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

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 œuvre 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

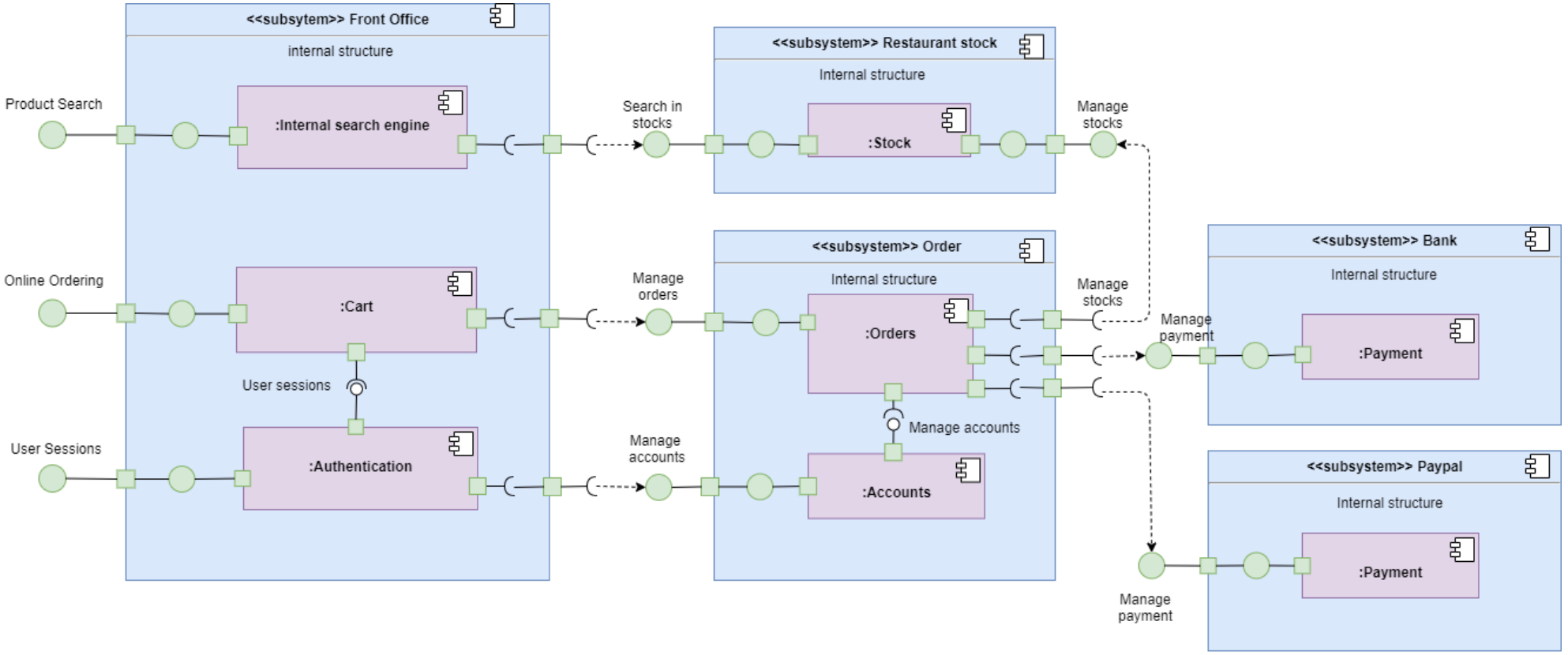
3 - Architecture de composants

3.1 - Composants de l'application

L'application est découpée en 5 sous-systèmes auxquels 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

