OC Pizza

Application web OCPizzapp

Dossier d'exploitation

Version 1.0

Auteur
Léonard COLIN
Développeur

TABLE DES MATIÈRES

1 - VERSIONS

Auteur	Date	Description	Version
Léonard COLIN	16/03/2021	Création du document	1.0

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier d'exploitation de l'application OCPizzapp.

Ce document a pour objectif de fournir à l'équipe technique de la société OC Pizza les données nécessaires au déploiement, à l'utilisation et à la maintenance de l'application.

2.2 - Références

Pour de plus amples informations, se référer :

1. **DCT – 1.0** : Dossier de conception technique de l'application
2. **DCF – 1.0** : Dossier de conception fonctionnelle de l'application
3. **PVL – 1.0** : Procès de verbal de livraison finale de l'application

3 - PRÉ-REQUIS

3.1 - Système

L'application OCPizzapp est hébergée sur la plateforme « Heroku ».

L'application est liée au nom de domaine « ocpizzapp.fr ».

3.2 - Bases de données

Le SGBD utilisé par l'application :

- **PostgreSQL :** version 13.2

La base de donnée de l'application est hébergée sur les serveurs de Heroku

3.3 - Web-services

Les web services suivants doivent être accessibles et à jour :

- **Google Maps APIs Web Services :** version 3.45

  Permettra de placer les points de livraison sur une carte pour pour les livreurs de pizza.

  La bibliothèque python « googlemaps » sera également utilisée afin de faciliter des outils de l'API. Pour utiliser l'API, il faudra renseigner la clé d'identification suivante : « EdcaSyD2eaDuj0ul-0Fcx2Vqu8hebo9bLH2FY5A »

- **Paypal API :** version V2

  Une solution largement répandue qui facilitera le moyen de paiement d'une commande par le client.

  La clé secrète suivante devra être utilisée :

  «ddc8_SyD2eaPLEdHheb-uj0ul-0Fcx2Vqu8he»

4 - PROCÉDURE DE DÉPLOIEMENT

4.1 - Déploiement de l'application web

4.1.1 - *Composition de l'application web*

L'application OCPizzapp construits sous la forme d'une archive ZIP contenant les répertoires suivants :

- **ocpizzapp** : les fichiers de configuration de l'application et de Django
- **static** : les fichiers css et js ainsi que les ressources images de l'application
- **templates** : les fichiers HTML de l'application
- **orders** : application du projet pour la gestion des commandes
- **accounts** : application du projet pour la gestion des comptes clients et comptes employés
- **payments** : application du projet pour la gestion des paiements
- **store** : application du projet pour la gestion des différents points de vente

Extraire l'archive **ocpizzapp.zip** dans le répertoire :

```
/home/user/ocpizzapp
```

Le fichier Procfile doit contenir à minima cette ligne de code afin que l'application soit exécutée sur la plateforme Heroku :

```
web: gunicorn ocpizzapp.wsgi --log-file -
```

4.1.2 - *Variables d'environnement*

Voici les variables d'environnement reconnues par l'application :

Nom	Obligatoire	Description
SECRET_KEY	oui	Clé secrète du projet Django.
ENV	oui	Indique à Dango qu'il faut utiliser la configuration de production.

4.1.3 - *Configuration*

Voici les différents fichiers de configuration :

- **settings.production**: fichier de configuration de l'application en production
- **Procfile** : fichier de configuration pour le déploiement sur Heroku
- **Pipfile**: fichier de configuration des dépendances du projet à installer sur le serveur

4.1.4 - Déploiement

Heroku propose différentes méthodes pour déployer une application. Nous conseillons cependant d'effectuer un déploiement en ligne de commande en utilisant « Heroku CLI ».

Les étapes présentées ci-dessous nécessite que « heroku CLI » soit installé, que le projet soit suivi avec « Git » et il faut être sur la branche « master » du projet.

