OC Pizza

Application pour OC Pizza

Dossier de conception technique

Version 1.0

Auteur
Gabrielle Barboteau
Développeuse Junior

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 -Composant User....8
3.1.1.1 -Classe utilisateur.... 8
3.1.1.2 -Classe rôle.... 8
3.1.2 -Composant Pizzeria....8
3.1.2.1 -Classe pizzeria.... 8
3.1.2.2 -Classe adress.... 8
3.1.2.3 -Classe pizzeria_stock.... 8
3.1.3 -Composant Commande....8
3.1.3.1 -Classe commande.... 8
3.1.3.2 -Classe status.... 8
3.1.3.3 -Classe productincommand.... 8
3.1.4 -Composant Product....9
3.1.4.1 -Classe product.... 9
3.1.4.2 -Classe productincommand.... 9
3.1.5 -Composant Recipe....9
3.1.5.1 -Classe recipe.... 9
3.1.5.2 -Classe ingredientinrecipe.... 9
3.1.6 -Composant Ingredient....9
3.1.6.1 -Classe ingredient.... 9
3.1.6.2 -Classe ingredientinrecipe.... 9
3.1.7 -Composants externes....9
3.1.7.1 -Google Maps.... 9
3.1.7.2 -Paypal.... 9
3.2 -Application Web.... 10
3.3 -Application Mobile.... 10
4 -Architecture de Déploiement....11
4.1 -Spécifications du serveur.... 11
5 -Architecture logicielle....12
5.1 -Principes généraux.... 12
5.1.1 -Les couches....12
5.1.2 -Les modules....12
5.1.3 -Structure des sources....13
6 -Points particuliers....14
6.1 -Gestion des connexion.... 14
6.2 -Environnement de développement.... 14

1 - Versions

Auteur	Date	Description	Version
Gabrielle Barboteau	23/04/20	Création du document	01/01/00

2 - Introduction

2.1 - Objet du document

Ce document a pour but de détailler les spécifications techniques du projet OC Pizza, afin de nous mettre d'accord entre prestataire et client sur les éléments à inclure dans la future application.

