OC Pizza

Application web OC Pizza

Dossier de conception technique

Version 1.0.0

Auteur
Decroix Nicolas
Développeur

Table des matières

1 - Versions

Auteur	Date	Description	Version
Decoix Nicolas	31/01/2020	Création du document	1.0.0

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 consiste en la définition des éléments permettant la mise en place du système informatique de gestion des points de vente du groupe OC Pizza.

Les éléments du présent dossier découlent :

- de la conception du dossier fonctionnel.
- de l'étude du domaine technique.

2.2 - Références

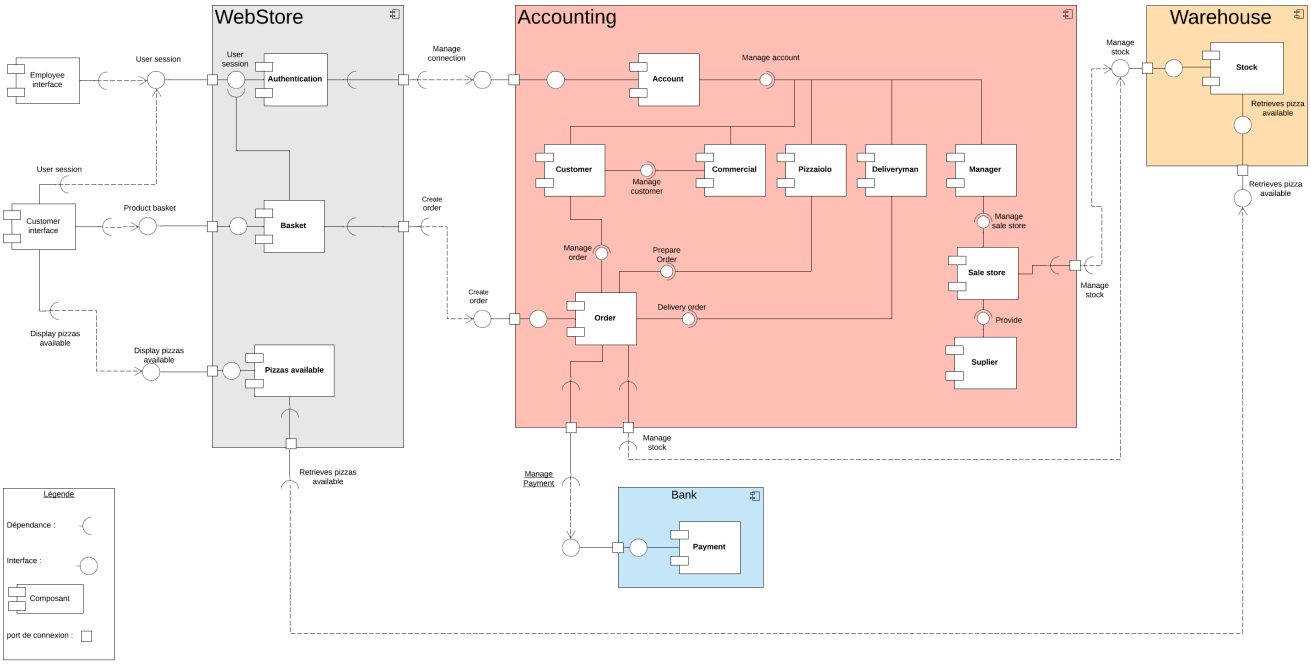
Pour de plus amples informations, se référer également aux éléments suivants:

1. **DCF – 1.0.0** : Dossier de conception fonctionnelle.
2. **DE – 1.0.0** : Dossier d'exploitation.

3 - Architecture Technique

3.1 - Composants

3.1.1 - Diagramme UML de Composants



3.1.2 - Composant Employee interface

Interface d'un employé du groupe OC Pizza. L'employé accède directement sur une page de connexion pour s'authentifier et avoir accès aux fonctionnalités dues à son rôle (Manager, Pizzaiolo, commercial, livreur).

