|  |
| --- |
| **OC PIZZA**  ProJet 9  Dossier de conception technique  Version 1.0 |
| **Auteur**  Yann Hamdi |

Table des matières

[1 - Versions 4](#_Toc64973996)

[2 - Introduction 5](#_Toc64973997)

[2.1 - Objet du document 5](#_Toc64973998)

[2.2 - Références 5](#_Toc64973999)

[3 - Le domaine fonctionnel 6](#_Toc64974000)

[3.1 - Référentiel 6](#_Toc64974001)

[3.2 - Diagramme de classe, explication 6](#_Toc64974002)

[3.3 - Application XXX... 8](#_Toc64974003)

[4 - ARCHITECTURE TECHNIQUE 9](#_Toc64974004)

[4.1 - Application Web 9](#_Toc64974005)

[5 - Architecture logicielle 13](#_Toc64974006)

[5.1 - Serveur de Base de données 13](#_Toc64974007)

[5.2 - Serveur Tiers Médian 13](#_Toc64974008)

[6 - ARCHITECTURE LOGICIELLE 15](#_Toc64974010)

[6.1 - Principes généraux 15](#_Toc64974011)

[7 - POINTS PARTICULIERS 18](#_Toc64974012)

[8 - Glossaire 19](#_Toc64974013)

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| Yann Hamdi | 21/02/2021 | Création du document | 1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## 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 réalisé́ avec le dirigeant de la société́ OC pizza du 18/09/2018.

• De l’analyse des besoins suite à cet entretien effectué par l’équipe IT EXPERTS

## Références

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

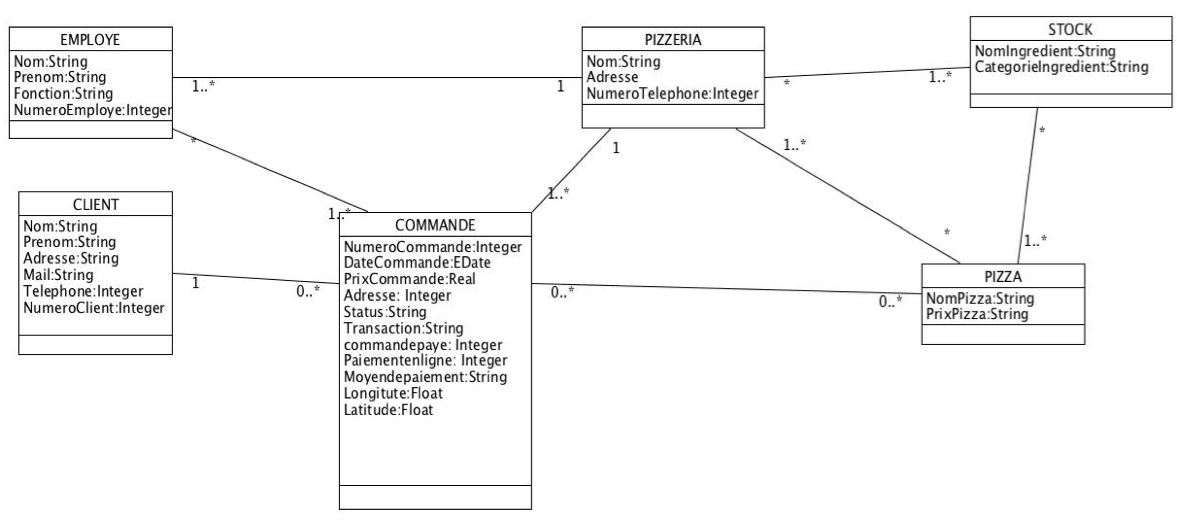
1. **DCT – 1.0** : Dossier de conception fonctionnelle
2. **DE – 1.0** : Dossier d’exploitation de l’application

# Le domaine fonctionnel

## Référentiel

Cette partie du document va présenter l’organisation et l’utilisation des données.

Après étude, nous obtenons le diagramme de classe suivant :



## Diagramme de classe, explication

Nous avons en premier lieu la classe « EMPLOYE » qui représentera tous les employés des pizzerias, cette classe est liée avec la classe pizzeria avec une relation 1..\* et 1 puisqu’un employé́ ne peut travailler quand dans une seule pizzeria par contre une pizzeria peut avoir 1 ou plusieurs employés qui y travaille.

Il y a également une relation entre la classe employé́ et la classe commande avec une relation \*, 1..\* puisqu’il y peut y avoir plusieurs employés qui travaillent sur la même commande (livreur et pizzaiolo par exemple) et un employé́ peut avoir travaillé́ sur plusieurs commandes.

