Proposition de projet OC Pizza

Dossier de conception technique

Auteur : Jean-François Subrini

1 mars 2018

Version : 1

TABLE DES MATIERES

1 - INTRODUCTION

1.1 - Objet du document

Le présent document constitue le **dossier de conception technique** de la future solution que PHAROS Consulting Services va développer pour OC Pizza.

Ce document met en évidence le **modèle fonctionnel du système** ainsi que le **modèle physique de données** qui servira à la conception de la base de données OC Pizza.
Les **composants internes et externes** seront aussi analysés afin d'expliciter les liens du système avec les applications externes (Google Maps, Banque).
Enfin, l'**architecture de déploiement** sera précisée afin d'identifier les éléments matériels ainsi que les éléments logiciels qui leurs sont rattachés.

En bref, ces spécifications décrivent comment la solution proposée sera implémentée du point de vue technique.

Pour de plus amples informations, se référer également au **Dossier de spécifications fonctionnelles** précédemment rédigé.

NB : Dans ce document, nous allons présenter différents diagrammes modélisés selon la méthode UML ainsi qu'un Modèle Physique de Données.

1.2 - Contexte

Les éléments du présent dossier découlent du cahier des charges que nous avons reçu, point de départ de notre travail.

Cahier des charges OC Pizza :

« 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. Mettre en place un système informatique, déployé dans toutes ses pizzerias et qui lui permettrait notamment :

- *d'être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation ;*
- *de suivre en temps réel les commandes passées et en préparation ;*
- *de suivre en temps réel le stock d'ingrédients restants pour savoir quelles pizzas sont encore réalisables ;*
- *de proposer un site Internet 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*
- *de proposer un aide mémoire aux pizzaiolos indiquant la recette de chaque pizza*

Suite à notre demande de précision, OC Pizza nous a confirmé que toutes les pizzerias actuelles et à venir du groupe se situaient dans l'**agglomération bordelaise**.

1.3 - Enjeux et Objectifs

Ce document est la conclusion de la **deuxième étape** d'une démarche qui se réalise, **en collaboration avec OC Pizza**, dans une optique de répondre au plus près à ses besoins et à ceux de ses clients utilisateurs.

Les réunions à venir nous permettront de mettre un point sur les spécifications techniques de la solution, en accord avec la version 2, précédemment validée par les deux parties, des **spécifications fonctionnelles** (cf. Dossier de spécifications fonctionnelles, v2).

2 - LE DOMAINE FONCTIONNEL

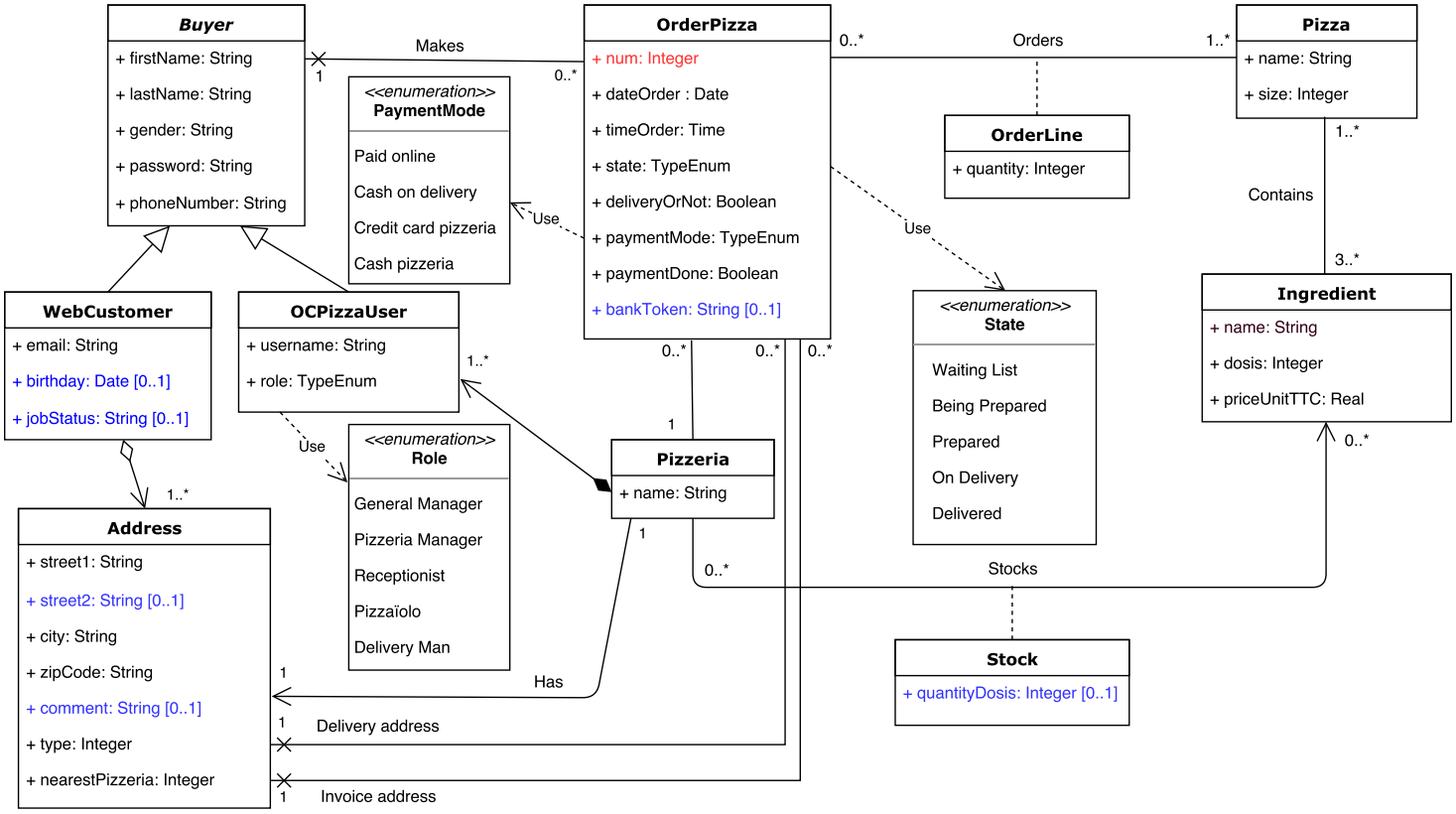
2.1 - Description

Le domaine fonctionnel du système OC Pizza établit l'ensemble des classes liées entre elles qui serviront de support à la programmation en Python et à la création du Modèle Physique de Données, base de la modélisation de la base de données relationnelle.
Le domaine fonctionnel est représenté par un diagramme UML : le Diagramme de classes.

2.2 - Diagramme de classes UML d'OC Pizza

Le diagramme de classes UML (*Class Diagram*) représente ainsi l'organisation de l'information grâce aux différentes classes et aux liens entre-elles.

