

# Dossier de conception technique



# IPIZZA

IT Consulting & Development

Mohamed JOUDAR

## Table des matières

1. Versions	3
2. Introduction	3
2.1. Object du document	3
2.2. Références	3
3. Architecture technique	4
3.1. Diagramme des composants	4
3.2. Composants généraux	5
3.2. Diagramme de classes détaillé	6
4. Architecture de déploiement	7
4.1. Diagramme de déploiement	7
4.2. Description du diagramme de déploiement	8
5. L'architecture logicielle	9
5.1. Principe généraux	9
5.2. Les couches	9

## 1. Versions

Version	Auteur	Date	Description
0.1	M. JOUDAR	13/05/2022	Création du document
0.2	M. JOUDAR	16/06/2022	Réajustement du domaine fonctionnel
0.3	M. JOUDAR	20/06/2022	Réajustement de l'architecture de déploiement
1.0	M. JOUDAR	23/06/2022	Mise en forme finale

## 2. Introduction

### 2.1. Object du document

Le présent document constitue le dossier de conception technique de l'application OC Pizza.

Objectif du document : Mise en place d'un nouveau système informatique pour l'ensemble des pizzerias du groupe.

Les éléments du présents dossiers découlent :

- du recueil des besoins et du cahier des charges fourni par le client
- de la présentation de la solution fonctionnelle au client lors de l'entrevue au lancement du projet
- de la présentation de la solution technique au développeur expérimenté

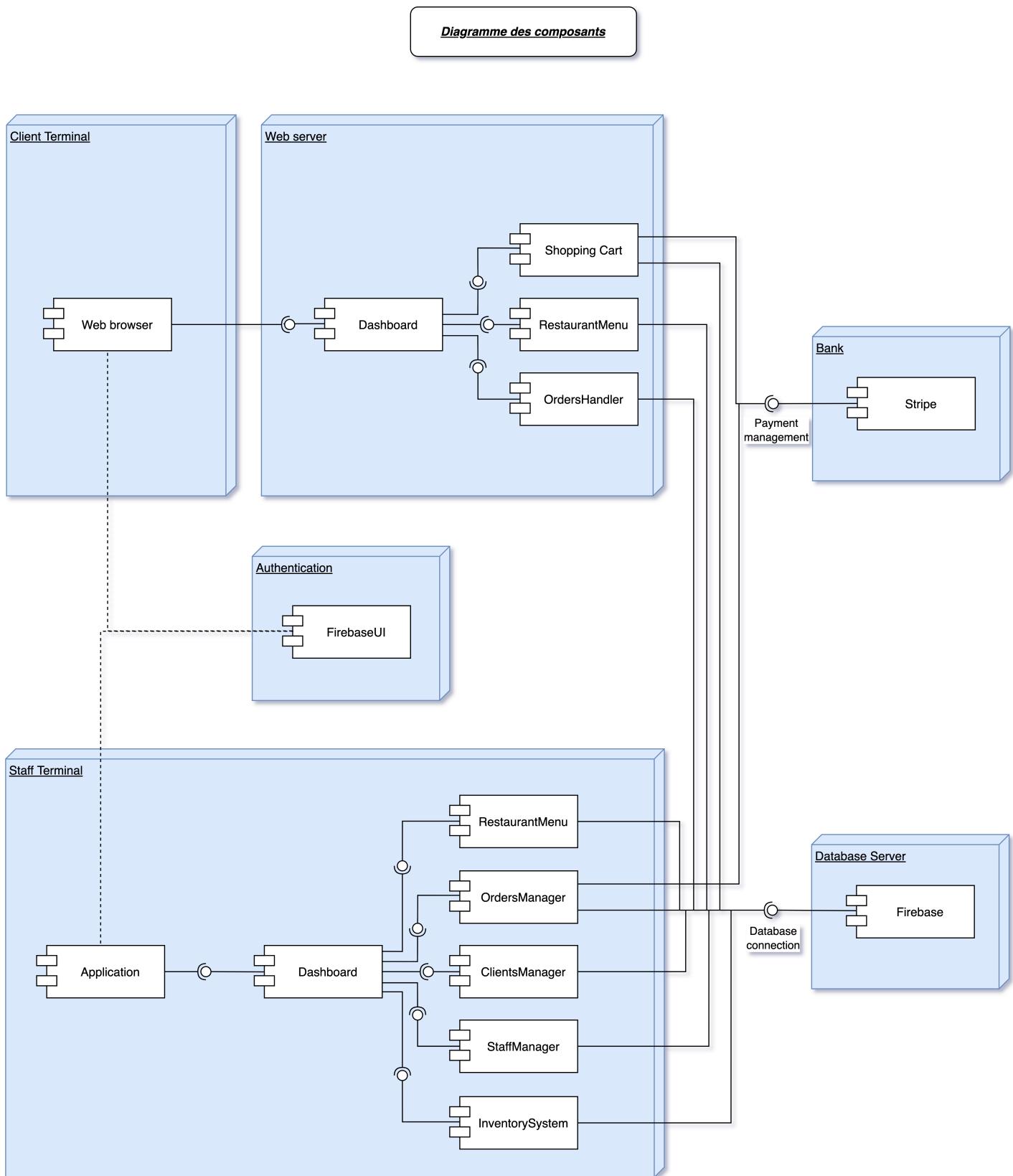
### 2.2. Références

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

1. MJ\_1\_conception\_fonctionnelle\_OCPizza.pdf : Dossier de conception fonctionnelle de l'application
2. MJ\_3\_dossier\_d'exploitation\_OCPizza.pdf : Dossier d'exploitation de l'application
3. MJ\_4\_PV\_de\_livraison\_finale\_OCPizza.pdf : Procès-verbal de livraison finale