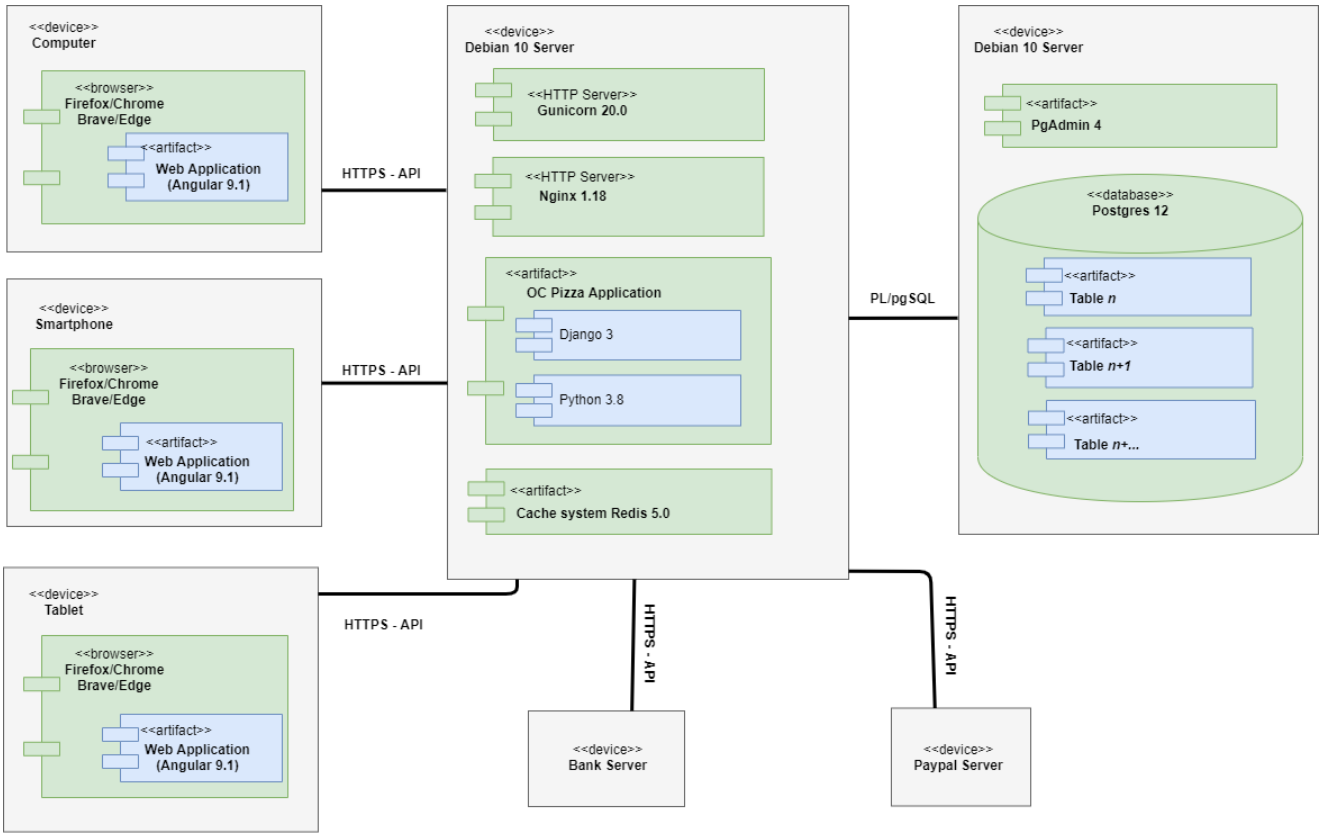
Le diagramme de composants ci-dessous décrit les différents sous-systèmes et composants de l'application ainsi que leurs interactions lorsque le client arrive sur l'application web pour passer commande.


<<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'exploitation 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?