Class Diagram for OC Pizza



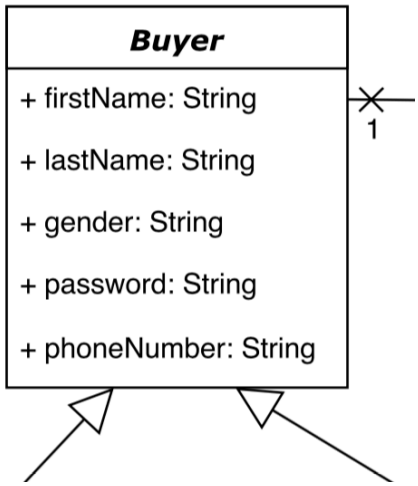
2.3 - Descriptif des classes

2.3.1 - Classe « Buyer »

Cette classe regroupe les attributs communs à tous les « Acheteurs ».

Cette **classe mère** est associée aux **classes filles** ***WebCustomer*** et ***OCPizzaUser*** qu'elle généralise.

Elle est aussi associée à la classe ***OrderPizza*** (***one-to-many***), car l' « Acheteur » fait (*Makes*) une commande. Chaque *Buyer* peut faire zéro ou une infinité de commandes (cardinalité **0..***) et chaque commande a obligatoirement un et un seul *Buyer* (cardinalité **1**).



Attribut	Description
firstName	Prénom de l'Acheteur
lastName	Nom de l'Acheteur
gender	Genre de l'Acheteur
password	Mot de passe de l'Acheteur
phoneNumber	Numéro de téléphone de l'Acheteur

2.3.2 - Classes « WebCustomer » et « OCPizzaUser »

Ces deux classes héritent de la classe *Buyer*. Ce sont des spécialisations de l' « Acheteur », association ***one-to-one***, avec dans le premier cas les attributs des *WebCustomer* (« Client Internet ») et dans le deuxième les attributs des *OCPizzaUser* (utilisateurs salariés d'OC Pizza).

Attribut	Description
email	Email du client Internet
birthday	Date de naissance du client Internet (attribut non obligatoire)
jogStatus	Statut professionnel du client Internet (attribut non obligatoire)

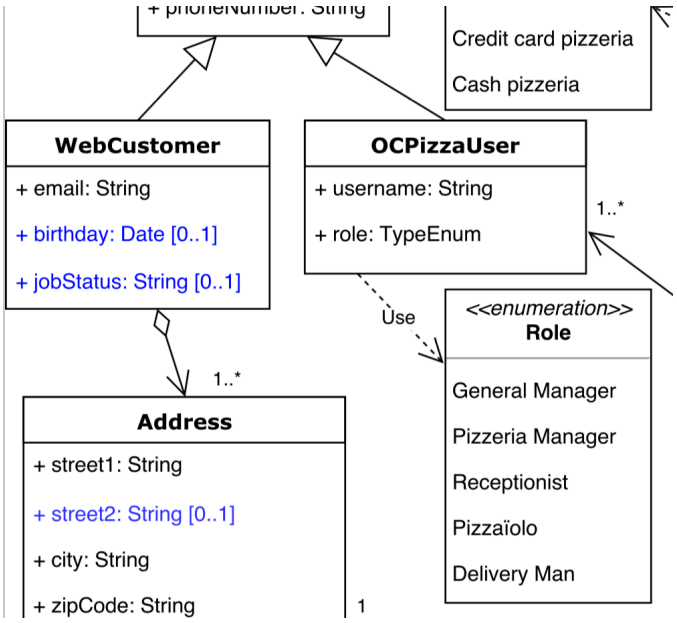
La classe ***WebCustomer*** est associée à la classe ***Address*** par une **association d'agrégation** (*one-to-many*) : les clients Internet possèdent et utilisent chacun une ou plusieurs adresses.

Attribut	Description
username	Nom d'utilisateur des employés d'OC Pizza
role	Qualification des postes des salariés d'OC Pizza (5 possibilités)

La classe ***OCPizzaUser*** est associée à la classe ***Pizzeria*** par une **association de composition** (*one-to-many*) : les différentes pizzerias du groupe OC Pizza regroupent et contiennent des utilisateurs salariés en leur sein.

La table <<*enumerate*>> définit les seules valeurs possibles de l'attribut *role* :

General Manager (Manager général), *Pizzeria Manager* (Manager de Pizzeria), *Receptionist* (Réceptionniste), *Pizzaïolo*, *Delivery Man* (Livreur).



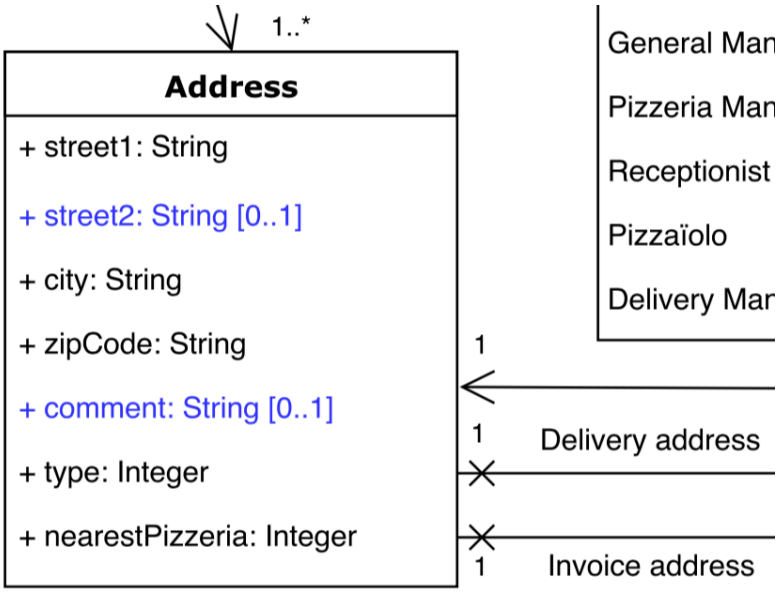
2.3.3 - Classe « Address »

Cette classe regroupe tous les attributs des adresses soit des clients Internet soit des pizzerias.

La classe ***Address*** a une association ***one-to-one*** avec la classe ***Pizzeria*** (chaque pizzeria possédant une adresse).

L'association avec *WebCustomer* est explicitée ci-dessus.

Elle a aussi deux associations ***one-to-many*** avec la classe ***OrderPizza*** : pour chaque commande, il y a en effet une (cardinalité **1**) adresse de facturation (*Invoice address*) et une (cardinalité **1**) adresse de livraison (*Delivery address*) qui peuvent être les mêmes ou différentes. Bien sûr, pour une adresse donnée, il peut y avoir aucune ou une infinité de commande (cardinalité : **0..***).