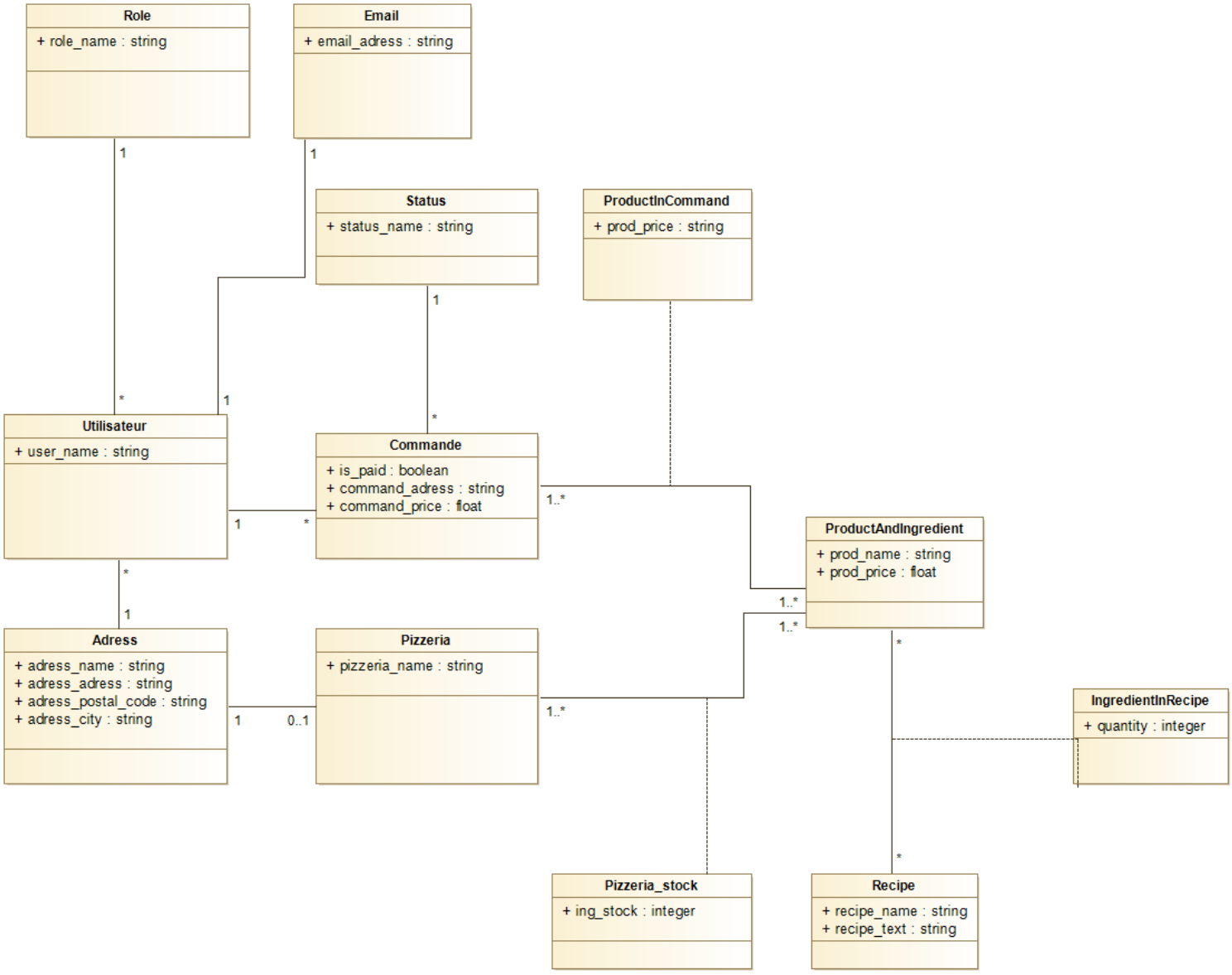
2.2 - Références

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

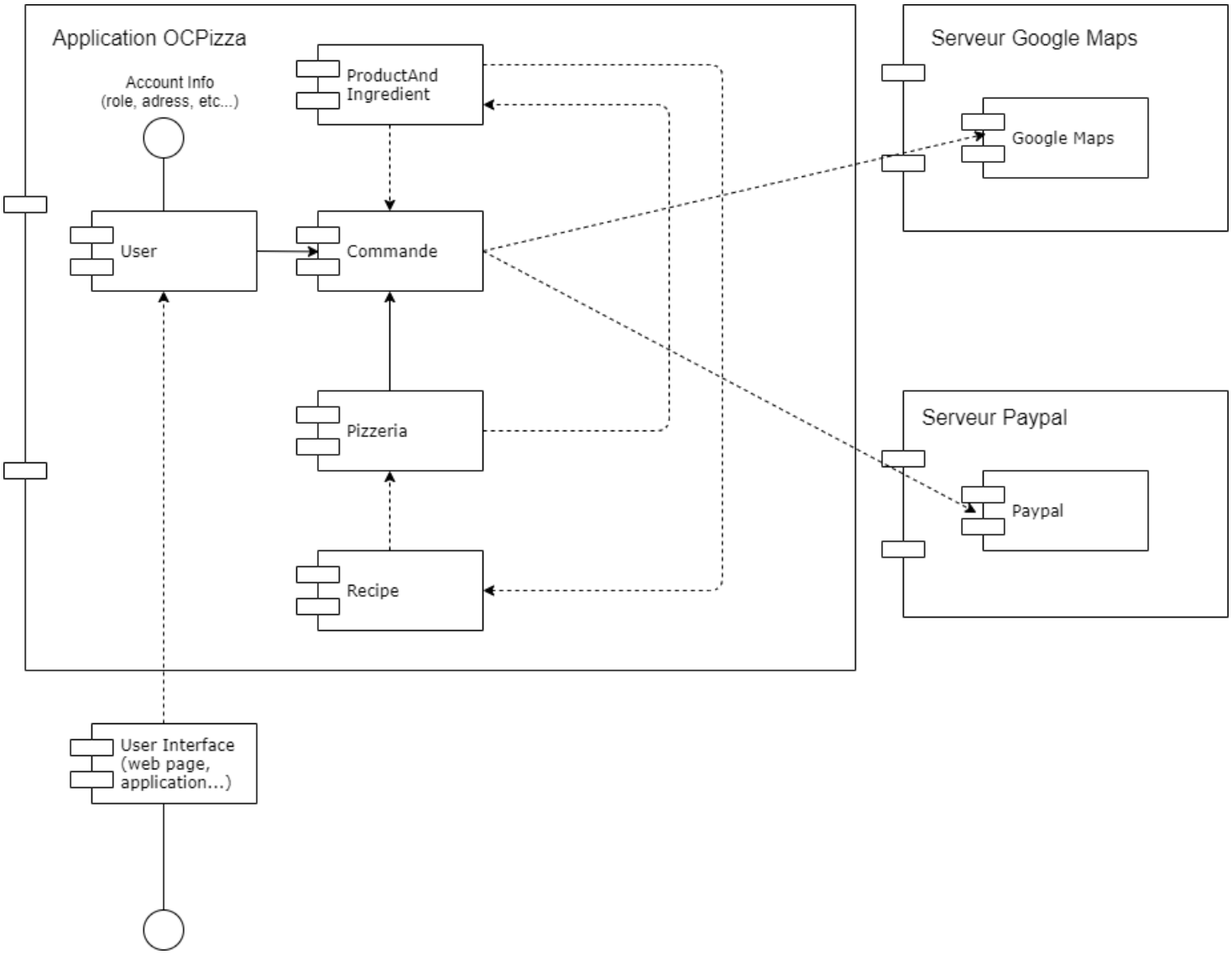
1. **PDOCPizza_01_fonctionnelle** : Dossier de conception fonctionnelle de l'application
2. **PDOCPizza_03_exploitation** : Dossier d'exploitation de l'application