Dépendance

- User session : Authentification de l'utilisateur.

3.1.3 - Composant Customer interface

Interface des clients et visiteurs lorsqu'ils accèdent au site de vente en ligne du groupe OC Pizza.

L'interface affiche le catalogue de pizzas et permet leur ajout dans le panier. C'est aussi par cette interface qu'il est possible de se connecter comme client.

Dépendances :

- User session : interface de connexion à une session utilisateurs.
- Product basket : Panier pour stocker les pizzas que l'on souhaite acheter.
- Display pizza available : Affichage du catalogue de pizzas.

3.1.4 - Composant WebStore

Le composant WebStore représente la logique de l'interface web de l'application OC Pizza. Il fournit une interface de base permettant la consultation des produits vendus par le groupe et gère l'authentification des acteurs.

Interfaces :

- user session : Interface pour l'authentification d'un utilisateur
- product basket : Interface d'enregistrement des pizzas par l'utilisateur.
- display pizzas available : affichage des pizzas disponibles à la vente.

Dépendances

- Manage connection : Accès aux comptes utilisateur pour lier un utilisateur à son compte lors de l'authentification de ce dernier.
- Create order : Créer une commande à partir du panier du client.
- Retrieves pizzas available : prends connaissance de toutes les pizzas disponibles à la production afin de les afficher comme disponible à la vente.

3.1.4.1 - Sous composant Authentication

Le composant authentification sert à l'identification des acteurs. Il permet au système d'afficher une interface utilisateur en fonction du type d'utilisateur connecté. Le visiteur n'ayant pour seule fonctionnalité le parcours du catalogue et la sélection des pizzas, il n'ait pas dans l'obligation de se connecter au système en tant que client, il sera toutefois considéré comme client non authentifié.

Interface :

- user session :Interface pour l'authentification d'un utilisateur.

Dépendance :

- Manage connection : pour la gestion du compte sur lequel l'utilisateur s'est authentifié.

3.1.4.2 - Sous composant Pizzas available

Le composant pizzas available affiche sur l'interface utilisateur des visiteurs, client et commercial « après sélection d'un client » les pizzas disponibles à la vente en fonction des stocks disponibles.

Interface :

- Display pizzas available : Affiche les pizzas disponibles.

Dépendance :

- Retrieves pizzas available : accès aux pizzas disponibles en stock.

3.1.4.3 - Sous composant Basket

Le composant basket permet l'enregistrement des pizzas sélectionnées par un visiteur, client, commercial. Si l'utilisateur est enregistré sur le système en tant que client, le panier « basket » est enregistré dans sa cession afin d'y avoir accès lors de sa prochaine connexion s'il décide de partir sans effectuer de commande.

Interface :

- Product basket : Panier pour l'enregistrement des produits.

Dépendances :

- User session : Récupère la session utilisateur pour récupérer les produits du panier de la dernière session utilisateur.
- Create order : Commencer le processus de commande à partir des produits du panier.

3.1.5 - Composant Accounting

Le composant accounting représente la partie comptabilité de l'application Web OC Pizza avec la gestion des comptes utilisateurs, commandes ainsi que des points de vente et fournisseur.

Interfaces :

- Manage connection : Accès à la gestion des comptes utilisateurs.
- Start order : Commencer une commande par l'interface du panier

Dépendances :

- Manage payment : Gestion du paiement des commandes.
- Manage stock : Gestion des stocks en fonction des commandes clients passées et des commandes fournisseurs

3.1.5.1 - Sous composant Account

Composant représentant le compte d'un utilisateur donnant accès aux informations du compte lié à son identification.

Interface :

- Manage connection : Donne accès aux éléments nécessaires à l'identification de l'utilisateur.

Dépendance :

- Manage account : Offre une interface de gestion du compte utilisateur.

