OC Pizza

Solution de gestion du groupe de restaurants OC_Pizza

Dossier de conception technique

Version 1.0

Auteur
Pierre Sempéré
Développeur

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
Pierre Sempéré	13/08/2021	Création du document et ébauche	0.1.1
Pierre Sempéré	26/08/2021	Impression pour revues des erreurs.	0.1.2
Pierre Sempéré	09/09/2021	Éditions majeures, correction des erreurs en vue d'entretient	0.2
Pierre Sempéré	21/12/2021	Revues et corrections finales	1.0

2 - INTRODUCTION

2.1 - Objet du document

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

L'objectif du document suivant est la description de la conception technique de l'application OC_Pizza_Web, permettant une explication du fonctionnement de l'application.

Les éléments du présent dossier découlent :

- D'un entretien avec le client et de ses retours à la suite de cet entretien.
- Des outils et connaissances technologiques à la disposition du groupe IT_Consulting.

2.2 - Références

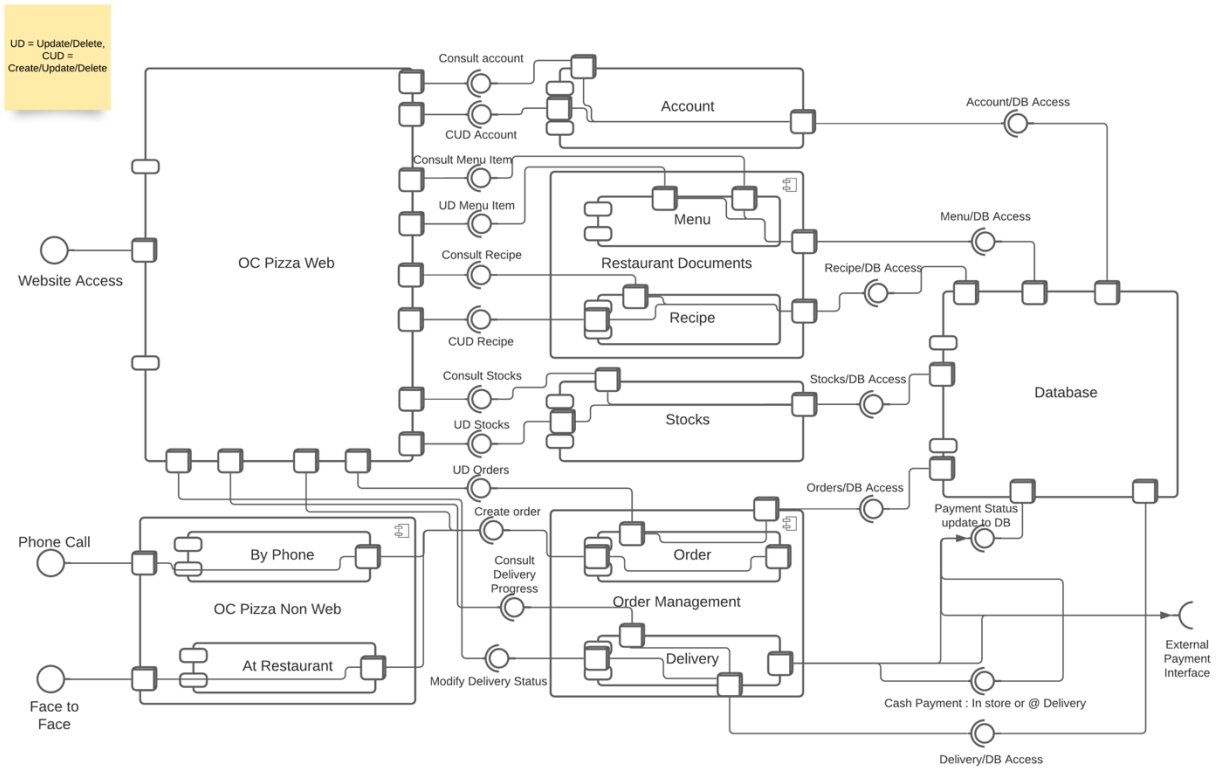
Pour de plus amples informations, se référer également aux éléments suivants :

1. Dossier de conception fonctionnelle de l'application
2. Dossier d'exploitation

3 - Architecture Technique

3.1 - Composants généraux

3.1.1 - Solution OC_Pizza



3.1.1.1 - Composant By Phone & At Restaurant

Permet au client de passer sa commande par téléphone ou directement au restaurant – Composant déjà présent dans l'organisation du restaurant intégrée à la solution.

3.1.1.2 - Composant Order

Représente la commande passée par un client ou HDC, on peut consulter la commande pour vérifier sa progression.

