OPENCLASSROOMS

My_solution

OC Pizzéria

Gestion de commande

Dossier de conception technique

Version 2.0

Auteur
Jean baptiste Servais
Developpeur

PLAN

1 Version .....p 3
2 Introduction.....p4
2.1 objet du document.....p4
2.2 Références.....p4
3 Architecture technique.....p5
3.1 application web.....p5
3.2 serveur de base de données.....p5
3.3 modele physique de données.....p6
4 architecture de déploiement.....p7
4.1 déploiment de l'application.....p7
5 architecture logicielle.....p8
5.1 principes généraux.....p8
5.1.1 application django.....p8
5.1.2 structure des sources.....p9
5.2 application web.....p11
6 points particuliers.....p13
6.1 ressources.....p13
6.1.1 application web.....p13
6.2 environement de développement.....p13
6.2.1 application web.....p13
6.3 procédure de packaging/livraisin.....p13

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

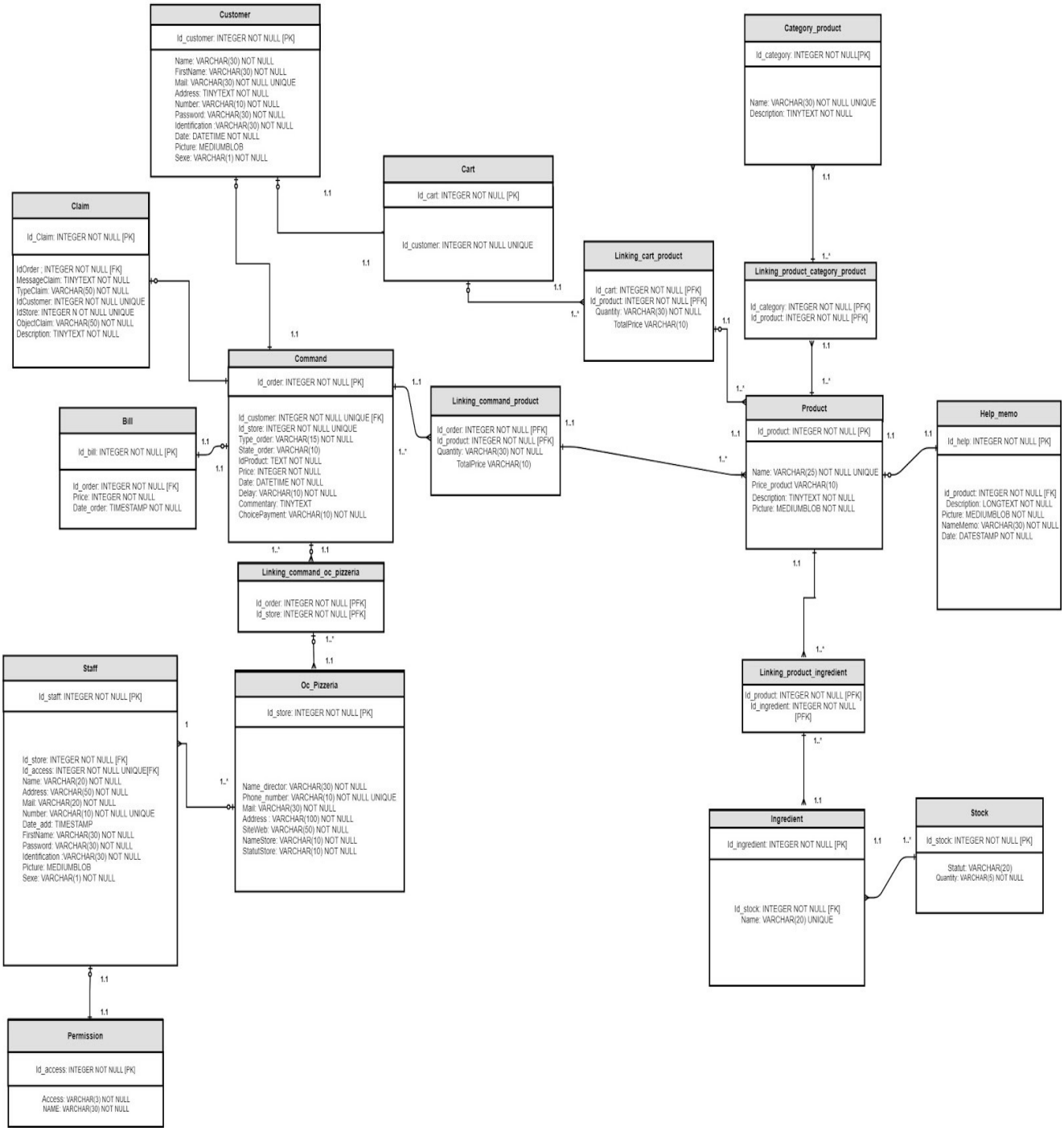
3.1 - Application Web

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

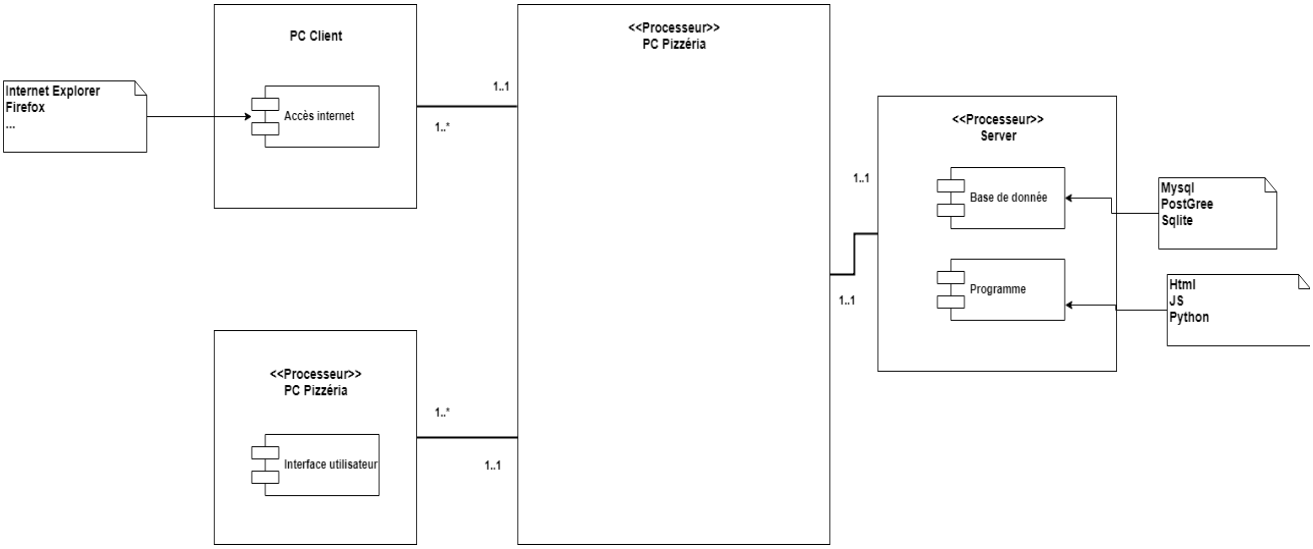
Nous déploieront notre base de donnée ainsi que notre site web sur le serveur d'Heroku. Heroku est un PassS (plateforme en tant que service).

3.3 - Modèle physique de données



4 - ARCHITECTURE DE DÉPLOIEMENT

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.

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.

5 - ARCHITECTURE LOGICIELLE

5.1 - Principes généraux

5.1.1 - 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.

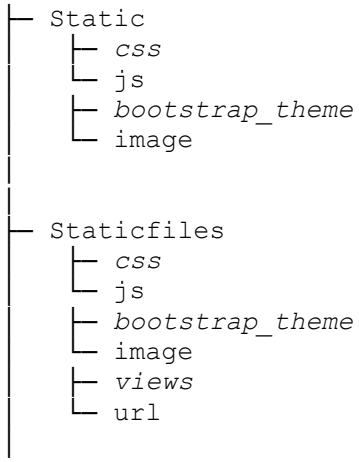
La views ou controleur permettra d'effectuer des opérations sur la base de donnée (soustraction du stock, ajout d'utilisateur en base de donnée, requete http ect...)