Attribut	Description
street1	Adresse (rue, cours, avenue, etc…)
street2	Complément d'adresse (attribut non obligatoire)
city	Ville
zipCode	Code postal
comment	Commentaires que peuvent laisser les clients sur le lieu de livraison. Exemple : code porte, étage, porte droite ou gauche… (attribut non obligatoire)
type	Qualification du type d'adresse (4 possibilités)
nearestPizzeria	Numéro de la pizzeria du groupe la plus proche de cette adresse du client

L'attribut *type* a 4 valeurs possibles : « *Adresse de facturation et de livraison habituelle* » (1) ; « *Adresse de livraison occasionnelle à enregistrer* » (2) ; « *Adresse de livraison occasionnelle à ne pas enregistrer* » (3) ; « *Adresse d'une pizzeria du groupe OC Pizza* » (4).

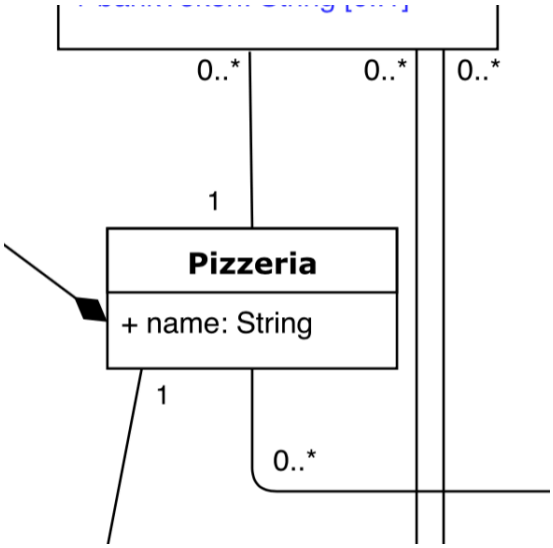
2.3.4 - Classe « Pizzeria »

Cette classe a pour seul attribut *name*.

Elle a des associations avec les classes *OCPizzaUser*, *Address*, *OrderPizza* et *Ingredient*.

Les deux premières associations ont été décrites ci-dessus.

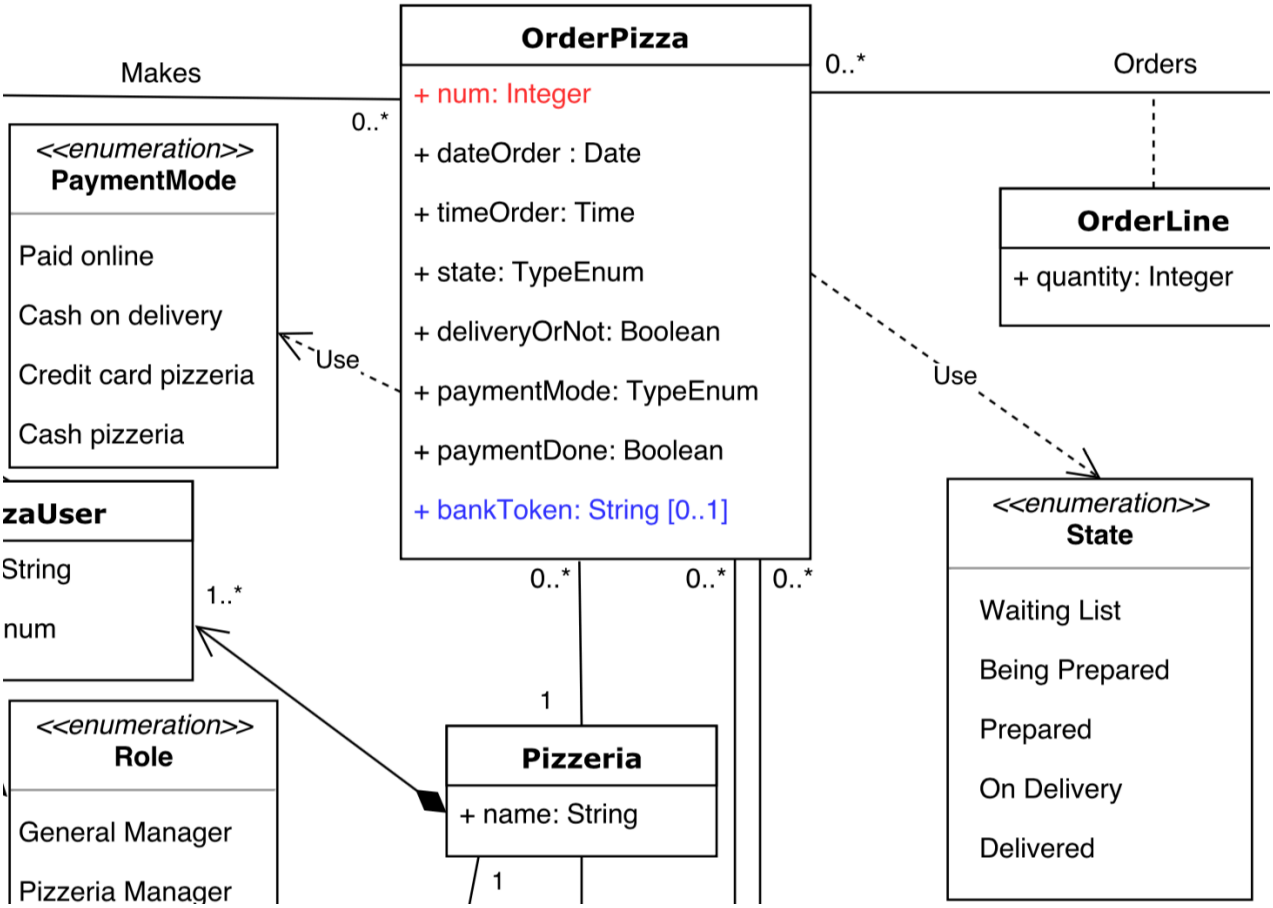
L'association avec ***OrderPizza***, de type ***one-to-many***, relie la commande à une pizzeria qui effectuera ladite commande. Il y a donc aucune ou une infinité de commandes passées à une pizzeria donnée du groupe (cardinalité **0..***) et chaque commande n'est affectée qu'a obligatoirement une seule pizzeria (cardinalité **1**).



Attribut	Description
name	Nom d'une pizzeria du groupe OC Pizza

2.3.5 - Classe « OrderPizza »

Cette classe regroupe les attributs nécessaires pour une commande de pizza(s).
Elle a des associations avec les classes *Buyer*, *Pizzeria*, *Address* et *Pizza*. Les trois premières associations ont été décrites ci-dessus.
L'association avec ***Pizza*** est de type ***many-to-many***, avec la cardinalité **0..***, car pour une pizza il peut y avoir 0 ou une infinité de commandes ; et la cardinalité **1..***, car pour une commande il y a forcément au moins une pizza.



