OC Pizza

Système de gestion de pizzeria en ligne

Dossier de conception technique

Version 1.0

Auteur
Clémence Robin
Analyste-programmeur

IT Consulting & Development
7 cité Paradis – 01 02 03 04 05 – contact@it-solutions.fr
www.it-solutions.fr
S.A.R.L. au capital de 1 000,00 € enregistrée au RCS de Xxxx – SIREN 999 999 999 – Code APE : 6202A

Table des matières

1 - Versions **3**
2 - Introduction **4**
2.1 - Objet du document 4
2.2 - Références 4
3 - Architecture de composants **5**
3.1 - Composants de l'application 5
4 - Architecture de Déploiement **7**
4.1 - Déploiement de l'application 7
4.1.1 - Matériel client *7*
4.1.2 - Serveur d'application *7*
4.2 - Serveur de Base de données 8
4.3 - Serveurs de paiement 10
5 - Architecture logicielle **11**
5.1 - Principes généraux 11
5.1.1 - Les couches *11*
5.1.2 - Les modules *11*
5.1.3 - Structure des sources *11*
6 - Points particuliers **14**
6.1 - Gestion des logs 14
6.2 - Fichiers de configuration 14
6.2.1 - Application web *14*
6.3 - Ressources 14
6.3.1 - Charte graphique *14*
6.3.2 - Données *14*
6.4 - Environnement de développement 14
6.5 - Procédure de packaging / livraison 15
6.5.1 - Serveur d'application web *15*
6.5.2 - Serveur de base de données *15*
6.5.3 - Dossier d'exploitation *15*
7 - Glossaire **16**

1 - Versions

Auteur	Date	Description	Version
Clémence R.	20/08/2020	Création du document	1.0

2 - Introduction

2.1 - Objet du document

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

L'objectif du document est de présenter les outils, les technologies, et les méthodes mises en oeuvre pour réaliser l'application.

Les éléments du présents dossiers découlent :

- de nos entretiens avec le client pour répondre à ses attentes ainsi que
- du document de spécifications fonctionnelles qui en est résulté

2.2 - Références

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

1. **DCF – 1.0** : Dossier de conception fonctionnelle de l'application
2. **DE – 1.0** : Dossier d'exploitation de l'application

