

# Spécifications techniques de l'implantation du système d'information de tous les acteurs de l'entreprise « OC- PIZZA »

Frédéric Pichot le 15/08/19

# Table des matières

## I. Introduction

**1.1 Le contexte** 4

**1.2 Les besoins** 4

## II. Domaine fonctionnel.

**2.1 Définition** 5

**2.2 Modélisation** 5

**2.3 Diagramme de Classes** 6

**2.4 Détail des Relations entre les Tables** 7

2.4.1 Utilisateur, Livreur, Accueil, Gérant, Pizzaiolo et Client 7

2.4.2 Utilisateur et Adresse 7

2.4.3 Pizzeria et Adresse 8

2.4.4 Pizzeria et Pizza 8

2.4.5 Pizza, Recette et Stock 9

2.4.6 Commande, Facture et Ligne De Commande 9

2.4.7 Commande, Statut Commande, Retrait, Paiement, Utilisateur et Pizza 10

## III. Organisation physique

**3.1 Définition** 11

<b>3.2 Exemple</b>	<b>11</b>
<b>3.3 Diagramme du Modèle Physique de Données</b>	<b>12</b>
3.3.1 Les tables utilisateur et livreur, accueil, gérant, pizzaiolo, client	13
3.3.2 Les tables commande, statut commande, retrait, paiement	13
3.3.3 Les tables pizza, recette, stock et ingrédient	14
3.3.4 Les tables pizzeria, formule et pizza	14
3.3.4 Les tables utilisateur, pizzeria et adresse	15
3.3.5 Les tables commande, pizza, facture et ligne de commande	15
<b>IV. Composants internes et externes</b>	
<b>4.1 Définition</b>	<b>16</b>
<b>4.2 Description des composants</b>	<b>17</b>
4.2.1 Le module Web Store	17
4.2.2 Le module OC_Pizza	18
4.2.3 Le module OC_Pizza	18
4.2.4 Le module Ordre de commande	19
<b>V. Architecture et déploiement</b>	
<b>5.1 Description</b>	<b>21</b>
<b>5.2 Diagramme de déploiement</b>	<b>22</b>

## I. INTRODUCTION

### 1.1 Le contexte

La société « OC Pizza » est un jeune groupe de pizzeria en plein essor et spécialisé dans les pizzas livrées ou à emporter. Il compte déjà 5 points de vente et prévoit d'en ouvrir au moins 3 de plus d'ici la fin de l'année. Pour avoir une visibilité d'ensemble un responsable du groupe nous a contacté pour que nous créions, installions et déployions un ensemble de systèmes informatiques multisupport permettant la mise en interconnexion de toutes les pizzerias. Au final la société souhaite proposer à la clientèle un ensemble d'accès à leur fabrication tant statique que mobile.

### 1.2 Les besoins

Vous souhaitez suivre en temps réel les commandes passées et en préparation ainsi que le stock d'ingrédient.

Vous souhaitez connaître le niveau d'avancement de chaque pizza afin d'aviser le client sur l'état de sa commande.

Vous souhaitez de proposer un site Internet et des applications mobiles pour que les clients puissent :

- passer leurs commandes, en plus de la prise de commande par téléphone ou sur place
- payer en ligne leur commande s'ils le souhaitent – sinon, ils paieront directement à la livraison
- modifier ou annuler leur commande tant que celle-ci n'a pas été préparée

Pour finir vous souhaitez proposer un aide mémoire aux pizzaiolos sur la recette de chaque pizza

## II. Domaine fonctionnel.

### 2.1 Définition

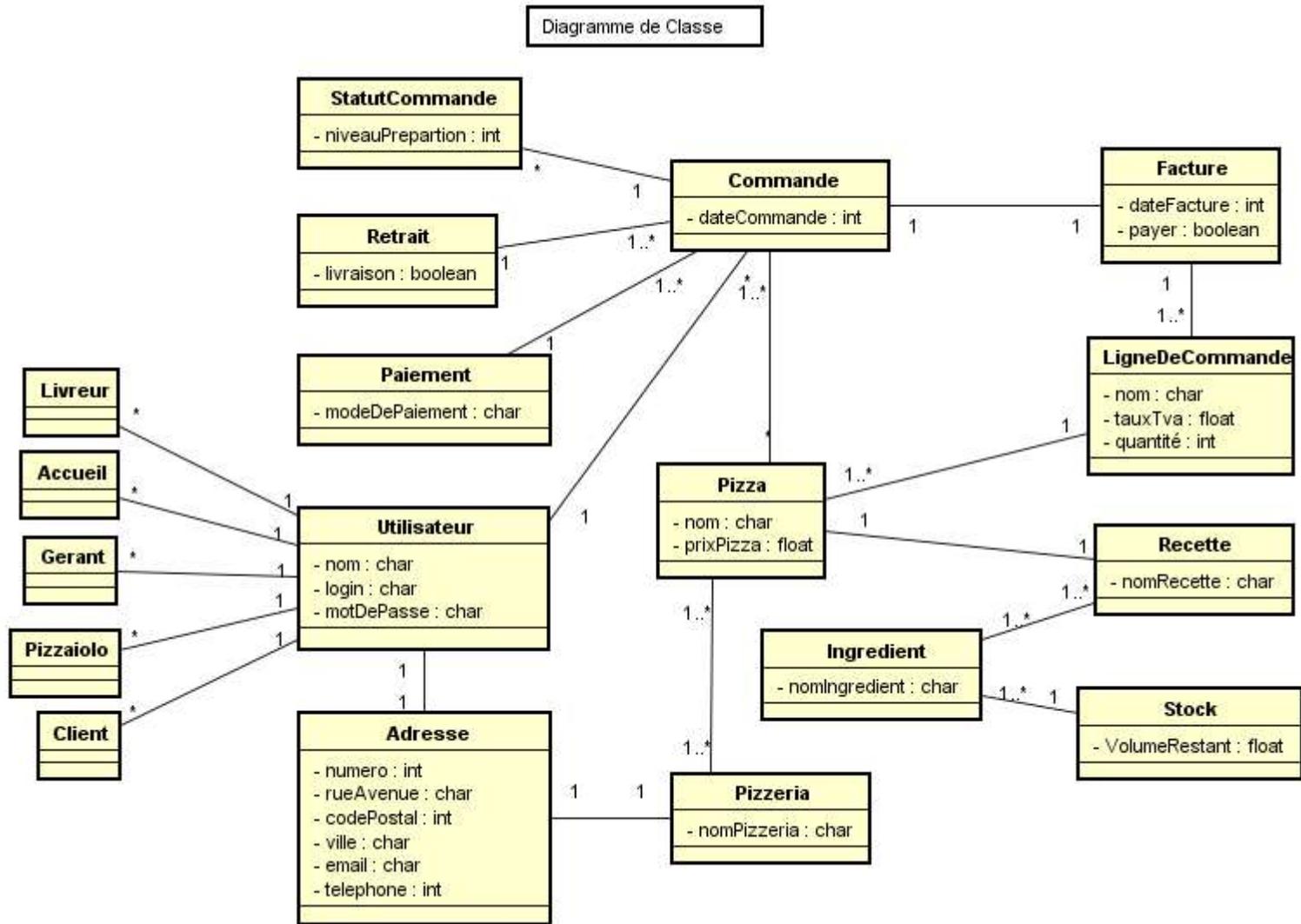
Le domaine fonctionnel est modélisé au travers d'un diagramme UML permettant de transposer le domaine sous la forme d'objet relié par un lien de jonction sur lequel on stipule le nombre d'occurrence de chacun des objets par rapport à l'autre.