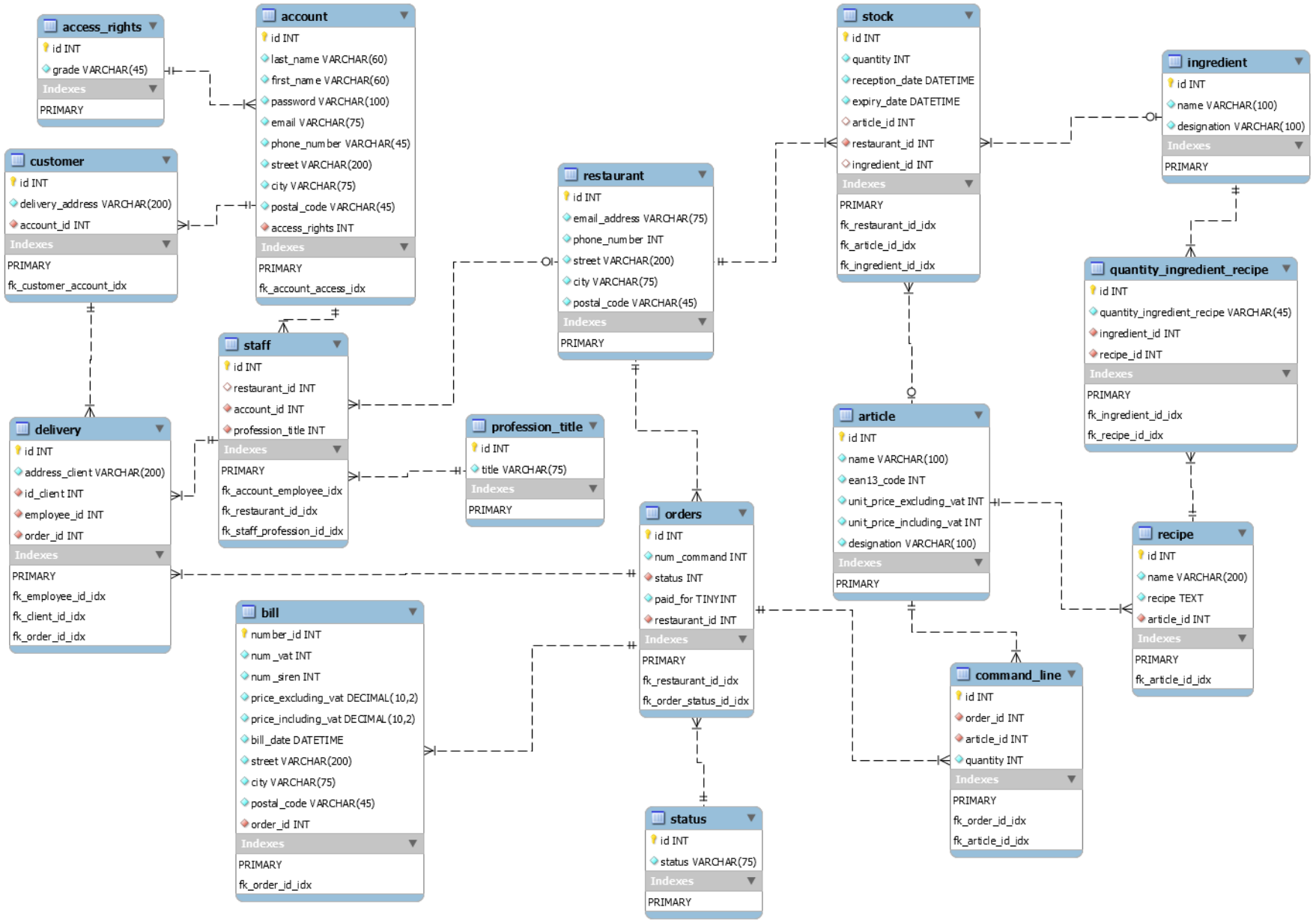
4.2 - Serveur de Base de données

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 - 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

4.2.1.1 - Descriptions du modèle physique de données

Nous revenons ici sur les tables principales et clarifions les clés étrangères qui les lient entre elles. Lorsqu'une clé est abordée dans la description d'une table A envers une table B, nous ne reviendrons par sur cette relations lors de la description de la table B.

4.2.1.1.1 Stock

Les champs articles_id et ingredient_id peuvent tous les deux être *null*. En effet, une instance du stock peut concerner un ingredient (matière première) ou un article (ex :cannette de soda). L'un des deux champs sera donc *null*.

La table contient trois clés étrangères :

- **fk_stock_restaurant_id** : entre le champ restaurant_id de la table Stock et le champ id de la table Restaurant
- **fk_stock_article_id** : entre le champ article_id de la table Stock et le champ id de la table Article
- **fk_stock_ingredient_id** : entre le champs ingredient_id de la table Stock et le champ id de la table Ingredient

4.2.1.1.2 Recipe

Une recette doit correspondre à un article préparé par le restaurant, donc à un instance de la classe Article. Elle aura donc une clé étrangère entre le champ article_id et le champ id de la table Article :

- **fk_recipe_article_id** : entre le champ article_id de la table Recipe et le champ id de la table Article

4.2.1.1.3 Orders

Chaque commande doit être associée à un restaurant et chaque commande doit être associée à un statut. La table Orders contient donc deux clés étrangères :

- **fk_order_restaurant_id** : entre le champ restaurant_id de la table Orders et le champs id de la table Restaurant
- **fk_order_status_id** : entre le champ status de la table Orders et le champs id de la table Status

4.2.1.1.4 Command_line

La table Command_line sert à préciser le nombre d'article commandé pour une commande donnée. Elle doit donc être associée à l'id de l'article et l'id de la commande. La table contient par conséquent deux clés étrangères:

- **fk_command_order_id** *:* entre le champ order_id de la table Command_line et le champ id de la table Orders
- **fk_command_article_id** *:* entre le champ article_id de la table Command_line et le champ id de la table Article