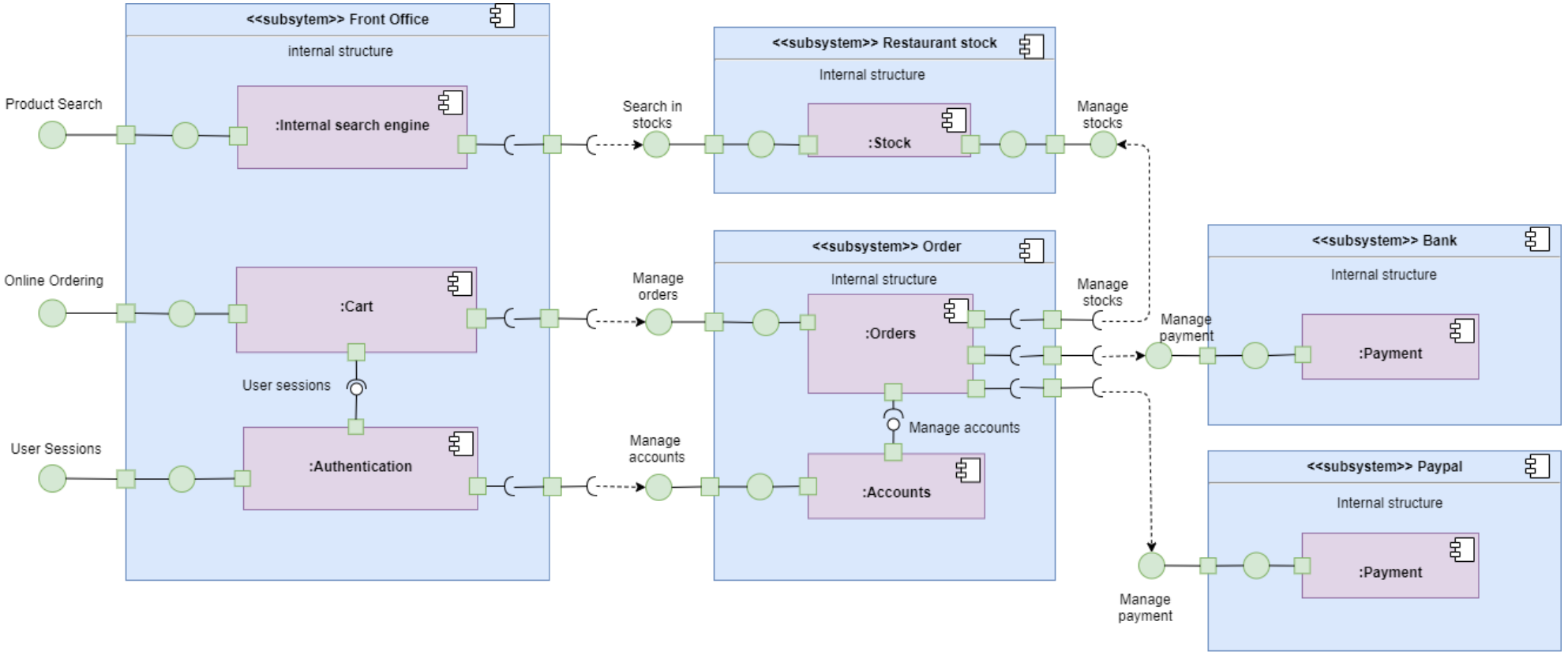
3 - Architecture de composants

3.1 - Composants de l'application

L'application est découpée en 5 sous-sytèmes auquels les composants sont intégrés :

- Front-office : représente le site web
  - Internal search engine : représente le moteur de recherche
  - Cart : représente le panier
  - Authentication : représente l'authentification de l'utilisateur à son compte
- Restaurant stock : représente les stocks du restaurant
  - Stock : représente les stocks
- Order : représente le processus de commande
  - Orders : représente les commandes des clients
  - Accounts : représente les comptes des utilisateurs
- Bank : représente la banque
  - Payment : représente le paiement
- Paypal : représente Paypal
  - Payment : représente le paiement