### 2.2 Modélisation

L'élément principale du schéma est une « Classe ». Celle-ci est basée sur un concepts que chaque composant qui nous entoure peut être représenté par une classe. Tout d'abord nommé, celle-ci décrit l'objet par une liste d'attributs qui sont sensés d'écrire l'objet ou une personne.

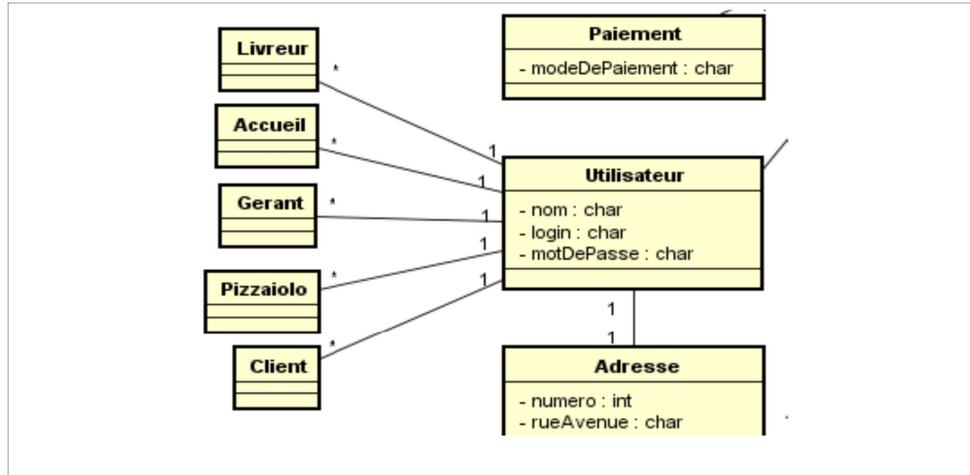
La liaison est un trait qui relie deux objets et qui signale à chacune de ses extrémités le nombre possible d'occurrence qui est possible lorsque le liaison est activée.

## 2.3 Diagramme de Classes



## 2.4 Détail des relations entre les Tables

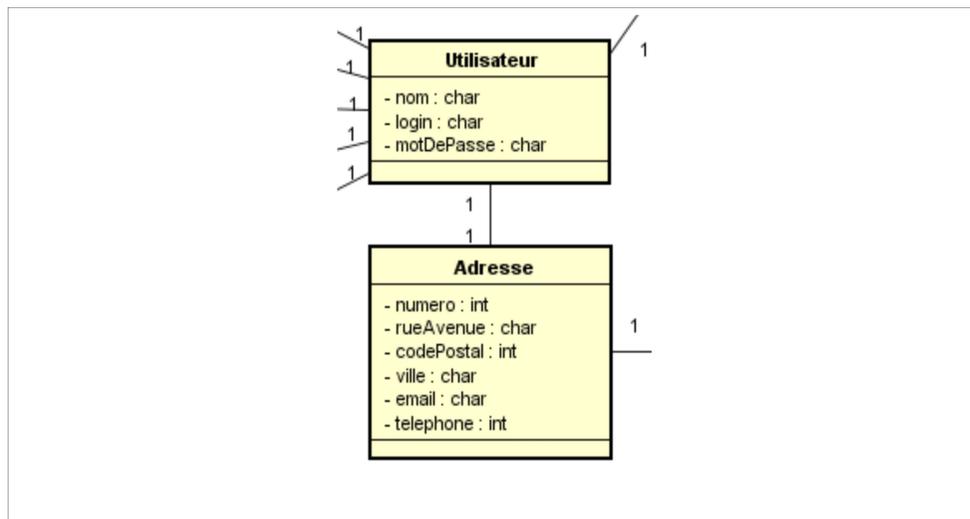
### 2.4.1 Utilisateur, Livreur, Accueil, Gérant, Pizzaiolo et Client



Pour permettre une modélisation chacun des protagonistes cités sont potentiellement est utilisateur du système.

La relation qui les unit, à savoir \*/1, stipule que chaque livreur peut être un utilisateur au moment de l'action, mais l'objet utilisateur peut avoir plusieurs objets Livreur.

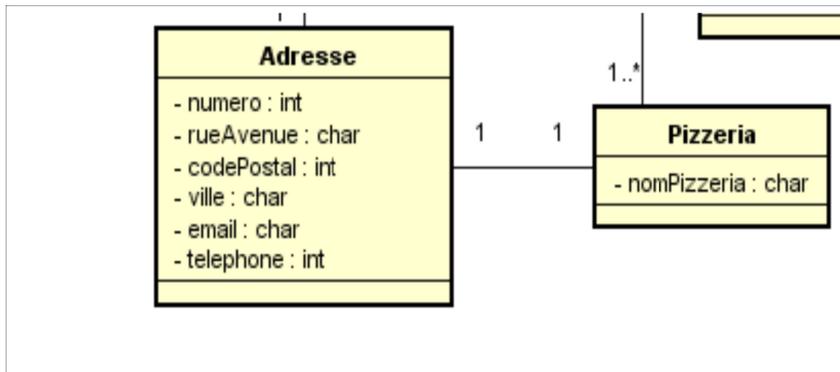
### 2.4.2 Utilisateur et Adresse



Cette partie du graphe mentionne que l'utilisateur est décrit par les attributs nom, login et motDePasse. En effet l'utilisateur pour pouvoir consommer devra s'enregistrer pour ainsi s'identifier à chaque fois qu'il se connecte. Les coordonnées de l'utilisateur s'obtiendront par le lien qui relie la table Utilisateur à celle d'Adresse.

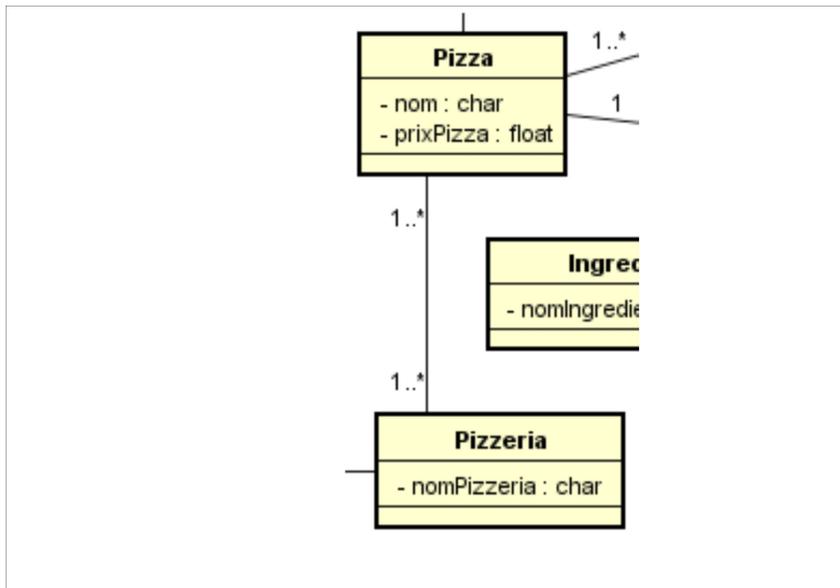
Cette liaison est du type 1/1 en effet chaque utilisateur n'a qu'une seule adresse et une adresse n'appartient qu'à un seul utilisateur. On peut remarquer que chaque attribut a une identification de son type de donnée qu'il sera amené à recevoir. Par exemple le numéro de l'adresse est un « int » qui veut dire « INTEGER » en anglais. Rue, Avenue sont des « char » qui est la traduction de caractère.