### 3. Architecture technique

#### 3.1. Diagramme des composants



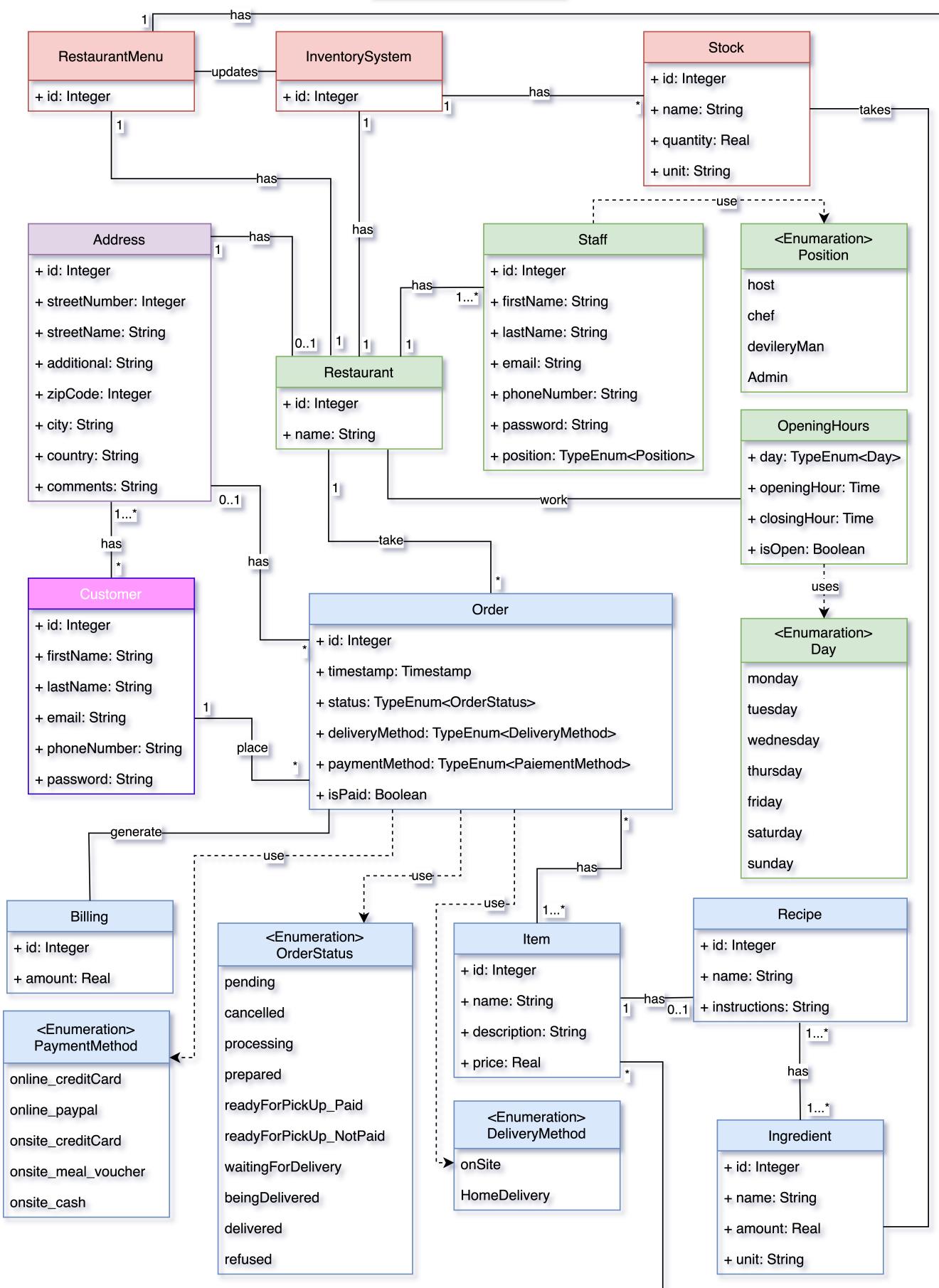
### 3.2. Composants généraux

- Un module **Authentication** qui permettra aux nouveaux *clients* de s’inscrire et aux clients déjà inscrits de se connecter. Cette étape est obligatoire pour passer des commandes. Le client peut s’authentifier avec son réseau social favori ou en utilisant une adresse mail et un mot de passe. Ainsi que permettre aux différents *employés du groupe* de se connecter sur l’application. Cette étape est obligatoire pour accéder aux fonctionnalités de l’application.
- Un **site web pour le Front office** (l’espace client pour commander, suivre ses commandes, gérer son compte...).
  - Le composant “*Web browser*” est un outil utilisé par les clients pour naviguer sur Internet et accéder au site web du restaurant.
  - Le composant “*Dashboard*” du **Front office (site web)** permettra aux clients de gérer leur espace personnel ainsi que d’accéder à toutes les fonctionnalités qui leurs sont proposées par le site du restaurant.
  - Le composant “*ShoppingCart*” permettra aux clients de constituer un panier, d’ajouter ou supprimer des articles et de passer commande.
  - Le composant “*RestaurantMenu*” du **Front office (site web)** permettra aux clients de consulter la liste des articles proposés par le restaurant. C’est à partir de ce composant que le panier pourra être rempli.
  - Le composant “*OrderHandler*” permettra aux clients de gérer et suivre leurs commandes, modifier ou supprimer les commandes en cours.

