



# OC Pizza

## Spécifications techniques

Alexandre Simões Mendes  
IT Consulting & Development  
Décembre 2019

# Table des matières

---

## **1. Introduction :**

- A. *Contexte*
- B. *Objectifs*

## **2. Le domaine fonctionnel :**

- A. *Le diagramme de classes*
- B. *Descriptif détaillé du diagramme*

## **3. Le modèle physique de données :**

- A. *Le MPD d'OC Pizza*
- B. *Descriptif détaillé des tables*

## **4. Les composants internes et externes du système :**

- A. *Le diagramme de composant*
- B. *Descriptif détaillé du diagramme*

## **5. L'architecture de déploiement**

- A. *Le diagramme de déploiement*
- B. *Descriptif détaillé du diagramme*

## 1. Introduction

---

### A. Contexte

« OC Pizza » est un jeune groupe de pizzeria en plein essor. Créé par Franck et Lola, le groupe est 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 6 mois. Le système informatique actuel ne correspond plus aux besoins du groupe car il ne permet pas une gestion centralisée de toutes les pizzerias. De plus, il est très difficile pour les responsables de suivre ce qui se passe dans les points de ventes. Enfin, les livreurs ne peuvent pas indiquer « en live » que la livraison est effectuée.

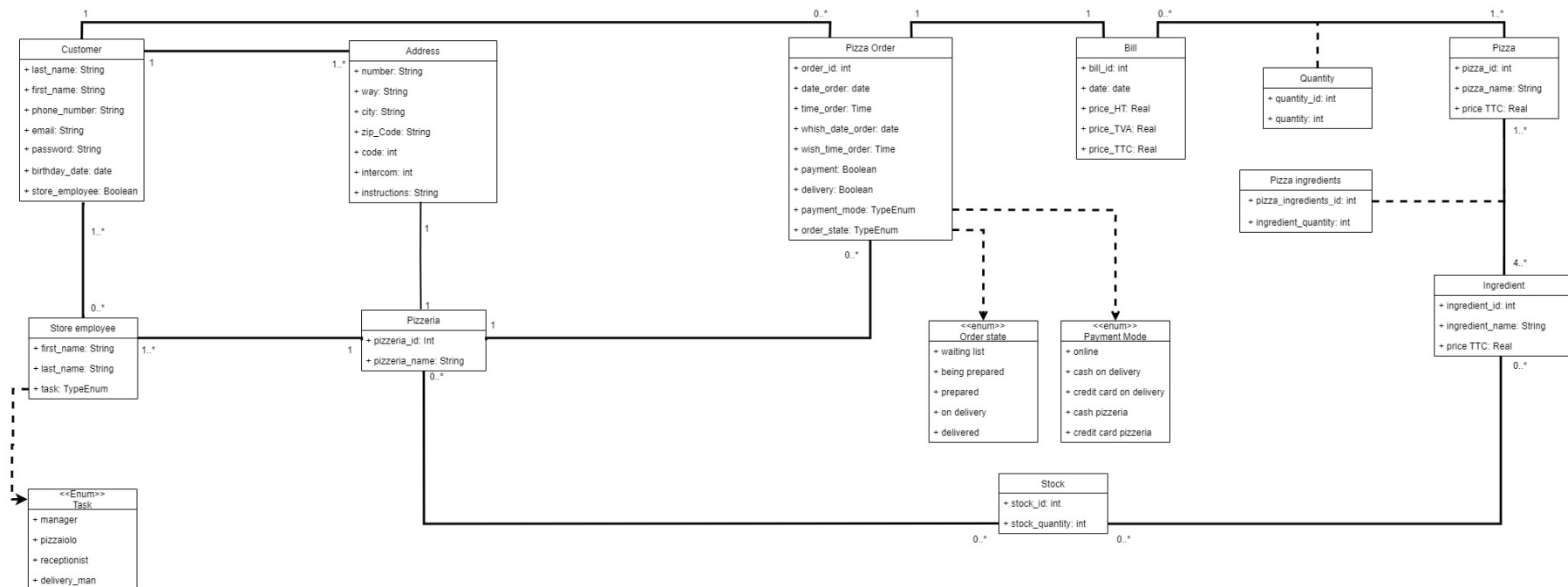
### B. Objectifs

- Être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation
- Suivre en temps réel les commandes passées, en préparation et en livraison
- Suivre en temps réel le stock d'ingrédients restants pour savoir quelles pizzas peuvent encore être réalisées
- 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 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
  - Proposer un aide-mémoire aux pizzaiolos indiquant la recette de chaque pizza

## 2. Le domaine fonctionnel

### A. Le diagramme de classe

Le domaine fonctionnel du système informatique d'OC Pizza présente l'ensemble des classes ainsi que leurs liens. Ces différents éléments serviront de support à la programmation en J2EE et à la création du Modèle Physique de Données (MPD), qui servira à la modélisation de la base de données d'OC Pizza. Le domaine fonctionnel est représenté par un diagramme UML (Langage de Modélisation Unifié, de l'anglais Unified Modeling Language) : le diagramme de classes.



## B. Descriptif détaillé du diagramme

### 1. « Pizza Order », commande.

Attribut	Description
order_id	Numéro unique de la commande du client
date_order	Date de la commande
time_order	Heure de la commande
wish_date	Date souhaitée pour la commande. Attribut non obligatoire.
wish_time	Heure souhaitée pour la commande. Attribut non obligatoire.
payment	L'acheteur a payé la commande (Boolean : True / False)
delivery	L'acheteur a demandé la livraison (Boolean : True / False)
payment_mode	Moyen de paiement (Enumération, 5 possibilités)
order_state	Statut de la commande (Enumération, 5 possibilités)

La classe « Pizza order » regroupe les attributs concernant les principales valeurs d'une commande. Elle est associée à trois différentes classes : « Bill », « Customer » et « Pizzeria ».

L'association avec « Bill » est de type one-to-one, avec une cardinalité de « 1 » car pour une commande il ne peut y avoir qu'une facture ; l'autre cardinalité est, elle aussi, de « 1 », car pour une facture il y ne peut y avoir uniquement qu'une seule commande. Les autres relations seront détaillées plus loin dans le document, afin d'éviter de trop nombreuses répétitions.

