# Solution technique

Dossier de conception technique

Version 1

**Auteur**
SIMONIN Cyril

# Table des matières

# 1 - Versions

| Auteur | Date | Description | Version |
|---|---|---|---|
| SIMONNIN Cyril | 29/04/2019 | Création du document | 1 |
| | | | |
| | | | |
| | | | |

# 2 - Introduction

## 2.1 - Objet du document

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

Objectif du document est de modéliser la conception technique de l'application.

Les éléments du présent dossier découlent:

- du dossier de conception fonctionnelle
- des besoins du client

## 2.2 - Références

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

1. **DCF – 3.2** : Dossier de conception fonctionnelle de l'application

# 3 - Architecture Technique

## 3.1 - Composants généraux

### *3.1.1 - Package Stock*

#### *3.1.1.1 - Composant stocks*

Ce composant permet de gérer le stock des ingrédients pour les différentes pizzerias, c'est le gestionnaire de stock ou le directeur général qui s'occupe d'ajouter les informations de stock, pour fonctionner il a besoin du composant ingrédient qui fournira la liste d'ingrédients.

### *3.1.2 - Package Administration*

#### *3.1.2.1 - Composant Utilisateurs*

Ce composant permet de créer, modifier et supprimer des utilisateurs, il a besoin du composant adresse qui stockera l'adresse ou les adresses de l'utilisateur, de typeAccounts qui permet de stocker le type de compte de l'utilisateur, si le type de compte est NULL il s'agit d'un utilisateur simple(client).

#### *3.1.2.2 - Composant Etablissements.*

Ce composant permet de gérer les établissements création, suppression et modification, il a besoin du composant adresse qui stocke les différentes adresses.

#### *3.1.2.3 - Composant Recettes.*

Ce composant permet de gérer les recettes disponibles création, suppression et modification, il utilise le composant ingrédient qui fournit les différents ingrédients disponibles.

#### *3.1.2.4 - Composant Ingredients*

Ce composant permet de gérer les ingrédients création, suppression et modification, d'ingrédient.

#### *3.1.2.5 - Composant Adresses.*

Ce composant permet de gérer les différentes adresses utilisateur ou d'établissements. création, suppression et modification.

#### *3.1.2.6 - Composant TypeAccounts*

Ce composant permet de gérer le type de compte des utilisateurs

## *3.1.3 - Authentification*

### *3.1.3.1 - Composant Utilisateurs*

Permets aux utilisateurs de s'authentifier sur le site, il utilise le composant adresse qui stockera l'adresse ou les adresse du client s'il en a plusieurs, ainsi que le type de compte si le type de compte est NULL c'est un utilisateur simple(client)