Attribut	Description
num	Numéro de la commande
dateOrder	Jour de validation de la commande
timeOrder	Heure de validation de la commande
state	Etat de la commande en cours (5 possibilités)
deliveryOrNot	Commande à livrer à domicile ou non : true (vrai) ou false (faux)
paymentMode	Mode de paiement (4 possibilités)
paymentDone	Paiement effectué (quelque soit le mode) : true (vrai) ou false (faux)
bankToken	Token renvoyé par la Banque après le paiement par carte bancaire (attribut non obligatoire, car le paiement peut se faire en liquide)

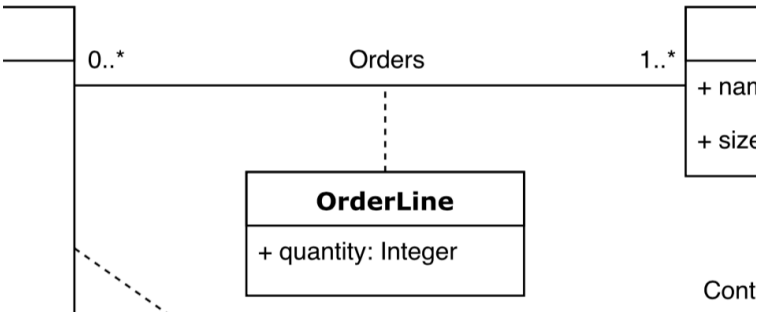
La table <<*enumerate*>> définit les seules valeurs possibles de l'attribut *state* : *Waiting List* (En attente), *Being Prepared* (Préparation en cours), *Prepared* (Commande prête), *On Delivery* (En cours de livraison), *Delivered* (Commande livrée).

La table <<*enumerate*>> définit les seules valeurs possibles de l'attribut *paymentMode* : *Paid online* (Payé en ligne), *Cash on delivery* (En liquide à la livraison), *Credit card pizzeria* (Par carte bancaire à la pizzeria), *Cash pizzeria* (En liquide à la pizzeria).

2.3.6 - Classe « OrderLine »

OrderLine est une **classe d'association**, permettant d'ajouter l'attribut *quantity* à l'association entre les classes ***OrderPizza*** et ***Pizza*** (type ***many-to-many***).

Cette classe permet de comptabiliser pour une commande combien de pizza de tel type font partie du panier d'achat.

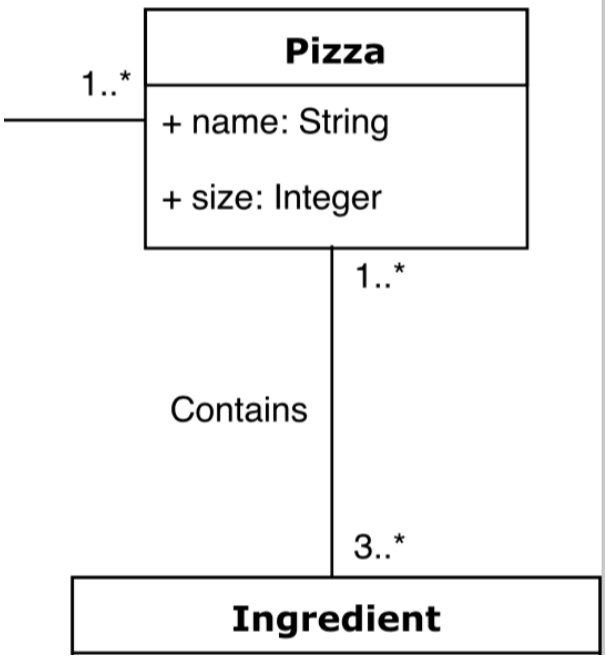


Attribut	Description
quantity	Nombre de pizza commandée pour une pizza donnée

2.3.7 - Classe « Pizza »

Cette classe regroupe les attributs d'une pizza : le *name* et la *size.*
Elle a des associations avec les classes *OrderPizza* et *Ingredient*. La première association a été décrite ci-dessus.
L'association avec ***Ingredient*** est de type ***many-to-many***, avec la cardinalité **1..***, car un ingrédient se retrouve dans au moins une recette de pizza ; et la cardinalité **3..***, car une pizza contient (*Contains*) au moins 3 ingrédients différents.

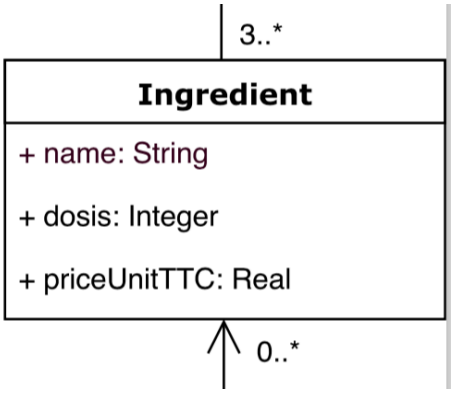
Pour la taille (*size*) des pizzas, la taille « Famille » est le double de la taille « Standard » et aura donc le double de dose de tous les ingrédients.



Attribut	Description
name	Nom de la pizza
size	Taille de la pizza : Standard (1) ou Famille (2)

2.3.8 - Classe « Ingredient »

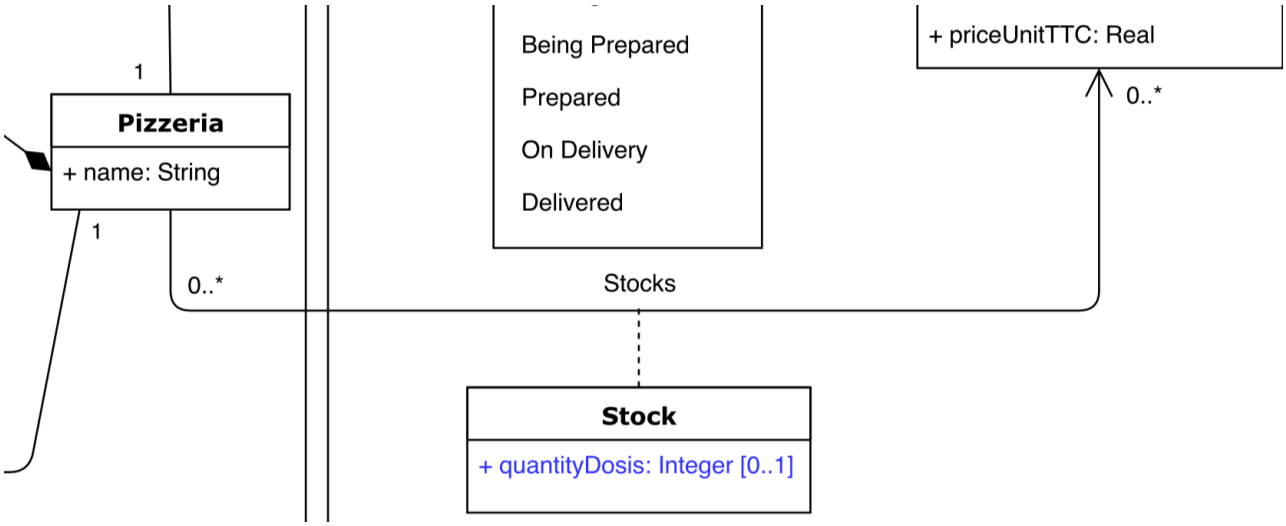
Cette classe regroupe les attributs des ingrédients d'une pizza.
Elle a des associations avec les classes *Pizza* (décrite ci-dessus) et *Pizzeria*.
L'association avec ***Pizzeria*** est de type ***many-to-many*** avec la cardinalité **0..*** dans les deux sens. En effet, chaque pizzeria (5 aujourd'hui mais potentiellement plus dans le futur) stocke (*Stocks*) tous les types d'ingrédients qui sont nécessaires à la préparation des pizzas. Un ingrédient peut n'être dans aucune pizzeria en cas de rupture de stock générale, de même, dans une pizzeria il peut n'y avoir aucun ingrédient.



