It Consulting
&
Development

OC PIZZA

OC_pizza

Dossier de conception technique

Version 1.0

Auteur
Jérémy Guyot

Table des matières

1 - Versions

Auteur	Date	Description	Version
Jeremy.G	10/07/2020	Création du document	1.0

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier de conception fonctionnelle de l'application OC pizza

Objectif du document est de présenter les besoins de l'utilisateur de décrire la solution qui va être implémentée pour répondre à ces besoins.

Les éléments du présent dossier découlent :

- de l'entretien réalisé avec le dirigeant de la société OC pizza du 20/03/2018.
- De l'analyse des besoins suite à cet entretien effectué par l'équipe, it consulting & development.

2.2 - Références

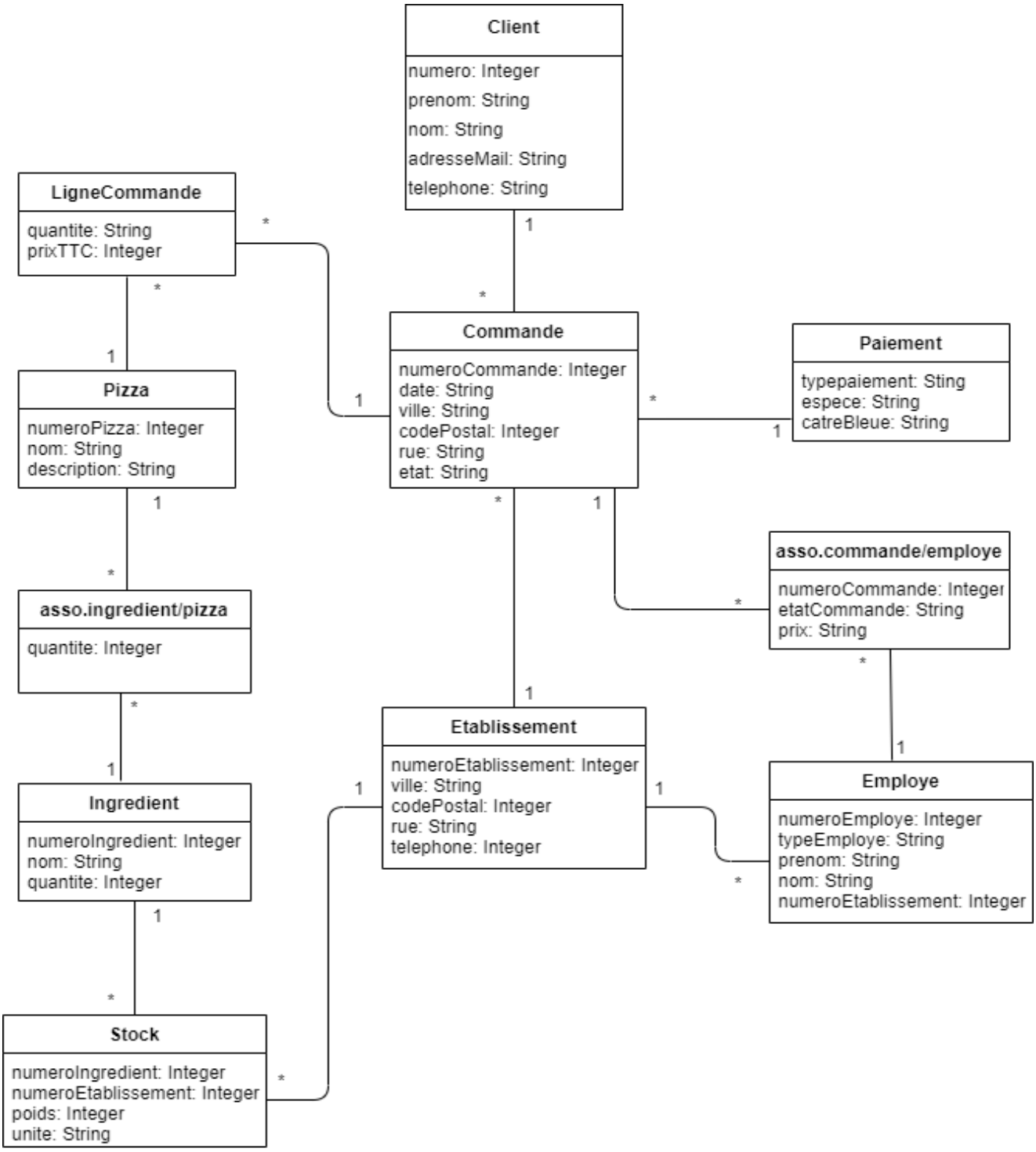
Pour de plus amples informations, se référer également aux éléments suivants:

1. **P9 _ DCF – 1.0** : Dossier de conception fonctionnelle de l'application.
2. **P9 _ DCT – 1.0** : Dossier de conception technique de l'application.
3. **P9 _ DE – 1.0** : Dossier d'exploitation de l'application.
4. **P9 _ PVL – 1.0** : Dossier de procès verbal de livraison.

3 - Le domaine fonctionnel

3.1 - Référentiel

Les paragraphes suivants ont pour objectif de présenter l'organisation et l'utilisation des données.
Après avoir étudié les différents aspects du projet, il en est ressorti le diagramme de classes suivant :



3.2 - Diagramme de classe, explication

Les classes représentées sur le schéma ci-dessus ont pour but de montrer les éléments suivants :

- **Client** : La classe client va représenter la fiche d'identité du client, son nom, prénom, adresse e-mail et son numéro de téléphone, ainsi que l'accès à sa commande via le site web.
- **Commande** : La classe commande est la classe qui regroupe l'ensemble des classes. Elle contient les informations de la date et l'heure, la ville où sera passée la commande, son code postale, la rue ou est implémentée le magasin, ainsi que l'état de la commande si elle est en cours ou préparée.
- **Paiement** : La classe paiement sert à finaliser la commande, soit en boutique, soit sur le site web ou quand le livreur arrive. Ils pourront régler en espèces ou carte bleue.
- **Pizza** : Chaque pizza sera matérialisée par une description du produit, d'un nom et du prix.
- **Ingrédient** : La classe ingrédient comprendra comme informations, le nom des ingrédients pour les utilisateurs, la fiche de préparations pour les pizzaiolos et la quantité qu'il leur faudrait pour chaque pizza.
- **Stock** : La classe stock sera répertoriée pour chacun des magasins avec le poids et l'unité de chaque ingrédient.
- **Établissement** : La classe établissement va représentée la ville, le code postale, la rue et le numéro de téléphone de chacune des pizzerias.
- **Employé** : La classe employée représente, le type de chaque employé, son prénom, son nom et dans quel établissement il est affilié.

