OPENCLASSROOMS

My_solution

OC Pizzéria

Gestion de commande

Dossier de conception technique

Version 2.0

Auteur
Jean baptiste Servais
Developpeur

1 - VERSIONS

Auteur	Date	Description	Version
Jean Baptiste servais	18/04/19	Création du document	01/02/00

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application gestion de commande OcPizzéria.

Objectif du document est de présenter les outils permettant la mise en oeuvre du site web. Par la base de donnée et le framework Django.

Les éléments du présents dossiers découlent :

- de reunions avec le personnel d'OcPizzéria,
- et de reunion avec les différents developpeurs qui vont travailler sur ce projet (font et back).

2.2 - Références

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

1. **DCF – 2.2** : Dossier de conception fonctionnelle de l'application
2. **DE – 2.2** : Dossier d'exploitation de l'application

3 - Architecture Technique

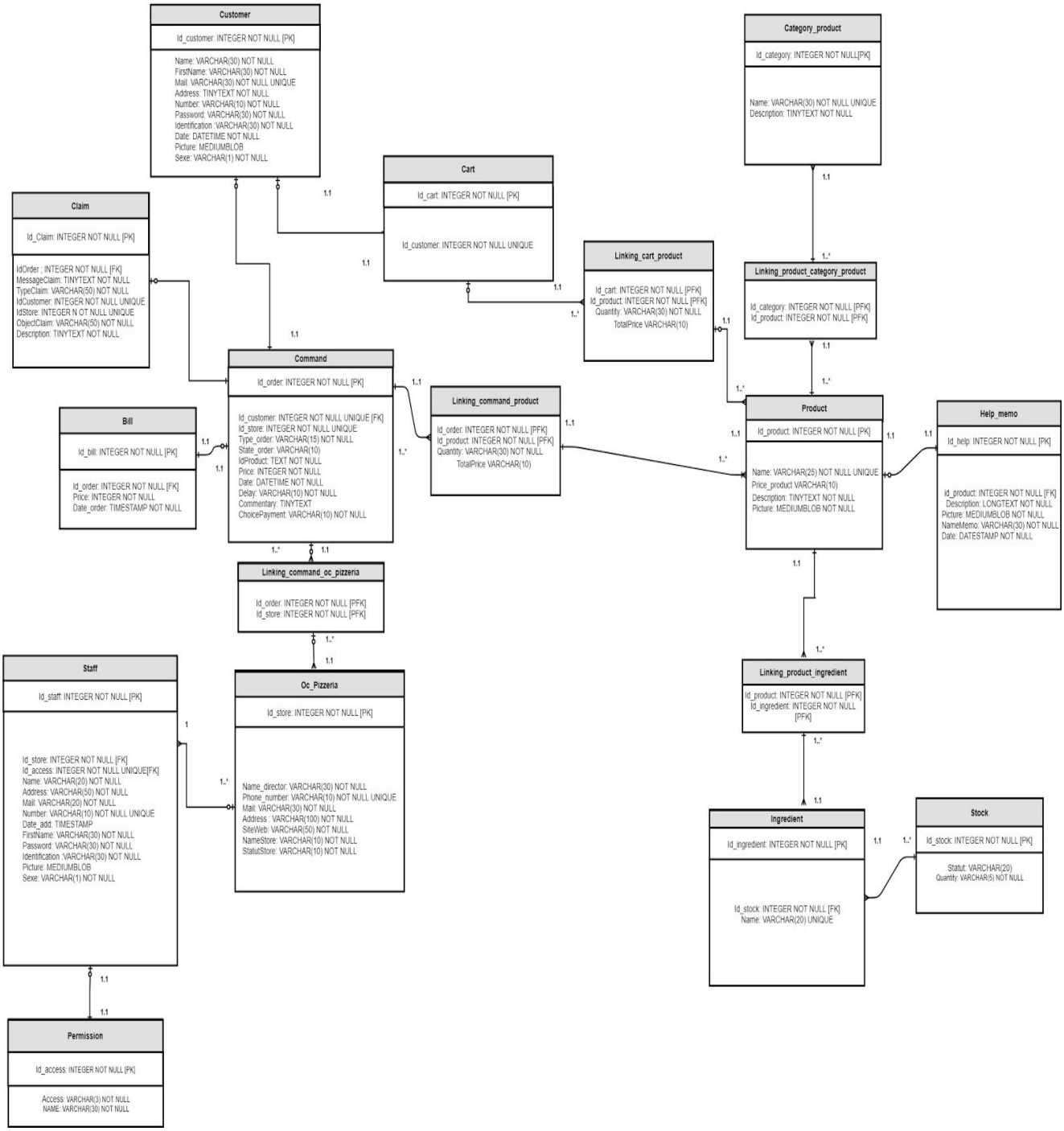
3.1 - Application Web