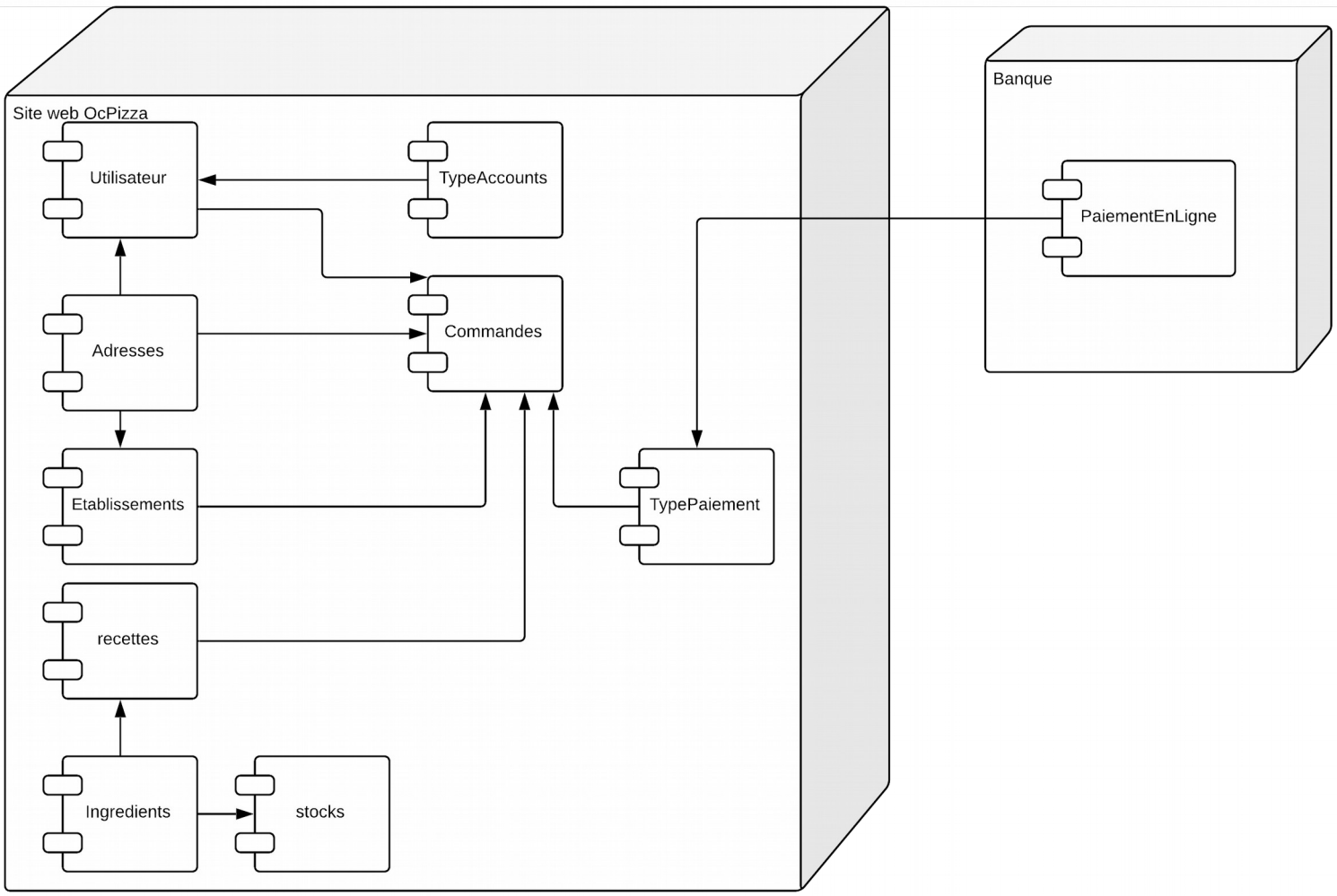
## *3.1.4 - Package Commande*

### *3.1.4.1 - Composant TypesPaiement*

Ce composant permet à l'utilisateur de choisir son type de paiement il peut payer en espèce(sur place ou à emporter) en CB (sur place, à la livraison ou en ligne), il est composé du composant paiementEnLigne de la banque si l'utilisateur décide de choisir le paiement en CB sur le site.