### 2.4.3 Pizzeria et Adresse



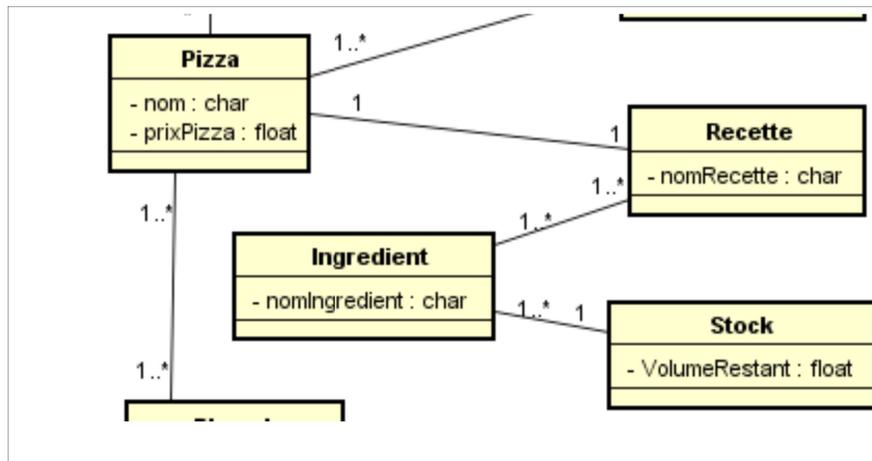
Dans le même principe que tout à l'heure une pizzeria n'a qu'une adresse et celle-ci n'appartient qu'à une seule pizzeria.

### 2.4.4 Pizzeria et Pizza



La relation qui unit Pizzeria et pizza est du type 1/\* car chaque pizzeria a plusieurs modèles de pizza à proposer comme d'ailleurs chaque pizza est fabriquée dans toutes les pizzerias du groupe.

## 2.4.5 Pizza, Recette et Stock

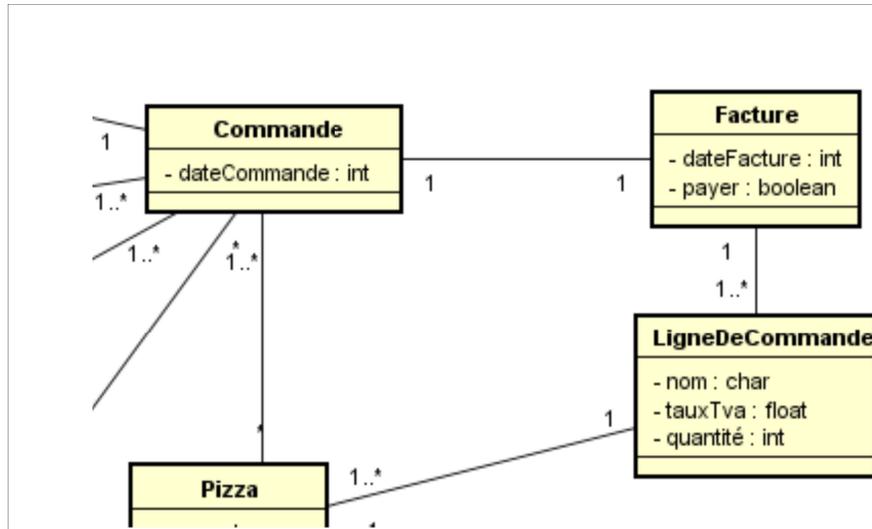


La pizza à un nom comme attribut, mais sa recette est dans une autre table car la composition regroupe plusieurs ingrédients. Pour autant, leur liaison est du type 1/1 car une pizza n'a qu'une seule recette et celle-ci définit une unique pizza.

La liaison entre recette et ingrédient est du type 1/\* dans les deux sens puisque chaque ingrédient peut être présent dans différentes recettes et en retour chaque recette contient plusieurs ingrédients.

Le stock quant à lui à plusieurs ingrédients mais en retour l'ingrédient n'a qu'un niveau de stock. On obtiendra le niveau du stock par le trait qui relie ingrédient et stock.

## 2.4.6 Commande, Facture et Ligne De Commande



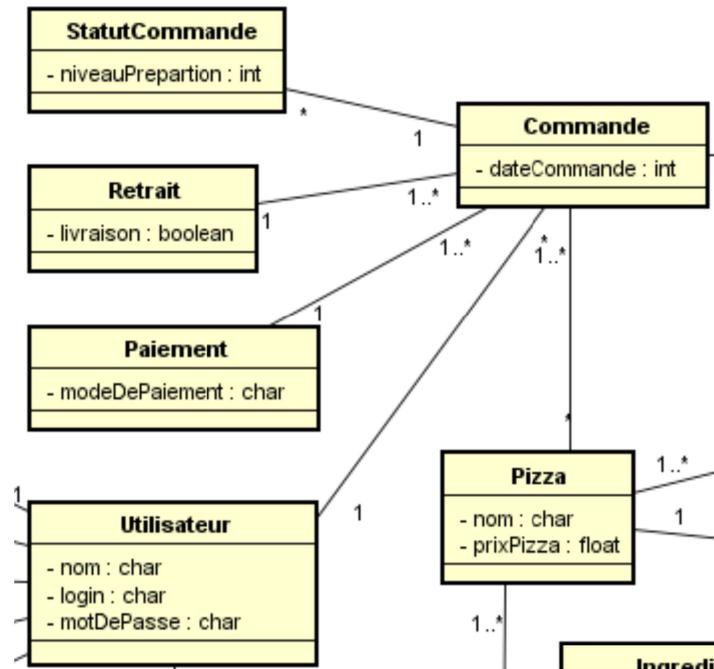
Cet capture d'écran regroupe la commande, la facture, la ligne de commande et la pizza. La liaison qui unit pizza et ligne de commande précise que sur une même ligne de commande il est possible d'avoir une ou plusieurs pizzas, mais le nom de la pizza ne peut-être inscrit qu'une seule fois.

La liaison facture ligne de commande suit les même règle que précédemment une ligne de commande ne peut se trouver qu'une seule fois par contre la facture peut avoir plusieurs lignes de commande.

Par contre une facture ne peut être attachée qu'à une seule facture et celle-ci ne peut faire éditer qu'une facture.

L'attribut booléen payé n'offre que deux statuts soit elle est réglée ou non.

## 2.4.7 Commande, Statut Commande, Retrait, Paiement, Utilisateur et Pizza



Cette dernière partie regroupe les classes Commande, Statut commande, Retrait, Paiement, Utilisateur et Pizza. La liaison entre les deux première est du type \*/1 en effet à un moment donné la commande peut avoir plusieurs statuts, mais pour chaque pizza le statut est unique.

L'attribut niveau préparation est imagé par un numéro qui pourrait aller de 1 à 5, dont le 1 serait la prise de commande, le deux la préparation et pour finir la livraison et le paiement.

Dans la classe Retrait, il y a un seul attribut qui indique si la commande est à livrer ou non, mais pour une même commande il n'y en aura qu'un.

