 - Introduction 4

2.1 - Objet du document 4

2.2 - Références 4

3 - Architecture Technique 5

3.1 - Composants généraux 5

3.2 - Composants Internes 6

3.2.1 - Composant Boutique en ligne 6

3.2.2 - Composant Commande 7

3.2.3 - Composant Pizzeria 7

3.3 - Composants Externes 9

3.3.1 – La base de données (BDD) 9

3.3.2 – Le système de paiement 9

3.4 - Application Web et Mobile 9

3.4.1 – Back End 9

3.4.2 – Front End 9

3.4.3 - Mobile 9

3.5 - Base de données (BDD) 10

4 - Architecture de déploiement 16

4.1 – Serveur de base de données 16

4.2 - Serveur Web 17

5 - Architecture Logicielle 18

5.1 – Principes Généraux 18

5.1.1 – Les couches 18

5.1.2 – Structures des sources 18

6 – Point particuliers 19

6.1 – Gestion des logs 19

6.2 – Ressources 19

6.2.1 – Charte graphique 19

6.2.2 - Données 19

6.3 – Environnement de développement 19

6.3.1 - Documentation 19

6.3.2 - Environnement 19

6.4 – Procédure de packaging/ livraison 20

7- Glossaire 21

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| A. TUFFERY | 16/06/2021 | Création du document | 1.0 |

# Introduction

## Objet du document

Le présent document constitue le dossier de conception technique de l'application XXX…

Objectif du document…

Les éléments du présent dossier découlent :

* Du recueil des besoins et du cahier des charges donné par le client
* De la présentation de la solution technique avec le chef de projet
* Du dossier fonctionnel

## Références

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

1. DCF – 1.0 : Dossier de conception fonctionnelle de l'application
2. CE – 1.0 : Dossier d’exploitation de l’application

# Architecture Technique

## Composants généraux

Le diagramme de composants illustre la relation entre les différents composants du système.

Nous retrouvons 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 (Front Office) et celui du personnel de la Pizzeria

(Back Office).

Avant de pouvoir commander le client doit se connecter.

Son panier sera lié à la composante « Commande » qui elle-même sera liée à la composante « Pizzeria » afin d’obtenir la disponibilité des produits.

Un panier validé par le client devient alors une commande en cours et active le suivi de commande. On peut aussi constater des dépendances comme la Livraison qui va dépendre du Suivi ou la Recette qui dépend du Produit.



Diagramme de composant UML

## Composants Internes

### 3.2.1 - Composant Boutique en ligne

On regroupe les fonctionnalités nécessaires pour la connexion au compte et à la prise d’une commande.

3.2.1.1 - Sous composant Authentification

Il permet l’Authentification de l’utilisateur. Il va permettre la l’inscription ou la connexion d’un utilisateur et va contenir les informations inhérentes au compte (Adresse, Numéro de téléphone, adresse email etc.)   
Par défaut lors de la création d’un compte, le rôle est défini sur client.

3.2.1.2 - Sous-Composant Panier

Il permet la création d’une commande par l’utilisateur.  
Les options seront plus ou moins poussées selon le rôle de l’utilisateur.

3.2.1.3 - Sous-Composant Point de vente

Lors de son arrivé sur le site, le client sélectionne le point de vente s’il ne l’avait pas attribué lors de la création de son compte (ou qu’il souhaite en sélectionner un autre) et le type de commande (à emporter, livraison).

### 3.2.2 - Composant Commande

Regroupe les fonctionnalités nécessaires au suivi des commandes pour un client ou une pizzeria.

3.2.2.1 - Sous-Composant Commande en cours

Il permet d’afficher une commande et son détail avec les lignes de commandes et le montant de la commande. Il va donc par conséquent permettre le suivi des commandes actuelles.

3.2.2.2 - Sous-Composant Statut de la commande

Il permet de donner les informations liées à une commande et de l’avancement de son état, il dépend de la commande en cours.

L’avancement du statut va permettre d’annuler ou modifier une commande quand cela est encore possible.

3.2.2.3 - Sous-Composant Historique des commandes

Il va contenir l’ensemble des commandes.  
Selon le cas, il peut s’agir de la Pizzeria, du client ou du livreur.

3.2.2.4 - Sous-Composant Livraison

Il permet la gestion des informations liées à la livraison et renvoie les informations à l’artefact suivi de la commande.

### 3.2.3 - Composant Pizzeria

Ce composant va contenir les informations d’un point de vente (PDV) et donc les produits disponibles à l’instant T dans celui-ci.

Il apporte également des informations supplémentaires dans la partie cuisine.

3.2.3.1 - Sous-Composant Produit

Il permet l’affichage des produits proposé par un point de vente.   
Il contient les informations détaillées des différents produits ainsi que leur prix de vente.  
Il est relié au stock afin d’afficher la disponibilité des produits.

3.2.3.2 - Sous-Composant Stock

Il permet la mise à jour en temps réel des composants qui sont encore disponibles afin d’informer sur les produits que peut proposer un PDV.

3.2.3.3 - Sous-Composant Recette

Il permet l’affichage d’une recette et des composants pour le Pizzaiolo.  
Il est relié aux produits afin d’afficher la recette correspondante ainsi que la quantité nécessaire à la création de celui-ci.

3.2.4 - Composant *Administration*

Ce package est spécifique au gérant.   
Il permet des manipulations avancées sur les comptes et un visuel global du groupe.

3.2.4.1 - Sous-Composant Gestion du Groupe

Ce sous-composant permet au gérant d’avoir une vue en temps réel sur l’activité du groupe (commande stock livraison).

3.2.4.2 - Sous-Composant Gestion des comptes

Il va permettre de créer des comptes utilisateurs pour la pizzeria ou le cas échéant de faires des modifications.

3.2.4.3 - Sous-Composant Gestion des droits

Comme indiqué plus haut, un utilisateur dispose par défaut du droit client par mesure de sécurité.   
Le gérant peut affecter un niveau de droit différent pour un utilisateur de manière à lui donner un accès plus spécifique (livreur, pizzaiolo...)