4.2.1.1.5 Bill

Chaque facture devant être associée à une commande, la table Bill contient donc une clé étrangère :

- **fk_bill_order_id** : entre le champ order_id de la table Bill et le champ id de la table Orders

4.2.1.1.6 Account

Les comptes pouvant être associés à des clients ou bien à des employés ayant des droits différents, chacun d'entre eux doit être associé à un droit spécifique. Nous avons donc une clé étrangère dans la table Account :

- **fk_account_access** : entre le champ access_rights de la table Account et le champ id de la table AccessRights

4.2.1.1.7 Quantity_ingredient_recipe

Cette table servant à indiquer les quantités d'un ingrédient donné pour une recette donnée, nous avons choisi de donner au champ quantit_ingredient_recipe le type VARCHAR : cela permet d'insérer d'autres types de données que des chiffres, par exemple : « Un zeste de » ou « Une cuillère à soupe de ... ». La table quantiy_ingredient_recipe faisant le lien entre un ingrédient et une recette, elle contient par conséquent deux clés étrangères :

- **fk_quantity_ingredient_id** : entre le champ ingredient_id de la table Quantity_ingredient_recipe et le champ id de la table Ingredient
- **fk_quantiry_recipe_id** : entre le champ recipe_id de la classe Quantity_ingredient_recipe et le champ id de la table Recipe

4.2.1.1.8 Customer

Chaque client passant commande doit avoir un compte associé. Nous avons donc une clé étrangère dans la table Customer :

- **fk_customer_account_id** : entre le champ account_id de la table Customer et le champ id de la table Account

4.2.1.1.9 Staff

Nous avons choisi de laisser la possibilité au champ restaurant_id d'avoir la valeur NULL, selon si un employé, comme un livreur, travaille dans un ou plusieurs restaurants. De plus, le directeur n'est pas associé à un restaurant en particulier.

Chaque membre du personnel possède un compte et est associé à un intitulé de profession. De même, si le champ restaurant_id n'est pas *null*, il doit être associé à un restaurant donné. Nous avons donc trois clés étrangères :

- **fk_staff_account_id** : entre le champ account_id de la table Staff et le champ id de la table Account
- **fk_staff_restaurant_id** : entre le champ restaurant_id de la table Staff et le champ id de la table Restaurant
- **fk_staff_profession_id** : entre le champ profession_id de la table Staff et le champ id de la table Profession_title

4.2.1.1.10 Delivery

Chaque livraison correspondant à une commande étant livrée à un client par un livreur donné, nous avons trois clés étrangères dans la table Delivery :

- **fk_delivery_employee_id** : entre le champ employee_id de la table Delivery et le champ id de la table Staff
- **fk_delivery_client_id** : entre le champ id_client de la table Delivery et le champ id de la table Customer
- **fk_delivery_order_id** : entre le champd order_id de la table Delivery et le champ id de la table Orders

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.

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 trouvant dans le fichier *models.py*
- une couche **modèle** : implémentation du modèle des objets métiers se trouvant dans le fichier *models.py*
- une couche **présentation** : interface client de l'application se trouvant dans le fichier *views.py*
- une couche accès **base de données** : les opérations de lecture écriture sur la base de données sont faites 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 chacune 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
│   │   ├── __init__.py
│   │   ├── local.py
│   │   └── production.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 dysfonctionnements.

6.1.1 - Création de compte chez Sentry

OC Pizza doit s'inscrire sur le site de Sentry.

6.1.2 - Configuration de Sentry

Lors de l'enregistrement du compte, Sentry affichera les informations à rajouter dans *oc_pizza/oc_pizza/settings/__init__.py*.

Un tableau de bord permettra ensuite d'afficher et de gérer plusieurs informations, telles que les logs, la performance ou encore la création d'alertes.

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 :

- __init__.py : rassemble toutes les configurations commune à l'environnement de production et à l'environnement de développement
- local.py : rassemble toutes les configurations propres au développement, comme l'activation du mode de débogage.
- production.py : rassemble toutes les configurations 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", qui permet d'avoir un CPU dédié et non partagé. 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
- 4vCPU
- 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
- 1vCPU
- 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
vCPU	CPU virtuel alloué à une machine virtuelle
RAM	Random Access Memory – Mémoire à accès aléatoire
API	Application Programming Interface – Interface de programmation d'application