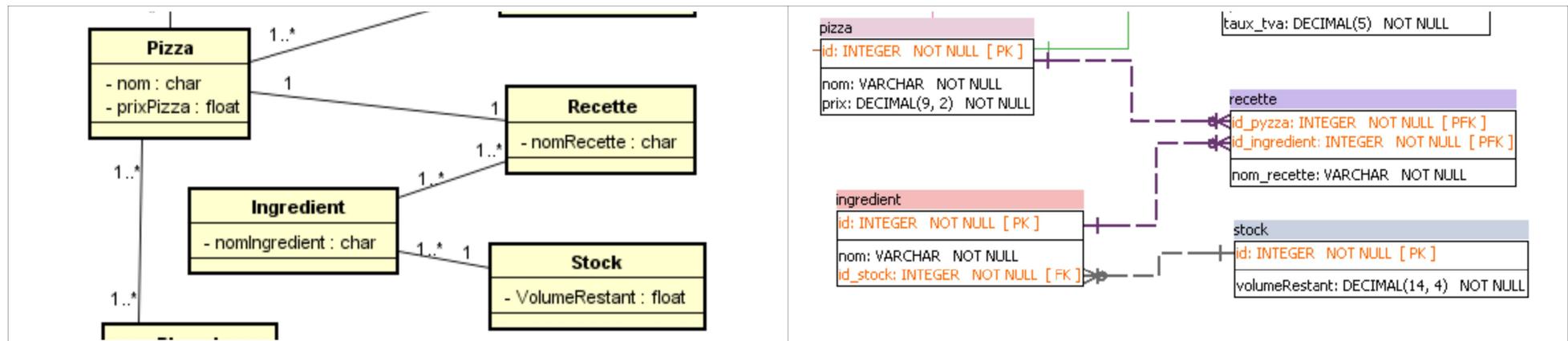
La Commande ne peut avoir qu'un seul utilisateur, mais celui-ci peut passer plusieurs commande. Le mode de paiement est un char pour permettre le réglé en cash, en chèque ou en carte voir d'autres qui seront à définir.

### III. Organisation physique des données

#### 3.1 Définition

Les objets ou classes que nous avons matérialisés précédemment ne peuvent pas encore être utilisables pour la conception d'une base de données. Bien que le diagramme de classe soit indispensable pour conceptualiser les différents acteurs, pour lister le plus exhaustivement possible les attributs et les affecter le plus pertinemment possible aux objets qui les caractérisent le mieux, pour autant, il manque deux éléments prépondérants pour constituer une base de donnée. Le premier est de permettre de rendre unique chacune des occurrences, le second est le moyen de matérialiser les liens qui relient les différentes tables.

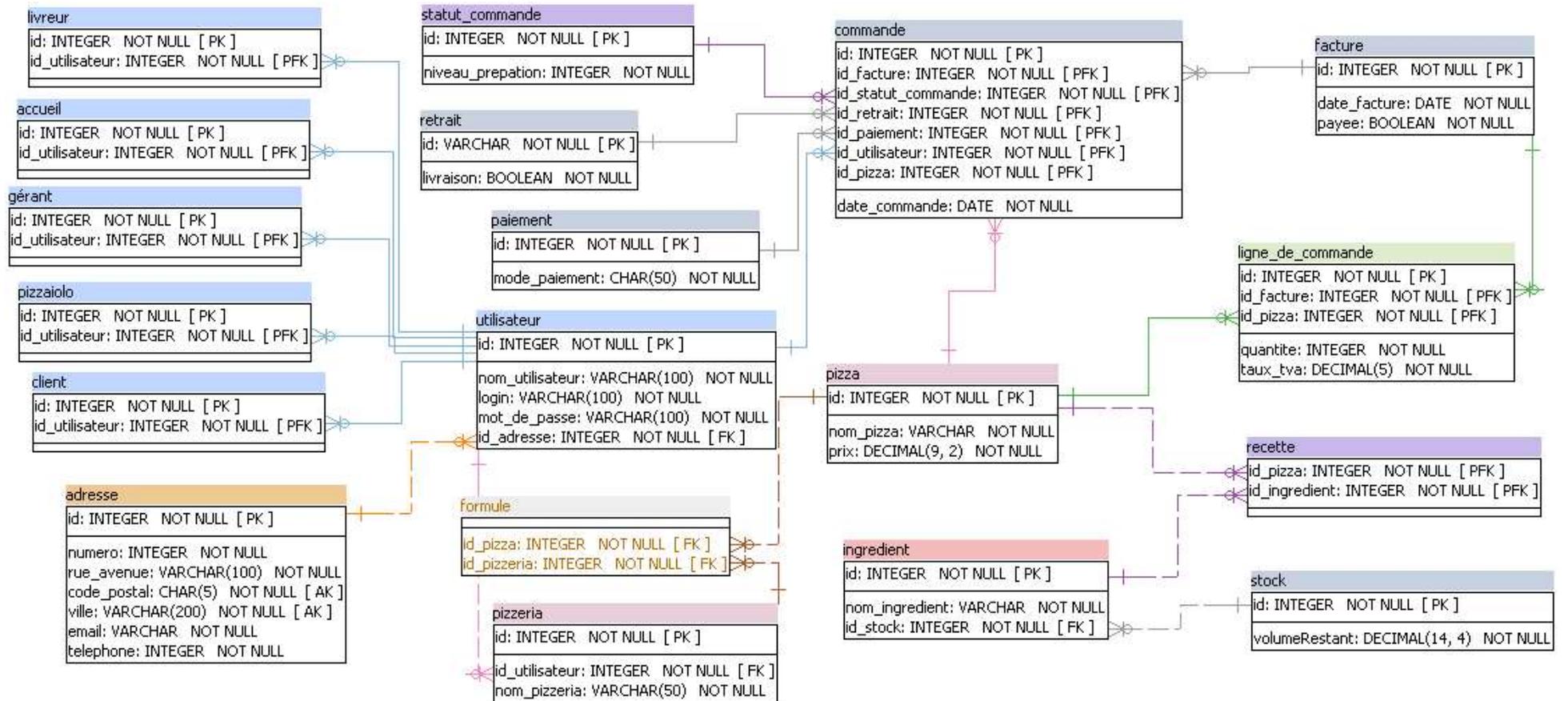
#### 3.2 Exemple : Comparons les liens autour de la pizza