3.1.5.2 - Sous composant Customer

Utilisateur identifié en tant que client pouvant effectuer des commandes.

Interface :

- Manage customer : Offre à un commercial l'accès au compte d'un client pour pouvoir effectuer une commande sur le compte de celui-ci.

Dépendances :

- Manage account : Gestion des données personnelles de l'utilisateur.
- Manage order : Gestion d'une commande (passer, modifier, annuler une commande).

3.1.5.3 - Sous composant Commercial

Utilisateur identifié en tant que commercial. Un commercial aura comme fonctionnalité la prise de commande pour un client.

Dépendances :

- manage account : Gestion de son propre compte.
- Manage customer : Passer une commande sur le compte d'un utilisateur.

3.1.5.4 - Sous composant Pizzaiolo

Utilisateur identifié comme pizzaiolo. Le pizzaiolo prépare les commandes des clients pour ensuite les mettre à disposition des livreurs.

Dépendances :

- Manage account : Gestion de son compte de pizzaiolo
- Prepare order: Accès aux commandes clients pour commencer la préparation des commandes en attente de préparation.

3.1.5.5 - Sous composant Deliveryman

Utilisateur identifié comme livreur. Le livreur a accès à toutes les commandes en attente de livraison afin de pouvoir effectuer leur livraison.

Dépendances :

- Manage account : Gestion de son compte de livreur
- Delivery order : Accès aux commandes en attentes de livraison.

3.1.5.6 - Sous composant Manager

Utilisateur identifié en tant que manager. Le manager est responsable d'un point de vente du groupe OC Pizza.

Dépendances :

- Manage account : Gestion de son compte manager.
- Manage sale store : Gestion du point de vente pour lequel il est responsable.

3.1.5.7 - Sous composant Order

Mécanisme de création et gestion d'une commande. Il offre des interfaces pour sa gestion par les acteurs de son élaboration et modifie les stocks en fonction de son état. Si une commande est préparée, les ingrédients nécessaires à sa préparation sont retirés des stocks ce qui permet d'avoir un suivi des stocks en temps réel.

Interfaces:

- Start order: Créer une commande à partir du panier du client ou par l'intermédiaire d'un commercial.
- Manage order : Suivi, modification, annulation et paiement d'une commande par le client ou le commercial en charge de la commande.
- Prepare order : Interface utilisée par le pizzaiolo pour avoir accès aux commandes en attente de préparation.
- Delivery order : Donne accès aux livreurs à toutes les commandes en attente de livraison.

Dépendances

- Manage payments : Permet d'accéder à l'interface de paiement de la commande.
- Manage stock : Modification des stocks en fonction des commandes effectuées.

3.1.5.8 - Sous composant Sale store

Point de vente du groupe oc pizza. Donne accès à toutes les données administratives du point de vente ainsi que de sa gestion par un manager et son approvisionnement par un fournisseur.

Interface :

- Manage sale store : Outils de gestion et de suivi du point de vente.

Dépendances :

- manage stock : Ajout de stock lors d'approvisionnements.
- Provide : Approvisionnement du point de vente par les fournisseurs.

3.1.5.9 - Sous composant Suplier

Fournisseur de produits nécessaires à la fabrication des pizzas vendus par le groupe OC Pizza ainsi que de tous autres produits nécessaires au fonctionnement d'un point de vente.

Interface :

- Provide : interface du fournisseur pour commander et recevoir des produits.

3.1.6 - Composant Bank

Composant qui représente les différents systèmes de paiement pour les commandes passées par les clients (Stripe, PayPal).

Inerface :

- Manage paiement : Interface d'accès aux API des moyens de paiement (Stripe, PayPal)

3.1.6.1 - sous composant Payment

interface de paiement d'une commande. Il offre la même interface que son composant parent.

3.1.7 - composant Warehouse

Entrepôt regroupant les stocks d'un point de vente. Les points de vente disposent d'un entrepôt qui contient un stock par produit.