Afin de pouvoir faire notre application nous allons nous servir d'une base de de type SGBD (système de gestion de base de donnée) Postgresql et du framework Django par les langages informatiques HTML, Javascript (pour les pages) et Python (pour le site web).

3.2 - Serveur de Base de données

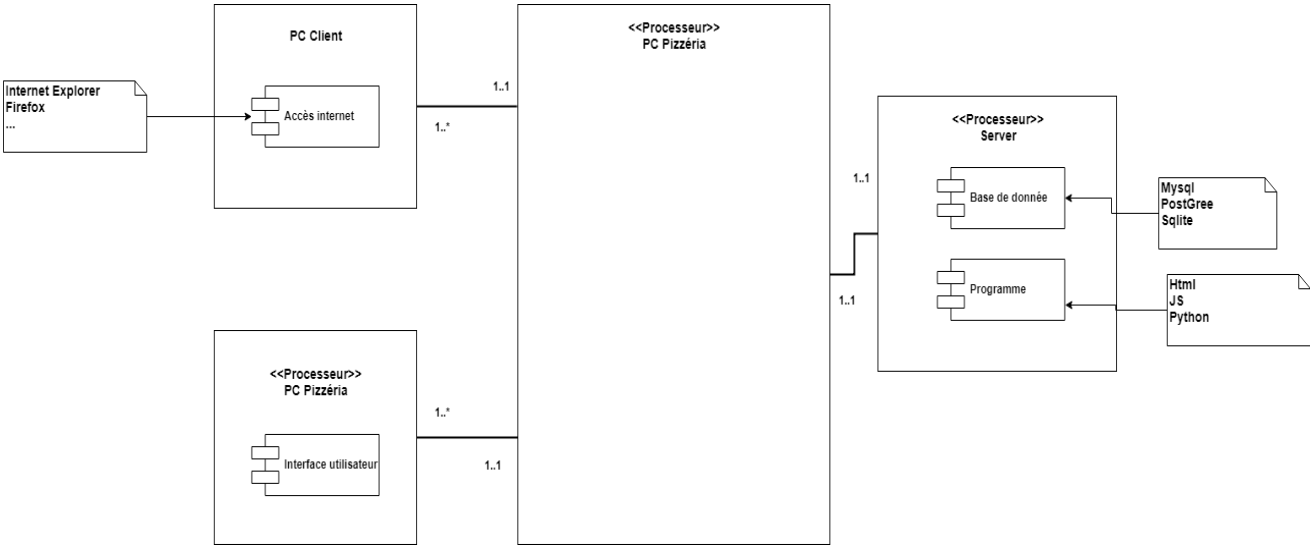
Nous déploieront notre base de donnée ainsi que notre site web sur le serveur d'Heroku. Heroku est un PassS (plateforme en tant que service).

3.3 - Modèle physique de données



4 - ARCHITECTURE LOGICIELLE

4.1 - Déploiment de l'application



Voici le diagramme de déploiement modélise l'architecture physique d'un système tant au niveau logiciel que physique. Ce diagramme possède quatre composants. Le pc des utilisateurs, que ce soit le client nécessitant un accès internet (que ce soit firefox, internet explorer…) ou du membre du personnel, doivent être en relation de dépendance avec le composant: processeur Oc pizzéria. Cet interface est en association avec le processeur serveur qui contient les composants de la base de donnée sous Mysql, Postgree ou bien Sqlite. En plus du composant programme composé de code HTML, Js, Python...