## Composants Externes

Notre application nécessite deux composant externe pour fonctionner

### 3.3.1 – La base de données (BDD)

Elle va contenir les informations nécessaires à la partie Authentification, Commande et Pizzeria  
Chaque composant a recours à la BDD pour fournir les informations nécessaires que ce soit pour vérifier les informations d’un compte utilisateur, d’afficher le détail d’une commande ou d’informer sur les produits disponibles ainsi que leur disponibilité.

### 3.3.2 – Le système de paiement

Le paiement d’une commande passera par le système de paiement qui va enregistrer le règlement de la commande

## 3.4 - Application Web et Mobile

La pile logicielle est la suivante :

• Debian « Buster » version 10.9 ou ultérieure OS Serveur

• NginX version 1.21.1 ou ultérieure Serveur Web

• Python version 3.9.5 ou ultérieure Back End

• Django version 3.2.3 ou ultérieure Back End

• HTML version 5.3 ou ultérieure Front End

• CSS version 4.15 ou ultérieure Front End

• JavaScript version ECMAScript2020 ou ultérieure Front End

• PostgreSQL version 9.6 ou ultérieure Base de données

• Swift version 5.4 ou ultérieure Application iOS

### 3.4.1 – Back End

Le Back-End est la partie invisible du site en charge du coté logique di site   
Celui-ci sera codé en Python avec un Framework Django   
Il sera en lien direct avec notre BDD.

### 3.4.2 – Front End

Le Front End est la partie visible du site web (notre interface client et équipe pizzeria).  
Pour ce faire, nous utiliserons les standards que sont HTML, CSS et JavaScript.

### 3.4.3 - Mobile

L’application mobile étant sous iOS, nous utiliseront le langage Swift pour la développer.

## 3.5 - Base de données (BDD)

3.5.1 – SGBD

Notre choix pour le système de gestion de base de données s’est tourné sur PostgreSQL.

3.5.2 – Modèle Physique de Données (MPD)

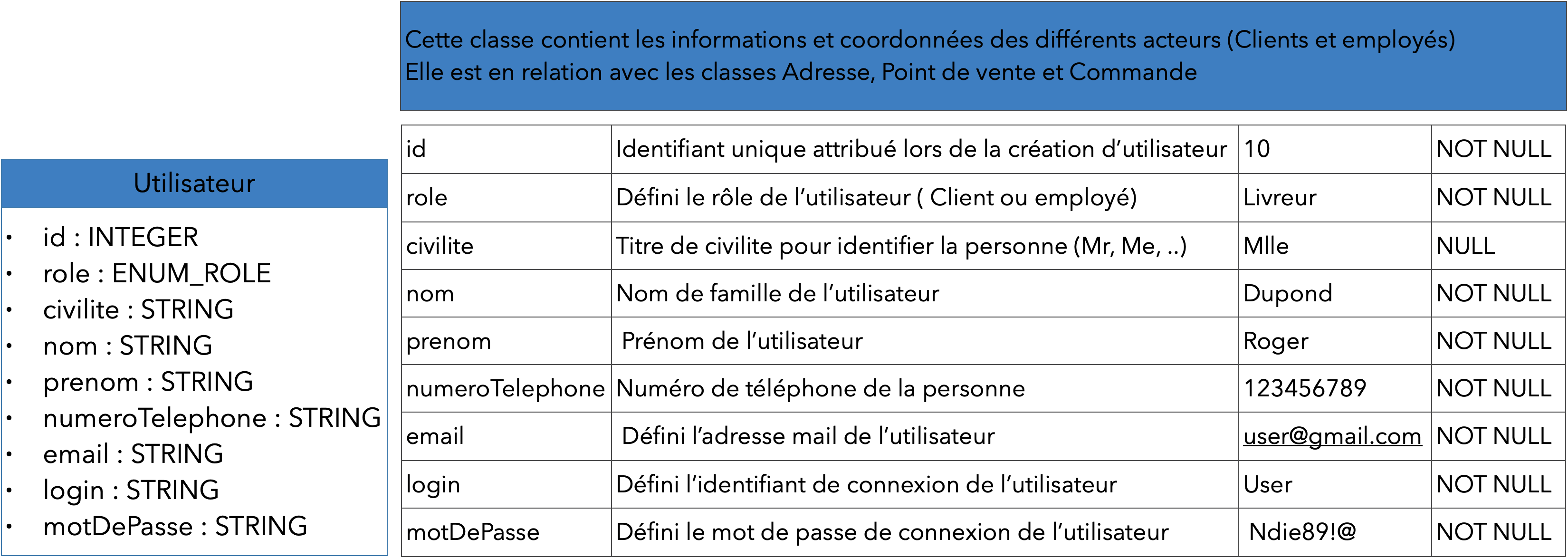
Le MPD représente notre base de données telle qu’elle a été conçue en se basant sur notre diagramme de classe visible dans la documentation fonctionnelle.  
Le modèle va représenter les relations entre les différentes tables.