Interfaces :

- manage : Interface de gestion des stocks.
- Retrieves pizzas available : retourne toutes les pizzas disponibles en fonction des stocks.

3.1.7.1 - Sous composant Stock

Stock d'un ingrédient dans l'entrepôt du point de vente. Il offre les mêmes interfaces que son composant parent.

3.2 - Application Web

La pile logicielle est la suivante :

- **Navigateur web** : (Firefox, Chrome, Opera, Yahoo...).
- **Plateforme Heroku**.
- **Framework Django - 3.0.2 :** Frameworkweb de haut niveau, rapide sécurisé et simple de maintenance.
- **Langage Python – 3.8.1**: Langage de programmation utilisé par le framework Django.
- **Heroku PostgreSQL - 11** :Système de gestion de base de données relationnel open source.
- **API Stripe** : solution de paiement en ligne.
- **API PayPal** : solution de paiement en ligne.

3.2.1 - Navigateur web

Le navigateur web permet d'accéder à l'interface de l'application web OC Pizza à partir d'un ordinateur, tablette, smartphone de l'utilisateur avec une connexion internet.

L'application web sera accessible par tout type de navigateur web mais optimisée sur les sites les plus populaires : Microsoft Edge, Safari, Chrome, Mozilla Firefox et Opera.

3.2.2 - Plateforme Heroku

Heroku est une plateforme permettant de déployer des applications sur le cloud AWS (Amazon Web Service) et ainsi bénéficier du savoir-faire d'Amazon. Il offre une interface d'utilisation simple et performante à l'aide de l'interface en ligne de commande Heroku (Heroku CLI) et utilise un tarif flexible en fonction des performances souhaitées.

3.2.3 - Framework Django

Django est un Framework web open source, puissant et de haut niveau permettant de créer des applications web en Python. Il comprend des dizaines de modules permettant la gestion des tâches courantes de développement web (L'authentification des utilisateurs, l'administration du contenu, gestion des plans du site, flux RSS...).

3.2.4 - Langage Python

Le Framework Django permet de développer des applications avec le langage Python qui est un langage de programmation orienté objet clair et puissant.

3.2.5 - Heroku PostgreSQL

Heroku PostgresSQL est un service de base de données SQL géré et fourni directement par Heroku. L'on accède à la base de données PostgreSQL à partir de n'importe quel langage à l'aide du driver PostgreSQL.

La version 11 de PostgreSQL est la version par défaut prit en charge par Heroku.

En plus de toutes les commandes de gestion disponibles avec Heroku CLI, Heroku PostgreSQL fournit une interface web permettant le partage de requêtes SQL à l'aide de Dataclips.