Attribut	Description
name	Nom de chaque ingrédient (pâte, sauce tomate, jambon, mozzarella…)
dosis	Nombre de dose de chaque ingrédient à mettre sur une pizza donnée
priceUnitTTC	Prix TTC de chaque ingrédient

2.3.9 - Classe « Stock »

Stock est une **classe d'association**, permettant d'ajouter l'attribut *quantityDosis* à l'association entre les classes ***Ingredient*** et ***Pizzeria*** (type ***many-to-many***). Cette classe permet de comptabiliser la quantité de doses de chaque ingrédient dans chaque pizzeria. Cette quantité peut être nulle en cas de rupture de stock.



Attribut	Description
quantityDosis	Quantité de doses d'un ingrédient donné dans chaque pizzeria du groupe

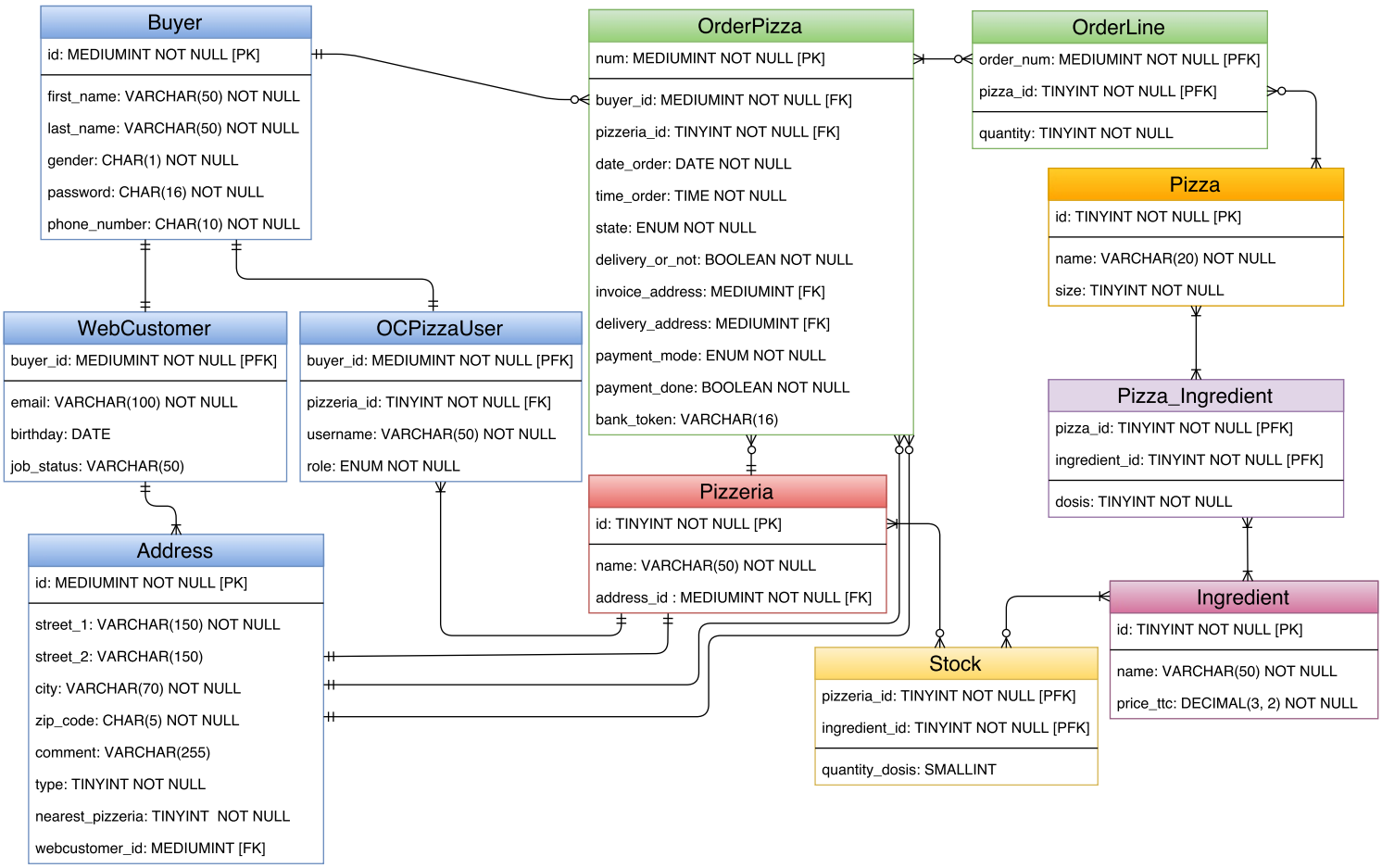
3 - LE MODELE PHYSIQUE DE DONNEES

3.1 - Description

Dérivé du Diagramme de classes UML précédent, le Modèle Physique de Données (*Physical Data Model*) va permettre de modéliser dans le détail la base de données relationnelle OC Pizza.
Nous décrirons les tables et les liens entre-elles, les types de données des différentes colonnes de chaque table et les clés primaires et étrangères.

3.2 - Modèle Physique de Données d'OC Pizza

Physical Data Model OC_Pizza



3.3 - Descriptif des tables

Les noms des colonnes des différentes tables reprennent le nom des attributs de la classe éponyme.

3.3.1 - Table « Buyer »

Cette table regroupe les informations de tous les « Acheteurs », en général.
Toutes les colonnes de cette table doivent être renseignées (**NOT NULL**).
La **clé primaire** est ***id***, avec un **AUTO_INCREMENT**, et de type **MEDIUMINT**.
Pour les autres colonnes, le type est :

- **VARCHAR**, avec **50** caractères pour le ***first_name*** et le ***last_name*** ;
- **CHAR**, avec **10** caractères pour ***phone_number***, **1** pour ***gender*** ('H' ou 'F'), et **16** caractères pour le ***password*** qui sera **encrypté**.

Logique métier :
Dans le cas où c'est un(e) réceptionniste qui entre une commande pour un client qui est dans la pizzeria pour un achat à emporter ou pour un client par téléphone, il devra obligatoirement :

- entrer le *first_name* et le *gender* de la personne ;
- entrer pour *last_name* et pour *password* : « xxx » ;
- entrer pour *phone_number* : « 0000000000 » pour un client de passage ; et bien sûr le bon numéro de téléphone pour un client au téléphone.