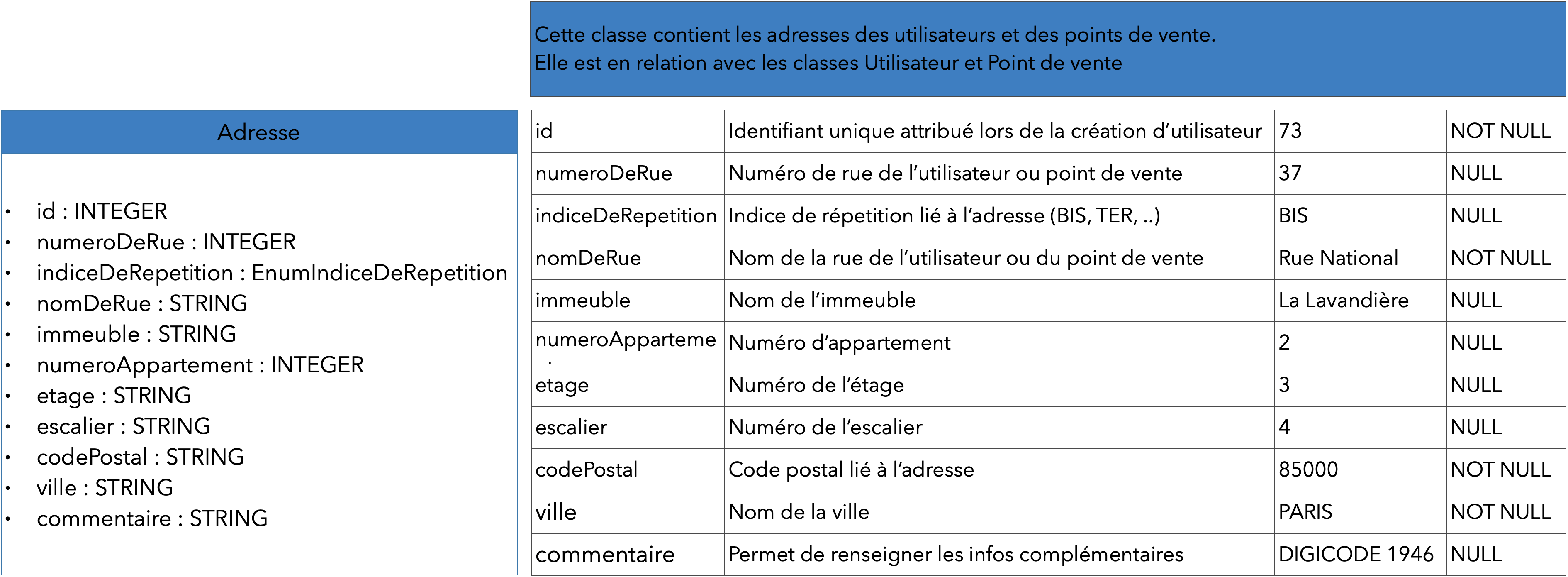


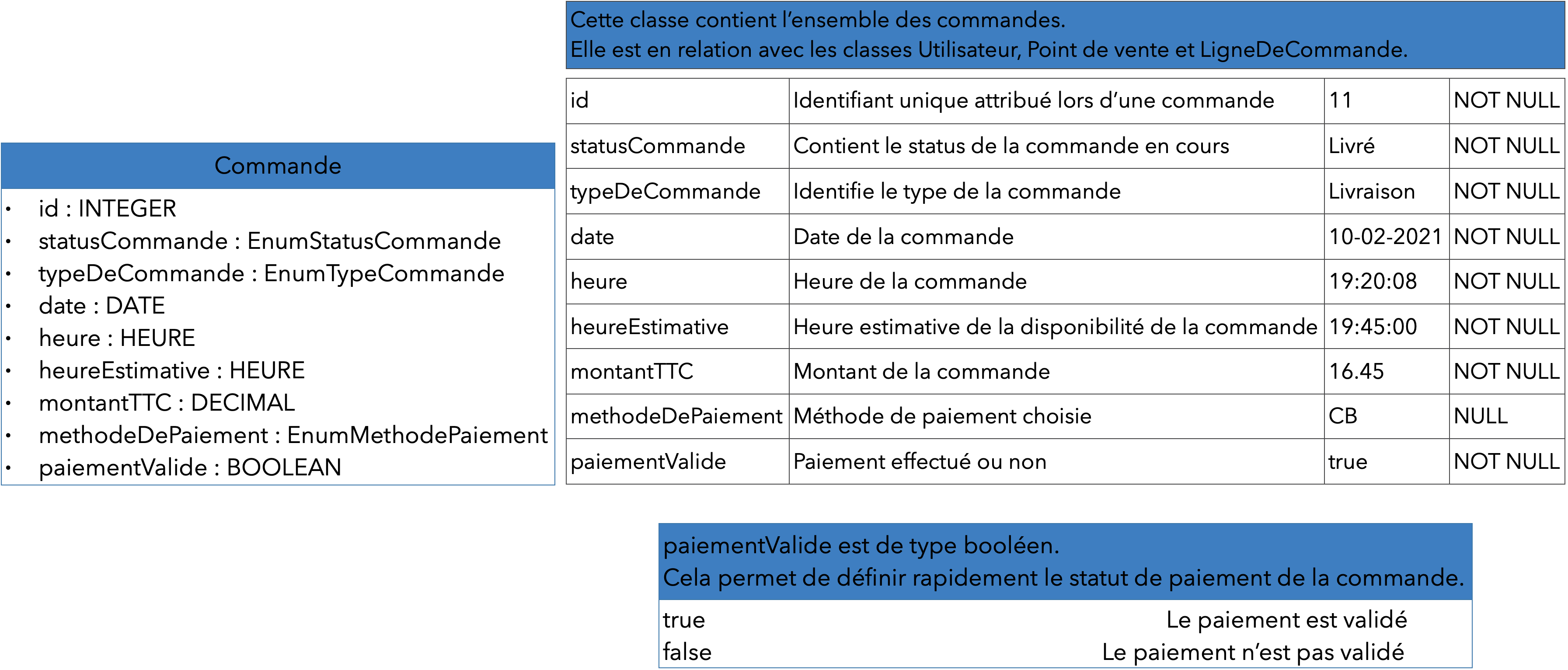
3.5.3 - Détail des tables

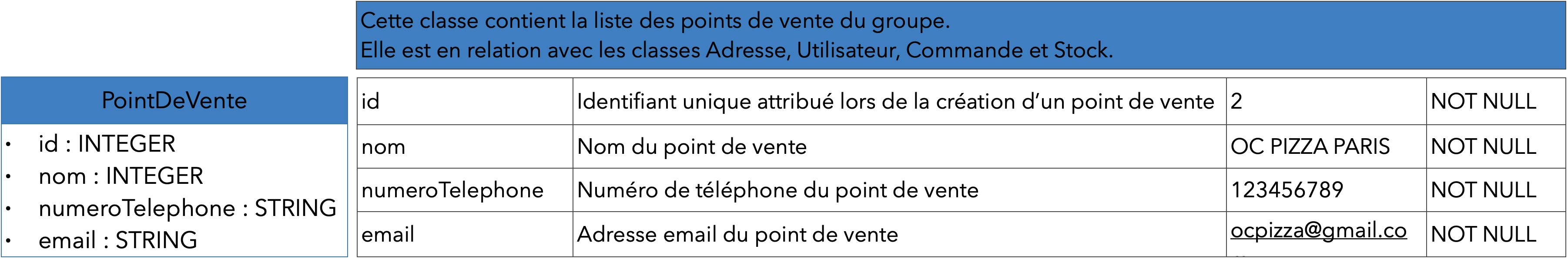
Les caractéristiques des différentes tables sont rappelées ci-dessous, nous retrouvons la table cible avec la précision des attributs.