5.1.2 - Structure des sources

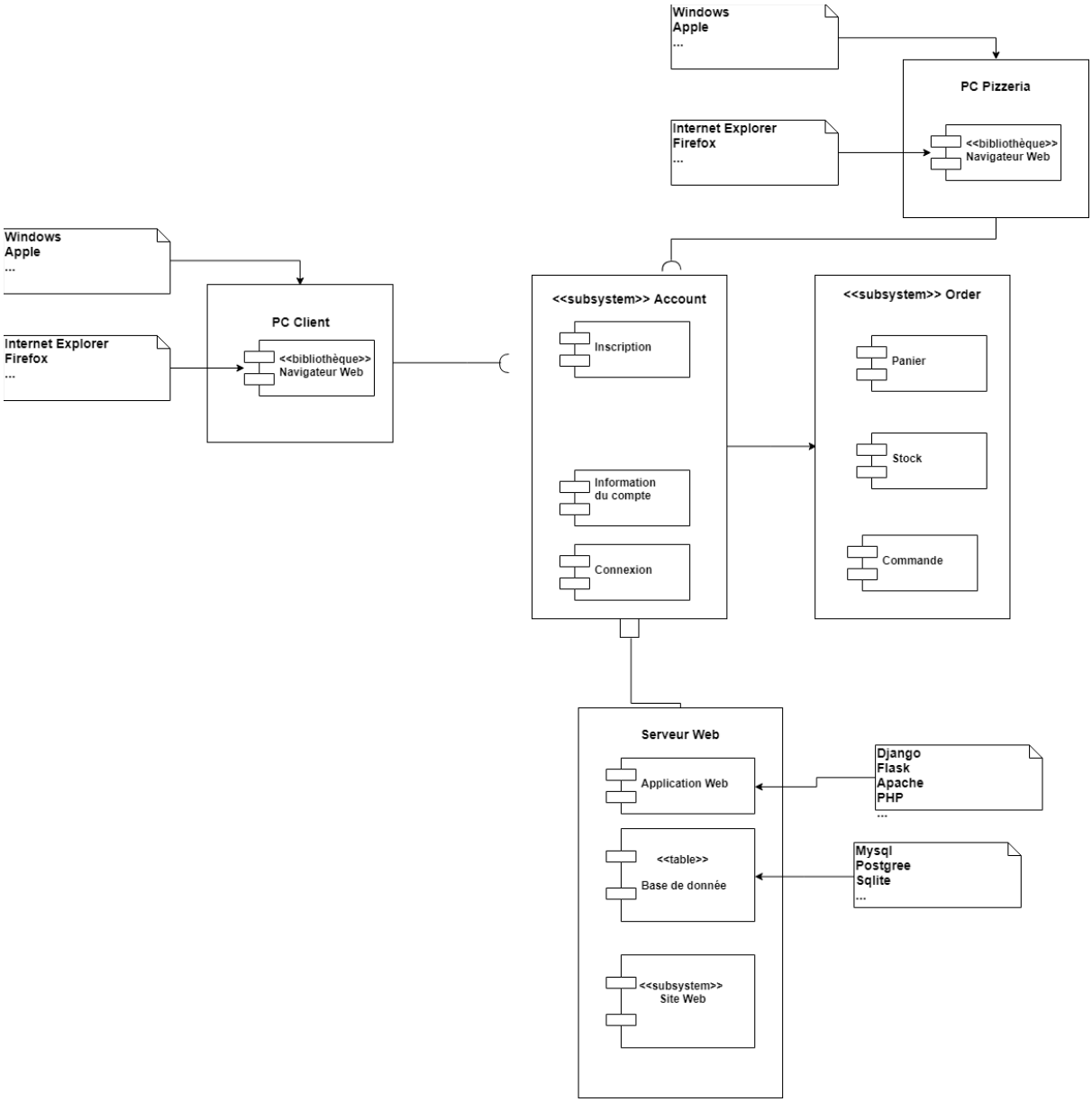
La structuration des répertoires du projet suit la logique suivante :