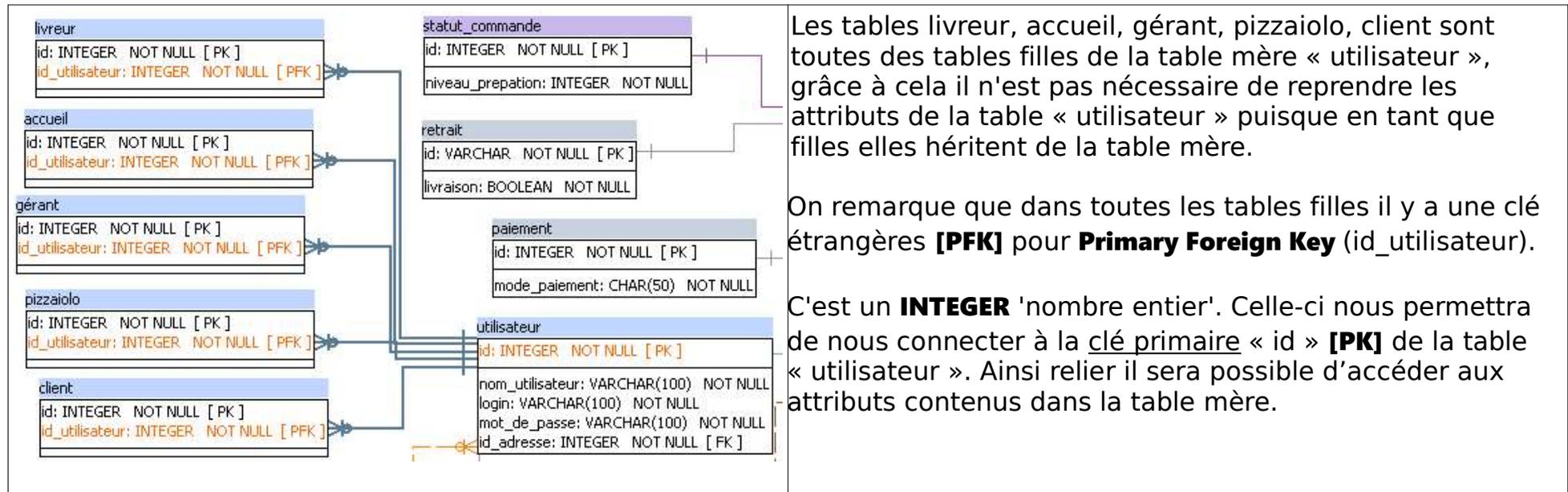
Il est entendu que tous les attributs sont les noms des colonnes dans la futur table. Maintenant pour différencier les occurrences, soit la succession des lignes entre elles et donc les rendre uniques, on intègre dans les attributs de la classe un identifiant (nommé « id ») qui s'incrémente de 1 à chaque fois que l'on entre une nouvelle occurrence, on s'en sert de clé primaire ou « Primary Key » symbolisé ainsi : [PK].

On peut remarquer le moyen que l'on utilise pour relier les deux tables pizza et recette, on intègre dans les attributs de la classe recette un identifiant qui reprend l'identifiant de la table pizza cela on l'appelle une clé étrangère.

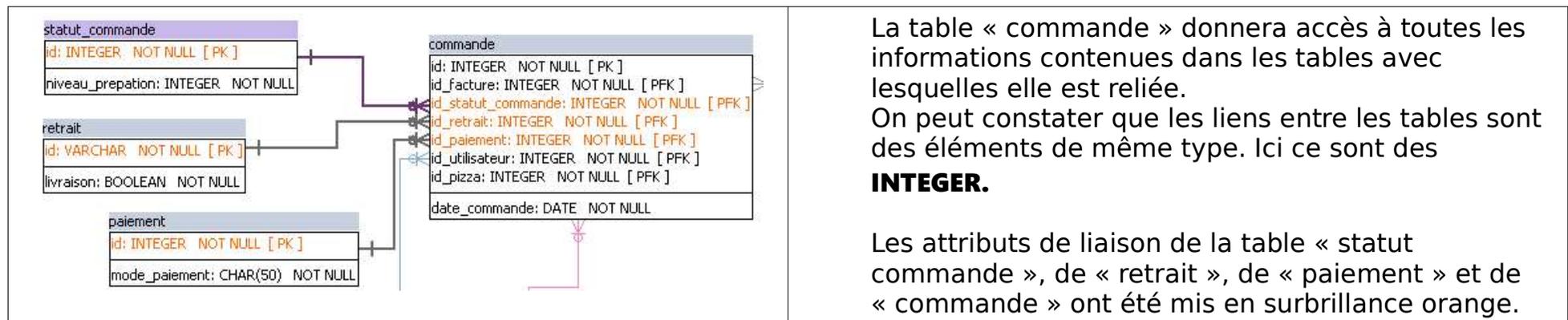
### 3.3 Diagramme du Modèle Physique de Données



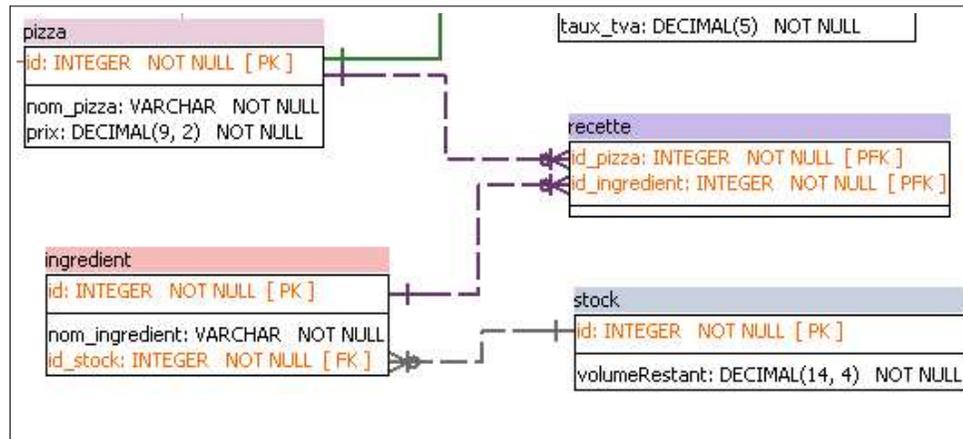
### 3.3.1 Les tables utilisateur, livreur, accueil, gérant, pizzaiolo, client



### 3.3.2 Les tables commande, statut commande, retrait, paiement



### 3.3.3 Les tables pizza, recette, stock et ingrédient

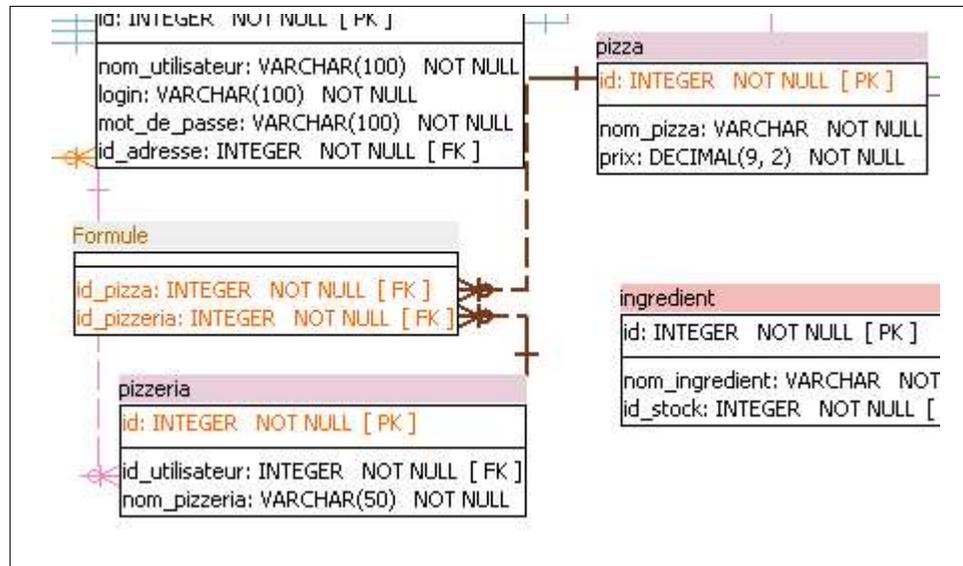


Dans cette partie nous avons la table recette qui est une table d'association, en effet elle est constituée de deux clés étrangères qui permettront de relier les tables « pizza » et « ingrédient ».

Cette dernière donnera accès au niveau de stock de l'ingrédient.