3.1.1.3 - Composant Delivery

Permet d'assurer la coordination entre les livreurs, restaurants et clients pour la livraison des commandes.

3.1.1.4 - Composant Menu

Ce composant permet au client de visualiser le catalogue et d'ajouter des produits a son panier.

3.1.1.5 - Composant Recipe

Permet au cuisinier de consulter la recette de n'importe quel plat présent sur le catalogue s'il en a besoin.

3.1.1.6 - Composant Stocks

Permet au personnel de visualiser les stocks et les mettre à jour pour avoir une vue d'ensemble sur la situation des restaurants.

A noter que IT Consulting et OC Pizza se sont accordés sur le fait d'avoir recours à des moyens de paiements externes, indépendants du groupe IT Consulting, ces composants ne sont donc pas connus du groupe et ne serons pas développés ici.

3.2 - Pile Logicielle

La pile logicielle est la suivante :

- Python v3.9
- PostgreSQL v13
- Gunicorn v20.1.0
- NginX v1.21.1
- Django et autres packages Python disponible dans le document requirements.txt

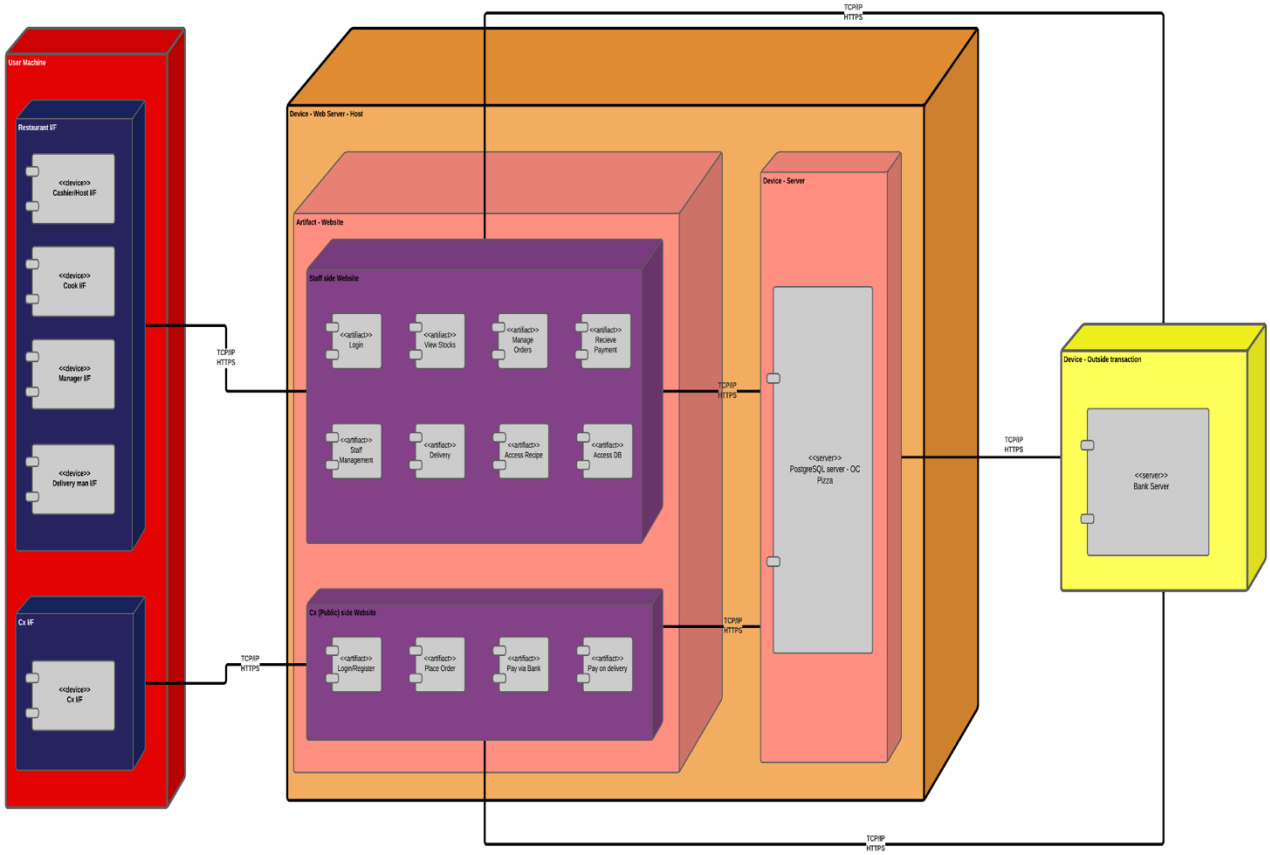
4 - Architecture de Deploiement

4.1 - Serveur de Base de données

PostgreSQL 13

Caractéristiques techniques : Serveur Linux Ubuntu Focal Fossa + PostgreSQL 13.X)