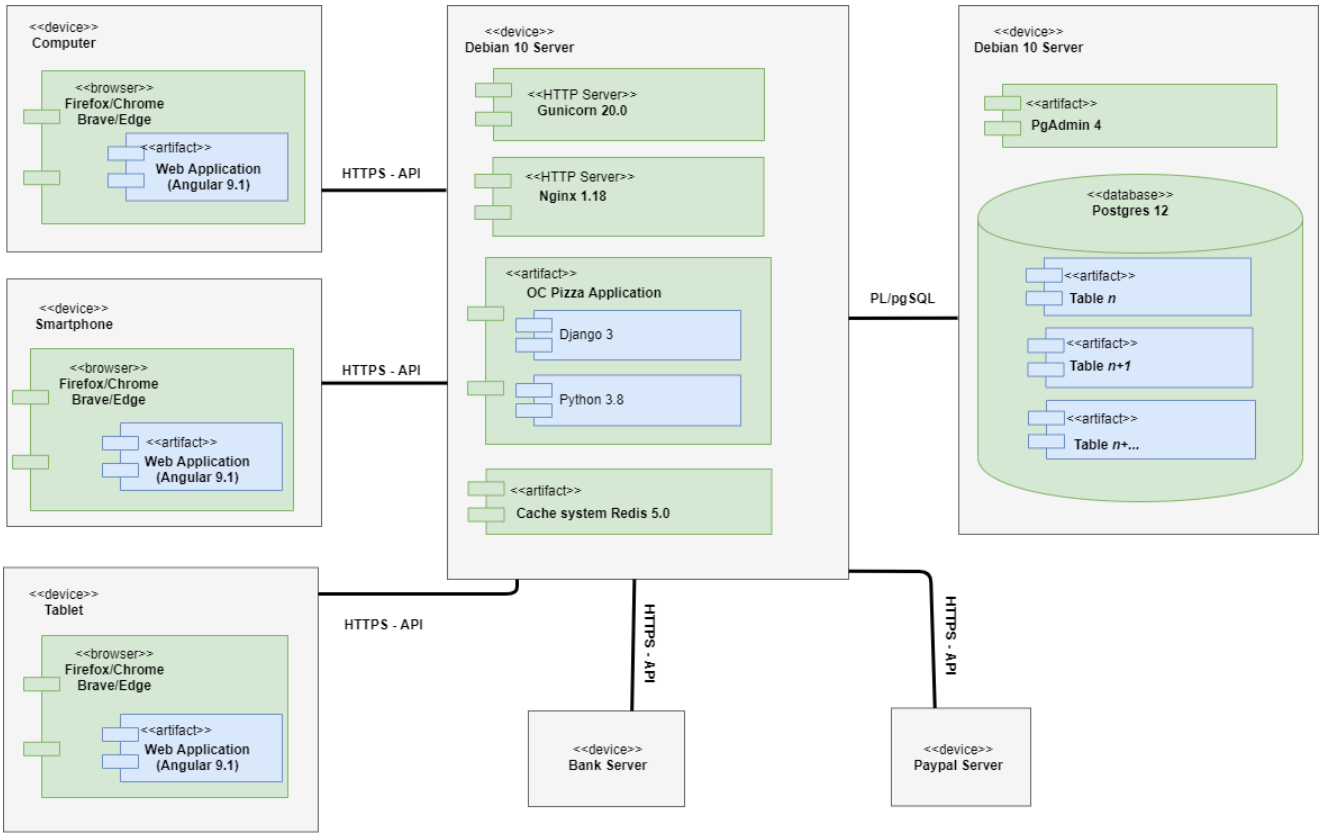
Le diagramme de composants ci-dessous décrit les différents sous-systèmes et composants de l'application ainsi que leurs intéractions lorsque le client arrive sur l'application web pour passer commande.


<<subsytem>> Front Office
internal structure
Product Search
:Internal search engine
Online Ordering
:Cart
User sessions
User Sessions
:Authentication
Search in stocks
<<subsystem>> Restaurant stock
Internal structure
:Stock
Manage stocks
Manage orders
<<subsystem>> Order
Internal structure
:Orders
Manage accounts
Manage accounts
:Accounts
Manage stocks
Manage payment
<<subsystem>> Bank
Internal structure
:Payment
<<subsystem>> Paypal
Internal structure
:Payment
Manage payment

4 - Architecture de Déploiement

4.1 - Déploiement de l'application

Le diagramme de déploiement ci-dessous décrit les différents éléments prenant par à la production de l'application.



4.1.1 - Matériel client

L'utilisateur accède à l'application web via son navigateur installé sur sa machine (ordinateur, tablette, smartphone). Le navigateur exécutera la partie front-end écrite avec Angular.

4.1.2 - Serveur d'application

L'application Django fonctionnera sur un serveur fonctionnant avec le système d'exploiration Debian 10. A cela s'ajoutera :

- un serveur web HTTP Nginx pour traiter les contenus statiques
- un serveur d'application HTTP Gunicorn à qui Nginx transférera les requêtes dynamiques

- Un système de cache Redis qui servira à réduire le temps de chargement de l'application lorsque la machine cliente aura déjà déjà les informations en mémoire cache?