La clé « id\_stock » est une clé étrangère noté **[FK]** pour **Foreign Key**

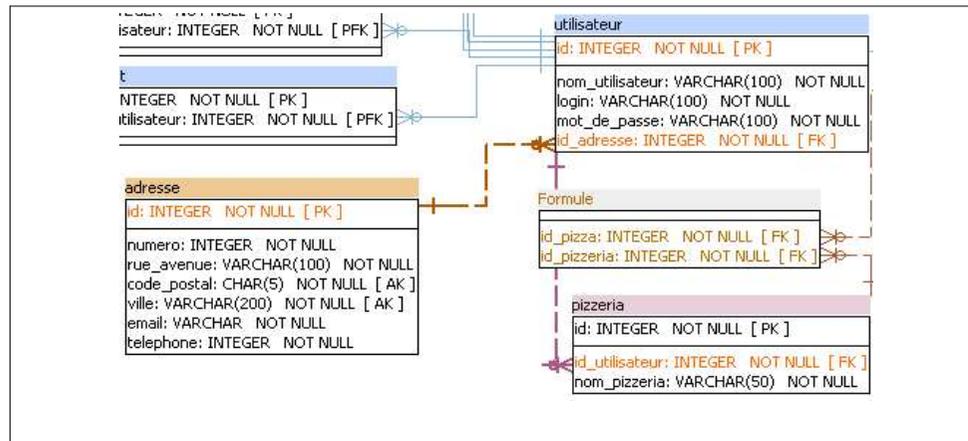
### 3.3.4 Les tables pizzeria, formule et pizza



Les tables « pizzeria » transitera par la table « utilisateur » en utilisant la [FK] « id\_utilisateur », et par rebond « id\_adresse » pour atteindre son adresse. Ce procédé est un excellent moyen pour éviter de stocker des doublons, ce qui est souvent un risque de glisser des erreurs de saisie, et donc d'avoir une table moins optimisée.

Les attributs nom de la table « pizzeria » est un **VARCHAR** 'chaîne de caractère'. Ce champs est paramétrable il a été limité à 50 caractères dans ce cas, mais dans la table « adresse » d'autres le sont à 200.

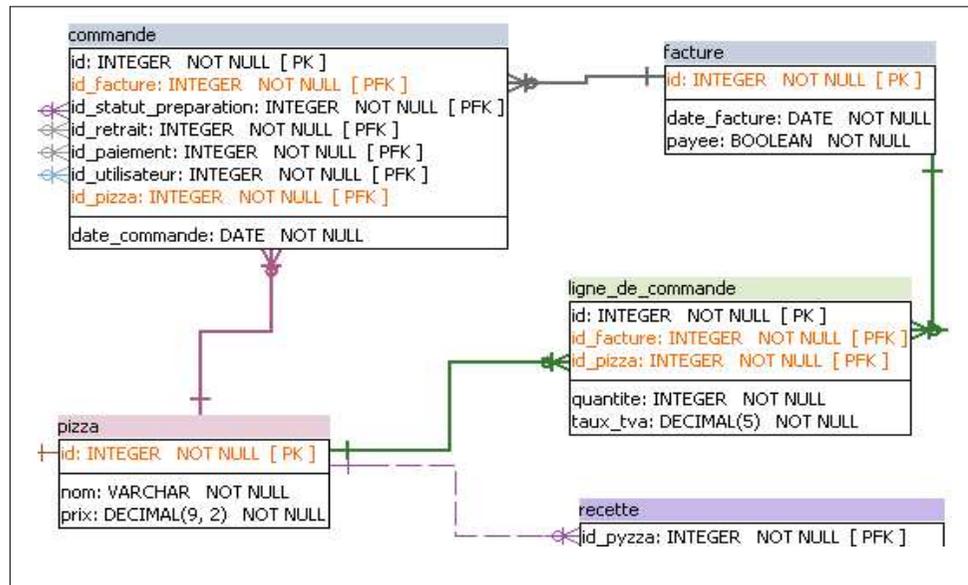
### 3.3.5 Les tables commande, pizza, facture et ligne de commande



Les tables « pizzeria » transitera par la table « utilisateur » en utilisant la [FK] « id\_utilisateur », et par rebond « id\_adresse » pour atteindre son adresse. Ce procédé est un excellent moyen pour éviter de stocker des doublons, ce qui est souvent un risque de glisser des erreurs de saisie, et donc d'avoir une table moins optimisée.

Les attributs nom de la table « pizzeria » est un **VARCHAR** 'chaîne de caractère'. Ce champs est paramétrable il a été limité à 50 caractères dans ce cas, mais dans la table « adresse » d'autres le sont à 200.

### 3.3.6 Les tables commande, pizza, facture et ligne de commande



La table « ligne\_commande » utilise deux clés primaires étrangères pour relier la table « facture » et la table « pizza ».  
Tant qu'à la table « commande » le lien avec la table « facture » est assuré.

## **IV. Composants internes & externes du système.**

### **4.1 Définition.**

Les diagrammes de composants représentent la structure du système logiciel, qui décrit les composants du logiciel, leurs interfaces et leurs dépendances. Ce type de diagramme prend en charge le développement à base de composants, dans lequel un système logiciel est divisé en composants et interfaces qui sont réutilisables et remplaçables.

Les diagrammes de composants sont utiles pour les raisons suivantes :

#### **Composants**

Ce sont des éléments de modèle qui représentent des éléments indépendants, interchangeables d'un système. Ils respectent et réalisent une ou plusieurs interfaces fournies et requises, qui déterminent le comportement des composants.

#### **Instances de composant**

Elles sont des éléments de modèle qui représentent des entités réelles dans un système.

#### **Packages**

Les packages regroupent des éléments de modèle associés de tout type, y compris d'autres packages.

