

OCPizza

OC PIZZA

**Pour une application web de vente de pizza en ligne
avec sa logistique**

Dossier de conception technique
Version 1.0



Angelique F.
Responsable Dev

OCPizza

TABLE DES MATIÈRES

1.Versions	3
2.Introduction	4
2.1.Objet du document	4
2.2.Références.....	4
3.Architecture Technique	5
3.1.Composants généraux	5
3.1.1.L'application.....	5
3.1.2.Le front-end	5
3.1.3.Le back-end	5
3.2.MPD	6
3.2.1.La présentation des tables	6
4.Architecture de Déploiement	9
4.1.Diagramme UML de déploiement	9
5.Architecture logicielle	10
5.1.Principes généraux.....	10
5.1.1.Les couches	10
5.1.2.Les modules.....	10
5.2.Diagramme de composant	10
5.2.1.Structure des sources.....	11
6.Points particuliers.....	12
6.1.Gestion des logs	12
6.2.Gestion des connexions.....	12
6.3.Environnement de développement.....	12
7.Glossaire	13

OCPizza

1. VERSIONS

Auteur	Date	Description	Version
Angelique Fourny	02/04/2021	Création du document	1.0

OCPizza

2. INTRODUCTION

2.1. Objet du document

L'objectif de ce document est d'aider les développeurs pour la conception, la maintenance et l'évolution de l'application.

2.2. Références

Pour de plus amples informations, se référer :

1. **PROJET_08_Dossier_fonctionnelle** : Dossier de conception fonctionnelle de l'application
2. **PROJET_08_Dossier_d_exploitation** : Dossier d'exploitation de l'application
3. **PROJET_08_Proces_verbal** : Procès-verbal de livraison finale

Les documents sont consultables sur Github sur OC_Pizza_P03_Complet_Tech :

https://github.com/fangel19/OC_Pizza_P03_Complet_Tech.git

OCPizza

3.ARCHITECTURE TECHNIQUE

3.1.Composants généraux OCPizza

3.1.1.L'application OCPizza

3.1.2.Le front-end

L'application sera réalisé en   et  NGULAR

3.1.3.Le back-end

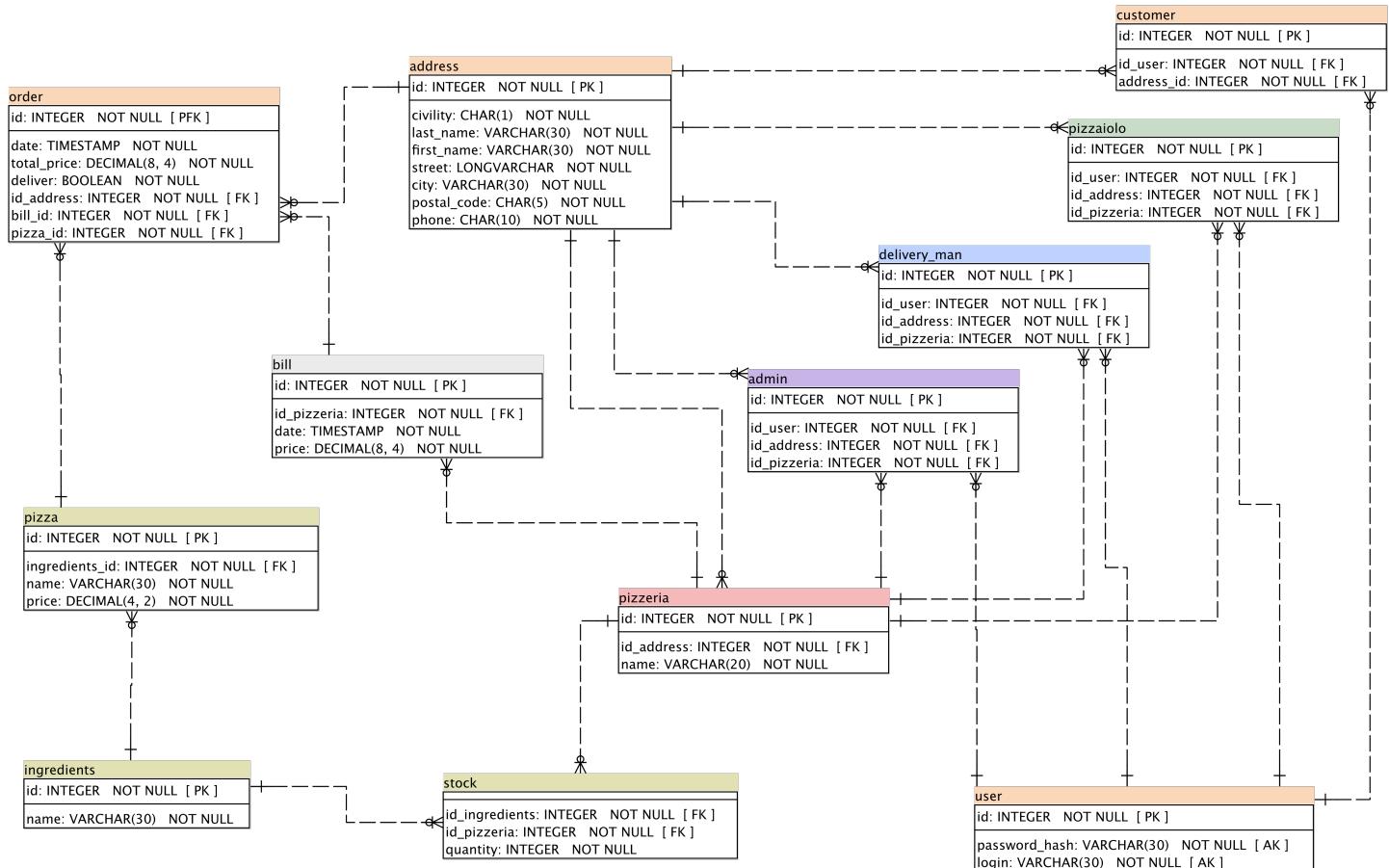
sera réalisé en language  python™ avec son framework  il communiquera avec  HEROKU