### A. « <<Enumerate>> Payment\_mode », liste des paiements possible chez OC Pizza

Attribut	Description
Online	En ligne (Paypal, CB)
Cash on delivery	En liquide à la livraison
Credit card on delivery	Par carte à la livraison
Cash pizzeria	En liquide en magasin
Credit card pizzeria	Par carte en magasin

La table <<Enumerate>> donne les cinq valeurs possibles pour l'attribut « payment\_mode », décrites dans le tableau ci-dessus.

### B. « <<Enumerate>> Order state », état de la commande.

Attribut	Description
Waiting list	En attente
Being prepared	En préparation
Prepared	Préparation terminée
On delivery	En livraison
Delivered	Commande livrée

La table <<Enumerate>> donne les cinq valeurs possibles pour l'attribut « order\_state », décrites dans le tableau ci-dessus.

2. « Bill », la facture.

Attribut	Description
bill_id	Numéro unique de la facture
date	Date d'émission de la facture
price_HT	Prix hors taxes de la commande
Price_TVA	Prix TVA de la commande
Price_TTC	Prix toutes taxes comprises de la commande

L'association avec la classe « Pizza » est de type « many-to-many », avec une cardinalité de « 0..\* », en effet, pour une pizza donnée, il peut n'y avoir aucune commande, ou au contraire, il peut y en avoir une infinité. De l'autre côté nous retrouvons une cardinalité de « 1..\* », puisque dans une commande il y a nécessairement une pizza au minimum, qui est sélectionnée par le client.

3. « Quantity », la quantité de pizza.

Attribut	Description
quantity_id	Numéro d'identification propre à chaque « quantité » demandée pour chaque client, et chaque pizza
quantity	Quantité désirée pour un type de pizza

« Pizza quantity » est une classe d'association, qui nous permet d'utiliser l'attribut « quantity » entre les classes « Bill » et « Pizza » (de type many-to-many, comme précisé plus haut) et donc de dénombrer, de comptabiliser la quantité d'une pizza, du même type, qui fait partie d'une commande (Ex : un client peut commander « x » quantité d'une pizza de type « reine »).

4. « Pizza », les différentes pizzas de l'enseigne.

Attribut	Description
pizza_id	Numéro d'identification propre à chaque pizza
pizza_name	Nom de la pizza, celui affiché sur le menu
price_TTC	Prix toutes taxes comprises de la pizza, affiché sur le menu

La classe « pizza » permet de lister les pizzas proposées par OC Pizza. Nous récupérons dans cette table son numéro d'identification, son nom mais aussi son prix. Elle est liée avec les classes « Bill », déjà décrite plus haut, mais aussi avec la classe « Ingredient » de type « many-to-many ». En effet, une pizza contient obligatoirement 4 ingrédients au minimum, donc une cardinalité de « 4..\* », tandis qu'un ingrédient est nécessaire au moins dans une recette, donc une cardinalité de « 1..\* ». Notons que l'attribut « size » permet d'établir la quantité nécessaire d'ingrédients dans la pizza, et que l'attribut « price\_TTC » permet quant à lui de renseigner prix de la pizza.

5. « Pizza Ingredients », la liaison entre une pizza et ses ingrédients.

Attribut	Description
pizza_ingredient_id	Numéro d'identification
Ingredient_quantity	Quantité à mettre sur la pizza

« Pizza ingredient » est une classe d'association qui nous permet d'utiliser l'attribut « pizza\_ingredient\_id » entre les classes « pizza » et « ingredient » (de type many-to-many, comme précisé plus haut) et donc de lister les ingrédients nécessaires à une pizza donnée.

6. « Ingredient », les ingrédients disponibles en magasin.

Attribut	Description
ingredient_id	Numéro de l'ingrédient
ingredient_name	Nom de l'ingrédient
price_TTC	Prix toutes taxes comprises de l'ingrédient

Cette classe regroupe les attributs des ingrédients d'une pizza. Elle a des associations avec les classes « Pizza », déjà décrites plus haut, et la classe « Stock ». L'association avec « Stock » est de type many-to-many avec la cardinalité 0..\* dans les deux sens car un ingrédient peut ne pas être disponible, comme il peut n'y avoir aucun ingrédient de la recette disponible.

7. « Stock », le stock d'ingrédients disponibles

Attribut	Description
stock_id	Numéro du stock
quantity	Quantité disponible d'ingrédient

La classe « Stock » permet de connaître en temps réel le stock disponible pour chaque ingrédient, dans chaque pizzeria. L'association avec « Pizzeria » est de type « many-to-many » avec la cardinalité « 0..\* » dans les deux sens, car là aussi, un stock d'ingrédient peut être en rupture uniquement dans une seule pizzeria, mais ce même stock peut être en rupture de stock dans toutes les pizzerias.

## 8. « Pizzeria »

Attribut	Description
pizzeria_id	Numéro d'identification de la pizzeria
Pizzeria_name	Nom de la pizzéria

Dans cette classe, il n'y a que deux attributs : « pizzeria\_id » qui renseigne un numéro d'identification et « pizzeria\_name » qui renseigne le nom de la pizzeria. Elle a plusieurs associations, avec les classes « Stock », « Address », « Pizza order » et « Store employee ». L'association avec la classe « Pizza order » est de type « one-to-many ». En effet, une pizzeria peut se voir attribuer une quantité innombrable de commande (« 0..\* »), alors qu'une commande ne peut être attribuée qu'à une seule pizzeria (« 1 »). L'association avec la classe « Stock » a déjà été explicitée, les deux autres le seront plus loin dans le document.

## 9. « Address »,

Attribut	Description
street_number	Numéro de voie
way	Voie (Rue, Boulevard, Avenue etc)
city	Ville
zip_code	Code postal
code	Code d'entrée. Attribut non obligatoire.
intercom	Nom ou numéro pour l'interphone. Attribut non obligatoire.
instructions	Consignes laissées par l'acheteur. Attribut non obligatoire.