Nous avons ensuite la classe **« PIZZERIA »** qui représente les différentes pizzerias du groupe. Cette classe est reliée à la classe **« EMPLOYE »**  comme expliqué plus haut puis nous avons d’autre relation.  
Elle est ensuite reliée au stock afin d’avoir un suivi du stock des différentes pizzerias, elle est reliée par une relation \*, 1..\* puisque que le même aliment du stock peut être présent dans plusieurs pizzeria à la fois et une pizzeria a plusieurs aliments du stock.

Ensuite, nous avons une relation entre la classe **« Pizzeria »** et la classe « Commande » avec une relation 1, 1..\* puisqu’une commande ne peut appartenir qu’a une seule pizzeria et une pizzeria peut avoir plusieurs commandes.

Puis également une relation entre la classe **« PIZZA»** et la classe  
**« PIZZERIA»** ce qui correspondrait au menu par pizzeria, nous avons une relation 1..\*, \* puisque qu’une pizza peut appartenir à̀ toutes les pizzerias et qu’une pizzeria peut avoir plusieurs pizzas au menu.

La classe « **STOCK**» correspond au stock disponible par pizzeria, elle joue le rôle de suivi du stock des aliments.  
Elle est reliée à la classe **« PIZZERIA »** comme déjà̀ expliqué précédemment puis à la classe **« PIZZA »** puisque qu’une pizza est composé de différents ingrédients disponibles dans le stock, nous avons une relation de cardinalité́ \*, 1..\*, effectivement un ingrédient du stock peut être présent sur plusieurs pizzas et une pizza peut avoir différents aliments par sorte de pizza.

La classe **« PIZZA »** correspond aux différentes pizzas disponibles au menu et cela par pizzeria. Elle est liée à la classe **« PIZZA »** et la classe **« STOCK »** (déjà̀ mentionnée dans la classe pizza et la classe stock)  
Elle est également liée à̀ la classe **« COMMANDE »** puisqu’une commande va être composée d’aucune ou plusieurs pizzas d’où̀ la relation 0..\*, 0..\*. Et une pizza peut faire partie d’aucune commande ou bien de plusieurs commandes.

La classe **« COMMANDE »** représente le détail de l’ensemble commandes présentes pour chaque pizzeria, la relation entre la classe  
**« PIZZERIA »** et la classe **« COMMANDE »** a déjà̀ été traitée dans la classe **« PIZZERIA ».**Nous avons une relation entre la classe **« COMMANDE »** et la classe  
**« CLIENT »** et cette relation est 1, 0..\*, évidemment, une commande ne peut appartenir qu’a un seul client à la fois tandis qu’un client peut avoir aucune commande à plusieurs commandes.

## Application XXX...

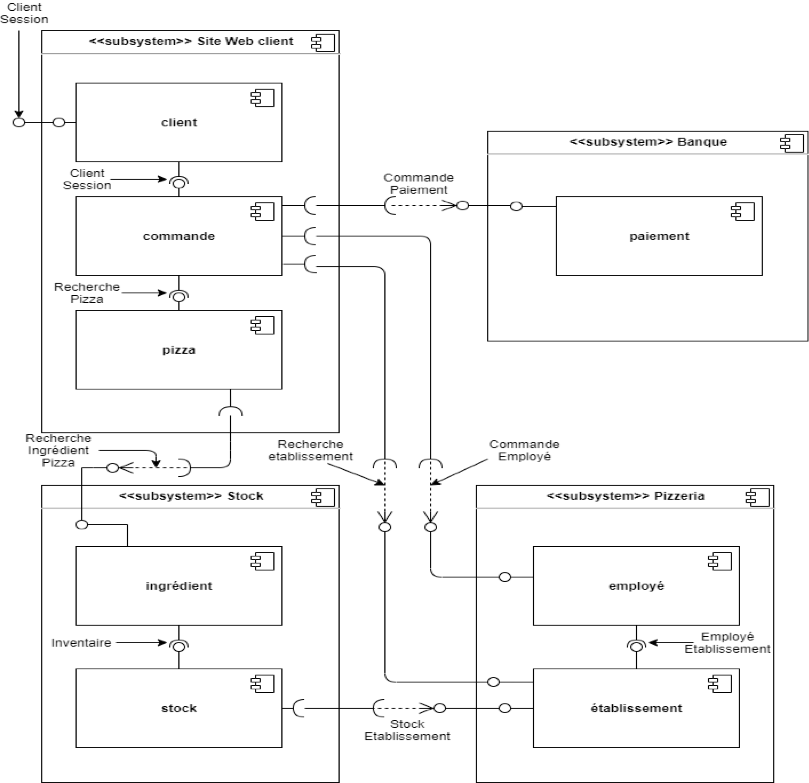
# ARCHITECTURE TECHNIQUE

## Application Web

La pile logicielle est la suivante :