3.2.6 - API Stripe

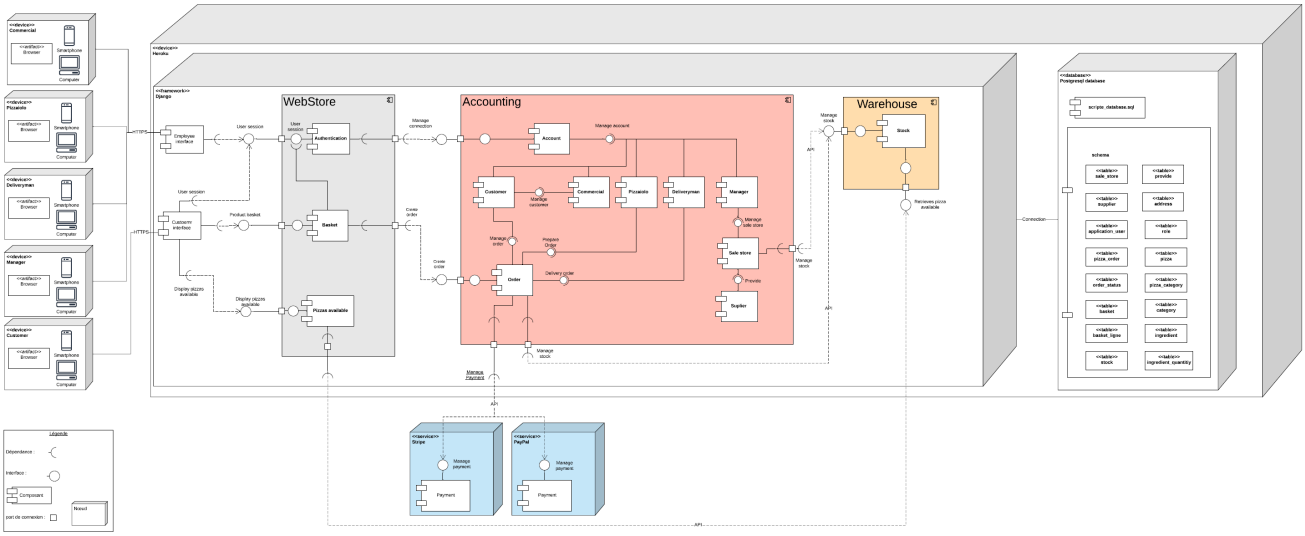
Stripe est une solution de paiement en ligne que l'on utilise à l'aide de son api REST.

Il utilise des URLs donnant accès aux ressources, accepte les formulaires encodés, retourne des réponses au format JSON et utilise les codes HTTP.

3.2.7 - API PayPal

PayPal est aussi une solution de paiement en ligne. Le plugiciel Django-PayPal permet d'implémenter facilement PayPal dans les projets Django.

4 - Architecture de Déploiement



Le diagramme de déploiement décrit le déploiement physique des composants du système. Il permet de montrer le déploiement des éléments logiciels par les éléments matériels et la nature des connexions entre les différents éléments matériels.

4.1 - Déploiement des nœuds

4.1.1 - Nœuds utilisateurs

Les nœuds des utilisateurs (Client, Commercial, Livreur, pizzaiolo, Manager) représentent les appareils permettant l'accès à l'interface utilisateur du site web OC Pizza par connexion HTTPS. Les types d'appareils par lesquels un utilisateur accède à l'interface du site OC Pizza peuvent être : un smartphone, une tablette, un ordinateur et un ordinateur portable.

4.1.2 - Nœud Heroku

La plateforme Heroku va recevoir l'application web OC Pizza développée sous le Framework Django.

Pour déployer l'application vous pouvez suivre ce lien ou les étapes qui suivent.

L'utilisation de la plateforme Heroku nécessite :

- la création d'un compte gratuit.
- Python version 3.8 installée en locale.
- PostgreSQL installé en local.

Pour le déploiement de l'application web nous allons avoir besoin de Heroku CLI qui nécessite la présence de Git.

- Installation de Git.
- Installation de Heroku CLI en fonction de votre OS.

Après l'installation sous Windows vous pouvez utiliser Heroku CLI directement à partir de votre interface de commande (cmd.exe) ou Powershell.

Pour se connecter sur Heroku CLI : **$ heroku login**

Vous devriez être redirigé sur la page de connexion du site Heroku pour rentrer vos identifiants de connexion.

Pour déployer le code de votre application sur Heroku il vous faut préparer Heroku à recevoir votre code, pour cela entrer la commande suivante : **$ heroku create app-oc-pizza**

Vous pouvez maintenant déployer l'application : **$ git push heroku master**

L'application web OC Pizza est maintenant déployée.
Vous pouvez vérifier son déploiement en `entrant la commande` : **$ heroku open**

4.1.3 - Nœud Django

Framework de développement de l'application web OC Pizza. Il contient les composants WebStore, Accounting et WhareHouse.

Pour son installation vous pouvez suivre le guide de ce lien ou la description qui suit.