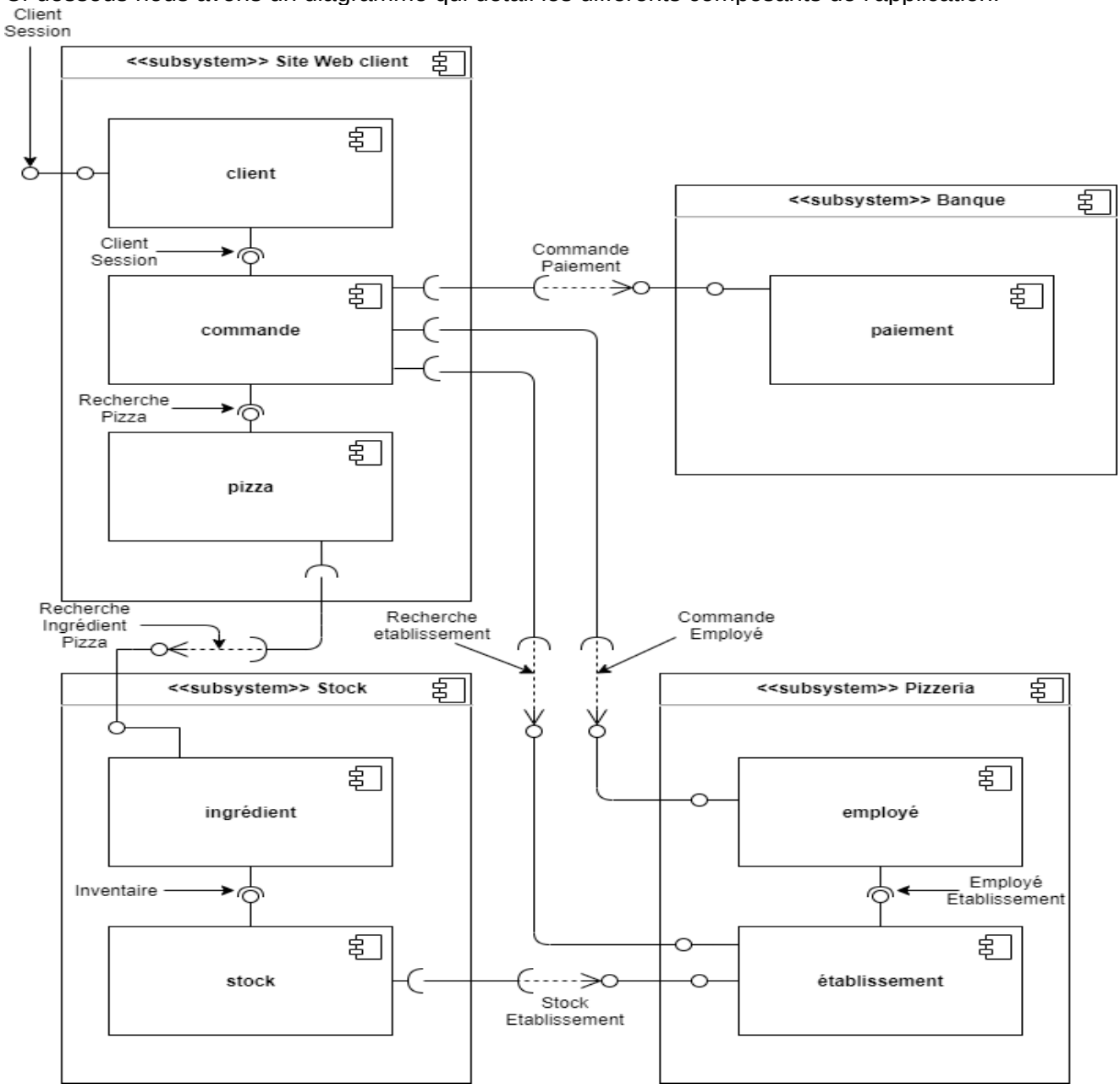
4 - Architecture Technique

4.1 - Application Web

La pile logicielle est la suivante :

- Application **Python** 3.8 (Django 3.0)
- Serveur d'application **Gunicorn** 20.0.4
- Serveur web **Nginx** 1.19.1
- Data Base **MySQL** 8.0.19

Ci-dessous nous avons un diagramme qui détail les différents composants de l'application.



4.1.1 - Composants clients

Clients : La partie client gère les données relatives aux clients qui comprend les données personnelles tel que l'adresse (rue, ville, code postal) le numéro de téléphone de la personne, le numéro de commande et l'e-mail.

4.1.2 - Composants produits

Produits : La partie produit, regroupe les données des ingrédients avec comme attribut un nom et un descriptif. Ils seront ensuite associés à des produits qui auront aussi un nom, une description et un prix TTC. Chaque instance de la classe produit, représentera un produit qui sera proposé aux clients et chaque produit sera associé à divers ingrédients. Pour associer les ingrédients aux produits nous devrons utiliser une table d'association pour pouvoir lier ces deux tables entre elles.

4.1.3 - Composants établissements

établissement : À chaque instance de la table établissement sont attribués plusieurs lots d'ingrédients ainsi que des employés qui seront affectés à différents corps de métier comme les livreurs, la caisse ou les pizzaïolos.

4.1.4 - Composants commandes

Commandes : La partie commande va regrouper, le numéro de la commande du client, la date et l'heure où sont passée les commandes, la ville suivante ou va se situer la personne et l'établissement concerné et l'état de la commande si elle est en préparation ou finie.

4.1.5 - Composants types de paiements

Types de paiements : Ce composant permet à l'utilisateur de choisir son type de paiement, il peut payer en espèce (sur place ou à domicile en se faisant livré) en CB (sur place, à la livraison ou en ligne)

4.1.6 - Composants employés

Employés : Ce composant permet de donner un numéro d'identification à chaque personne employée. De dire dans quel domaine elle travaille (pizzaïolo, livreur, vendeur, directeur), de donner son prénom, nom et de préciser dans quel établissement il ou elle travaille.