Cette classe regroupe les divers attributs concernant les adresses, soit des clients des différentes pizzerias du groupe OC pizza, soit l'adresse de ces dernières. L'association avec la classe « Pizzeria » est de type « one-to-one ». En effet, une pizzeria n'a qu'une adresse (« 1 »), et une adresse ne correspond qu'à une pizzeria (« 1 »). Elle possède aussi une association de type « one-to-many » avec « Customer ». En effet, une adresse ne correspond qu'à un client (« 1 »), alors qu'un client peut avoir renseigné plusieurs adresses (« 1..\* »).



10. « Customer », client.

Attribut	Description
last_name	Nom de famille de l'acheteur
first_name	Prénom de l'acheteur
phone_number	Numéro de téléphone de l'acheteur
email	Adresse email de l'acheteur (identifiant de connexion)
password	Mot de passe du compte de l'acheteur
birthday_date	Date d'anniversaire de l'acheteur. Attribut non obligatoire.
employee	L'acheteur a-t-il passé commande à travers un employé du magasin (Boolean : True / False)

Cette classe regroupe les attributs concernant les principales informations clients. La classe « Customer » est liée à la classe « Store employee ». En effet, si un client passe une commande à travers un employé du magasin, que ce soit lors d'une visite en magasin ou par téléphone, cela permet de savoir quel est le salarié qui l'a pris en charge. Cela se traduit par une association de type « many-to-many » ; en effet un employé du magasin peut n'avoir pris aucune commande en charge à la pizzeria, ou au contraire, il peut avoir dû prendre une quantité infini de commande (« 0..\* »), alors qu'une commande peut n'avoir été prise en charge par aucun salarié en magasin, par exemple lors d'une commande sur internet (« 1 »).

« Customer » est aussi en relation avec la classe « Pizza order », par une association de type « one-to-many » ; un client peut ne passer aucune commande, ou au contraire, une quantité infini de commande (« 0..\* »), alors qu'une commande ne possède qu'un seul et unique client (« 1 »)

11. « Store employee », employé du magasin.

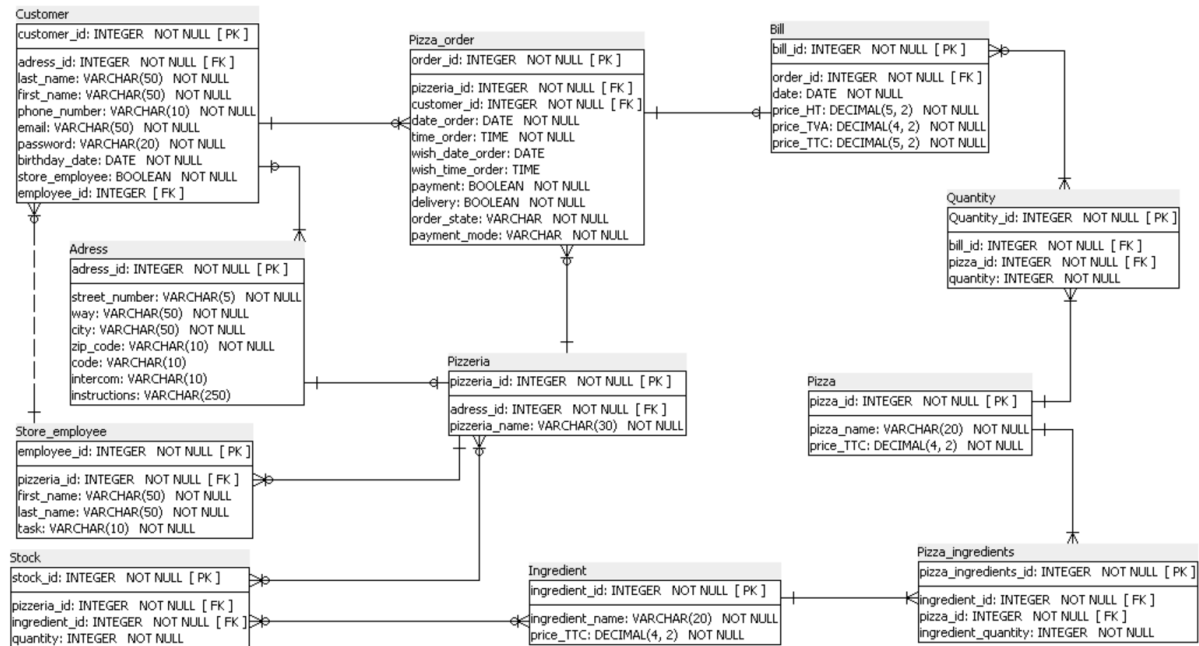
Attribut	Description
last_name	Nom de famille de l'employé du magasin
first_name	Prénom de l'employé du magasin
task	Poste de l'employé au sein du magasin

Cette classe regroupe les informations concernant les employés du restaurant. Cette classe n'est sollicitée que si la commande d'un client, qu'elle soit par téléphone ou à la caisse de la boutique, est prise en charge par un employé d'OC Pizza. Cela permet de me suivre la prise en charge d'une commande et pouvoir mieux tracer une commande.

12. « <<Enumerate>> Task », poste de l'employé.

Attribut	Description
Manager	Manager du magasin
Pizzaiolo	Pizzaiolo du magasin
Receptionist	Vendeur du magasin
Delivery man	Livreur

La table <<Enumerate>> donne les quatre valeurs possibles pour l'attribut « Task », décrites dans le tableau ci-dessus.



## B. Descriptif détaillé des tables

Table « Pizza order »

Pizza_order	
order_id: INTEGER	NOT NULL [ PK ]
pizzeria_id: INTEGER	NOT NULL [ FK ]
customer_id: INTEGER	NOT NULL [ FK ]
date_order: DATE	NOT NULL
time_order: TIME	NOT NULL
wish_date_order: DATE	
wish_time_order: TIME	
payment: BOOLEAN	NOT NULL
delivery: BOOLEAN	NOT NULL
order_state: VARCHAR	NOT NULL
payment_mode: VARCHAR	NOT NULL

Presque toutes les colonnes de cette table doivent être renseignées (« NOT NULL »), les seules exceptions étant « wish\_date\_order » (date souhaitée de livraison) et « wish\_time\_order » (heure souhaitée de livraison) .

Nous retrouvons au sein de cette table l'attribut « order\_id », qui est la clé primaire de cette table. Cette clé primaire servira de clé unique qui ne correspondra uniquement qu'à une commande. Ainsi, chaque commande ne possèdera qu'un numéro d'identification. Le type de cet attribut est « INTEGER ».