3 - Architecture Technique

3.1 - Composants généraux



Ci-dessus se trouve un schéma du domaine fonctionnel, suivi du détail des composants des différents packages. Ces classes sont regroupés en différents composants, détaillés dans le diagramme de composants ci-dessous (à noter que deux composants externes interragissent avec le composant commande : Paypal, pour les paiements en ligne, et Google Maps, pour que la position du livreur soit transmise en temps réel).



Les interactions se font de la manière suivante :

- L'utilisateur utilise le site via une interface (web, app mobile...), et se connecte : le composant User est ainsi utilisé (diverses infos sont nécessaire pour identifier dans quelle catégorie l'utilisateur se situe).
- L'utilisateur passe ensuite commande (composant Commande), commande que prend en charge la pizzeria (composant Pizzeria).
- Mais avant de passer commande, il a fallu la remplir de produits : c'est là que le composant ProductAndIngredient entre en jeu. Attention : il nécessite le composant Pizzeria afin de vérifier les stocks.
- Lors de la préparation de la commande, le pizzeria aura potentiellement besoin du composant Recipe, lui permettant d'accéder aux recettes des pizzas (composant utilisant lui-même le composant ProductAndIngredient).
- Enfin, deux composants externes interviennent sur le package commande : Paypal si paiement en ligne il y a, et Google Maps pour indiquer au client la position du livreur.

Notez que les produits et les ingrédients ont volontairement été rassemblés au sein d'une même catégorie, ProductAndIngredient, afin de gérer plus facilement les stocks et permettre une plus grande flexibilité dans la gestion de l'enseigne (il est ainsi possible d'utiliser des produits vendus aux consommateurs dans la confection de recettes, ainsi que de vendre aux clients les ingrédients utilisés, par exemple dans des kits pour à pizza maison).

3.1.1 - Composant User

Il regroupe les classes User, Role, Adress et Email. Il permet d'identifier un utilisateur, et l'autorise par la suite à effectuer certaines actions.

3.1.1.1 - Classe utilisateur

Il permet de gérer les utilisateurs et leurs informations (historique de commande, adresse de livraison, etc...).

3.1.1.2 - Classe rôle

Il permet d'identifier quels utilisateurs peuvent accéder à quelles parties du sites en fonction de leur privilèges (cette partie est abordée dans le document de spécifications fonctionnelles).

3.1.2 - Composant Pizzeria

Il regroupe les classes Pizzeria, Adress et Pizzeria_Stock. Il permet d'identifier une pizzeria, et à ce magasin bien précise de faire une série d'action (consulter le stock, traiter des commandes...).

3.1.2.1 - Classe pizzeria

Liste et gère les différentes pizzerias de l'enseigne.

3.1.2.2 - Classe adress

Permet de stocker les adresses (à la fois des utilisateurs et des pizzerias) dans un format précis (numéro et rue, code postal, etc...).

3.1.2.3 - Classe pizzeria_stock

Gère les stocks des différents ingrédients dans chaque pizzeria.

3.1.3 - Composant Commande

Il regroupe les classes Commande, Status, et Product_in_command. Il permet de gérer les différents aspects d'une commande (le statut, qui l'a passé, etc...).

3.1.3.1 - Classe commande

Gère l'ensemble des commandes, et ce qui en découle (status, etc...). Permet aussi de savoir si la commande a été payée ou pas, pour que le livreur sache quoi faire à la livraison.

3.1.3.2 - Classe status

Permet de suivre les diverses étapes d'une commande, et de proposer différentes actions à différents acteurs en fonction (exemple : si la commande est « en attente », le client peut l'annuler ou la modifier).

3.1.3.3 - Classe productincommand

Liste les produits faisant partis d'une commande donnée.