- Une ***application Android pour le Back office*** (l'interface du personnel pour recevoir, préparer, livrer les commandes, gérer le restaurant...).
  - Le composant “*Dashboard*” du *Back office* (*l'application staff*) permettra aux employés d'accéder à toutes les fonctionnalités qui leur sont autorisées par l'application du Back office.
  - Le composant “*RestaurantMenu*” du *Back office* (*l'application staff*) permet aux employés de consulter la liste des articles proposés par le restaurant ainsi que leurs aide-mémoire-recette, mais aussi, pour les employés autorisés, de modifier cette liste.
  - Le composant “*OrderManager*” permet aux employés de gérer les commandes reçues (modifier ou supprimer des articles d'une commande, modifier les coordonnées d'une commande, supprimer une commande...) selon leurs autorisations.
  - Le composant “*ClientManager*” permet aux employés autorisés de gérer les comptes client (modifier ou supprimer un compte).
  - Le composant “*StaffManager*” permet aux employés autorisés de gérer les comptes staff (modifier ou supprimer un compte).
  - Le composant “*InventorySystem*” permet aux employés autorisés de gérer l'inventaire du restaurant.
- Une ***base de données Firebase*** qui permettra la transition des données collectées et manipulées.

### 3.2. Diagramme de classes détaillé

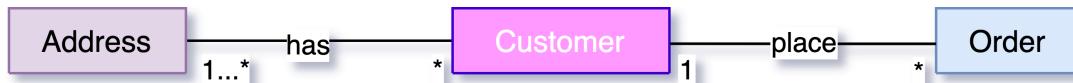
Diagramme des classes



### 3.3. Description des classes

#### Customer

Cette classe regroupe les attributs associés à un client du restaurant.

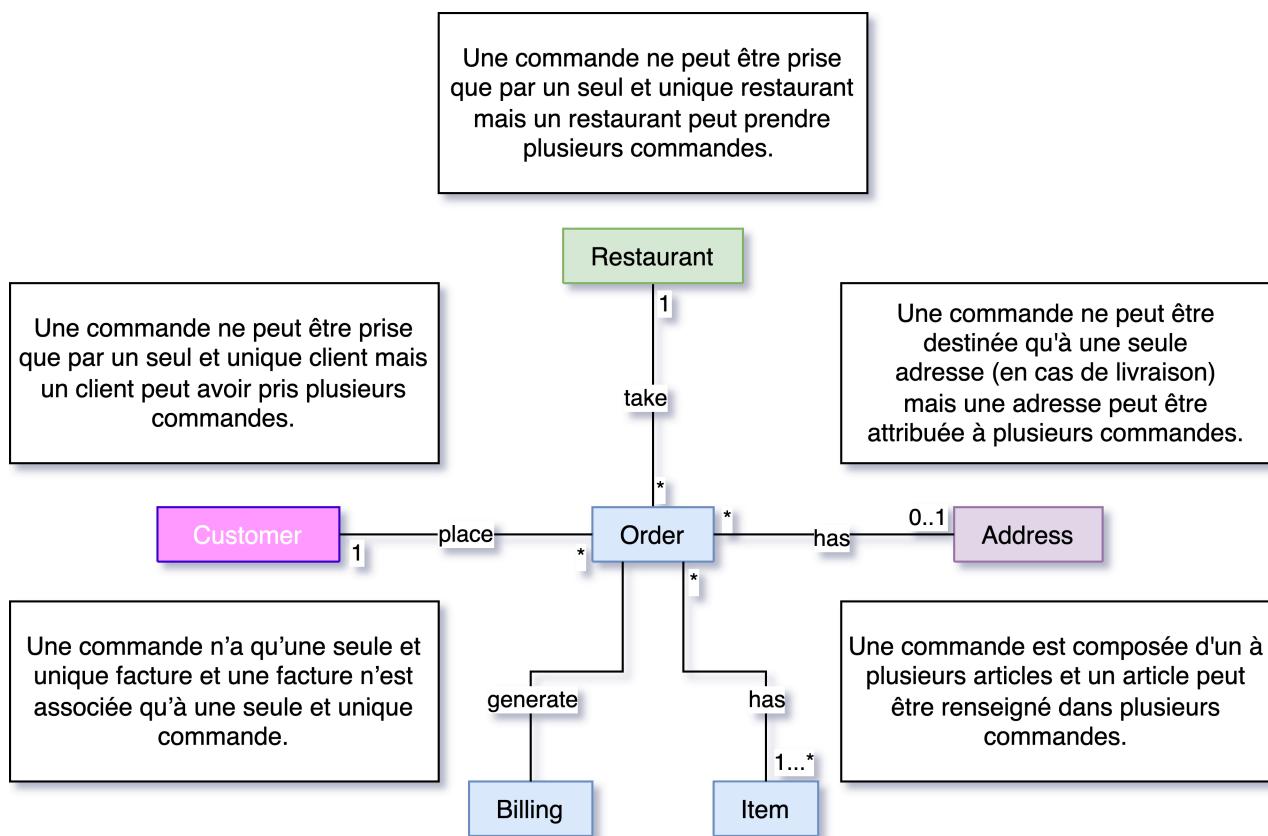


Une adresse peut appartenir à plusieurs clients (eg. membres d'une famille, colocataires) et un client peut avoir une à plusieurs adresses (eg. domicile, travail).