La base de données en SGBP (système de gestion de base de données) est utilisés avec  PostgreSQL et gère avec  mongoDB.

OC Pizza

3.2.MPD

A partir du diagramme de classes présenté dans **PROJET_08_Dossier_fonctionelle**, nous avons élaboré le MPD (Modèle Physique de Données).



3.2.1.La présentation des tables

La Table **address** :

Elle regroupe toutes les informations pour **les adresses**. Toute la Table sera NOT NULL.

Une clé primaire PK « id », en auto-incrémentation et de type INTEGER (Int)

- Civility de type CHAR(1) = 1 caractère.
- last_name, first_name et city VARCHAR(30) <= 30 caractères
- street LONGVARCHAR = caractères illimités
- postal_code CHAR(5) = 5 caractères (ex 75000)
- Phone CHAR(10) = 10 caractères (0231961919)

OCPizza

Les Table **customer**, **pizzaiolo**, **delivery_man**, et **admin** :

Elles auront toutes une clé primaire « id », en auto-incrémentation et de type INTEGER (Int) NOT NULL. Avec des associations clés étrangères FK sur **id_user**, **id_address**, et pour tous sauf **customer** une association avec **id_pizzeria**

La Table **user** :

Elle regroupe toutes les informations pour la connexion de chaque utilisateur. Toute la Table sera NOT NULL. La PK « id », en auto-incrémentation et de type INTEGER (Int)

- password_hash de style VARCHAR(30) = le mot de passe <= 30 caractères
- Login VARCHAR(30) = identifiant <= 30 caractères

La Table **order** :

Elle regroupe toutes les informations concernant les **commandes**. Toute la Table sera NOT NULL.

Une clé primaire PK « id », en auto-incrémentation et de type INTEGER (Int)

- date de type TIMESTAMP = pour la date et l'heure
- total_price DECIMAL(8, 4) = il y aura 8 chiffres en tout
- Délivre BOOLEAN = pour avoir le choix du livreur

Avec des associations FK **id_address**, **bill_id**, **pizza_id**

La Table **bill** :

Elle regroupe toutes les informations de chacune des **factures**. Toute la Table sera NOT NULL. La PK « id », en auto-incrémentation et de type INTEGER (Int)

- date de type TIMESTAMP = pour la date et l'heure
- price DECIMAL(8, 4) = il y aura 8 chiffres en tout pour le montant de la facture

Elle aura en association FK **id_pizzeria**

La Table **pizzeria** :

Elle regroupe toutes les informations des **pizzéries**. Toute la Table sera NOT NULL. La PK « id », en auto-incrémentation et de type INTEGER (Int)

- name VARCHAR(20) <= 20 caractères pour le nom

Elle aura en association FK **id_address**

OCPizza

La Table **pizza** :

Elle regroupe toutes les informations pour les **pizzas**. Toute la Table sera NOT NULL.

Une clé primaire PK « id », en auto-incrémentation et de type INTEGER (Int)

- name VARCHAR(30) <= 30 caractères pour le nom de chacune des pizzas
- price DECIMAL(4, 2) = il y aura 4 chiffres pour le prix de chacune des pizzas

Avec une association clé étrangère FK **ingredients_id**

La Table **ingredients** :

Elle regroupe les informations pour les **ingrédients** utilisés. Toute la Table sera NOT NULL.