3.1.4 - Composant Product

Il regroupe les classes Product et Product_in_command. Gère toutes les informations liées à un produit (sa disponibilité, etc...).

3.1.4.1 - Classe productandingredient

Liste les produits que l'utilisateur peut acheter.

3.1.4.2 - Classe productincommand

Liste les produits faisant partis d'une commande donnée.

3.1.5 - Composant Recipe

Il regroupe les classes Recipe et Ingredient_in_recipe. Gère toutes les informations liées à une recette (comment la préparer, quels aliments y a-t-il...).

3.1.5.1 - Classe recipe

Liste les différentes recettes des pizzas de l'enseigne (agit comme aide-mémoire).

3.1.5.2 - Classe ingredientinrecipe

Liste les ingrédients nécessaires à une recette, avec le nombre de portions nécessaires (exemple : 2 tomates, 3 x 100g de fromage, etc...).

3.1.6 - Composant Ingredient

Il regroupe les classes Ingredient et Ingredient_in_recipe. Gère toutes les informations liées aux ingrédients.

3.1.6.1 - Classe productandingredient

Liste tous les ingrédients utilisables au sein de l'enseigne.

3.1.6.2 - Classe ingredientinrecipe

Liste les ingrédients nécessaires à une recette, avec le nombre de portions nécessaires (exemple : 2 tomates, 3 x 100g de fromage, etc...).

3.1.7 - Composants externes

Il y en a deux, qui interagissent avec le composant Commande.

3.1.7.1 - Google Maps

Permet d'avoir la position d'un livreur en temps réel.

3.1.7.2 - Paypal

Permet de gérer les paiements en ligne.

3.2 - Application Web

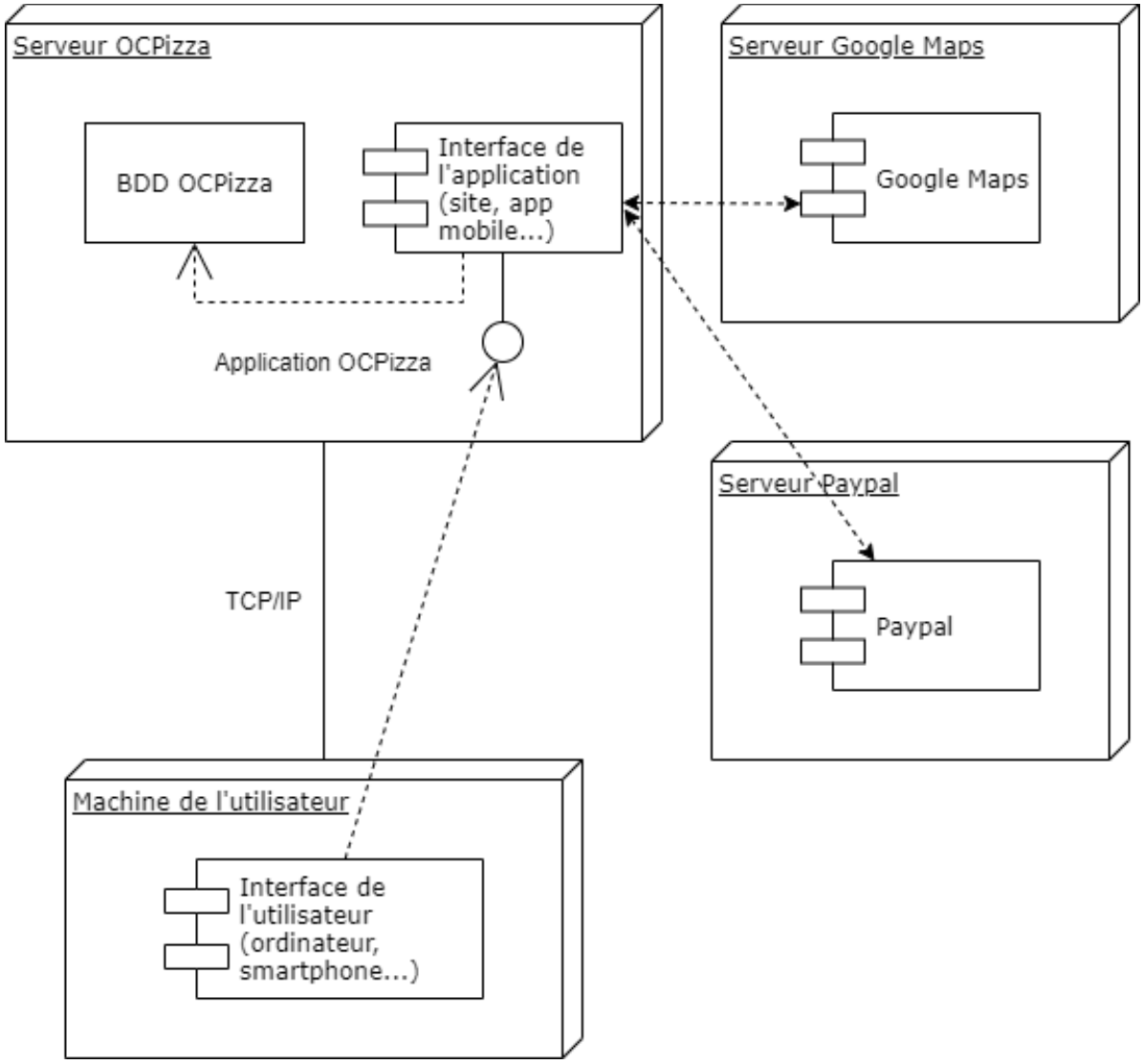
La pile logicielle est la suivante :