Une commande ne peut être prise que par un seul et unique client mais un client peut avoir pris plusieurs commandes.

#### Order

Cette classe regroupe tous les attributs associés à une commande effectuée.



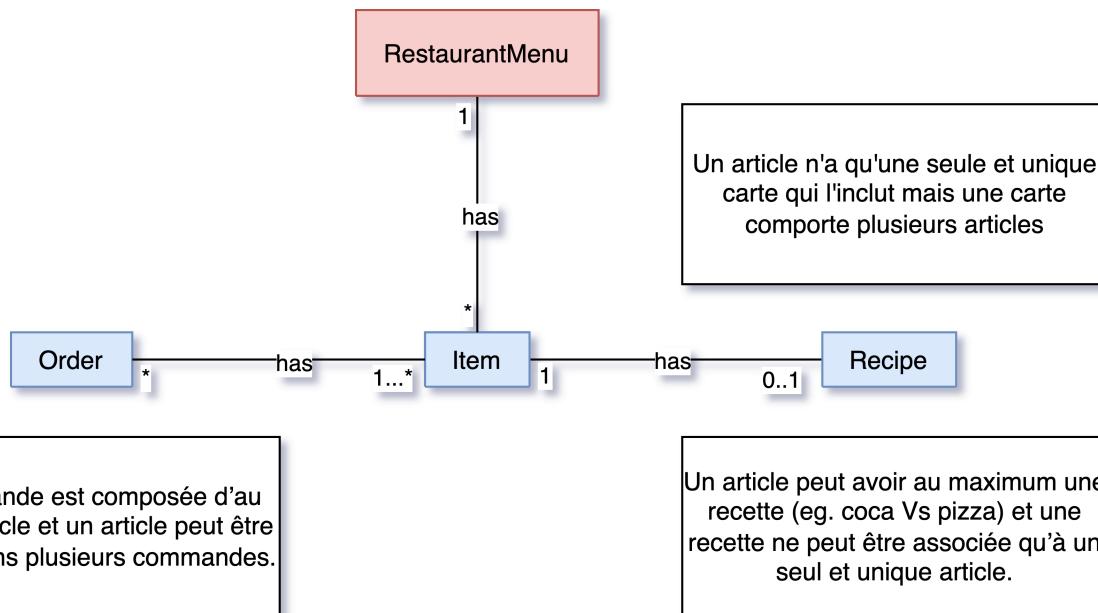
### Billing

Cette classe regroupe tous les attributs associés à une facturation.



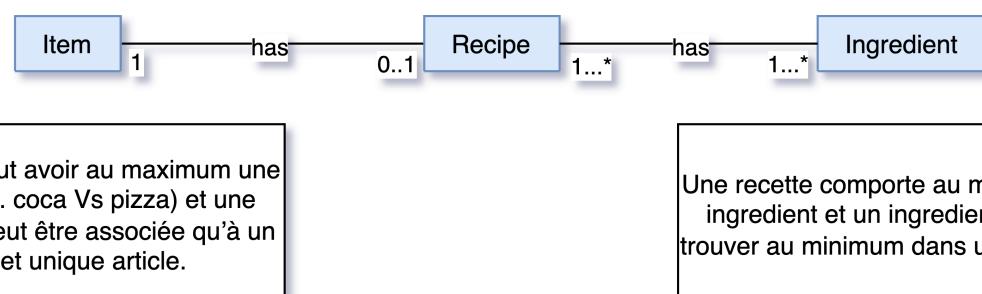
### Item

Cette classe regroupe les attributs spécifiques d'un article proposé (eg. pizza, salade, boisson...).



### Recipe

Cette classe regroupe tous les attributs associés à une recette.



## Ingredient

Cette classe regroupe les attributs spécifiques d'un ingrédient.



Une recette comporte au minimum un ingrédient et un ingrédient doit se trouver au minimum dans une recette.

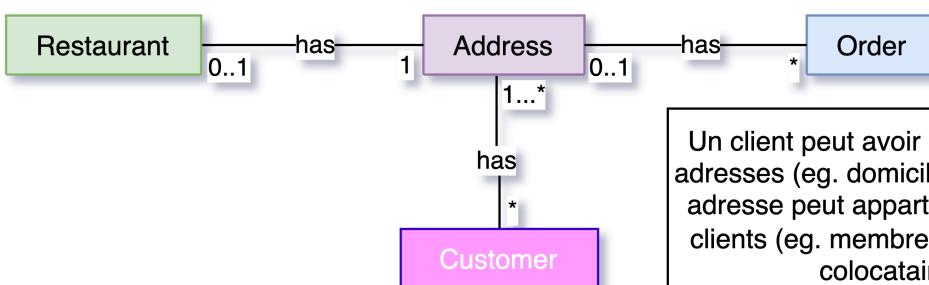
L'ingredient fait partie du stock, c'est une micro quantité du stock.