* Application Python 3.8
* Serveur d’application Gunicorn 20.0.4
* Serveur web Nginx 1.19.1
* Database MySQL 8.0.19

Ci-dessous nous avons notre 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 proposé́ 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 à̀ différents corps de métier comme les livreurs, la caisse ou les pizzaiolos.

***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 passées 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 à̀ l’utilisateur de choisir son type de paiement, il peut payer en espèce (sur place ou à̀ 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 (pizzaiolo, livreur, vendeur, directeur), de donner son prénom, nom et de préciser dans quel établissement il ou elle travaille.

***4.1.7 - Composants pizzas***

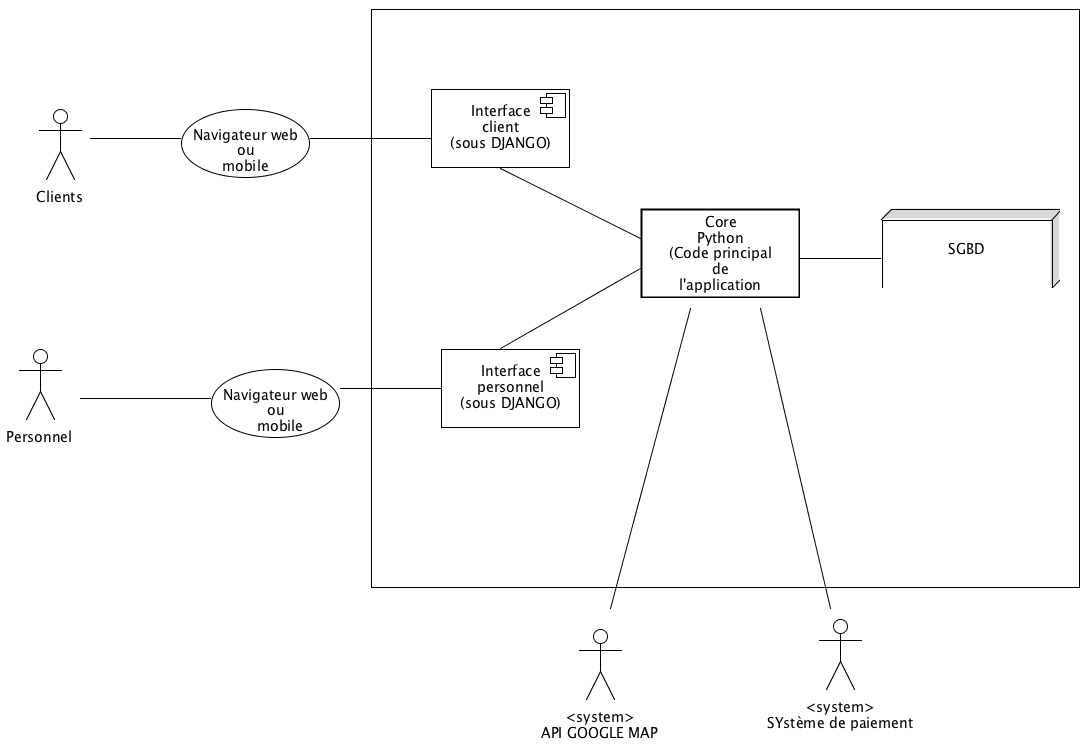
**Pizzas :** Ce composant va permettre d’avoir le numéro des différentes pizzas proposées, son nom pour les clients avec une description des ingrédients utilisés et son prix à l’unité́ hors promotion.

***4.1.8 - Composants stock***

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

# Architecture logicielle

Dans cette partie, nous retrouvons le diagramme de déploiement qui nous donnera une vue d’ensemble de l’architecture de production de notre application OC Pizza.



## 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 à̀ Londres

## 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.

## Application Xxx

…

# ARCHITECTURE LOGICIELLE

## 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 • employé  
• 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 │  
├───employé  
│ │ 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

# 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.

# Glossaire

|  |  |
| --- | --- |
| **GIT** | Git est un logiciel de versions décentralisé. |
| **Pip** | Pip est un gestionnaire de paquets utilisé pour installer et gérer les 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 ». |