L'attribut « date\_order », de type « DATE », nous permet quant à lui de définir la date à laquelle la commande a été passée par le client au format « YYYY-MM-DD ».

L'attribut « wish\_date », est de même type, mais n'est utilisée que si le client le souhaite, dans le cas d'une livraison qui n'est pas immédiate dans la journée, voir qui n'est pas du tout pour le jour même. Il va de pair avec « wish\_time » qui permet, lui, de définir l'heure à laquelle le client souhaite recevoir sa commande. « wish\_time », est de type « TIME », comme « time\_order », qui nous permet de définir l'heure à laquelle la commande a été passée par le client, au format « HH:mm:ss »

Nous avons aussi au sein de cette table, l'attribut « payment », qui est de type « BOOLEAN ». Celui-ci permet de connaître si oui ou non, la commande d'un client a déjà été réglée. Les seules réponses possibles pour cet attribut sont donc « TRUE » ou « FALSE » (vrai ou faux). Un autre attribut de ce type fait partie de cette table, « delivery », qui permet quant à lui de savoir si le client souhaite une livraison ou un retrait en magasin. Là aussi, deux uniques réponses sont possibles « TRUE » ou « FALSE ».

Les deux derniers attributs de la table « Pizza order » ont la particularité de renvoyer tous les deux à des énumérations. En effet, « order\_state » qui permet de connaître l'état de la commande à un instant « t » et « payment\_mode » qui lui, nous renseigne sur le moyen de paiement sélectionné par le client, renvoient tous les deux aux tableaux « <<Enumerate>> Order State » et « <<Enumerate>> Payment Mode » vu un peu plus tôt dans ce document. Les deux attributs sont donc de type « VARCHAR », configuré avec une capacité de 20 caractères.

## Table « Customer »

Customer	
customer_id: INTEGER	NOT NULL [ PK ]
adress_id: INTEGER	NOT NULL [ FK ]
last_name: VARCHAR(50)	NOT NULL
first_name: VARCHAR(50)	NOT NULL
phone_number: VARCHAR(10)	NOT NULL
email: VARCHAR(50)	NOT NULL
password: VARCHAR(20)	NOT NULL
birthday_date: DATE	NOT NULL
store_employee: BOOLEAN	NOT NULL
employee_id: INTEGER	[ FK ]

Dans cette table aussi, presque toutes les colonnes doivent être renseignées (« NOT NULL »), la seule exception étant « employee\_id » (numéro d'identification de l'employé) qui n'est utilisé que si le client est passé par un employé du magasin pour passer sa commande.

La clé primaire de cette table est « customer\_id », un numéro d'identification unique pour chaque client du restaurant. Notons, que dans cette table, nous avons aussi trois clés étrangères : « order\_id », « adress\_id » et « employee\_id ». Voyons plus en détails ces trois clés ci.

L'attribut « order\_id », est une clé étrangère qui permet d'associer un client avec un numéro de commande établi auparavant dans la table « Pizza order ». La deuxième clé étrangère, comme nous l'avons vu, est « adress\_id », qui permet d'attribuer à un client une ou plusieurs adresses, qu'il aura préalablement défini sur son compte ou en magasin. Cette clé, permet de faire le lien avec la table « Address », que nous verrons plus en détails plus tard. La dernière clé étrangère est donc « employee\_id », qui permet d'associer la classe « Customer » avec la classe « Store\_employee ». Ces trois attributs sont de type « INTEGER ». Revenons quelques instants sur cette dernière clé étrangère. Elle a la particularité de n'être utilisée que si l'attribut « Store\_employee », de type « BOOLEAN » s'est vu attribuer une valeur « TRUE ». Ainsi, si la commande est prise par un employé du magasin et non passée par internet, nous pouvons connaître la personne étant intervenu dans le processus de commande. Bien entendu, nous retrouvons ici des attributs qui permettent de renseigner des informations sur le client, comme « last\_name », qui permet de récupérer le nom de famille du client, mais aussi comme « first\_name », qui permet lui de connaître le prénom d'un client. Les deux sont de types « VARCHAR », avec une capacité totale de 50 caractères chacun, afin de permettre aux clients de renseigner le mieux possible leurs informations personnelles. Toujours concernant les informations personnelles, l'attribut « birthday\_date », de type « DATE » permettra de recueillir la date d'anniversaire de la personne, afin de lui transmettre des offres ou des promotions la semaine de cet évènement, par exemple.

Pour ce qui est des informations de contact, l'attribut « email » nous est utile afin de récupérer le mail de l'utilisateur. Son type est de « VARCHAR » avec une capacité de 50 caractères, ce qui nous semble l'idéal dans l'optique où un utilisateur aurait utilisé tous ses noms et prénoms pour composer son adresse mail. L'attribut « phone\_number » permet quant à lui de pouvoir avoir un numéro de téléphone de contact, que ce soit pour le magasin ou le livreur. Son type est « VARCHAR » là aussi, mais avec une capacité de 10 caractères, suffisant que ce soit pour un numéro fixe ou mobile.

Enfin, l'attribut « password », de type « VARCHAR » lui aussi, mais avec une capacité de 20 caractères, permettra de récupérer le mot de passe défini par l'utilisateur.

## Table « Address »

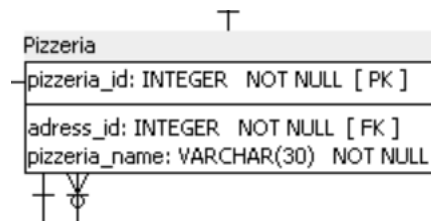
Adress	
adress_id: INTEGER	NOT NULL [ PK ]
street_number: VARCHAR(5)	NOT NULL
way: VARCHAR(50)	NOT NULL
city: VARCHAR(50)	NOT NULL
zip_code: VARCHAR(10)	NOT NULL
code: VARCHAR(10)	
intercom: VARCHAR(10)	
instructions: VARCHAR(250)	

Dans cette table aussi, presque toutes les colonnes doivent être renseignées (« NOT NULL »), les seules exceptions étant « code », « intercom » et « instructions »

La clé primaire de cette table est « adress\_id », un numéro d'identification unique pour les adresses renseignées. Notons, que dans cette table, nous avons aussi une clé étrangère : « pizzeria\_id », qui permet d'associer une pizzeria à une adresse. Celle-ci est de type « VARCHAR ».