#### **Interfaces**

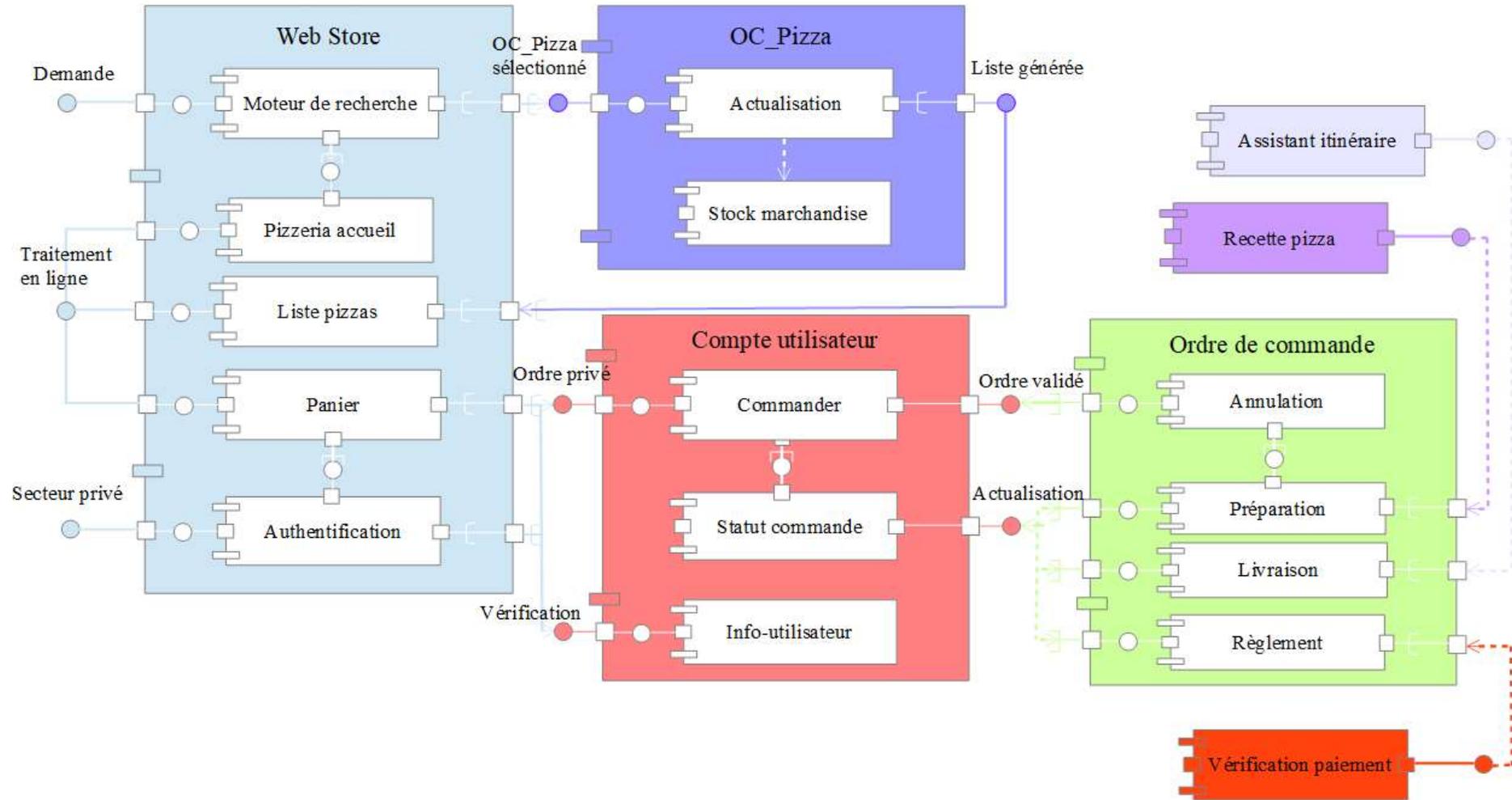
Elles sont des éléments de modèle qui définissent des ensembles d'opérations que d'autres éléments de modèle, tels que des classes ou des composants, doivent implémenter. Un élément de modèle d'implémentation réalise une interface en remplaçant chacune des opérations que l'interface déclare.

#### **Relations dans les diagrammes de composants**

En langage UML, une relation est une connexion entre des éléments de modèle. Une relation UML est un type d'élément de modèle qui ajoute une sémantique à un modèle en définissant la structure et le comportement entre les éléments de modèle.

## 4.2 Diagramme des composants.

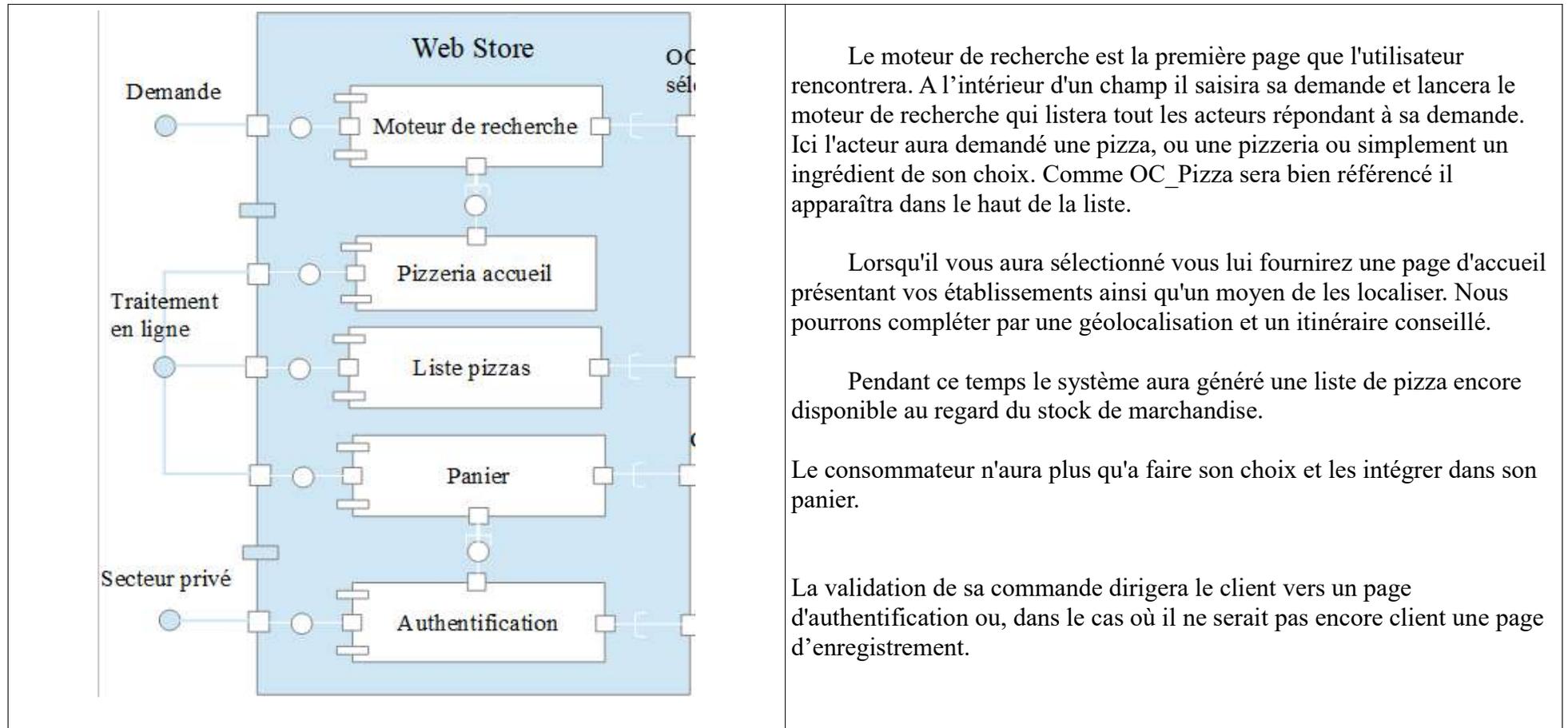
DIAGRAMME DES COMPOSANTS



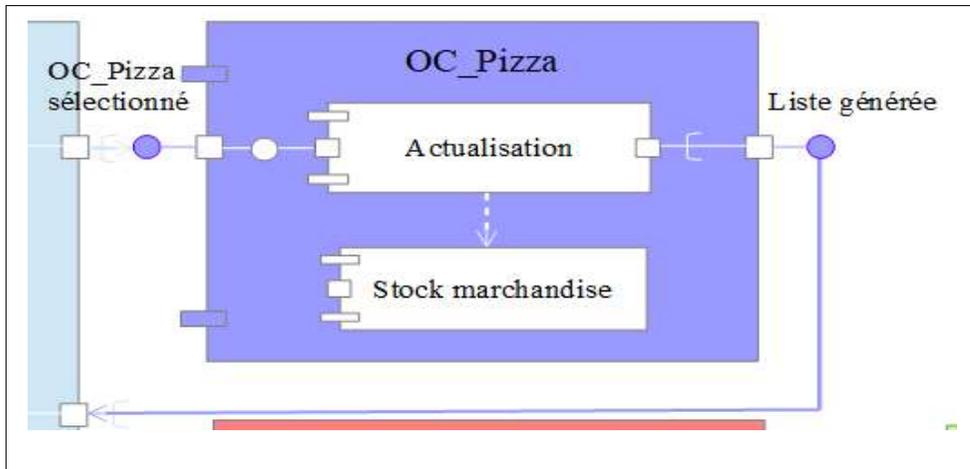
## 4.2 Description des composants.