## Address

Cette classe regroupe les attributs des adresses des clients, de livraison et des restaurants.

Un restaurant ne peut être localisé qu'à une seule et unique adresse et une adresse ne peut être attribuée qu'à un seul restaurant.

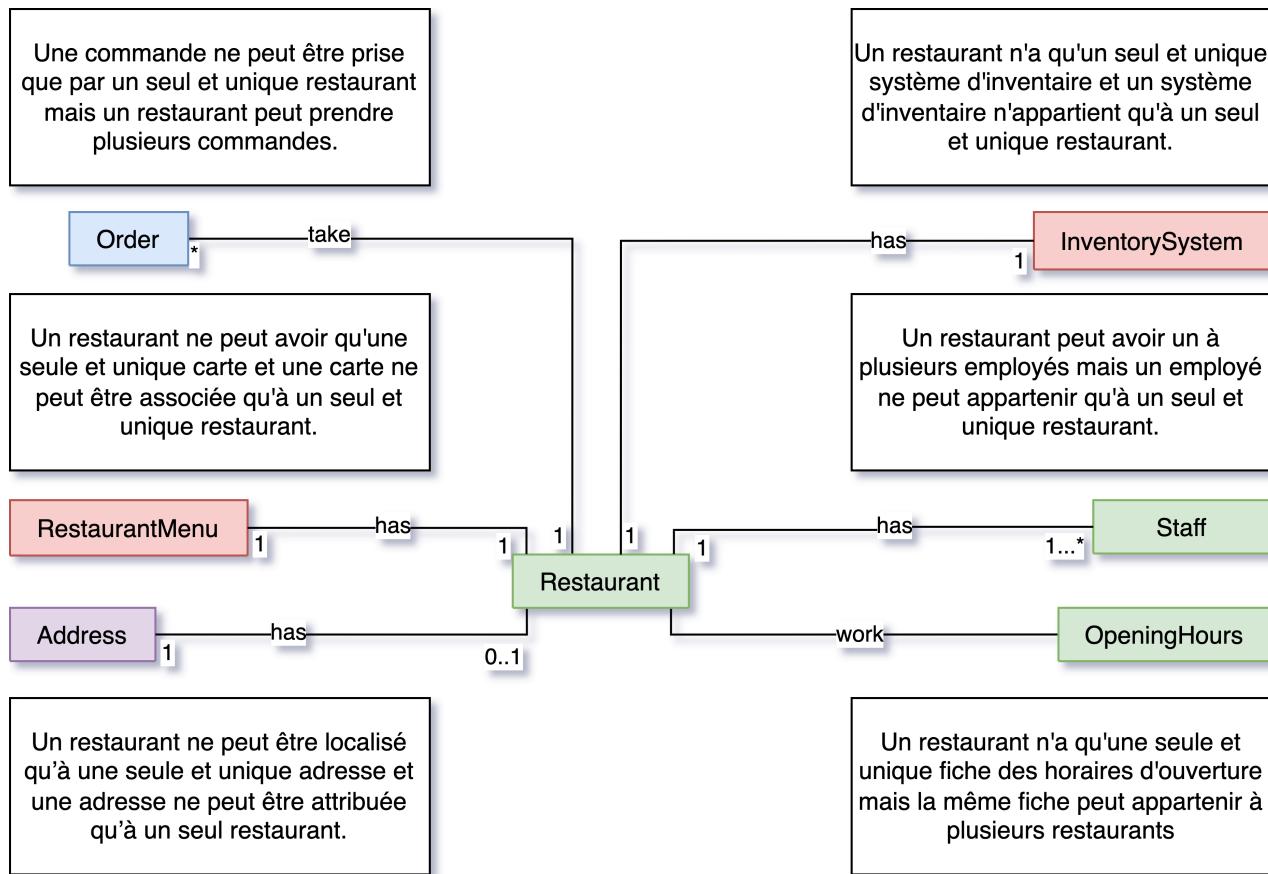
Une commande ne peut être destinée qu'à une seule adresse (en cas de livraison) mais une adresse peut être attribuée à plusieurs commandes.



Un client peut avoir une à plusieurs adresses (eg. domicile, travail) et une adresse peut appartenir à plusieurs clients (eg. membres d'une famille, colocataires).

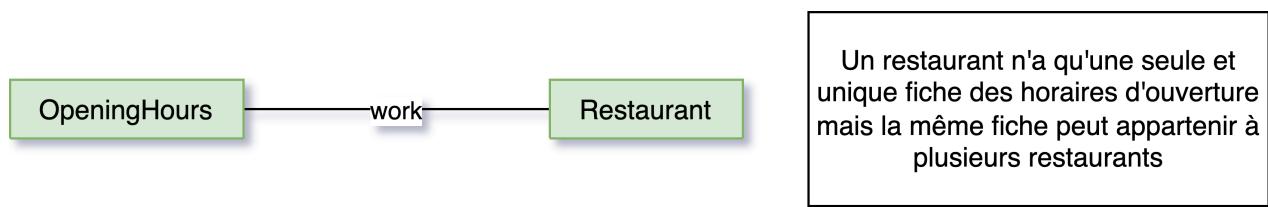
## Restaurant

Cette classe regroupe les attributs des différentes restaurants du groupe.