Pour éviter de lourdes répétitions, partons du postulat que les prochains attributs concernent aussi bien le domicile du client que celui d'un des restaurants. L'attribut « street\_number » permet de pouvoir renseigner le numéro de l'habitation, « way » quant à lui permet de renseigner la voie, « city » lui la ville et finalement « zip\_code », le code postal. Avec ces quatre attributs, tous de type « VARCHAR » nous avons déjà de quoi pouvoir récupérer les principales informations concernant une adresse. Pour permettre une livraison à domicile, dans le cas où cela est nécessaire, nous pouvons récupérer le « code » d'entrée l'immeuble, l'« intercom » ou interphone, que ce soit un nom ou un numéro, mais surtout « instructions » qui permet de saisir des instructions précises concernant la livraison. Là aussi, tous sont de type « VARCHAR » ; notons que les instructions peuvent contenir jusqu'à 255 caractères, ce qui laisse la possibilité à l'utilisateur d'être le plus précis possible.

## Table « Pizzeria »



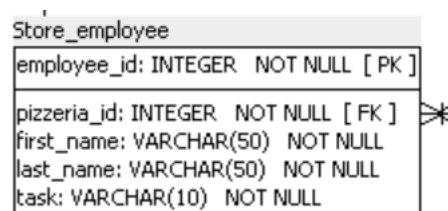
Toutes les colonnes doivent être renseignées (« NOT NULL ») pour cette table.

La clé primaire de cette dernière est l'attribut «pizzeria\_id» qui permet à la table « Pizzeria » d'être associée à la table « Address ». Ainsi, lorsqu'un client passe une commande, selon son adresse, le restaurant le plus proche de chez lui, lui sera attribué.

Cette table possède aussi une clé étrangère, « order\_id » de type « INTEGER » qui renvoie au numéro de commande attribué par la table « Pizza order » .

Enfin, « pizzeria\_name » permet d'avoir la raison sociale de l'établissement concerné, son type est donc évidemment de type « VARCHAR », avec une capacité de trente caractères, afin de parer toute éventualité.

## Table « Store Employee »



Pour cette table aussi, toutes les colonnes doivent être renseignées (« NOT NULL »).

La clé primaire de « Store\_employee » est « employee\_id », un « INTEGER » qui permet d'attribuer un numéro unique à chaque employé du réseau OC Pizza. Une seule clé étrangère est ici présente, « pizzeria\_id » qui indique pour un numéro d'employé donné, la pizzeria qui l'emploie. Nous avons ici affaire avec un attribut de type « VARCHAR ».

Les attributs « last\_name » et « first\_name », comme dans la table « Customer », permettent de récupérer le nom et prénom, mais ici, d'un employé, en fonction de son numéro d'identification. Les deux attributs sont de type « VARCHAR » avec une limite de 50 caractères.

Enfin, l'attribut « task », permet de connaître la fonction de l'employé du magasin. Il renvoie à une énumération de quatre possibilités, comme vu lors du chapitre précédent.

### Table « Bill »

Bill	
bill_id: INTEGER NOT NULL [ PK ]	
order_id: INTEGER NOT NULL [ FK ]	
date: DATE NOT NULL	
price_HT: DECIMAL(5, 2) NOT NULL	
price_TVA: DECIMAL(4, 2) NOT NULL	
price_TTC: DECIMAL(5, 2) NOT NULL	

Toutes les colonnes doivent être renseignées (« NOT NULL ») pour cette table aussi.

La clé principale de cette table est « bill\_id » qui permet de connaître le numéro de la facture, un numéro unique. Son type est donc logiquement un « INTEGER ». Type partagé avec la clé étrangère, « order\_id » qui renvoie au numéro de la commande à laquelle fait référence la facture. L'attribut « date » permet de connaître la date à laquelle a été établie la facture par le magasin, son type est donc « DATE ».

Enfin, pour établir une facture, connaître les différents prix est une obligation ; ainsi, grâce à « Price\_HT » qui permet de connaître le prix total hors taxe, « Price\_TVA » qui renseigne le prix de la TVA sur la commande, et « Price\_TTC » qui renseigne le montant total de la commande nous pouvons récupérer les informations nécessaires. Les trois attributs sont de même type, « DECIMAL ».

### Table « Quantity »

Quantity	
quantity_id: INTEGER NOT NULL [ PK ]	
bill_id: INTEGER NOT NULL [ FK ]	
pizza_id: INTEGER NOT NULL [ FK ]	
quantity: INTEGER NOT NULL	

Toutes les colonnes doivent être renseignées (« NOT NULL »).

Cette table permet d'établir pour une commande client, la quantité demandée pour une pizza. Ainsi, sa clé primaire n'est autre que « quantity\_id », et comme tous les numéros d'identifications ici, il est de type « INTEGER ». Les deux clés étrangères de cette table permettent de récupérer le numéro d'identification unique et donc la facture du client (« bill\_id ») et le numéro d'identification des pizzas demandées par le client.

Enfin, l'attribut « quantity » permet de connaître exactement la quantité pour chaque pizza d'une commande.



### Table « Pizza »

Pizza	
pizza_id: INTEGER NOT NULL [ PK ]	—
pizza_name: VARCHAR(20) NOT NULL	—
price_TTC: DECIMAL(4, 2) NOT NULL	

Toutes les colonnes doivent être renseignées (« NOT NULL »).

Pour la table « Pizza », la clé primaire est « pizza\_id » qui permet de connaître le numéro d'identification unique de la pizza. Il n'y a pas de clé étrangère au sein de cette table, qui permet de lister l'ensemble des pizzas proposés par OC Pizza.

Nous retrouvons aussi au sein de cette table, l'attribut « pizza\_name », de type « VARCHAR » qui nous permet de connaître le nom d'une pizza, tel qu'il est indiqué sur le menu, tant sur internet, que sur les divers supports du restaurant.

Enfin, nous avons aussi l'attribut « Price\_TTC », qui permet de connaître le prix de la pizza.