4.1.7 - Composants pizzas

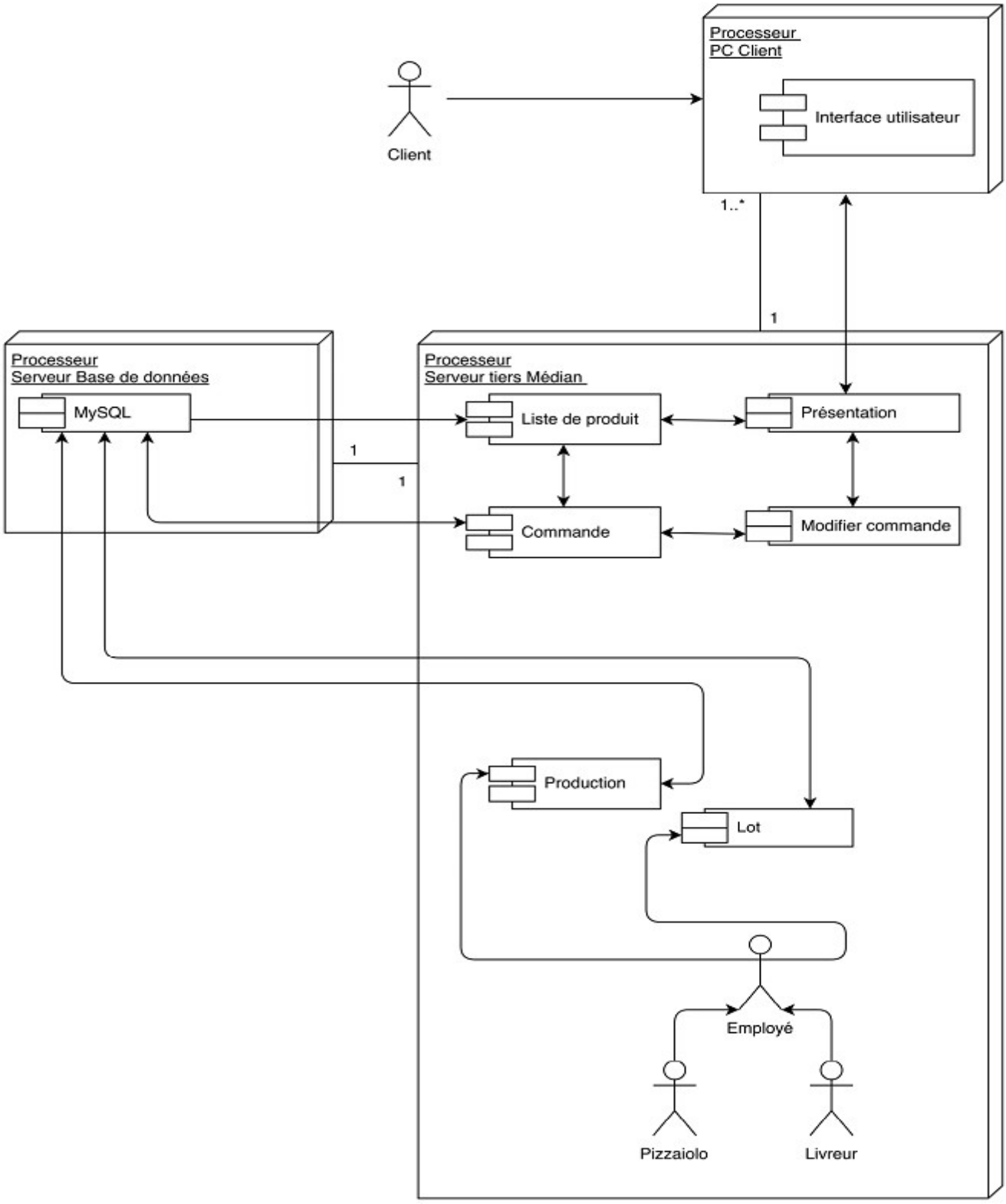
Pizzas : Ce composant va permettre d'avoir le numéro des différentes pizzas proposées, son nom pour les clients avec une description des ingrédients utilisés et son prix à l'unité hors promotion.

4.1.8 - Composants stock

Stock : Ce composant va permettre de pouvoir savoir quel type de produits on a en stock, ainsi que sa quantité et le numéro de l'établissement qui lui est attribué.

5 - Architecture de Déploiement

Dans cette section vous pouvez retrouver le diagramme de déploiement qui donne une vue d'ensemble de l'architecture de production de notre application OC Pizza.



Vous pourrez retrouver les explications de ce diagramme ci-dessous.