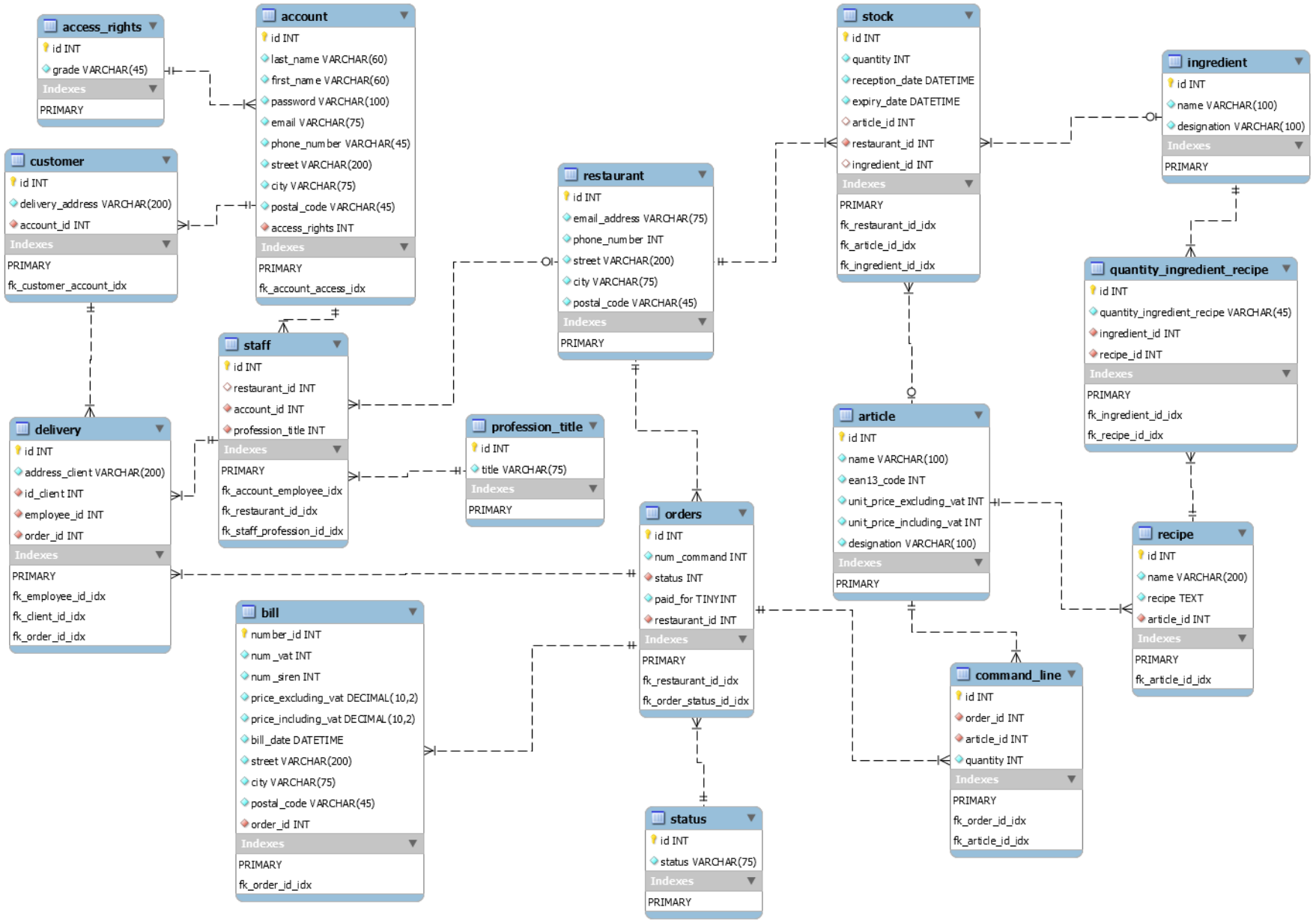
4.2 - Serveur de Base de données

Le SGBDR PostgreSQL 12 fonctionnera sur un serveur fonctionnant avec le système d'exploitation Debian 10.

Le système d'administration de base données PgAdmin 4 sera aussi installé sur ce serveur pour faciliter la gestion de la base de données.