### Table « Ingredient »

Ingredient	
ingredient_id: INTEGER NOT NULL [ PK ]	—
*ingredient_name: VARCHAR(20) NOT NULL	
price_TTC: DECIMAL(4, 2) NOT NULL	

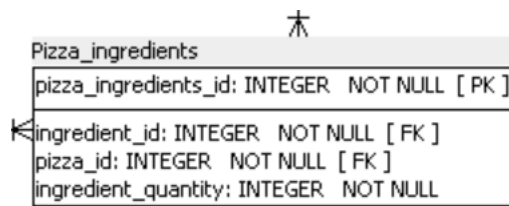
Toutes les colonnes doivent être, là aussi, renseignées (« NOT NULL »).

Pour la table « Ingrédient », la clé primaire est « ingredient\_id » qui permet d'attribuer un numéro d'identification unique pour un ingrédient, pour une pizza lors d'une commande. Nous n'avons aucune clé étrangère au sein de cette table, qui permet de lister l'ensemble des ingrédients dont dispose chaque magasin OC Pizza.

Nous retrouvons aussi au sein de cette table, l'attribut « ingredient\_name », de type « VARCHAR » qui nous permet de connaître le nom d'un ingrédient, tel qu'il est indiqué sur le menu, tant sur internet, que sur les divers supports du restaurant.

Enfin, nous avons aussi l'attribut qui permet de connaître le prix de cet ingrédient à travers « price\_TTC ».

### Table « Pizza Ingredients »

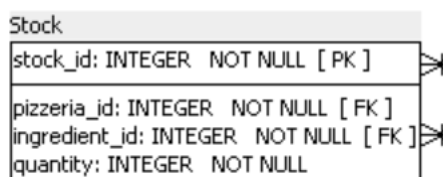


Pizza_ingredients	
pizza_ingredients_id: INTEGER NOT NULL [ PK ]	
ingredient_id: INTEGER NOT NULL [ FK ]	
pizza_id: INTEGER NOT NULL [ FK ]	
ingredient_quantity: INTEGER NOT NULL	

Toutes les colonnes doivent être renseignées (« NOT NULL »).

La clé primaire de cette table est là aussi un numéro d'identification, ici, « pizza\_ingredients\_id ». Le reste de la table est composé uniquement de deux clés étrangères, « pizza\_id » et « ingredient\_id », permettant d'associer à une pizza, les ingrédients nécessaires à sa préparation. Enfin, « ingredient\_quantity » permet de connaître la quantité nécessaire d'un ingrédient. Tous les attributs sont ici des « INTEGER ».

### Table « Stock »



Stock	
stock_id: INTEGER NOT NULL [ PK ]	
pizzeria_id: INTEGER NOT NULL [ FK ]	
ingredient_id: INTEGER NOT NULL [ FK ]	
quantity: INTEGER NOT NULL	

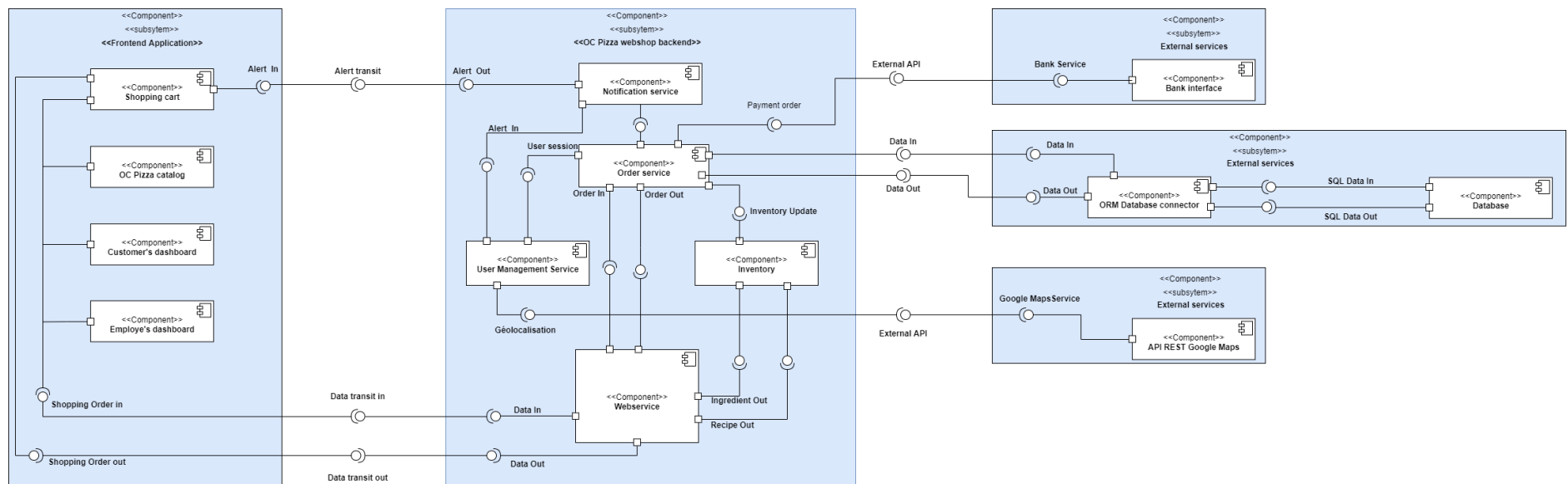
Toutes les colonnes doivent être renseignées (« NOT NULL »)

La clé primaire de cette table est « stock\_id », un « INTEGER » qui est aussi un numéro d'identification. Les deux clés étrangères de la table sont aussi de ce type, que ce soit « pizzeria\_id » ou « ingredient\_id ». Idem pour « quantity » qui permet de connaître la quantité disponible d'un ingrédient (grâce à « ingredient\_id » ) dans une pizzeria donnée (grâce à « pizzeria\_id »).

## Les composants internes et externes du système

### A. Le diagramme de composant

Les diagrammes de composants (Component Diagrams en anglais) décrivent l'organisation du système du point de vue des éléments logiciels. Ils mettent en avant 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 et dans notre cas une base de données hébergée sur un serveur distant à travers une offre de l'entreprise OVH).