La PK « id », en auto-incrémentation et de type INTEGER (Int)

- name VARCHAR(30) <= 30 caractères pour le nom des ingrédients

La Table **stock** :

C'est une Table association entre la Table **ingrédients** et **pizzeria**

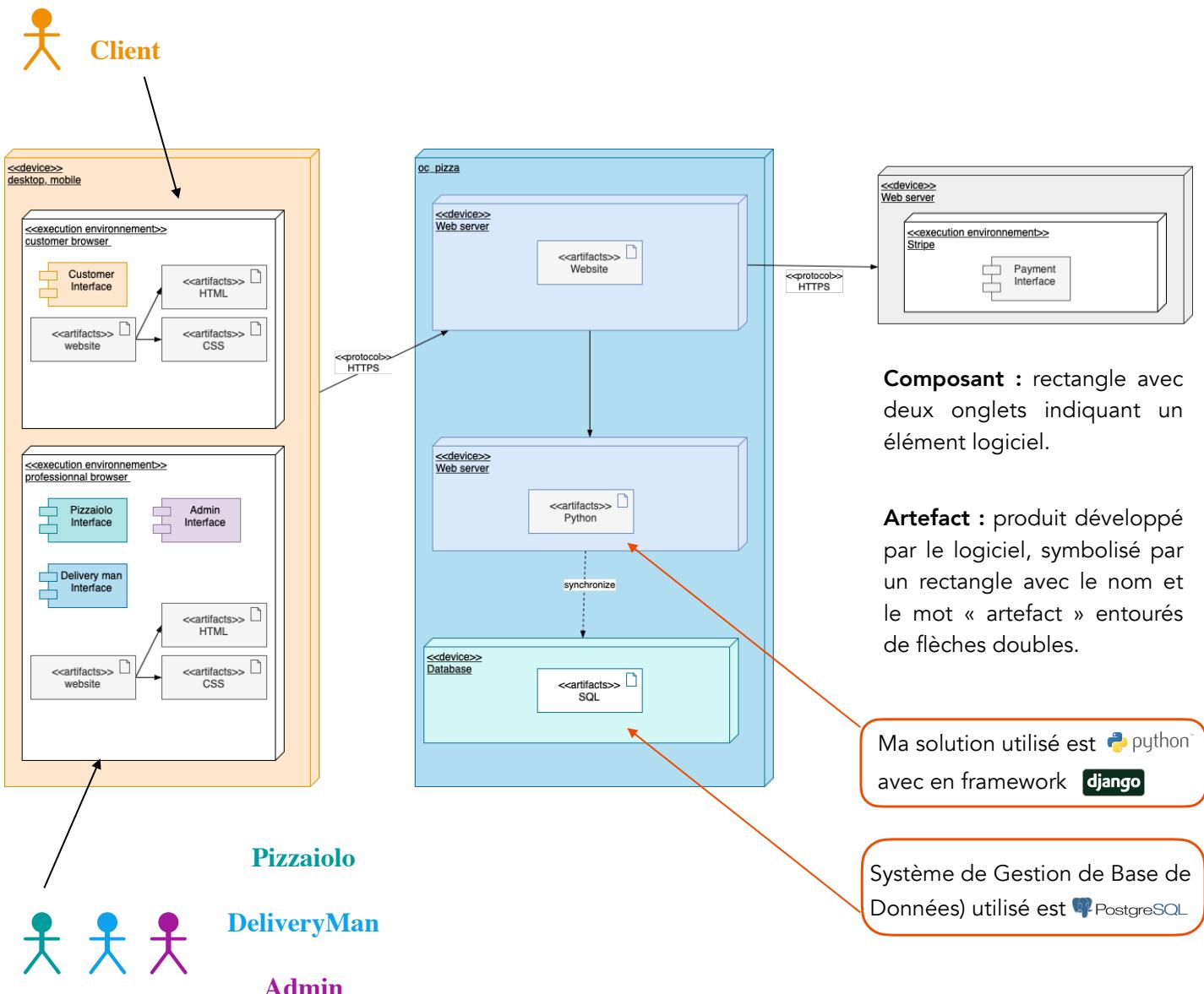
- quantity INTEGER = quantité en int. Associée à **l'id_ingredients** et **id_pizzeria**

OCPizza

4.ARCHITECTURE DE DÉPLOIEMENT

4.1.Diagramme UML de déploiement

Il décrit le déploiement physique des informations générées par le logiciel sur des composants matériels. On appelle artefact l'information qui est générée par le logiciel



OCPizza

5.ARCHITECTURE LOGICIELLE

5.1.Principes généraux

Les sources et versions du projet sont gérées par **Git** et hébergées sur **Github**.