3.3.2 - Table « WebCustomer »

Cette table regroupe les informations spécifiques d'une spécialisation des « Acheteurs » que sont les « Clients Internet ».
La colonne ***email*** (**VARCHAR(100**) doit bien sûr être obligatoirement renseignée (**NOT NULL**) contrairement aux colonnes sur la ***birthday*** (**DATE**) et le ***job_status*** (**VARCHAR(50)**). A ce propos, on pourrait définir une liste de choix à sélectionner (à définir avec OC Pizza). Dans ce cas, on mettra un type ENUM avec tous les choix possibles.
La colonne ***buyer_id*** est une **clé étrangère** se référant à l'*id* correspondant dans la table **Buyer**.

3.3.3 - Table « OCPizzaUser »

Cette table regroupe les informations spécifiques d'une spécialisation des « Acheteurs » que sont les employés d'OC Pizza.
Ici, **toutes les colonnes** sont à renseigner (**NOT NULL**).
Le ***username*** est un **VARCHAR(50)**.
Le ***role*** est un **ENUM** avec comme choix : **General Manager**, **Pizzeria Manager**, **Receptionist**, **Pizzaïolo** et **Delivery Man**, représentant les 5 postes possibles au sein du groupe qui interagiront avec le système.
La colonne ***buyer_id*** est une **clé étrangère** se référant à l'*id* correspondant dans la table **Buyer**.

3.3.4 - Table « Address »

Les colonnes ***street_1***, ***city***, ***zip_code***, ***type***, ***nearest_pizzeria*** doivent être renseignées (**NOT NULL**).
Les colonnes *street_2* et *comment* ne sont pas obligatoirement renseignées.
La **clé primaire** est ***id***, avec un **AUTO_INCREMENT**, et de type **MEDIUMINT**.
Pour les colonnes, le type est :

- **VARCHAR**, avec **150** caractères pour ***street_1*** et ***street_2*** ;
- **VARCHAR**, avec **70** caractères pour ***city*** ;
- **CHAR**, avec **5** caractères pour le ***zip_code*** ;
- **VARCHAR**, avec **255** caractères pour ***comment*** ;
- **TINYINT**, pour ***type*** avec 4 possibilités (cf. ci-dessous) ;
- **TINYINT**, pour ***nearest_pizzeria*** reprenant l'*id* de la Pizzeria pour la plus proche ;

Pour *type*, les possibilités sont :

- **1** : pour l'**adresse de facturation et de livraison habituelle** ;
- **2** : pour une **adresse de livraison occasionnelle à enregistrer** ;
- **3** : pour une **adresse de livraison occasionnelle à ne pas enregistrer** ;
- **4** : pour l'**adresse d'une pizzeria du groupe OC Pizza**.

Enfin, la colonne ***webcustomer_id*** est une **clé étrangère** se référant à l'*id* correspondant dans la table **WebCustomer**, afin de relier chaque adresse à un client Internet (celui-ci pouvant avoir plusieurs adresses).

3.3.5 - Table « Pizzeria »

La colonne ***name***, *de type* **VARCHAR**, avec **50** caractères, doit être renseignée (**NOT NULL**).
La **clé primaire** est ***id***, avec un **AUTO_INCREMENT**, et de type **TINYINT**.
La colonne ***address_id*** est une **clé étrangère** se référant à l'*id* correspondant dans la table **Address**, afin de récupérer l'adresse de chaque pizzeria.

3.3.6 - Table « OrderPizza »

Toutes les colonnes, à part *bank_token* (car le paiement n'est pas forcément par carte bancaire), doivent être renseignées (**NOT NULL**).
La **clé primaire** est ***num***, avec un **AUTO_INCREMENT**, et de type **MEDIUMINT**.
Pour les colonnes, le type est :

- **DATE**, pour ***date_order*** ;
- **TIME**, pour ***time_order*** ;
- **ENUM**, pour le ***state*** *avec 5 possibilités (cf. ci-dessous) ;*
- **BOOLEAN**, pour ***delivery_or_not*** *(true ou false si la commande est à livrer à domicile ou non) ;*
- **ENUM**, pour ***payment_mode*** avec 4 possibilités (cf. ci-dessous);
- **BOOLEAN**, pour ***payment_done*** (true ou false si le paiement a été effectué ou non) ;
- **VARCHAR**, avec **16** caractères pour ***bank_token*** qui est une donnée encryptée.

Pour ***state***, les possibilités sont :
- ***Waiting List***, « En attente » ;
- ***Being Prepared***, « Préparation en cours » ;
- ***Prepared***, « Commande prête » ;
- ***On Delivery***, « En cours de livraison » ;
- ***Delivered***, « Commande livrée ».

Pour ***payment_mode***, les possibilités sont :
- ***Paid online***, « Paiement en ligne » ;
- ***Cash on delivery***, « En liquide à la livraison » ;
- ***Credit card pizzeria***, « Par carte bancaire à la pizzeria » ;
- ***Cash pizzeria***, « En liquide à la pizzeria ».

Enfin, il y a **4 clés étrangères** :
- la colonne ***buyer_id*** se référant à l'*id* correspondant dans la table **Buyer**, afin de relier chaque commande à un « Acheteur ».
- la colonne ***pizzeria_id*** se référant à l'*id* correspondant dans la table **Pizzeria**, afin de relier chaque commande à une pizzeria du groupe. Le choix de la pizzeria dépend du résultat du calcul de géolocalisation entre le lieu de livraison et les différentes pizzerias : la plus proche sera choisie.
- la colonne ***invoice_address*** se référant à l'*id* correspondant dans la table **Address**, afin de relier chaque commande à l'adresse de facturation du client (non renseigné pour les ventes à apporter).
- la colonne ***delivery_address*** se référant à l'*id* correspondant dans la table **Address**, afin de relier chaque commande à l'adresse de livraison du client (non renseigné pour les ventes à apporter).

3.3.7 - Table « OrderLine »

La table **OrderLine**, issu de la classe d'association éponyme, est une **table d'association** (association *many-to-many*).
La colonne ***quantity***, de type **TINYINT**, doit être renseignée (**NOT NULL**). C'est la quantité de chaque type de pizza commandée.
Il y a **2 clés étrangères** :
- la colonne ***order_num*** se référant à *num* correspondant dans la table **OrderPizza**, afin de relier chaque ligne de commande à une commande.
- la colonne **pizza_*id*** se référant à l'*id* correspondant dans la table **Pizza**, afin de relier chaque ligne de commande à une pizza.

3.3.8 - Table « Pizza »

La colonne ***name***, de type **VARCHAR**, avec **20** caractères, doit être renseignée (**NOT NULL**).
La colonne ***size***, de type **TINYINT**, doit être renseignée (**NOT NULL**). Elle prend la valeur **1** pour les pizzas de taille « **Standard** » et **2** pour les pizzas de taille « **Famille** ».
La **clé primaire** est ***id***, avec un **AUTO_INCREMENT**, et de type **TINYINT**.