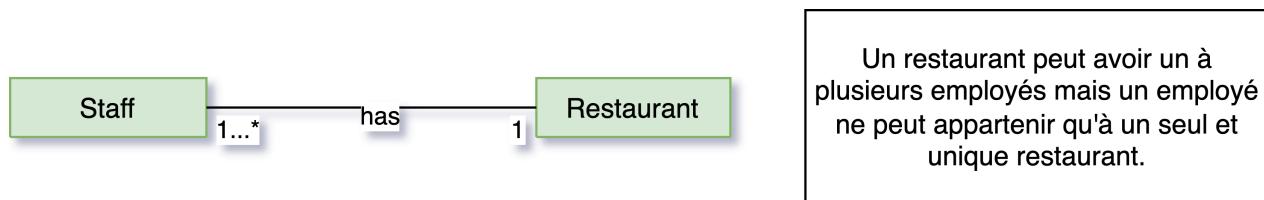
## OpeningHours

Cette classe regroupe les attributs des horaires d'ouverture des restaurants.



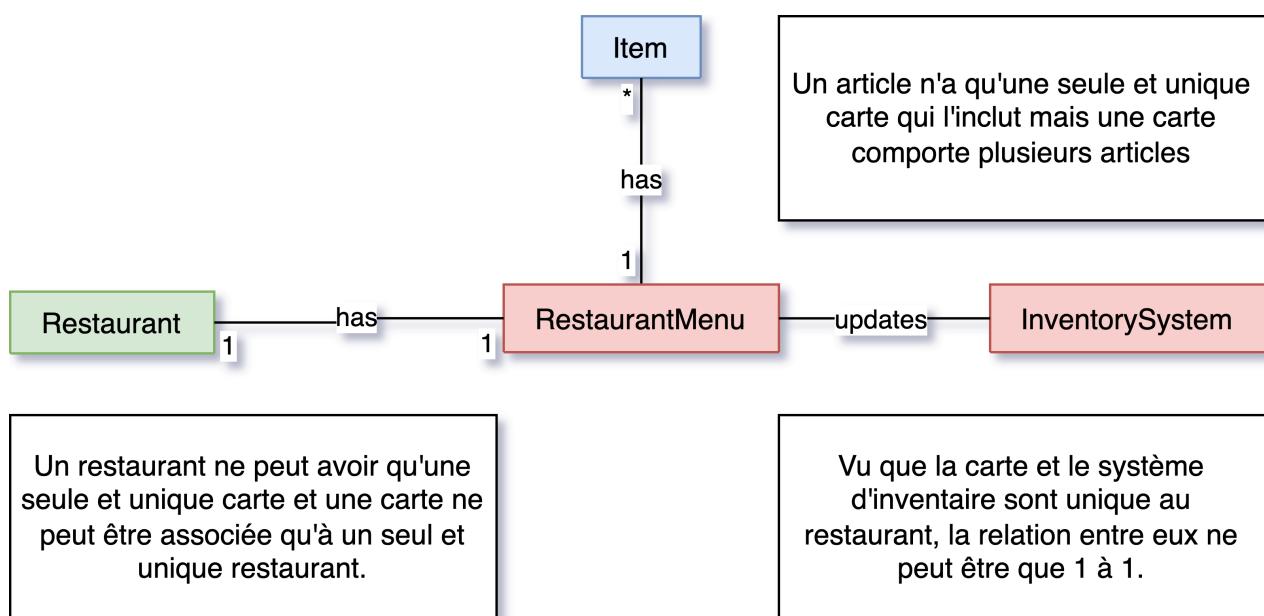
## ***Staff***

Cette classe regroupe tous les attributs associés à un membre du personnel.