L'utilisation de Django nécessite :

- Installation de Python 3.8.1
- Installation de pip

Après l'installation de python et pip vous pouvez installer Django à l'aide de la commande suivante en fonction de votre OS:

```
...\> py -m pip install Django
```

```
$ python -m pip install Django
```

Django est maintenant installé sur votre poste de travail.

4.1.4 - Nœud PostgreSQL

La base de données PostgreSQL est intégrée et gérée par Heroku. Heroku PostgreSQL offre un large éventail de plans/forfaits modifiables au besoin.

Commençons par approvisionner notre application avec la commande suivante :

```
$ heroku addons:create heroku-postgresql:<PLAN_NAME>
```

Remplaçons <PLAN_NAME> par heroku-postgresql:standard-0.

Pour utiliser PostgreSQL comme base de données dans notre application nous allons avoir besoin de l'adaptateur psycopg2 :

```
$ pip install psycopg2-binary
```

Pour effectuer une connexion sur la base de données renseignons le code qui suit :

```
import os
import psycopg2

DATABASE_URL = os.environ['DATABASE_URL']

conn = psycopg2.connect(DATABASE_URL, sslmode='require')
```

Pour se connecter avec Django nous devons installer l'utilitaire dj-database-url avec pip :

```
$ pip install dj-database-url
```

Et enfin ajouter le code qui suit dans le ficher settings.py afin de rendre la variable DATABASE_URL compréhensible par Django.

```
import dj_database_url
DATABASES['default'] = dj_database_url.config(conn_max_age=600, ssl_require=True)
```

4.1.5 - Nœud API Stripe

L'API Stripe va nous permettre d'ajouter un moyen de paiement à notre application Web. Elle dispose d'une librairie permettant d'utiliser son API avec l'aide d'une clé API.

Vous pouvez suivre les étapes de ce lien ou celles qui suivent.

Pour installer Stripe, entrons la commande : **$ pip install stripe**

Utiliser Stripe :

```
1  import stripe
2  stripe.api_key = "sk_test_4eC39HqLyjWDarjtT1zdp7dc"
```

4.1.6 - Nœud API PayPal

L'API PayPal permet elle aussi l'intégration d'un moyen de paiement. Le plugiciel Django-PayPal permet d'utiliser son API. Nous utiliserons le standard IPN afin d'avoir une meilleure gestion des logs.

Nous allons utiliser le plugiciel Django-PayPal pour implémenter PayPal dans notre application.

Lien pour la documentation.

Commande d'installation : **$ pip install django-paypal**

Ajouter django-paypal dans la liste INSTALLED_APPS du fichier settings.py :

```
#...

INSTALLED_APPS = [
    #...
    'paypal.standard.ipn',
    #...
]
```

5 - Architecture logicielle

5.1 - Principes généraux

Les sources et versions du projet sont gérées par **Git** quant à l'application web elle est développée avec Django sous forme de plusieurs modules qui constituent une app.

Le téléchargement et l'installation des paquets se font avec pip.

5.1.1 - Les couches

L'architecture applicative est la suivante :

- **une couche-modèle** : implémentation du modèle des objets métiers.
- **une couche vue** : Traitement des requêtes des utilisateurs et le renvoie des réponses.
- **Une couche gabarit** : Syntaxe pour le rendue des informations.