3.3.9 - Table « Ingredient »

La colonne ***name***, de type **VARCHAR**, avec **50** caractères, doit être renseignée (**NOT NULL**).
La colonne ***price_ttc***, *de type* **DECIMAL**, de **3** chiffres dont **2** après la virgule, doit être renseignée (**NOT NULL**).
La **clé primaire** est ***id***, avec un **AUTO_INCREMENT**, et de type **TINYINT**.

3.3.10 - Table « Pizza_Ingredient »

La table **Pizza_Ingredient**, n'apparaît pas dans le Diagramme de classes. Mais c'est une **table d'association** qui doit être créée du fait de l'association *many-to-many* entre les tables **Pizza** et **Ingredient**.
La colonne ***dosis***, de type **TINYINT**, doit être renseignée (**NOT NULL**) et prend la valeur **1** pour les pizzas « **Standard** », et **2** pour les pizzas « **Famille** ». C'est la quantité de dose de chaque ingrédient que nécessite chaque pizza.
Il y a **2 clés étrangères** :

- la colonne ***pizza_id*** se référant à l'*id* correspondant dans la table **Pizza**, afin de relier chaque ingrédient à une pizza donnée.
- la colonne **ingredient_id** se référant à l'*id* correspondant dans la table **Ingredient**, afin de relier chaque pizza à un ingrédient donné.

3.3.11 - Table « Stock »

La table **Stock**, issu de la classe d'association éponyme, est une **table d'association** (association *many-to-many*).
La colonne ***quantity_dosis***, de type **SMALLINT**, doit être renseignée mais peut être de valeur nulle (fin de stock d'un ingrédient). C'est la quantité de doses que chaque pizzeria possède de chaque ingrédient.
Il y a **2 clés étrangères** :

- la colonne ***pizzeria_id*** se référant à l'*id* correspondant dans la table **Pizzeria**, afin de relier chaque pizzeria donnée à un stock d'ingrédient.
- la colonne **ingredient_id** se référant à l'*id* correspondant dans la table **Ingredient**, afin de relier un stock d'ingrédient donné à chaque pizzeria.

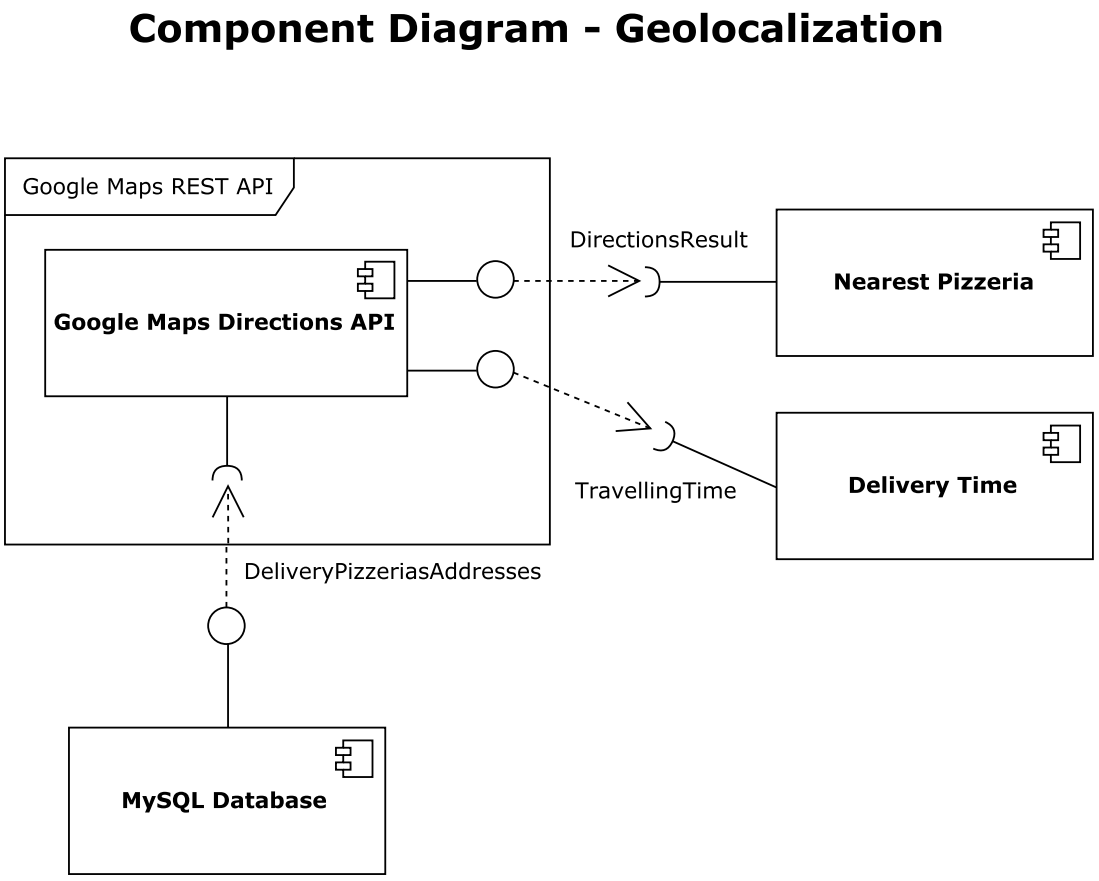
4 - COMPOSANTS INTERNES ET EXTERNES DU SYSTEME

4.1 - Description

Les Diagrammes de composants (*Component Diagrams*) décrivent l'organisation du système du point de vue des éléments logiciels. Ils mettent en évidence les dépendances entre les composants, et décrivent ici les interfaces entre les composants internes du système OC Pizza et les composants externes (Banque, Google Maps).