- Le front-end sera réalisé en HTML, CSS et Javascript, à l'aide du framework Bootstrap (version 4.4).
- Le back-end utilisé Python 3.7 et Django 2.2 (les versions à la fois les plus récentes et les plus stables), ainsi qu'une série de modules divers (entre autres gunicorn 20.0 et mysql-connector-python).
- La base de donnée sera créée et gérée avec MySQL 8.0.
- Le serveur sera configuré via nginx.

3.3 - Application Mobile

Afin de limiter les coûts de développement et de maximiser la maintenance sur le long terme, les applications mobiles (iOS et Android) seront des *progressive web apps* (PWA).

4 - Architecture de Déploiement



Notre application et notre base de données seront hébergés sur le même serveur, pour des raisons de coût et d'efficacité. Ci-dessous se trouve le diagramme de déploiement illustrant la communication entre ces deux éléments et l'utilisateur.

4.1 - Spécifications du serveur

L'hébergement de la solution se fera sur un serveur dédié, nous préconisons d'en choisir un chez OVH ou Digital Ocean, avec 1To de stockage et 8Go de RAM minimum.

Nous utiliserons Ubuntu 18.04 (la dernière version LTS) pour faire fonctionner ce serveur. La version 8.0 de MySQL sera installé dessus pour faire fonctionner la base de donnée.

Python 3.7 et Django 2.2 sera évidemment installé, et nous utiliserons nginx, gunicorn, et supervisor pour faire la liaison entre application et serveur, ainsi que contrôler le bon déroulement des opérations.

5 - Architecture logicielle

5.1 - Principes généraux

Les sources et versions du projet sont gérées par **Git**, l'environnement, les dépendances et les packages par **Pip** et **VirtualEnv**.

5.1.1 - Les couches

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

5.1.2 - Les modules

Nous utiliserons Gunicorn pour gérer l'application une fois sur serveur, ainsi que mysql-connector pour pouvoir interagir avec la base de donnée depuis notre application.

5.1.3 - Structure des sources

Les répertoires sources sont crées de façon à respecter la philosophie Django (l'utilisation du modèle Model/View/Template)

```
root
├── manage.py
├── ocpizzaapp
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── admin.py
│   ├── models.py
│   ├── tests.py
│   ├── urls.py
│   ├── views.py
│   ├── migrations
│   │   ├── __init__.py
│   │   ├── 0001_initial.py
│   │   └── ...
│   ├── static
│   │   ├── ocpizzaapp
│   │   │   ├── css
│   │   │   │   ├── bootstrap.css
│   │   │   │   └── custom.css
│   │   │   ├── images
│   │   │   │   └── ...
│   │   │   ├── js
│   │   │   │   ├── bootstrap.js
│   │   │   │   └── custom.js
│   ├── templates
│   │   ├── admin
│   │   │   ├── base_site.html
│   │   │   └── ...
│   │   ├── ocpizzaapp
│   │   │   ├── index.html
│   │   │   └── ...
│   │   ├── 404.html
│   │   ├── 500.html
│   │   └── ...
│   │
└── project
    ├── init.py
    ├── asgy.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

6 - Points particuliers

6.1 - Gestion des connexion

Les connexions seront gérés par Django lui-même, ayant un système d'identification des utilisateurs intégré.

6.2 - Environnement de développement

Nous travaillerons sur Windows 10, en utilisant un environnement virtuel sous VirtualEnv.

Pour les tests, nous utiliserons un de nos serveurs sur Ubuntu, dont la configuration est similaire à celle préconisée dans la partie 4.1.