### *Le descriptif détaillé du diagramme*

La gestion des données se fera grâce à une base de données (SGBD) stockée sur un cloud, Pour l'ensemble de ces solutions (server et SGBD) nous passerons par l'intermédiaire d'un prestataire : OVH. Leurs nombreuses offres groupées permettent d'obtenir les outils nécessaires quant à la mise en place de la solution proposée et cela à un coût attractif, notamment un partenariat avec Microsoft pour la mise en place d'une base de données, avec MS SQL Server, ainsi que la possibilité de bénéficier des services nécessaires en cloud, ce qui présente de nombreux avantages. Le « Webshop » communiquerait ainsi avec une base de données à distance, en cloud. La solution du cloud est privilégiée car elle permet un gain de place non négligeable, mais surtout d'éviter tous les problèmes que pourraient subir des serveurs dit « physiques », comme les pannes, les incidents (incendies, inondations ou tout autre problème du même type) qui pourraient causer la perte des données. Le cloud permet aussi de mieux gérer l'afflux de clients sur le site internet, et donc d'éviter tout incident relatif à un trop plein de visiteurs.

Pour la gestion des paiements en ligne, nous passerons là aussi par une API externe, celui de la banque avec laquelle OC Pizza travaille. Le client remplira d'abord un formulaire afin de renseigner toutes les informations nécessaires afin de procéder au règlement, en ligne, par carte bancaire, comme le numéro de carte, le type de carte, le nom, la date d'expiration et le cryptogramme. Et cela à partir du site internet ou de l'application OC Pizza. L'API externe communiquera ainsi les informations renseignées par l'utilisateur à la banque partenaire, qui validera ou non, instantanément, le paiement. Si ce dernier est validé, la commande l'est aussi.

L'utilisation de l'API REST Google MAPS quant à elle permet de géolocaliser un client. Ainsi, lors de son inscription sur le site internet, ou en magasin, lorsque l'adresse de ce dernier est renseignée, l'API permet de connaître le magasin le plus proche de son domicile et le client est donc « rattaché » à celui-ci. Cette API est aussi pratique pour les livreurs, qui pourront consulter l'itinéraire, le temps de trajet jusqu'au domicile du client, et même se servir de ce service sur leur téléphone portable comme GPS à travers l'interface de leur application. Ces informations sont aussi complétées par le temps de trajet estimé, qui s'ajoute alors au temps de préparation de la commande, ce qui permet au final d'estimer un temps de livraison dont le client sera informé sur son « tracker » disponible sur son espace personnel, sur la page de la commande concernée.

Le Frontend de l'application ou du site internet est composé de quatre éléments distincts. D'abord le « Shopping cart », qui n'est autre que le panier de l'utilisateur. Ce dernier peut ainsi décider d'ajouter ou supprimer des produits au sein de son panier. Lorsque le panier est validé, une alerte est envoyée au composant « Notification Service », faisant partie du backend.

Nous retrouvons aussi « OC Pizza catalog », permet d'afficher l'ensemble du catalogue des restaurants OC Pizza sur le site web ou l'application mobile. C'est à partir du catalogue que le client va pouvoir constituer son panier, où que le vendeur va pouvoir constituer un panier pour un client. Ensuite nous retrouvons le « Customer's dashboard », qui n'est autre que le tableau de bord du client. C'est à travers ce composant que l'utilisateur peut modifier ses informations personnelles préalablement renseignées lors de son inscription, consulter son historique de commande, suivre la préparation de sa commande etc.

Enfin, le « Employee's dashboard » quant à lui, est le tableau de bord des différents employés du magasins. C'est à travers ce composant qu'ils peuvent consulter les différentes commandes passées par les clients, suivre la préparation d'une commande, retrouver un compte client, consulter les informations d'un client et les modifier etc.

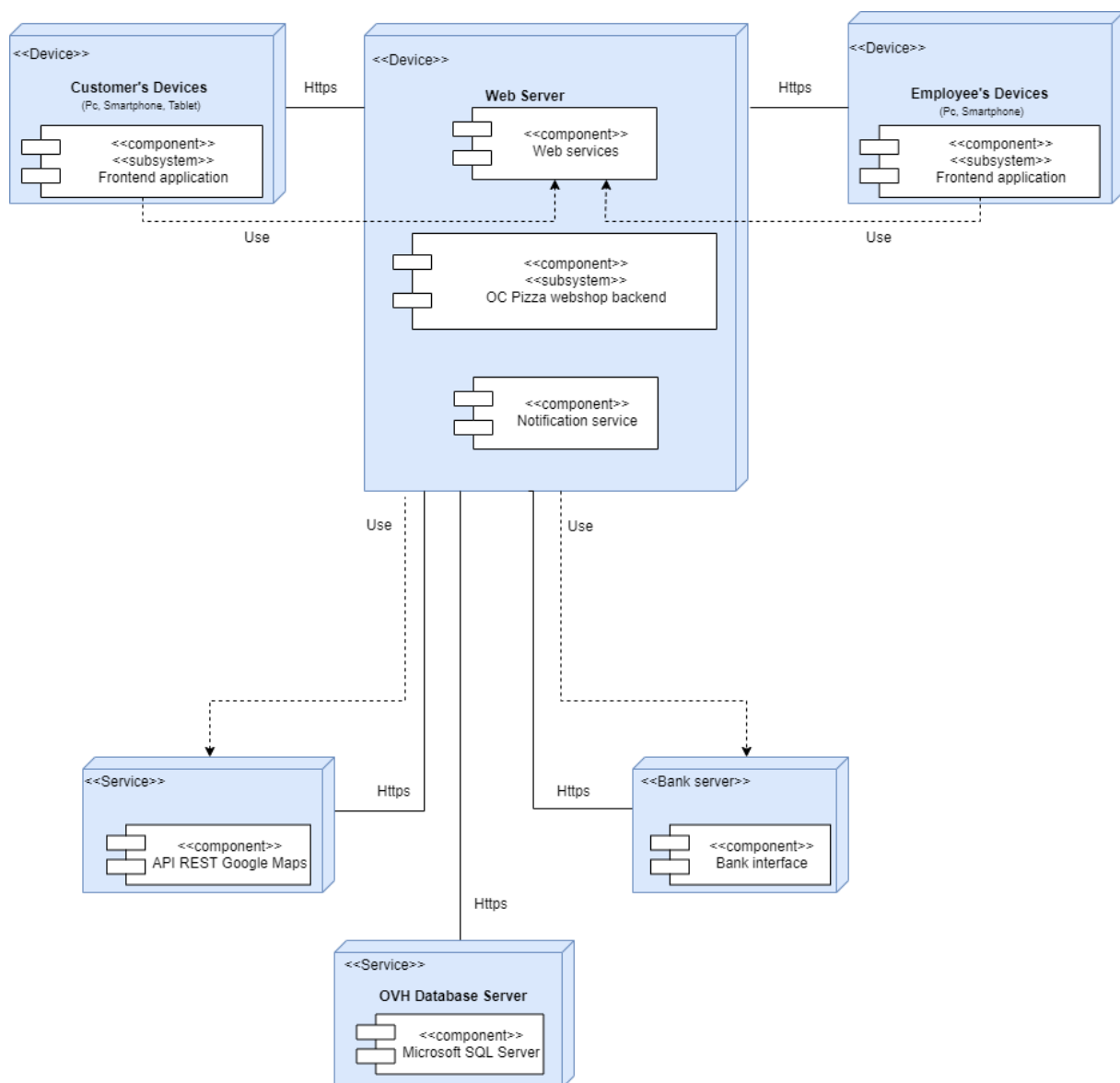
De l'autre côté nous avons le Backend du webshop est composé de cinq éléments. Commençons par « Notification service », dont nous avons déjà parlé précédemment, qui permet de recevoir des alertes lorsque le panier d'un client est validé et que la commande est elle aussi validée.