4.2 - La géolocalisation

4.2.1 - Diagramme de composants - Géolocalisation



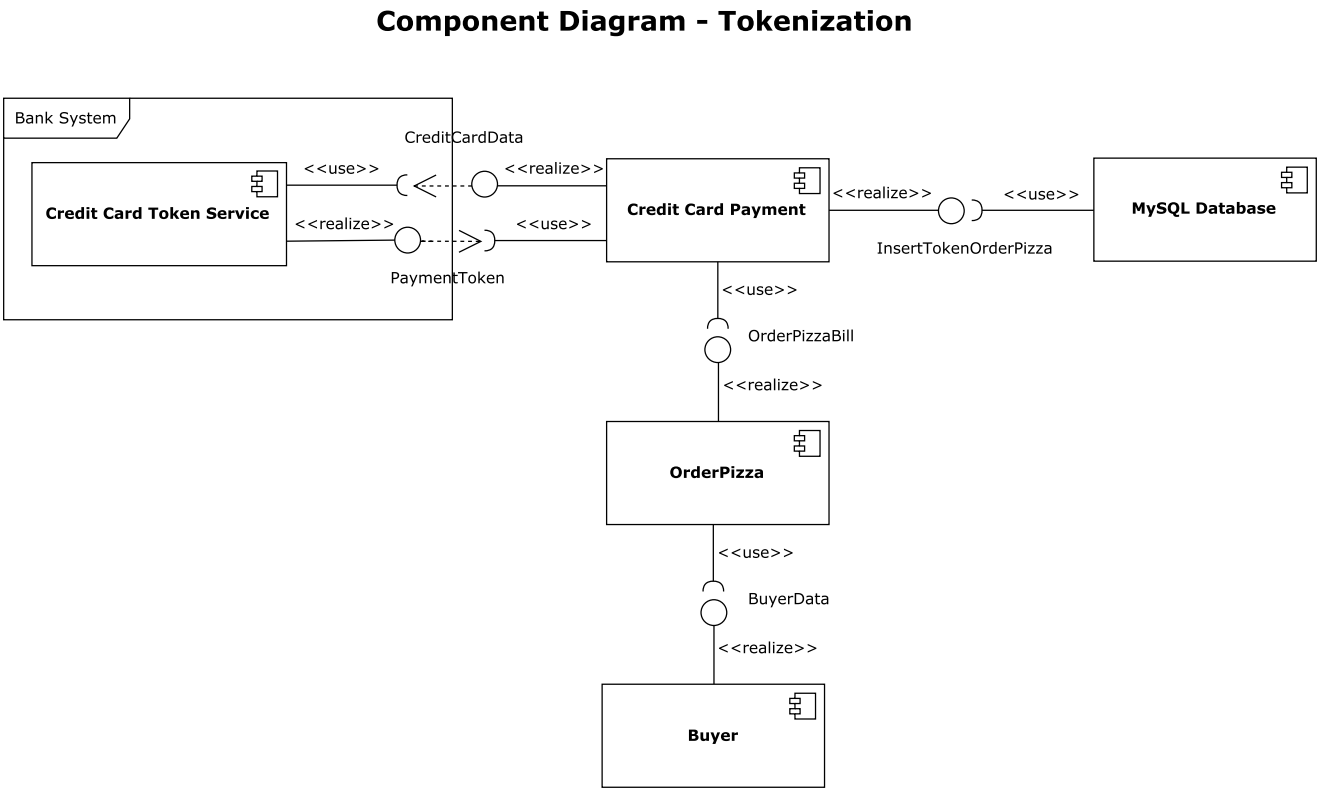
4.2.2 - Descriptif des composants

Le composant **MySQL Database**, par son interface offerte envoie à l'API REST **Google Maps Directions** les informations sur l'adresse de livraison d'un client et l'adresse des différentes pizzerias du groupe OC Pizza. Ce composant externe calcul le temps de parcours et l'itinéraire à privilégier, pour un véhicule, entre cette adresse et celles des pizzerias et envoie l'information à l'interface requise du composant **Nearest Pizzeria** qui va retenir la pizzeria la plus près du lieu de livraison. Cette information (la pizzeria la plus proche) sera stockée dans la base de données dans la colonne *nearest_pizzeria* de la table *Address* pour l'adresse de livraison donnée.

Le composant **Google Maps Directions API** fournit aussi au composant **Delivery Time** le temps prévu du trajet. Celui-ci s'additionnera au temps d'attente puis au temps prévu de fabrication de la commande en cours et fournira une heure prévue de livraison au client, avant qu'il ne valide sa commande.

4.3 - Le paiement en ligne

4.3.1 - Diagramme de composants - Tokenisation



4.3.2 - Descriptif des composants

Le composant **Buyer**, par son interface offerte envoie au composant **OrderPizza** les informations sur la commande et celui-ci calcul la facture totale. Ce montant est requis par le composant **Credit Card Payment** qui récupère aussi le nom du client.
Ce composant va échanger avec le système bancaire via le composant **Credit Card Token Service** des informations sur le client et le montant à régler. Le client aura directement accès au service de carte bancaire de la banque pour rentrer ses coordonnées personnelles (numéro de carte, date d'expiration…).
En retour, ce composant externe renvoie un *token* au composant interne avec l'information cryptée sur la réalisation du paiement.
C'est cette information que gardera en base de données, via le composant **MySQL Database**, le système d'OC Pizza.

5 - ARCHITECTURE DE DEPLOIEMENT

5.1 - Description

Après les Diagrammes de composants, le Diagramme de déploiement UML complète la conception architecturale en identifiant les éléments matériels, physiques, et leurs connexions entre eux ainsi que la disposition des exécutables, les composants, sur ces éléments matériels.

Le Diagramme de déploiement décrit donc la disposition des ressources matérielles, appelés nœuds, qui composent le système et montre la répartition des composants sur ces matériels, ainsi que les chemins de communication entre les nœuds.

5.2 - Matériels des utilisateurs

Le premier nœud, **User's devices,** représentent les matériels (*devices*) utilisés par les acteurs « Acheteurs », de l'application OC Pizza, pour passer une commande. Il y a les clients qui se connectent au site web via leur ordinateur, tablette ou smartphone : ce sont nos « Clients Internet » (***Webcustomer***). Il y a aussi les employés des différentes pizzerias du groupe OC Pizza, les « Réceptionnistes » (***Receptionist***), qui passent commande via le site web pour les clients de passage dans la pizzeria ou les clients par téléphone.

Il y a aussi les employés d'OC Pizza occupant les postes de **Pizzaïolo**, Livreur (***Delivery Man***), Manager général et Manager de pizzeria (***Manager***), qui utilisent le moniteur pour le premier, la tablette pour le deuxième ou l'ordinateur pour les derniers, afin d'interagir avec l'application.

Tous ces utilisateurs utilisent un navigateur web (***Web browser***) qui, via les requêtes **HTTP ou HTTPS**, se connectent au deuxième nœud, le Web Server.

A noter aussi le smartphone des clients, nœud **Buyer's smartphone**, qui va recevoir les SMS de la couche d'application (**Application Layer**), informant des divers événements liés à la commande.

5.3 - Serveur Web et Couche d'application

Le chemin de communication du nœud User's devices le relie au nœud **Web Server**, matériel décentralisé où se situe le site Internet (***Website***) qui utilise comme ressource de backend le **framework Django**.

Ce site est lié à la couche d'application (**Application Layer**) qui, programmée en **Python/Django**, intégrant les logiques métiers (**Business Logic Layer**), et interagissant avec les services externes comme **Google Maps REST API** et le système bancaire (**Bank**), permet de gérer tous les processus des commandes.
Le dernier composant logiciel de ce nœud est l'ORM (**Data Access Layer**) qui permet l'interface avec la base de données.

5.4 - Serveur de Base de Données

Le chemin de communication **TCP/IP** permet à la couche d'application de communiquer avec le nœud serveur de base de données (**Database Server**) qui est situé sur un autre matériel, externe.
Ce serveur héberge la Base de données (*Database*) qui tourne sous **MySQL 5.7.20**, version actuelle.

5.5 - Diagramme de déploiement (Deployment Diagram)