4.2.1.1 - Modèle physique de données

Nous pouvons voir ci-dessous les colonnes de chaque table, leur type et les liens entre les tables.


access_rights
id INT
grade VARCHAR(45)
Indexes
PRIMARY
account
id INT
last_name VARCHAR(60)
first_name VARCHAR(60)
password VARCHAR(100)
email VARCHAR(75)
phone_number VARCHAR(45)
street VARCHAR(200)
city VARCHAR(75)
postal_code VARCHAR(45)
access_rights INT
Indexes
PRIMARY
fk_account_access_idx
stock
id INT
quantity INT
reception_date DATETIME
expiry_date DATETIME
article_id INT
restaurant_id INT
ingredient_id INT
Indexes
PRIMARY
fk_restaurant_id_idx
fk_article_id_idx
fk_ingredient_id_idx
ingredient
id INT
name VARCHAR(100)
designation VARCHAR(100)
Indexes
PRIMARY
customer
id INT
delivery_address VARCHAR(200)
account_id INT
Indexes
PRIMARY
fk_customer_account_idx
restaurant
id INT
email_address VARCHAR(75)
phone_number INT
street VARCHAR(200)
city VARCHAR(75)
postal_code VARCHAR(45)
Indexes
PRIMARY
quantity_ingredient_recipe
id INT
quantity_ingredient_recipe VARCHAR(45)
ingredient_id INT
recipe_id INT
Indexes
PRIMARY
fk_ingredient_id_idx
fk_recipe_id_idx
staff
id INT
restaurant_id INT
account_id INT
profession_title INT
Indexes
PRIMARY
fk_account_employee_idx
fk_restaurant_id_idx
fk_staff_profession_id_idx
profession_title
id INT
title VARCHAR(75)
Indexes
PRIMARY
delivery
id INT
address_client VARCHAR(200)
id_client INT
employee_id INT
order_id INT
Indexes
PRIMARY
fk_employee_id_idx
fk_client_id_idx
fk_order_id_idx
article
id INT
name VARCHAR(100)
ean13_code INT
unit_price_excluding_vat INT
unit_price_including_vat INT
designation VARCHAR(100)
Indexes
PRIMARY
orders
id INT
num _command INT
status INT
paid_for TINYINT
restaurant_id INT
Indexes
PRIMARY
fk_restaurant_id_idx
fk_order_status_id_idx
recipe
id INT
name VARCHAR(200)
recipe TEXT
article_id INT
Indexes
PRIMARY
fk_article_id_idx
bill
number_id INT
num _vat INT
num _siren INT
price_excluding_vat DECIMAL(10,2)
price_including_vat DECIMAL(10,2)
bill_date DATETIME
street VARCHAR(200)
city VARCHAR(75)
postal_code VARCHAR(45)
order_id INT
Indexes
PRIMARY
fk_order_id_idx
command_line
id INT
order_id INT
article_id INT
quantity INT
Indexes
PRIMARY
fk_order_id_idx
fk_article_id_idx
status
id INT
status VARCHAR(75)
Indexes
PRIMARY

4.3 - Serveurs de paiement

Lors du paiement, le serveur d'application communiquera avec des serveurs externes à notre infrastructure :

- les serveurs bancaires
- les serveurs de Paypal

5 - Architecture logicielle

5.1 - Principes généraux

Les sources et versions du projet sont gérées par Git. La méthode de travail AGILE,
transcrit dans un tableau Trello les différentes stories, et dispatche le travail en plusieurs branches, réintégrées au fils du temps sur les branches de master de Git.

Le développement de l'application se présente sous la forme d'un projet Django, lui-même divisé en deux « Applications » : Ventes et Production.
Chaque application respecte le pattern Model Vue Template (MVT).

5.1.1 - Les couches

L'architecture applicative est la suivante :

- une couche **métier** : responsable de la logique métier du composant se trouve dans le fichier models.py
- une couche **modèle** : implémentation du modèle des objets métiers se trouve également dans le fichier models.py
- la couche **client** : interface client de l'application se trouve dans le fichier views.py qui affiche le template html
- la couche **base de données** : les opérations de lecture écriture sur la base de données se fait par l'ORM de Django qui communique avec la base de donnée

5.1.2 - Les modules

L'application est divisée en 4 modules gérant chaqu'une une couche de l'application.