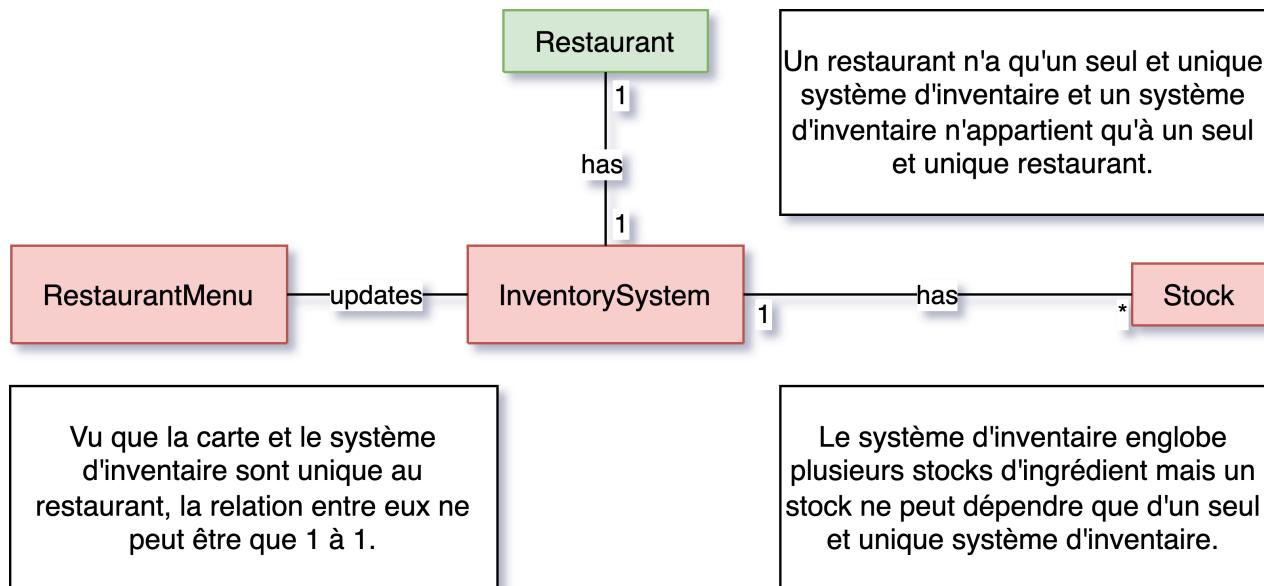
## ***RestaurantMenu***

Cette classe regroupe tous les attributs associés à la carte du restaurant.



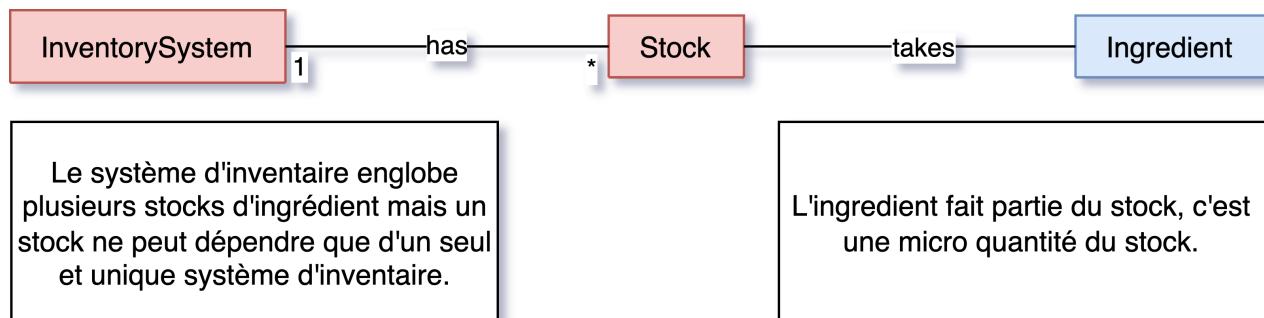
### InventorySystem

Cette classe regroupe tous les attributs associés au système d'inventaire du restaurant.