5.1.2 - Structure des sources

La structuration des répertoires du projet suit la logique suivante :

```
ocpizza/
    manage.py
    ocpizza/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

Description :

- ocpizza/ : Dossier qui contient tout le projet.
- manage.py : Utilitaire en ligne de commande utile à l'interaction avec le projet.
- ocpizza/ : Sous répertoire correspondant au paquet Python du projet.
- __init__.py/ : Identifie le répertoire parent comme un paquet.
- settings.py/:Réglage et configuration du projet.
- urls.py/ :Déclaration des URL du présent projet.
- asgi.py/ Point d'entrée du déploiement pour les serveurs Web compatibles ASGI.
- wsgi.py/ :Point d'entrée du déploiement pour les serveurs Web compatibles WSGI.

6 - Points particuliers

6.1 - Gestion des logs

La gestion des logs s'effectues avec le module logging directement intégré à Python.

6.1.1 - Configuration

Une configuration de journalisation Python consiste en quatre parties :

1. Journaliseurs
2. Gestionnaires
3. Filtres
4. Formateurs

6.1.1.1 - Journaliseur

Réceptacle nommé dans lequel les messages sont écrits en vue de leur traitement.

Il dispose de 5 niveaux de journalisation :

- `DEBUG`: information système de bas niveau à des fins de débogage.
- `INFO`: information système générale.
- `WARNING`: information décrivant la présence d'un problème mineur.
- `ERROR`: information décrivant la présence d'un problème majeur.
- `CRITICAL`: information décrivant la présence d'un problème critique.

6.1.1.2 - Gestionnaires

Le gestionnaire est le moteur déterminant ce qui doit arriver à chaque message d'un journaliseur. Il indique le comportement du message comme : son écriture à l'écran, dans un fichier ou dans un connecteur réseau.

6.1.1.3 - Filtres

Un filtre permet d'ajouter un contrôle supplémentaire sur la sélection des enregistrements de journal lorsqu'ils sont transmis du journaliseur au gestionnaire.

6.1.1.4 - Formateurs

Les formateurs spécifient le format du texte des enregistrements. Un formateur correspond à une chaîne de format python contenant des attributs LogRecord. Il vous est possible d'en écrire vous-même.

6.1.2 - Fichier de configuration

Le fichier de configuration « settings.py » de l'application OC Pizza est un module Python.

Il contient toute la configuration de notre projet Django.

Heroku nécessite aussi un fichier de configuration qui doit être présent dans notre projet sous le nom Procfile.

6.2 - Ressources

Python : https://docs.python.org/3/

Django : https://docs.djangoproject.com/en/3.0/

Django-paypal : https://django-paypal.readthedocs.io/en/stable/

Heroku : https://devcenter.heroku.com/categories/python-support

Stripe : https://stripe.com/docs/api

git : https://git-scm.com/doc

PostgreSQL : https://www.postgresql.org/docs/12/index.html

6.3 - Environnement de développement

Pour l'environnement de développement seul les versions des éléments cités dans ce dossier doivent être respectées ainsi que l'utilisation de Git pour la gestion de versions.

Exemple : https://developer.mozilla.org/fr/docs/Learn/Server-side/Django/development_environment.

6.4 - Procédure de livraison

La procédure de livraison s'effectuera en 4 étapes :

1. Création des comptes admin pour les responsables du groupe OC Pizza.
2. Formation des principaux acteurs du groupe OC Pizza à l'utilisation des interfaces de l'application.
3. Déploiement de l'application avec base de données sur Heroku.
4. Remise du dossier d'exploitation à l'équipe technique du groupe.

7 - GLOSSAIRE

REST	***representational state transfer*** : style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services web. Wikipedia.
JSON	***JavaScript Object Notation*** :est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Wikipedia.
codes HTTP	Permet de déterminer le résultat d'une requête ou d'indiquer une erreur au client. Wikipedia.
pip	Pip installs packages ou Pip installs python
ASGI	***Asynchronous Server Gateway Interface*** **:** interface standard entre les serveurs web, les frameworks et les applications Python asynchrone.
WSGI	**Web Server Gateway Interface :** La plate-forme principale de déploiement Django est WSGI, le standard Python pour les serveurs et applications Web.
Framework	Un framework désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel. Wikipedia.
Dataclips	Vous pouvez créer des requêtes sur votre base de données Heroku PostgreSQL et partager le résultat avec vos collègues qui peuvent au besoin les télécharger au format JSON et CSV.
Plugiciel	Plugiciel = Plugin. Paquet qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités. wikipedia