- L'authentification
- Le stock
- L'administration
- Les commandes

5.1.3 - Structure des sources

La structuration des répertoires du projet suit la logique suivante :

– les répertoires sources sont crées de façon à respecter les conventions de Django :

```
oc_pizza/
├── administration
│   ├── admin.py
│   ├── apps.py
│   ├── forms.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
```

```
│   ├── models.py
│   ├── static
│   ├── templates
│   ├── tests.py
│   ├── urls.py
│   └── views.py
├── auth
│   ├── admin.py
│   ├── apps.py
│   ├── forms.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── static
│   ├── templates
│   ├── tests.py
│   ├── urls.py
│   └── views.py
├── manage.py
├── oc_pizza
│   ├── asgi.py
│   ├── __init__.py
│   ├── settings
│   │   ├── base.py
│   │   ├── development.py
│   │   └── prod.py
│   ├── urls.py
│   └── wsgi.py
├── order
│   ├── admin.py
│   ├── apps.py
│   ├── forms.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── static
│   ├── templates
│   ├── tests.py
│   ├── urls.py
│   └── views.py
├── README.md
├── requirements.txt
├── staticfiles
├── stock
│   ├── admin.py
│   ├── apps.py
│   ├── forms.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── static
│   ├── templates
│   ├── tests.py
│   ├── urls.py
│   └── views.py
```

```
└── tests
```

6 - Points particuliers

6.1 - Gestion des logs

Le projet Django Sentry est une application de monitoring qui permet de gérer les logs et d'être averti des disfonctionnements

6.2 - Fichiers de configuration

6.2.1 - Application web

La configuration de l'application se trouve dans le dossier settings, composé de trois fichiers :

- base.py : rassemble toutes les configurations commune à l'environnement de production et à l'environnement de développement
- development.py : rassemble toutes les configurations propres au développement, comme l'activation du mode de débogage.
- prod.py : rassemble toutes les configrations propres à l'environnement de production.

6.3 - Ressources

6.3.1 - Charte graphique

Le cahier des charges graphique nous est fourni par OC Pizza.

6.3.2 - Données

Les données nécessaires au fonctionnement de la base de données (comptes, recettes, etc) nous sont fournies par OC Pizza.

6.4 - Environnement de développement

Le développement de l'application ne nécessite pas l'utilisation d'un IDE en particulier. Cependant, l'édition professionnelle de PyCharm rencontre tous les avantages liés à un développement couplant Django et Python.
Un environnement virtuel sera créé pour s'assurer de la compatibilité entre tous les composants de l'application.

6.5 - Procédure de packaging / livraison

6.5.1 - Serveur d'application web

L'application fera l'object d'un déploiement sur un serveur loué chez Digital Ocean.
Digital Ocean permet d'obtenir une machine virtuelle où l'on peut installer Debian. De plus, ressources s'adaptent au besoin de l'application.

Le produit qui correspond à nos besoin est dans le catalogue "General Purpose Droplet". Il est conçu pour les applications web avec un certain niveau de trafic, les sites de commerce en ligne, ainsi que les bases de données de taille moyenne.

Si les besoins d'OC Pizza évolue, la machine adaptera ses ressources et le prix s'adaptera en conséquence.
Notre machine partira donc avec ces caractéristiques :

- 8GB de RAM
- 4vCPUs
- 5 TB de transfert de données
- 160 GB d'espace disque
- Coût : 40$

6.5.2 - Serveur de base de données

Comme l'application web, la base de donnée sera aussi déployée sur un serveur loué chez Digital Ocean. La base de donnée sera automatiquement mise à jour et sauvegardée quotidiennement. La base de donnée est joignable avec l'API.

Notre machine aura ces caractéristiques :

- 1GB de RAM
- 1vCPUs
- 10 GB d'espace disque
- Coût : 15$ par mois.

Il est possible d'allouer plus de ressources si l'évolution de la base de données le nécessite.

6.5.3 - Dossier d'exploitation

OC Pizza se verra remettre un dossier d'exploitation permettant la continuité de l'utilisation de l'application.

7 - Glossaire

SGBDR	Système de gestion de base de données
MVT	Model View Template – Modèle Vue Template