### *3.1.4.2 - Composant commandes*

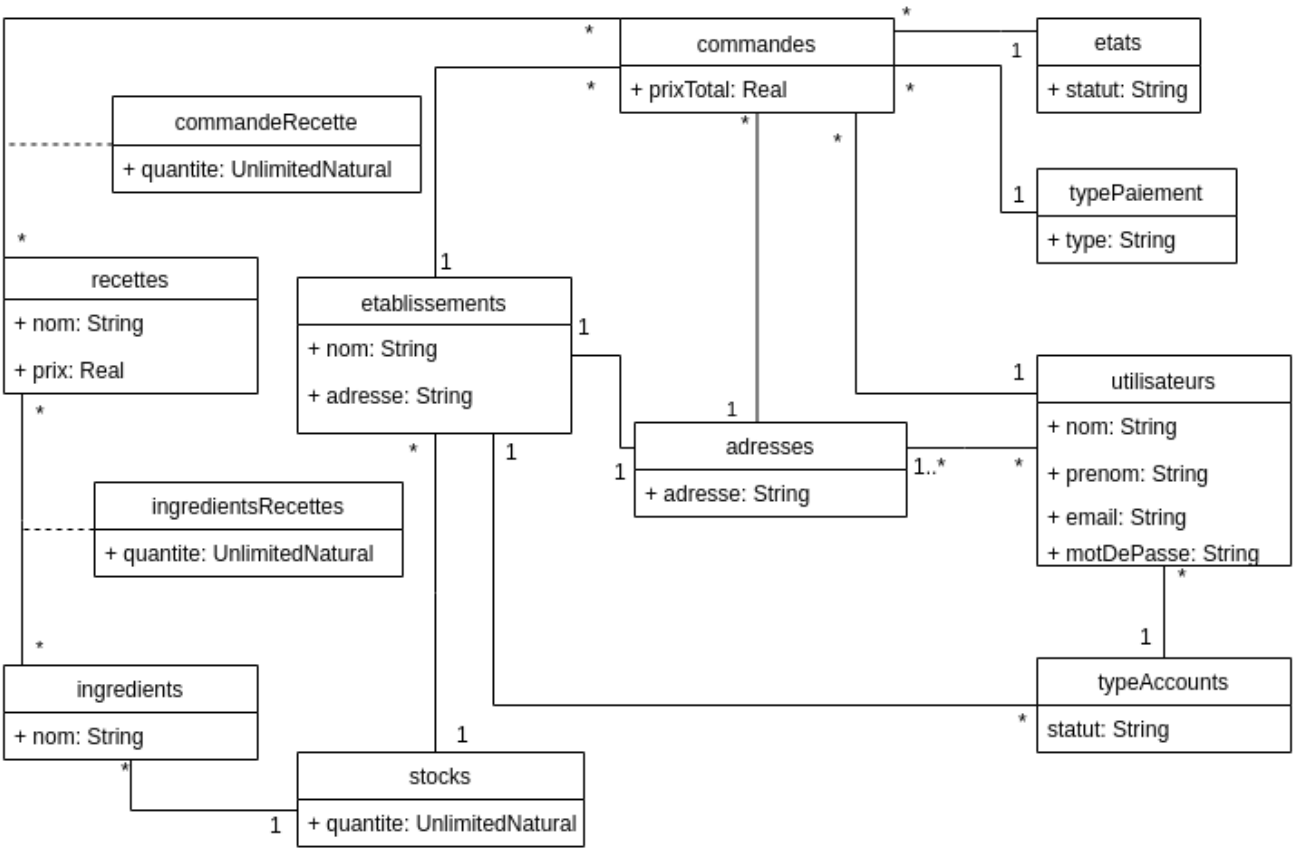
Ce composant enregistre les commandes passées par les clients, il utilise le composant utilisateur pour définir qui est l'utilisateur qui a commandé, le composant adresse pour l'adresse de livraison, si l'adresse est NULL alors c'est une commande à emporter à l'adresse de l'établissement, du composant établissement qui permet de définir sur quel établissement a été fait la commande, du composant recette qui permettra d'indiquer  les pizzas commandées, et tu types de paiement qui indique si l'utilisateur paie en CB ou en espèce.