5.1 - Serveur de Base de données

Caractéristiques techniques (Serveur Linux Ubuntu (LTS) x64 + MySQL 8.0.19)

Le système de gestion de base de données utilisé sera MySQL

Caractéristiques techniques du serveur :

Serveur : Digital Ocean 1 vCPU (private CPU)

RAM : 1 Go Mémoire

Stockage : 1 SSD 25 Go

Transfert : 1To

Datacenter sera à Londre

5.2 - Serveur Tiers Médian

On aura 2 serveurs sur le serveur tiers médian :

Il y a le serveur **Nginx** qui est le serveur web (HTTP), qui va surtout concerner la partie pour les clients et les vendeurs. Il va être utilisé si un fichier statique et demandé, il va l'afficher sans passer par l'application et sinon il va rediriger le trafic vers l'application **Django**.

Il y a le serveur **Gunicorn** qui est le serveur (HTTP Python) pour **Unix**, qui va faire vivre l'application **Django**. Il s'agit d'une librairie Python en source ouverte.

6 - Architecture logicielle

6.1 - Principes généraux

Les sources et versions du projet sont gérées par **Git**, les dépendances et le packaging par **Pipenv**.

6.1.1 - Les couches

L'architecture applicative suit l'architecture standard d'un projet **Django** qui se nomme :

- une couche **views** : La vue sert à recevoir une requête HTTP et d'y répondre de manière intelligible par le navigateur.
- une couche **model** : Le modèle interagit avec la base de données. Sa mission est de chercher dans une base de données les items correspondant à une requête et de renvoyer une réponse facilement exploitable par le programme.
- une couche **template** : Un template est un fichier HTML qui peut recevoir des objets Python et qui est lié à une vue .

6.1.2 - Les modules

Les apps suivants seront nécessaires pour implémenter l'application OC pizza :

- commande
- employe
- paiement
- pizzas
- stock
- users

6.1.3 - Structure des sources

La structuration des répertoires du projet suit la logique suivante :

– les répertoires sources sont créés de façon à respecter la philosophie Maven (à savoir : « convention plutôt que configuration »)

```
|   manage.py
|
├───commande
|   |   apps.py
|   |   ...
|   |
|   └───migrations
|           __init__.py
|
├───employe
|   |   views.py
|   |   ...
|   |
|   └───migrations
|           __init__.py
|
├───ocpizza
|       asgi.py
|       settings.py
|       urls.py
|       wsgi.py
|       __init__.py
|
├───paiement
|   |   apps.py
|   |   ...
|   |
|   └───migrations
|           __init__.py
|
├───pizzas
|   |   apps.py
|   |   ...
|   |
|   └───migrations
|           __init__.py
|
├───stock
|   |   models.py
|   |   ...
|   |
|   └───migrations
|           __init__.py
|
└───users
    |   views.py
    |   ...
    |
    └───migrations
            __init__.py
```

7 - Points particuliers

7.1 - Gestion des logs

On va utiliser **Sentry** pour monitoré les logs de l'application, vois-ci un liens :

https://docs.sentry.io/

7.2 - Fichiers de configuration

7.2.1 - Application web

On va configurer l'application avec plusieurs fichiers settings pour le local et la production, ces fichiers seront situés dans le répertoire. (OCpizza/settings)

7.3 - Ressources

Documentation **Django** : https://docs.djangoproject.com/fr/3.0/

Documentation **Python** : https://docs.python.org/fr/3/

Documentation **Sentry** : https://docs.sentry.io/

Documentation **MySQL** : https://dev.mysql.com/doc/

7.4 - Environnement de développement

On a utilisé l'application et est développée à l'aide du serveur de développement Django

(**python manage.py runserver**) ainsi que (**MySQL 8.0**) pour la base de données.

7.5 - Procédure de packaging / livraison

Pour le packaging on va directement se servir dans le dépôt GitHub qui servira à héberger les sources du projet.
On va en particulier avoir une branche production sur lequel se trouvera le code disponible sur le serveur.

8 - Glossaire

GIT	Git est un logiciel de gestion de versions décentralisé.
Pip	pip est un gestionnaire de paquets utilisé pour installer et gérer des paquets écrits en Python. De nombreux paquets peuvent être trouvés sur le dépôt Python Package Index (PyPI). pip est un acronyme récursif qui correspond à la fois à « Pip Installs Packages » ou à « Pip Installs Python ».