Nous avons donc vu le diagramme de déploiement de l'infrastructure physique et logiciels des composants. Au niveau du PC du client qui nécessite un accès internet afin de pouvoir communiquer avec le processeur OC pizzéria contenant le composant interface utilisateur lui même en interaction avec le composant serveur web contenant la base de donnée ainsi que le programme.

4.2 - Déploiment de l'application

Nous déploieront notre application sur Heroku qui est un PaaS c'est à dire une plateforme as a service. En effet, le service est gratuit et répond à nos besoins.

4.3 - Principes généraux

4.3.1 - Application Django

Le framework Django est est un outil python de haut niveau gratuit et opensource. Nous nous servieront de se servir afin de developper notre application. Nous feront 2 applications l'une située client et l'autre situé personnel. Les deux seront reliées par notre base de donnée. Nous utiliseront le principe de MVT modèle, vue, templates.

Nos modèles servieront à faire les comptes des utilisateurs tant au niveau du client qu'au niveau du personnel, le stockage des produits, la liste des produits disponibles en stock pour l'utilisateur, les prix des produits ainsi que de leur détail (image, prix, constitution...).

Nos templates ou gabarit seront l'interface utilisateur. Nous avons choisi un thème bootstrap après conertation avec le directeur d'OcPizzéria.

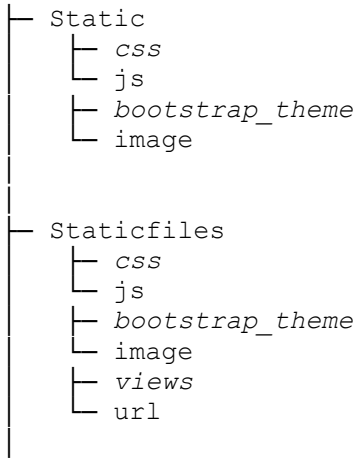
La views ou controleur permettra d'effectuer des opérations sur la base de donnée (soustraction du stock, ajout d'utilisateur en base de donnée, requete http ect...)

4.3.2 - Structure des sources

La structuration des répertoires du projet suit la logique suivante :

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

```
racine
 ├─ Procfile
 ├─ requirements.txt
 ├─ .gitignore.txt
 ├─ runtime.txt
 ├─ manage.py
 │
 ├─ Platefome
 │    ├─ __init__.py
 │    └─ prod_setting.py
 │    ├─ Setting
 │    └─ url
 │    ├─ views
 │    └─ wsgi.py
 │    ├─ tests.py
 │    ├─ models.py
 │    ├─ forms.py
 │    ├─ migrations.py
 │
 ├─ Personnel
 │    ├─ __init__.py
 │    └─ prod_setting.py
 │    ├─ Setting
 │    └─ url
 │    ├─ views
 │    └─ wsgi.py
 │    ├─ tests.py
 │    ├─ fonctions_associées
 │    ├─ models.py
 │    ├─ forms.py
 │    ├─ migrations.py
 │
 ├─ Client
 │    ├─ __init__.py
 │    └─ prod_setting.py
 │    ├─ Setting
 │    └─ url
 │    ├─ views
 │    └─ wsgi.py
 │    ├─ tests.py
 │    ├─ fonctions_associées
 │    ├─ models.py
 │    ├─ forms.py
```



4.4 - Application Web

...

Si besoin, diagramme UML de composants pour monter les différents modules et leur inter-dépendances

4.5 - Application Xxx

...

5 - Points particuliers

5.1 - Gestion des logs

...

5.2 - Fichiers de configuration

5.2.1 - Application web

...

5.2.1.1 - Datasources

...

5.2.1.2 - Fichier xxx.yyy

...

5.2.2 - Application Xxx

...

5.3 - Ressources

...

5.4 - Environnement de développement

5.5 - Procédure de packaging / livraison

5.6 - XXX

...

6 - GLOSSAIRE