- 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
```


Static
css
js
bootstrap_theme
image
Staticfiles
css
js
bootstrap_theme
image
views
url

5.2 - Application Web



Plusieurs composants entrent en jeu dans notre site Web. Le premier composant est le moyen avec lequel le client et le membre du personnel va entrer en jeu avec le site Web. Pour cela, l'utilisateur doit pouvoir utiliser une navigation web, une bibliothèque, que ce soit internet explorer, firefox, google chrome… De plus son ordinateur doit disposer d'un système d'exploitation comme Apple, Windows…

Le client et le membre du personnel à travers leur ordinateur vont devoir entrer en interaction avec le subsystem Account. En effet, chaque utilisateur a à disposition un compte avec ses propres informations qui sont d'une part de connexion et d'autres part liées à ses informations du compte. Par exemple son adresse, nom nom, son numéro de téléphone… Le subsystem

Account est donc composé de trois éléments. Le premier est le composant inscription afin de pouvoir créer son compte dans le cas d'un client. Le deuxième est: information du compte pour la livraison par exemple ou pour la rémunération dans le cas d'un employé de la pizzéria. Et le troisième composant est la connexion afin de pouvoir de connecter et ainsi commander ou recevoir et traiter la commande.

Nous avons donc vu que le premier composant est "Pc client" et "Pc du personnel". Ces derniers composants doivent pouvoir recourir au composant Account afin de pouvoir faire une requête de commande dans le cas d'un client ou pour pouvoir la traiter dans le cas d'un employé de la pizzéria. L'élément Account lui, doit aussi entrer en interaction avec d'autres composants comme le serveur Web afin de pouvoir recourir aux autres fonctionnalités du site.

De fait, c'est le serveur web qui va gérer l'application web (qu'elle soit sous django, flask, apache, PHP…) . Mais c'est aussi le serveur web qui va intégrer le composant de la base de donnée sous Mysql, Postgree ou bien Sqlite. Enfin le serveur web héberge aussi le site web.

Nous avons donc vu que nous avions un composant serveur web. Ce composant la dispose de plusieurs éléments liés au site web, au programme du site web et à sa base de donnée. Pour rappel, le composant serveur web est en liaison avec le composant account. Et que ce composant est aussi en relation avec le composant subsystem Order.

Le composant Order possède trois composants. Le premier est Panier dans lequel le client va pouvoir faire la première démarche à la commande. Le deuxième est stock. En effet afin de pouvoir constituer son panier par la constitution de produit, le client doit pouvoir les choisir via une liste de pizza affichant des éléments selon l'état des stocks. Et enfin la dernière partie est le composant commande.

Nous avons donc vu que notre système était composé de plusieurs composants comme les pc clients/personnels disposant de bibliothèque. Recourant à différents composants dit "subsystem" Account et Order tout deux composés de composants. Et enfin un dernier composant: le serveur web composé du programme, du site et de la base de donnée.

6 - Points particuliers

6.1 - Ressources

6.1.1 - Application web

Les ressources graphiques tant au niveau des templates (bootstrap thème) la police sont fournis par OcPizzéria. Cependant la disposition du site nous est libre. Pour cela notre developpeur front sera aider par une personne ayant des connaissances en neuromaketing.

Les données de base (produits, prix, ingrédients, images) sont fournis par OcPizzéria.

6.2 - Environement de développement

6.2.1 - Application web

L'utilisation d'un IDE n'est pas imposé. Cepdant nous collaboreront sur Github et par visioconférences.

6.3 - Procédure de packaging / livraison

La procédure de livraison se fera après la realisation des tests finaux ainsi que le dernier déploiement sur Heroku. Le dossier d'exploitation sera alors remis au directeur ainsi que nos coordonées dans d'éventuels problèmes ou d'incompréhension.

7 - GLOSSAIRE

Heroku	Service PaaS Pateforme as Service. Cloud permettant de déployer des applications.
SGBD	Système de gestion de base de données.
Python	Langage informatique orienté objet.
HTML	Langage de balisage pour la création de pages web.
JAVASCRIPT	Langage informatique utilisé pour la dynamique des pages web.