Nous retrouvons aussi le composant « Order Service » qui permet de gérer les différentes commandes, passées et en cours grâce aux informations communiquées par le « Webservice », de changer leurs statuts mais aussi de communiquer avec la base de données pour la mettre à jour, mais aussi avec le service bancaire externe afin de procéder au paiement des commandes. Ce composant permet aussi de mettre à jour le composant « Inventory » en actualisant les stocks, les ingrédients et les recettes disponibles. Ce dernier permet d'ailleurs d'avoir accès aux différents inventaires et stocks, magasin par magasin, ingrédient par ingrédient, grâce aux informations communiquées par « Order Service ».

Enfin, « User Management Service » permet de son côté de gérer différents processus comme l'authentification d'un client, la modification d'un mot de passe, la modification d'informations relatives au compte d'un client. C'est aussi ce composant qui nous permet de géolocaliser un client, et donc de lui attribuer un restaurant, ou simplement de fournir un itinéraire au livreur du magasin.

### A. Le diagramme de déploiement

Le diagramme de déploiement permet d'identifier les différents éléments matériels, physiques, ainsi que leurs connexions, leurs relations.



## B. Descriptif détaillé du diagramme

Comme vu dans le document précédent qui consistait à analyser les besoins de notre client pour son groupe de pizzerias et la mise en place d'un nouveau système informatique, la solution retenue est codée en J2EE sur le framework Spring. Le site internet devra être parfaitement adapté aux navigateurs mobiles (Responsive Web Design) afin de toucher le plus d'utilisateurs possibles mais aussi d'offrir un confort de navigation non négligeable. Les clients auront accès au « Frontend » de ce site soit la devanture du magasin, les autres acteurs auront accès quant à eux au « Backend » soit l'arrière-boutique de celui-ci.

Grace au site internet développé pour s'adapter parfaitement à une utilisation sur navigateur mobile, nous avons décidé de partir sur une autre solution, complémentaire avec le site, une application mobile simple, mais efficace. En effet, dès que l'utilisateur lancera l'application « OC Pizza », le site s'affichera directement dans celle-ci. Cette application sera donc une application « CRUD » (« Create, Read, Update, Delete » / « Créer, Lire, Mettre à jour, Supprimer »). Elle sera développée grâce à Angular, un framework qui permet de créer des applications web dynamiques et immersives en utilisant TypeScript comme langage de programmation.

Maintenant, intéressons-nous un peu plus en détail au diagramme de déploiement. Nous pouvons constater qu'il y a deux nœuds qui représentent les différents matériels que les différents acteurs peuvent utiliser lorsqu'ils utilisent le site internet ou l'application mobile OC Pizza. D'un côté, nous avons le nœud « Customer's devices », qui représente les appareils utilisés par les clients du restaurant pour passer une commande. Ces utilisateurs ont accès à une « Frontend Application ». De l'autre côté nous retrouvons le nœud « Employee's devices », qui quant à lui représente les supports matériels utilisés par les employés des différents magasins pour accéder aux différentes interfaces à leur disposition.

Que ce soit à travers le site internet ou l'application, c'est à travers des requêtes HTTPS (HyperText Transfer Protocol Secure, ou protocole de transfert hypertextuel sécurisé) qu'ils communiquent tous deux avec un troisième nœud, le « Web Server ».

Ce serveur web, comporte plusieurs composants tel que le « Web Service » qui permet aux différents éléments du diagramme de communiquer entre eux et d'échanger des données. « OC Pizza webshop Backend » est, comme son nom l'indique, le composant qui représente « l'arrière-boutique » du restaurant, du moins de son site internet, uniquement utilisé par les différents salariés du magasin.

Le dernier composant de « Web Server » est « Notification service » qui est quant à lui le composant qui permet d'informer, à travers le système informatique, les différents acteurs, comme par exemple, à travers un « tracker » qui suit l'évolution de la préparation d'une commande, d'une livraison.

Pour finir, nous pouvons aussi dire que le « Web Server » permet de communiquer, là aussi en HTTPS, avec trois autres nœuds, qui représentent ici trois services externes. Tout d'abord l'utilisation d'une API REST, ici Google Maps, qui sera utilisée pour les informations relatives de près ou de loin à la géolocalisation des clients et des restaurants OC Pizza. Ensuite l'accès à l'interface et aux services de la banque partenaire de la chaîne de restaurants, afin de pouvoir procéder à la vérification des données renseignées par les clients et relative au paiement de leurs commandes, interface qui permettra donc de valider ou non le paiement de ces dernières. Enfin, l'accès aux serveurs loués chez OVH et donc l'accès aux bases de données des magasins.