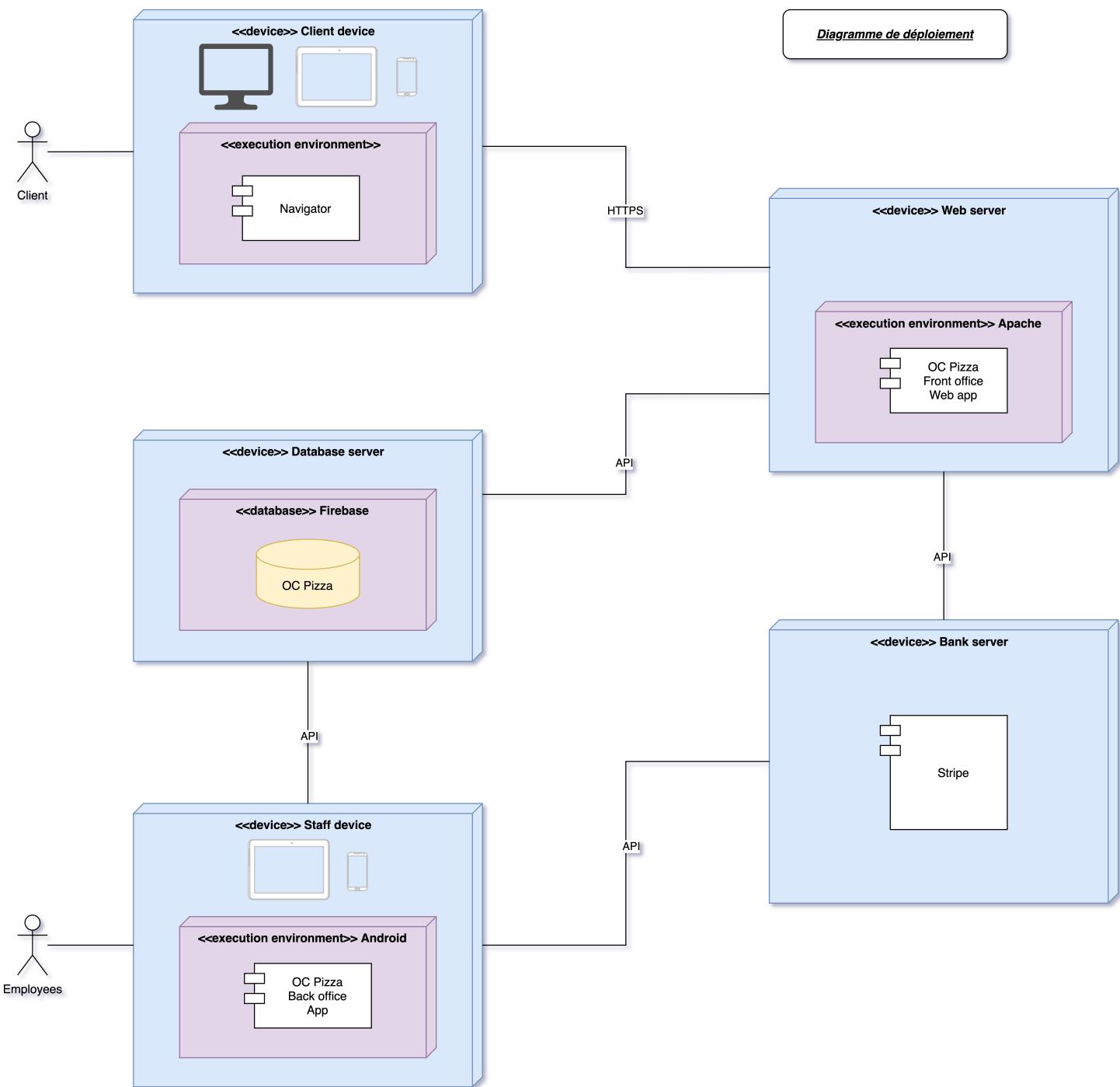
### Stock

Cette classe regroupe tous les attributs associés à un stock d'un ingrédient donné (eg. farine, mozzarella...).



## 4. Architecture de déploiement

### 4.1. Diagramme de déploiement



## 4.2. Description du diagramme de déploiement

Dans ce diagramme chaque entrée représente un nœud du déploiement du système. La communication entre ces noeuds se fait à travers des API sécurisées.

### Client device

Le terminal à travers lequel le client, usant d'un navigateur, accède au site web (front office) du restaurant et à ses fonctionnalités.

### Web server

Le serveur Apache qui héberge le site web (front office) du restaurant. Il est proposé que le site web soit Shopify, déployé sur un serveur Apache. Le serveur communique directement avec la base de donnée Firestore et avec le service de paiement Stripe via des API sécurisées afin d'enregistrer les informations liées aux clients ou aux commandes, ainsi que les transactions bancaires.

### Staff device

Le terminal utilisé par le personnel du restaurant, un appareil Android où l'application OC Pizza (back office) du restaurant y est installée. Il communique directement avec la base de donnée Firestore et avec le service de paiement Stripe via des API sécurisées afin d'enregistrer les informations liées à la gestion des commandes (eg. suivi d'une commande, modification, remboursement...) et à la gestion du restaurant (stock, carte du restaurant...).

Le matériel devra au minimum embarquer l'API 21 d'Android

### Bank server

Le service Stripe, externe au système, qui permettra de gérer et sécuriser les paiements.

### Database server

Le service Firebase, externe au système, qui hébergera la base de données Firestore. Celle-ci prendra en charge toutes les données liées au système OC Pizza. Ce service embarquera la base de données et le système d'API qui permettra d'établir la communication avec les applications (Web et Android de OC Pizza).

## 5. L'architecture logicielle

### 5.1. Principe généraux

Le code source du projet et son versioning sont gérés par Git.

L'application Android OC Pizza sera développée suivant le pattern MVC (Model-View-Controller).

- **Model** : Le modèle est la partie de l'application qui exécute toute la partie logique de l'application comme les appels réseaux ou encore les opérations arithmétiques.
- **View** : C'est la partie graphique de l'application, elle n'a pour but que d'afficher des éléments à l'écran.
- **Controller** : Cette partie permet de relier le modèle à la vue.

L'application Web sera développée quant-à-elle suivant une template de Shopify.

### 5.2. Les couches

L'architecture applicative est la suivante :

- **Une couche business** : responsable de la logique métier du composant
- **Une couche model** : implémentation du modèle des objets métiers
- **Une couche presentation** : responsable de l'interface graphique de l'application.

## 6. Conclusion

Afin de répondre aux besoins exprimés par OC-Pizza, la solution proposée est la suite; mettre en place un système OZ Pizza qui sera composé des éléments ci-dessous :

- Un site web du restaurant destiné aux clients pour:
  - Afficher la carte du restaurant;
  - Proposer un espace personnel client;
  - Effectuer des commandes.
- Une application Android destinée au personnel du restaurant pour:
  - Gérer les commandes en temps réel, de leur réception à leur livraison;
  - Gérer le restaurant, le stock et le personnel.
- Une base de données Firebase pour:
  - Structurer les données mises en jeu par le système,
  - la gestion des données en temps réel,
  - la gestion de leurs persistances.

Nous avons, dans un premier temps, présenté le domaine fonctionnel, en annonçant l'architecture de notre système, sous la forme de diagrammes de classes et de composants.

Dans un second temps, nous avons clairement défini le modèle physique de données et les éléments constituants la base de données.

Pour finir nous avons établi la base de données qui contient un jeu de données de démo, pour simuler une commande factice.