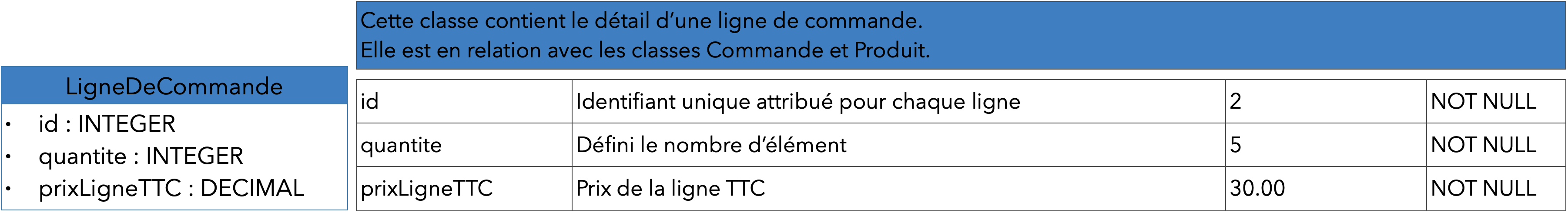
L’encart de droite vient expliquer le rôle de chaque attribut préciser si la valeur peut être NULL et donner un exemple d’utilisation.

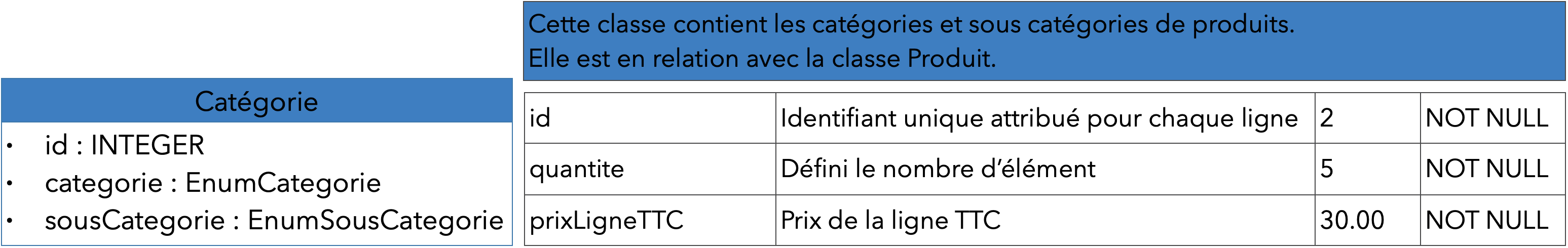


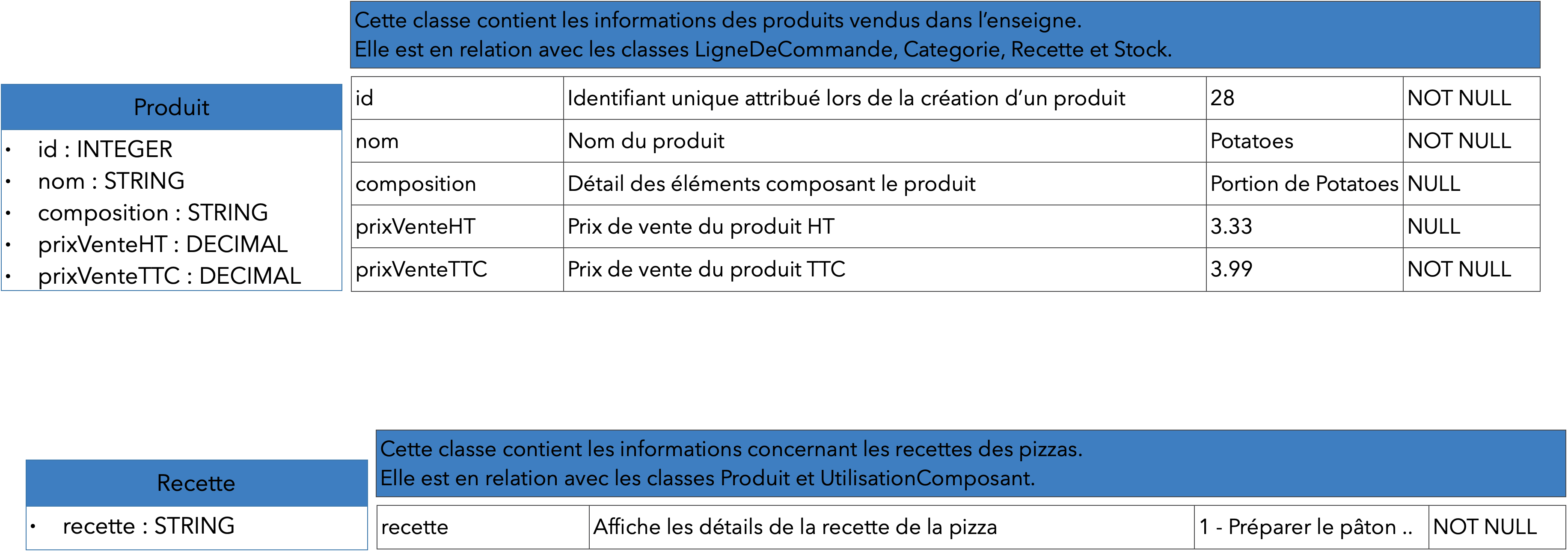


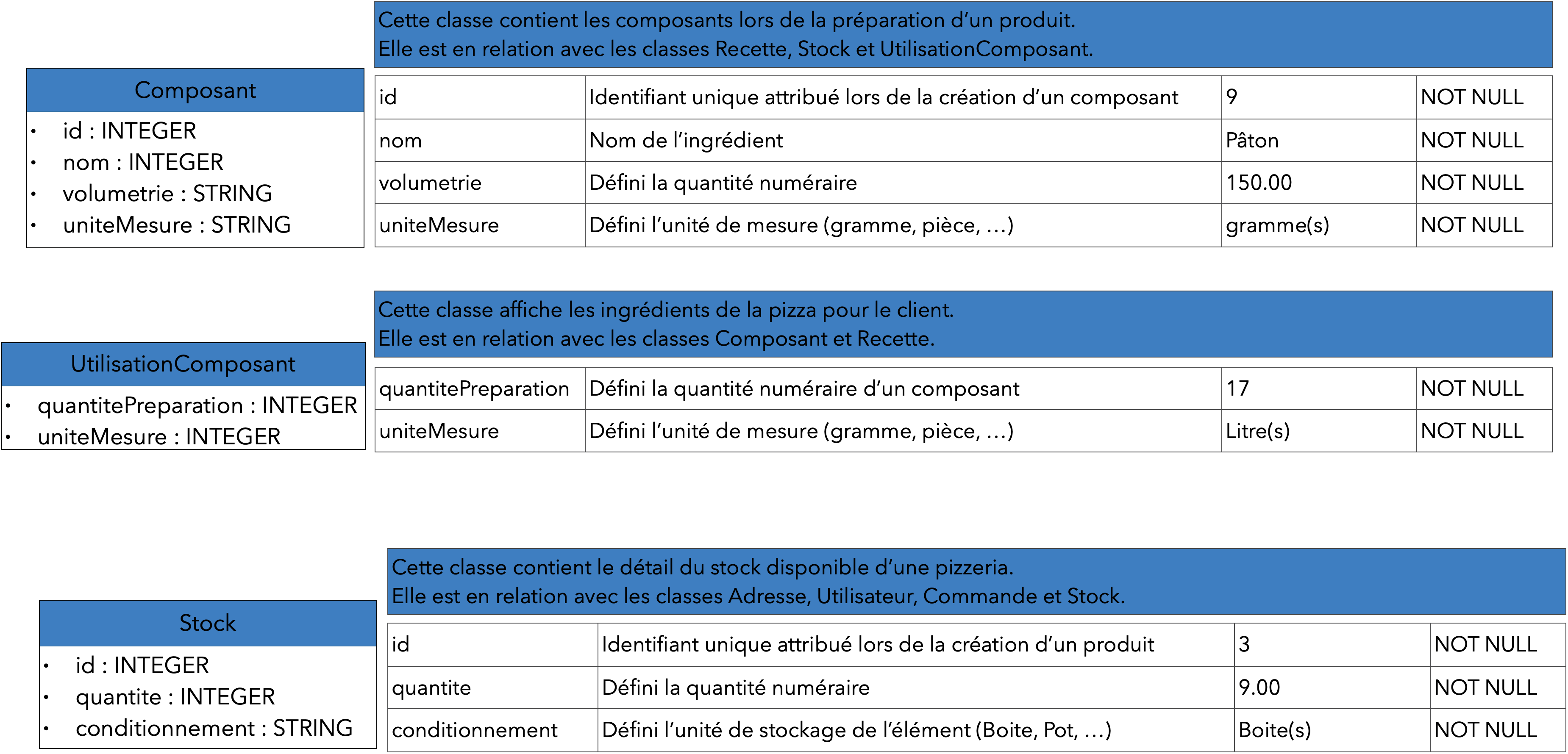






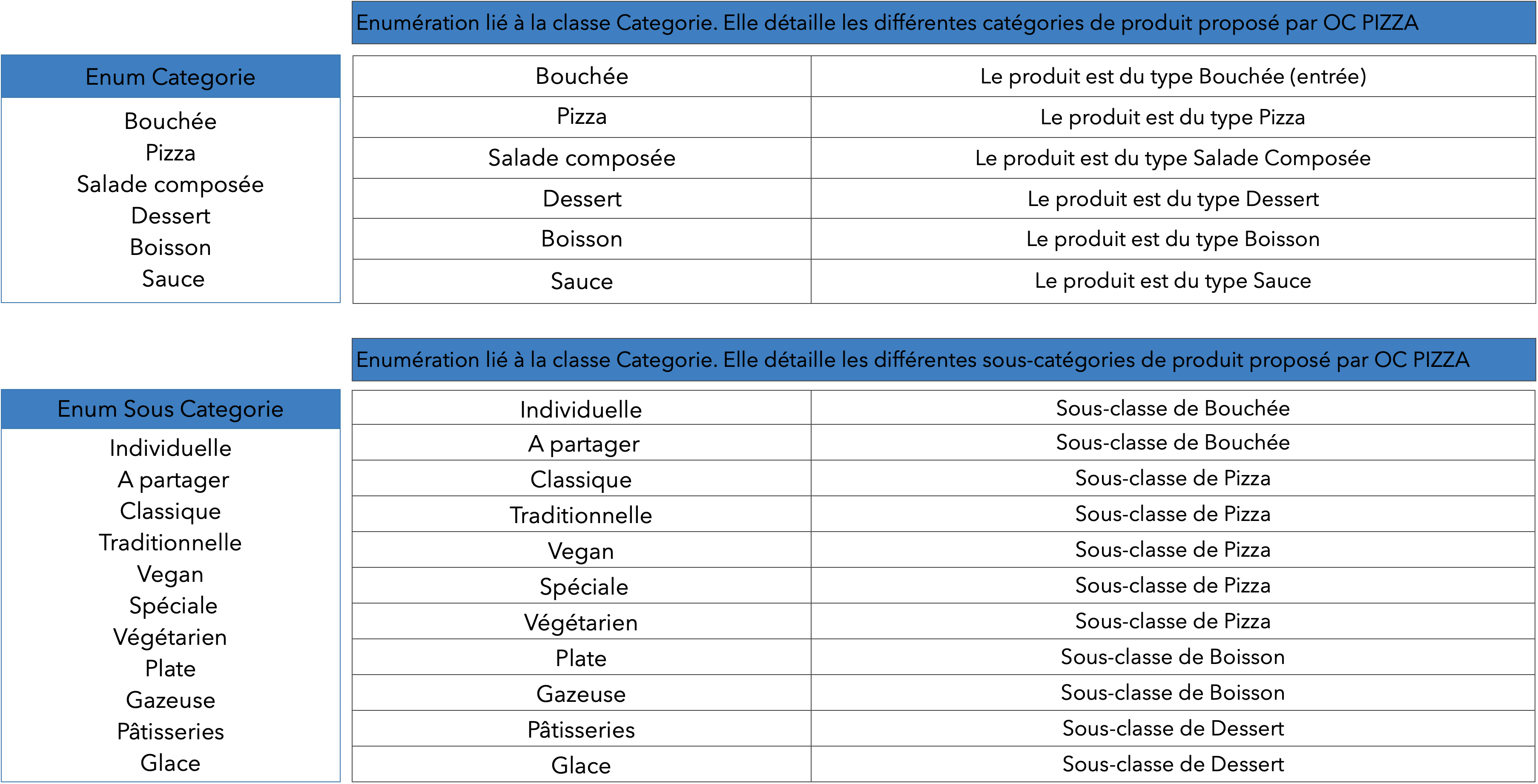


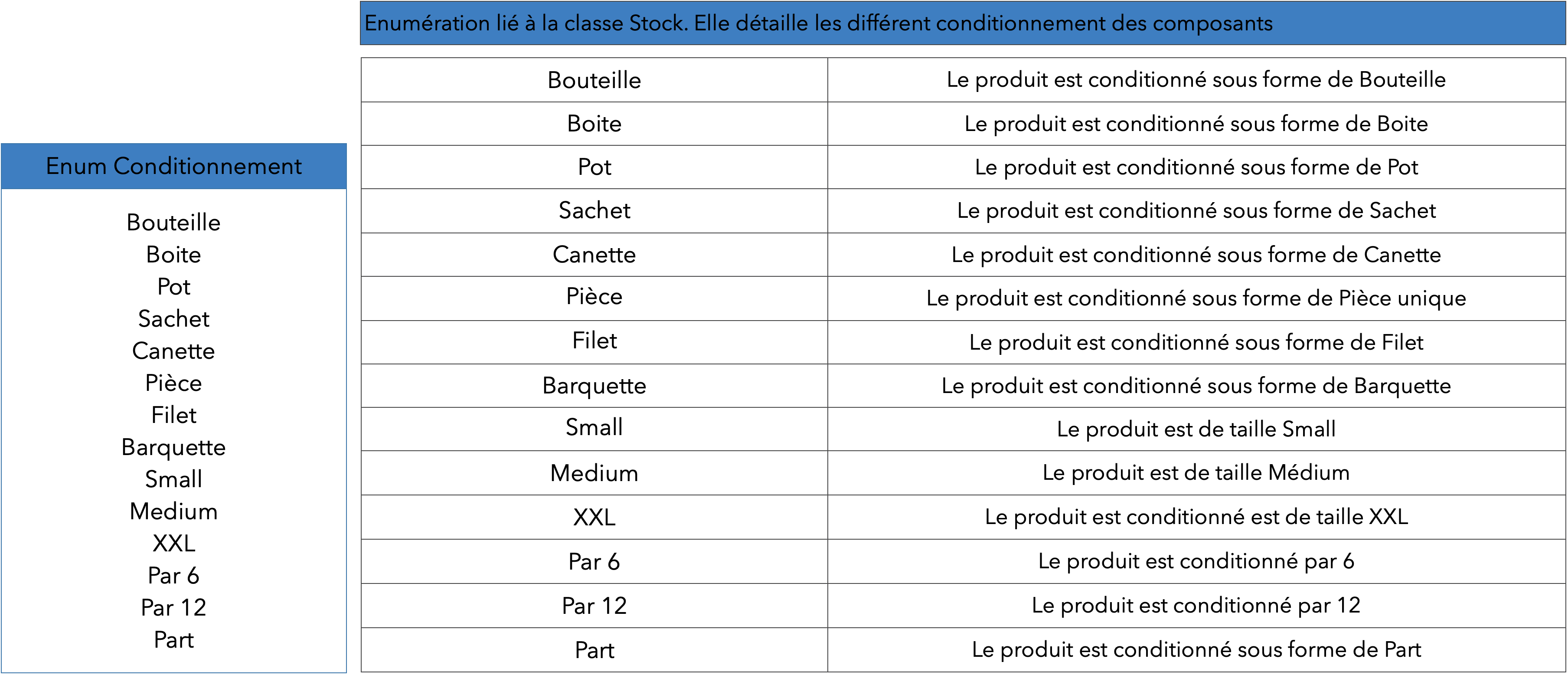


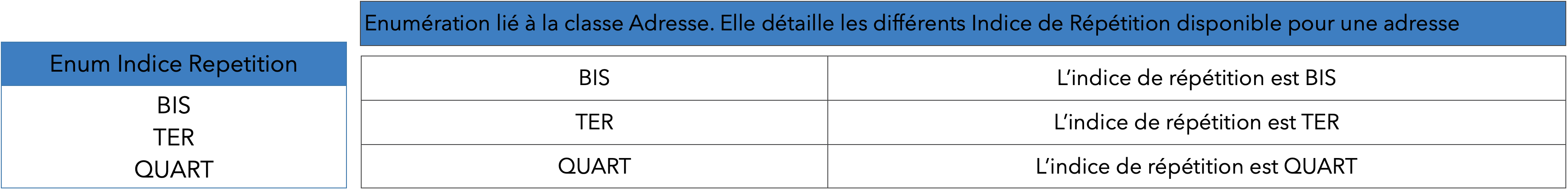


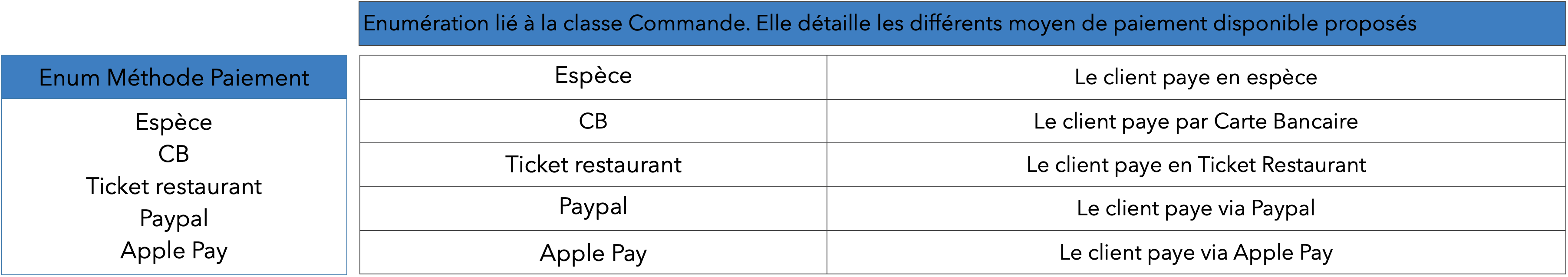
3.5.4 - Détail des énumérations

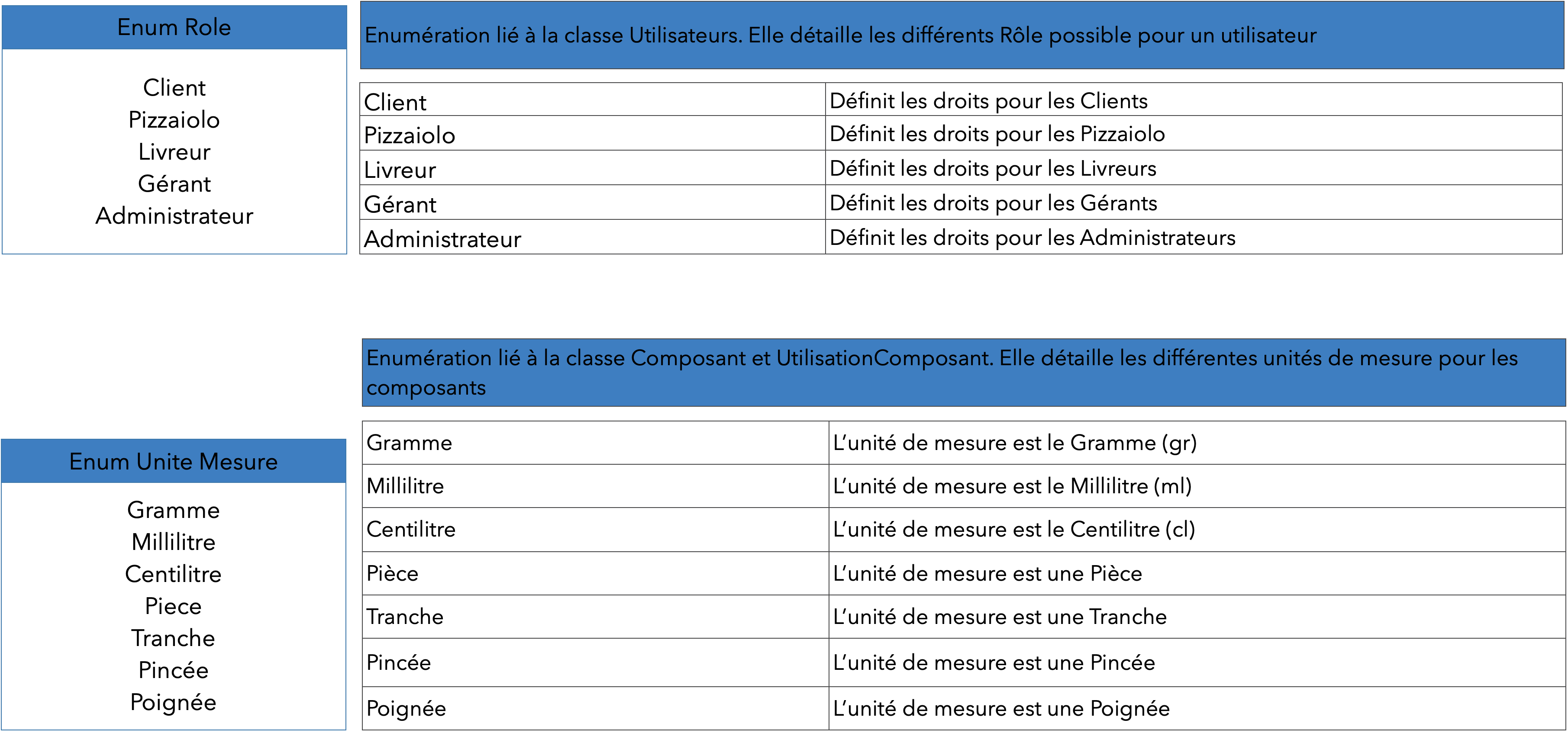