5.1.1.Les couches

L'architecture applicative est la suivante:

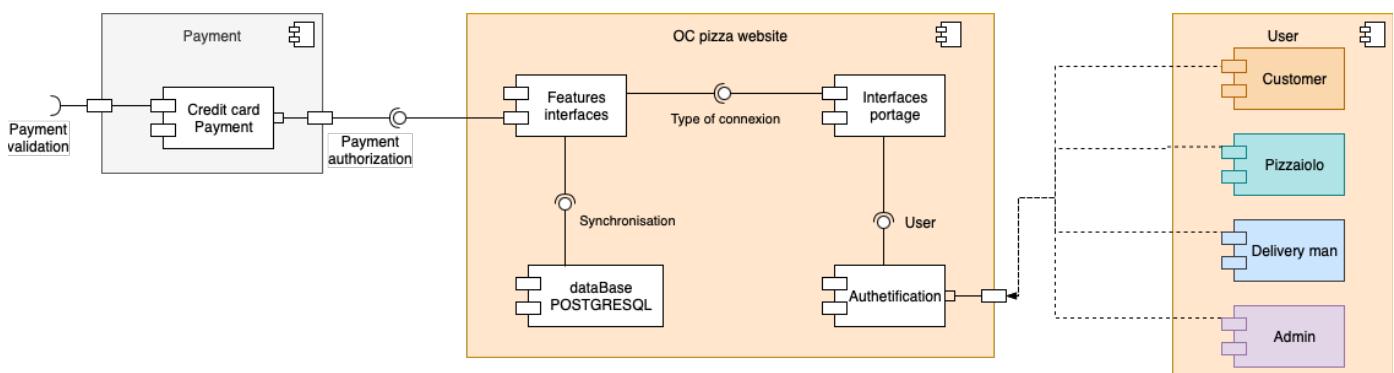
- Une couche **model** : implémentation du modèle des objets métiers
- Une couche **template** : représente les gabarits des données renvoyées
- Une couche **vue** : donne accès au modèle et au template adapté à la requête

5.1.2.Les modules

Django-Cron : Permet de créer des tâches « Cron » pouvant s'exécuter à intermédiaire régulier, comme une mise à jour de la base de données vis-à-vis d'une source externe.

L'application étant développé avec Django, nous tirerons partie de ses forces et adopteront un modèle Model/View/Template.

5.2.Diagramme de composant

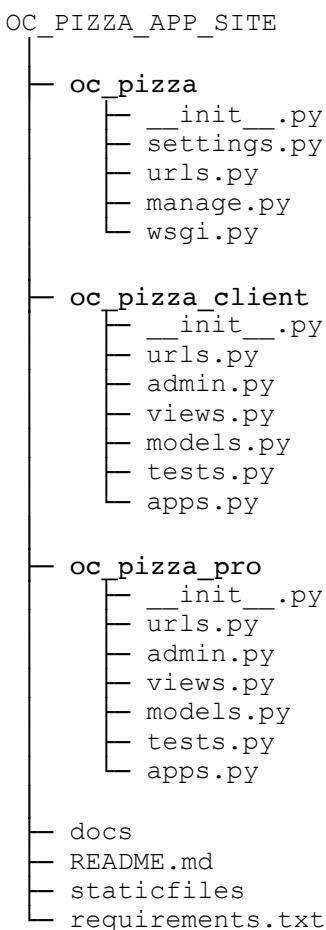


OCPizza

5.2.1. Structure des sources

La structuration des répertoires du projet suit la logique suivante :

- L'application `oc_pizza` : logique de l'application web.
- une application client : l'offre de vente de pizzas en ligne
- une application pro : les fonctions de gestion destinées aux employés et dirigeants



- `__init__.py` permet de déclarer un dossier comme un package importable
- `settings.py` contient la configuration du site web
- `urls.py` contient une liste de patterns d'urls utilisés
- `manage.py` est un script qui aide à gérer ou maintenir le site
- `wsgi.py` norme qui sert à définir comment un serveur Python et son application peuvent communiquer
- `admin.py` permet d'ajouter, modifier ou supprimer des données
- `views.py` contient les vues
- `models.py` permet de définir les objets
- `staticfiles` contient les fichiers html, css, javascript et les images
- `requirements` maintient une liste des dépendances

OCPizza

6.POINTS PARTICULIERS

6.1.Gestion des logs

La gestion des lois peut se faire sur  HEROKU

6.2.Gestion des connexions

Les connexions seront gérés par  django

6.3.Environnement de développement

Environnement de développement sublime text  et  POSTMAN

Pour tester le routage HTTP

OCPizza

7.GLOSSAIRE

SGDB	Système de Gestion de Base de Données
MPD	Modèle Physique de données
MVT	Model - View - Template
Python	Python est un langage de programmation multiplateformes
Django	Django est un cadre de développement web (framework) open source en Python.