Nous avons identifié 17 composants, vous remarquerez sans doute l'absence de la base de donnée, mais en faite elle omniprésente puisque tous les composants font appel à elle à un moment ou un autre du processus.

### 4.2.1 Le module Web Store :



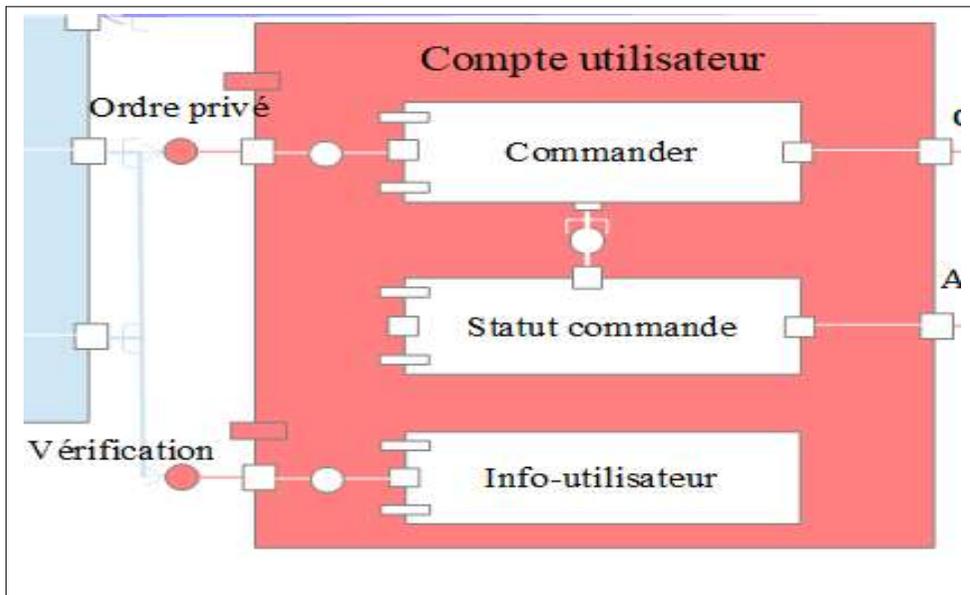
#### 4.2.2 Le module OC\_Pizza



Le client vient de vous sélectionner et consulte votre page d'accueil.  
Ici nous organisons la génération de la liste des pizzas en ayant interrogé le stock des ingrédients.

Une fois terminé nous la chargeons dès que le client aura cliqué le lien consulter la carte.

#### 4.2.3 Le module OC\_Pizza

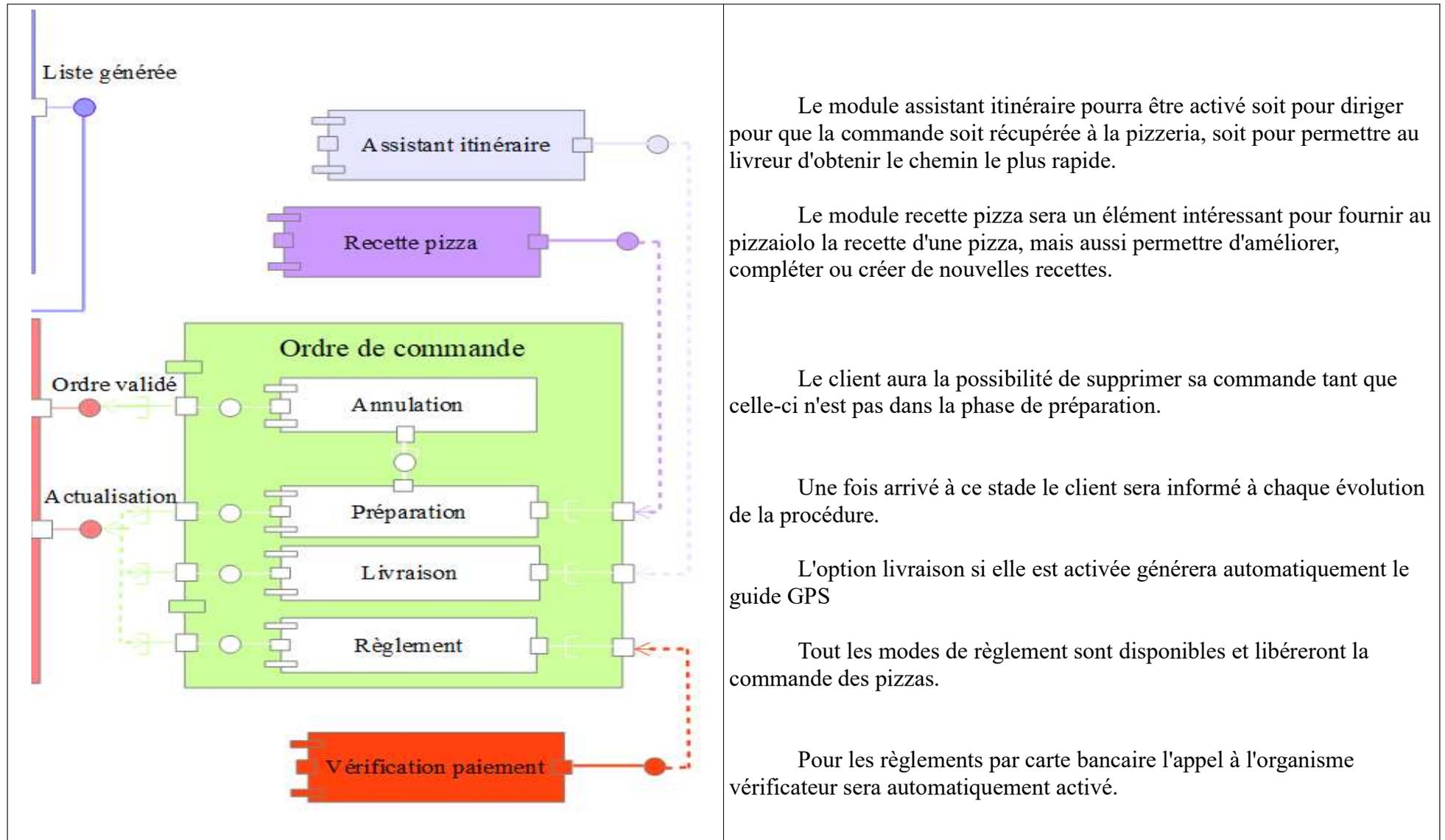


Le client rentre dans la phase de validation de son panier. Pour commander il devra s'être authentifié.

Le statut de la commande tient au courant le client de l'état d'avancement de sa commande.

A tout moment l'utilisateur pourra mettre à jour son compte client afin que ses moyens de contact et de livraison soit garantis.

#### 4.2.4 Le module Ordre de commande



## V. Architecture et déploiement

### 5.1 Description

Nous aurions pu réaliser un diagramme de déploiement ordinaire, mais nous avons souhaité rendre plus accessible celui-ci en y intégrant des modules factuels. Les connections sont remplacées par des vecteurs orientés. La disposition est respectée en unissant tous ceux qui se positionnent ensemble. Les symboles représentent la réalité, les nœuds devant représenter chaque ressource est seulement suggérée.

## 5.2 Diagramme de déploiement