Tout comme les tables nous détaillons les énumérations de la même manière.













# 4 - Architecture de déploiement

Notre architecture de déploiement est donc la suivante. Elle explique la disposition des composants du système sur les infrastructures physiques.



Diagramme UML de déploiement

Notre serveur est composé d’un Front-end et d’un Back-end qui vont communiquer ensemble sur notre serveur.

Nos utilisateurs accèdent à la partie Front End du Site Web qui va permettre d’afficher les informations voulues.

Notre partie Back End quant à elle, va communiquer avec la base de données via des requêtes SQL.

Notre application iOS va elle aussi être en lien avec le Back-end.

Enfin nous avons recours à des API pour la partie système de paiement et le système de localisation.

## 4.1 – Serveur de base de données

Point central de notre système la base de données sera sous PostgreSQL.

La base de données sera hébergée sur un serveur OVH doté de :

• 2 CPU

• 4 Go de RAM

Le schéma de la base de données a été vu dans la partie 3.5.2

## 4.2 - Serveur Web

Pour le serveur Web nous avons choisi l’offre d’OVH « Cloud Web », qui inclus :

• 2 CPU

• 4 Go de RAM

A cela s’ajoute d’autres fonctions utiles comme :

• Un nom de domaine personnalisé  
• Le **SSL Let’s Encrypt** qui inclus la certification du nom de domaine en activant le protocole HTTPS

Le coût mensuel est de 22.80€ TTC par mois.

L’offre est sécurisée et facilement modifiable.

# 5 - Architecture Logicielle

## 5.1 – Principes Généraux

Les sources et versions du projet sont gérées par Git.