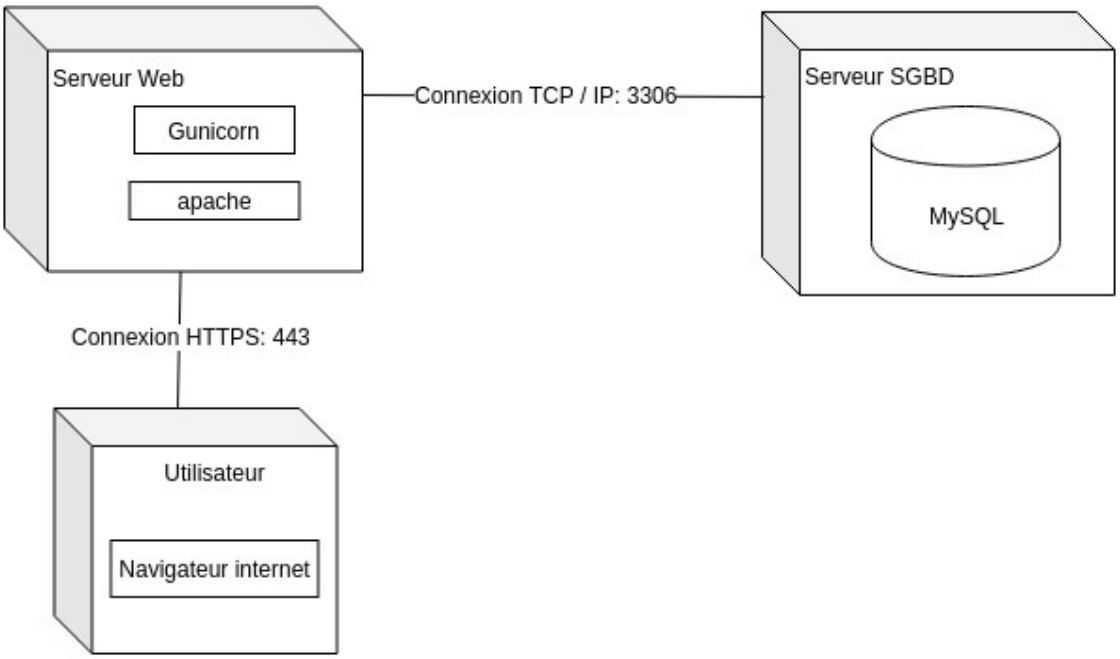

## 3.2 - Application Web

La pile logicielle est la suivante :

- Seveur d'application Application **Django** (V2.2) / **Python** (V3.7)/ **gunicorn** (V 19)
- Serveur web **Apache** (V2.4)
- Serveur de base de donnée **MySQL**(V8)


commandes
+ prixTotal: Real
etats
+ statut: String
commandeRecette
+ quantite: UnlimitedNatural
typePaiement
+ type: String
recettes
+ nom: String
+ prix: Real
etablissements
+ nom: String
+ adresse: String
utilisateurs
+ nom: String
+ prenom: String
+ email: String
+ motDePasse: String
adresses
+ adresse: String
ingredientsRecettes
+ quantite: UnlimitedNatural
ingredients
+ nom: String
typeAccounts
statut: String
stocks
+ quantite: UnlimitedNatural

# 4 - Architecture de Déploiement




## 4.1 - Serveur de Base de données

Le serveur de base de données permet de stocker les données du site web

Caractéristiques techniques du serveur:

**OS:** Debian Stretch(V9.9)

**Ram:** 4Go

**Processeur:** 1Core

**Stockage:** 40 Go(extensible)

Informations importantes:

Nous préconisons la mise en place d'un CRONTAB afin de faire des sauvegardes régulières de la base de données, éventuellement avec sauvegarde externe sur support physique.

## Serveur Web

Le serveur Web il permet l'interaction entre l'utilisateur et la base de données, ainsi que l'affichage du site web.

Caractéristiques techniques du serveur:

**OS:** Debian Stretch(V9.9)

**Serveur HTTP:** Apache(V2.4)

**Ram:** 4Go

**Processeur:** 1Core

**Stockage:** 40 Go(extensible)

# 5 - Architecture logicielle

## 5.1 - Principes généraux

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

### *5.1.1 - Les couches*

L'architecture applicative est la suivante :

- Une couche **Model**: Permet la comunication avec la BDD.
- Une couche **View**: Permet la visualisation de l'information.
- Une couche **Template**: Permet à la couche view une interaction avec la couche model.

### *5.1.2 - Les modules*

Modules mysql-connector-python permet à l'application de communiqué avec la BDD MySQL

### *5.1.3 - Structure des sources*

La structuration des répertoires du projet suit la logique suivante :

– les répertoires sources sont crées de façon à respecter la philosophie Django

```
Racine
├── manage.py
├── mon_app
│   ├── admin.py
│   ├── forms.py
│   ├── __init__.py
│   ├── models.py
│   ├── static
│   │   ├── css
│   │   │   └── style.css
│   │   └── js
│   │       └── behavior.js
│   ├── templates
│   │   └── mon_app
│   │       └── index.html
│   ├── tests.py
│   ├── urls.py
│   └── views.py
└── project
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

# 6 - Points particuliers

## 6.1 - Gestion des logs

La gestion des logs sera faite par Django à l'aide du module logging intégré à python, les logs seront conserver pour une periode d'un mois.

## 6.2 - Environnement de développement

Trois environnement seront utilisé **DEV**, **TEST** et **PROD.**

- L'environnement de **DEV** permettra le developpement de l'application
- L'environnment **TEST** permettra de mettre en post-production l'appplication et de vérifier son bon focntionnement
- L'environnement de **PROD** sera utilisé pour la mise en production de l'application après validation des tests.