4.2 - Serveur OC_Pizza_web_DB



5 - ARCHITECTURE LOGICIELLE

5.1 - Principes généraux

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

5.1.1 - Les couches

L'architecture applicative est la suivante :

- Une couche **présentation** (Template en terme propres à Django). Cette couche représente ce que l'utilisateur (client ou employé) voit. Cela correspond aux gabarits HTML ainsi que les fichiers CSS et Javascript auxquels les gabarits HTML peuvent faire appel ; les fichiers staticfiles de Django, c'est-à-dire les éventuels médias et autres composants utilisés pour le style de l'application.
- Une couche **métier et contrôle**. Cela représente tout le code, ici en Python, faisant entre la couche présentation et la couche d'accès aux données. Cela prend en compte les scripts Python écrits pour que les deux couches citées précédemment communiquent entre elles. Dans le vocabulaire propre à Python et Django, ce sont les vues, les routeurs et leurs classes, les éventuels signaux etc.
- Une couche **d'accès aux données** qui sont ici les classes de **l'ORM** Django définies dans les fichiers models.py de chaque application. Cette couche est responsable de la gestion de base de données, ici PostreSQL 13.

5.1.2 - Les modules

Les modules et dépendances Python sont gérés par Pip, EasyInstall et PyPi.

5.1.3 - Structure des sources

Structure Classique projet Django

La structuration des répertoires du projet suit la logique suivante :

```
oc_pizza_project
  ├─ manage.py
  ├─ <administration>
  |    ├─ admin.py
  |    ├─ apps.py
  |    ├─ forms.py
  |    ├─ models.py
  |    ├─ signals.py
  |    ├─ view.py
  |    ├─ tests.py
  |    ├─ migrations
  |    |   ├─ 0001_initial.py
  |    ├─ templates
  |    |   ├─ administration
  |
  ├─ <client_interface>
  |    ├─ admin.py
  |    ├─ apps.py
  |    ├─ forms.py
  |    ├─ models.py
  |    ├─ urls.py
  |    ├─ view.py
  |    ├─ tests.py
  |    ├─ migrations
  |    |   ├─ 0001_initial.py
  |    ├─ templates
  |    |   ├─ client_interface
  ├─ <oc_pizza>
  |    ├─ asgi.py
  |    ├─ urls.py
  |    ├─ wsgi.py
  |    ├─ settings
  |    |   ├─__init__.py
  |    |   ├─production.py _
```

6 - POINTS PARTICULIERS

6.1 - Gestion des logs

Sentry sera utilisé comme package de monitoring de l'application Django. Différents niveaux de retours peuvent être configurés pour que les retours de Sentry soient le plus pertinents possibles.

6.2 - Fichiers de configuration

La solution étant créée via Django, tous les éléments de configuration sont présents dans les réglages de Django définis par la variable DJANGO_SETTINGS_MODULE. En développement cela prendra en compte le fichier settings.py présent dans le dossier *oc_pizza_project/oc_pizza/settings.py,* en production, les éléments de configuration seront places dans un fichier *oc_pizza_project/oc_pizza/settings/production.py* qui importe les éléments de configuration présent dans le fichier __init__.py non versionnée pour des raisons de sécurité.

Datasources : Un dump sql (db_dump.sql) sera fourni pour permettre d'initialiser la base de données avec la configuration souhaitée et des valeurs prédéfinies.

6.3 - Ressources

Les ressources utilisées par l'application seront présentes dans le dossier 'staticfiles' conformément aux recommandations officielles de Django.

6.4 - Environnement de développement

Le projet sera développé en continu sur GitHub (branche staging) pour versionner le code de l'application tout en permettant au client d'appliquer lui-même les mises à jour.

6.5 - Procédure de packaging / livraison

La livraison de la solution sera une archive tar.gz (facile à décompresser sous linux) contenant tous les fichiers de configuration nécessaires, les fichiers statics et le code source. Cela sera envoyé au client via un FTP sécurisé à déterminer entre OC Pizza et IT Consulting.

7 - GLOSSAIRE

Pip/Easy Install/PyPi	Gestionnaires des dépendances de paquets Pythons, c'est-à-dire les librairies externes dont le code a besoin pour s'exécuter correctement.
ORM	Acronyme d'Object-Relational Mapping. Dans le cas d'une application, c'est un utilitaire permettant d'interagir avec une base de données dans le langage Python, évitant ainsi de mélanger les langages SQL et Python dans les mêmes documents.