L’application est construite selon le pattern MVC.

Les dépendances et le packaging par Swift Package Manager.

### 5.1.1 – Les couches

L'architecture applicative est la suivante :

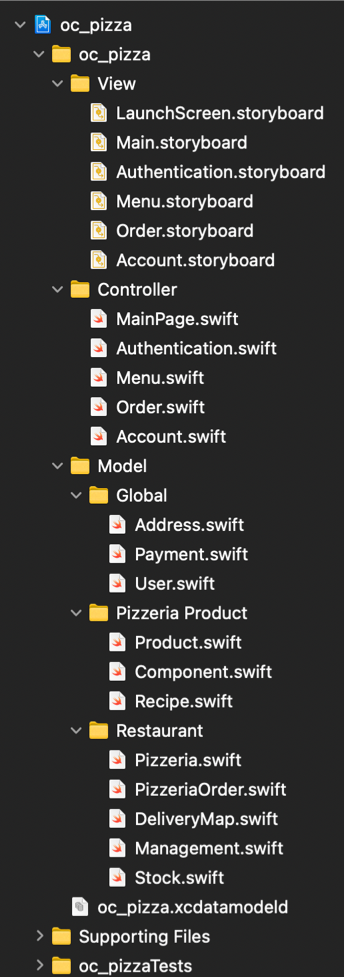
• une couche Modèle : responsable de la logique métier du composant

• une couche Vue : gestion de l’interface utilisateur

• une couche Contrôleur : qui permet la communication entre le modèle et la Vue

### 5.1.2 – Structures des sources

La structuration des répertoires du projet suit la logique suivante :

- les répertoires sources sont créés de façon à respecter l’architecture MVC

# 6 – Point particuliers

## 6.1 – Gestion des logs

Les logs serveurs seront dans le dossier /var/logs

Ce dossier regroupe les logs des différents services utilisés (Apache, MySQL, …)

Afin de visualiser les logs on utilisera Grafana avec Loki et Promtail.

## 6.2 – Ressources

### 6.2.1 – Charte graphique

Les éléments graphiques (polices, logos, photos des produits, jeux de couleur, …) sont fournis par OC Pizza.

### 6.2.2 - Données

Les données pour la création de la base de données sont-elles aussi fournies par OC Pizza, cela comprend notamment : les produits, les données utilisateurs, les adresses, …

Mais aussi les références bancaires pour la partie encaissement.

## 6.3 – Environnement de développement

### 6.3.1 - Documentation

Les documentations techniques sont listées ci-dessous :

• Debian https://www.debian.org/doc/index.fr.html

• Python https://docs.python.org/3/

• Django https://docs.djangoproject.com/en/3.2/

• HTML https://html.spec.whatwg.org/multipage/

• CSS https://www.w3.org/Style/CSS/

• JavaScript https://developer.mozilla.org/fr/docs/Web/JavaScript

• PostgreSQL https://docs.postgresql.fr

• Swift https://swift.org/documentation/

• Google Maps <https://developers.google.com/maps/documentation?hl=fr>

### 6.3.2 - Environnement

Le choix de l’environnement reste libre à chaque développeur néanmoins il existe des

Prérequis et des préconisations.

• L’IDE pour le développement de l’application iOS sera Xcode et nécessitera un

Environnement MacOs

• L’IDE recommandé pour le développement de la partie Site Web est Visual Studio Code

• L’outil de gestion pour PostgreSQL reste libre mais pgAdmin est préférable

• Le versionning passera par un client Git qui sera interfacé avec GitHub

## 6.4 – Procédure de packaging/ livraison

Le système fera l’objet d’un déploiement sur OVH et sur l’App Store au moment de la livraison finale.

En parallèle le code source du site sera donnée dans un zip au client.

De même pour le code de l’application qui aura son propre zip. A cela nous ajoutons un zip qui va contenir le dump de la base de données.

# 7- Glossaire

|  |  |
| --- | --- |
| **API** | Une API pour « Application Programming Interface », que l'on traduit par interface de programmation d'application.  Elle permet d'utiliser les ressources, données ou fonctionnalités, d'une application web depuis une autre application. |
| **Back End** | Le back end (arrière-plan) en opposition au front end, fait référence à l'ensemble des éléments qui ne sont pas visibles directement sur le site web mais qui va permettre le bon fonctionnement d’un site internet.  Le back end est exécuté côté serveur.  Le back end utilise principalement Python, Java, PHP. |
| **Back Office** | Le back office regroupe un ensemble de tâches administratives ou logistiques liées à la vente (, mise à jour des stocks, livraison, moyens de paiement, etc.)  C’est la partie du site visible par le personnel de la pizzeria. |
| **Front End** | Le front end ou (web frontal) est la partie d’une page internet ou d’une application qu’un utilisateur peut voir et avec lesquelles il peut interagir directement.  Le front end est exécuté côté client (PC, Smartphone, Tablette)  Le front end utilise principalement HTML, CSS et JavaScript. |
| **Front Office** | Le front office est l'ensemble des actions, fonctions ou tâches liées à la vente se faisant en contact avec le client.  C’est la partie du site visible par le client |
| **Grafana** | Grafana est une plate-forme open-source spécialisé dans l'analyse et la surveillance. Il fonctionne spécifiquement sur des données chronologiques provenant de sources telles que Prometheus et Loki. |
| **Git** | Logiciel de gestion de version |
| **IDE** | Un environnement de développement informatique est un ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui développent des logiciels. |
| **Logs** | Le terme log désigne un type de fichier, ou une entité équivalente, dont la mission principale consiste à stocker un historique des événements. |
| **Loki** | Loki est un système d'agrégation de journaux scalable horizontalement, hautement disponible et inspiré de Prometheus.  Loki va s’occuper de la centralisation des logs. |
| **MVC** | Le MVC est un motif d'architecture logicielle destiné aux interfaces graphiques.  Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs. |
| **Promtail** | Promtail est un agent qui expédie le contenu des journaux locaux vers une instance Loki privée ou Grafana Cloud. Il est généralement déployé sur chaque machine sur laquelle des applications doivent être surveillées. |
| **SGBD** | Système de Gestion de Base de données |