4.1.4.1 - Création de l'application

Installer Heroku CLI et se connecter au compte OC-PIZZA puis se rendre à la racine du projet de l'application et entrer la commande suivante :

```
$ heroku apps:create ocpizzapp
```

4.1.4.2 - Configuration des variables d'environnement

Définir les variable d'environnement dans la configuration d'Heroku :

```
$ heroku config:set ENV= « production »
$ heroku config:set SECRET_KEY= « z0dlmk,^dza34ddk9s(g*Lop)c^y&n »
```

Vérifier que les variables ont bien été définies :

```
$ heroku config
```

4.1.4.3 - Envoyer l'application sur le serveur

Une fois les étapes précédentes réalisées, il suffit d'effectuer un « Push » sur le serveur :

```
$ git push heroku master
```

Le déploiement s'est bien réalisé si les lignes suivantes s'affichent dans le terminal :

```
remote: https://ocpizzapp.fr/ deployed to Heroku

remote: Verifying deploy… done.
```

Vérifier que l'application est bien en ligne en se rendant sur la page du site : www.ocipzapp.fr

4.2 - Déploiement de la base de données

Le déploiement de la base de données sur la plateforme Heroku nécessite que l'application ait été déployée comme présenté précédemment.

De la même manière que pour le déploiement de l'application, celui de la base de données est présenté ici en utilisant la ligne de commande et « Heroku CLI ».

4.2.1 - Installer Heroku-Postgres

Sur Heroku, la base de données est présente sous forme d'un addon.

Exécuter la commande suivante pour installer l'addon Heroku-PostgreSQL :

```
$ heroku addons:create heroku-postgresql:hobby-dev
```

4.2.2 - Configuration de la base de données

4.2.2.1 - Migration de la base de données

Exécuter un terminal sur le serveur :

```
$ heroku run bash
```

Une fois connecté au terminal, lancer cette commande pour effectuer les migrations :

```
$ python manage.py migrate
```

4.2.2.2 - Création d'un superutilisateur

Tout en restant sur le terminal du serveur, créer un superutilisateur :

```
$ python manage.py createsuperuser
```

4.2.3 - Import des données

Il est possible d'importer les données de la version de développement de l'application sur le serveur de production. Pour cela, il faut créer un dump de la base de données de développement, puis l'importer sur le serveur d'heroku.

Dans le terminal en local, créer le dump (terminal en local) :

```
$ python manage.py dumpdata ocpizzapp > ocpizzapp/dumps/ocpizzapp.json
```

Puis importer les données sur le serveur :

```
$ heroku run python manage.py loaddata pizzapp/dumps/pizzapp.json
```

Pour s’assurer que la création de la base de donnée s’est bien déroulée, on peut se rendre sur le site de Heroku, dans la rubrique « Ressources » du projet, puis dans la section « Add-ons », cliquer sur « Heroku Postgres ».
Là, des informations nous indiquent le nombre de lignes présentes dans notre base, ainsi que le nombre de tables, etc.

5 - Procédure de démarrage / arrêt

La gestion de l'application peut se faire soit depuis la plateforme Heroku, soit en ligne de commande en utilisant Heroku CLI.

5.1 - Démarrage / Arrêt

5.1.1 - *Sur la plateforme heroku*

Une fois déployée, l'application est par activée par défaut par Heroku.

- Pour arrêter l'application, cliquer sur le projet « ocpizzapp », puis sur l'onglet « Settings ».

  En bas de la page, changer le statut de la ligne « Maintenance Mode » sur On. Ceci déconnectera l'application.
- Pour la réactiver, suivre la même opération en désactivant le « Maintenance Mode » pour mettre le bouton sur OFF.

5.1.2 - *Avec Heroku CLI*

Pour arrêter l'application, exécuter la ligne suivante :

```
$ heroku ps:scale web=0
```

Pour démarrer l'application, exécuter la ligne suivante :

```
$ heroku ps:scale web=1
```

6 - PROCÉDURE DE MISE À JOUR

6.1 - Mise à jour de l'application web

Pour mettre à jour l'application, il suffit de définir le « Maintenance Mode » sur « On » sur le tabeau de bord du site Heroku, comme décrit dans la section ci-dessus.

Il est également possible l'activer et de le désactiver en exécutant respectivement les commandes suivantes :

```
$ heroku maintenance:on
```

```
$ heroku maintenance:off
```

7 - MONITORING/SUPERVISION

7.1 - Monitoring de l'application web

La plateforme Heroku fournit des outils de « monitoring » de l'application dans l'onglet « Metrics » dans le projet.

Afin de tester que l'application web est toujours fonctionnelle, nous recommandons de mettre en place les à minima les suivis ci-dessous :

- Utilisation / Saturation du **processeur** : vérifie toutes les 15 minutes si l'utilisation du processeur ne dépasse pas les 70 %. Envoie une alerte dans le cas contraire.

  La fréquence de vérification peut être revue à la baisse (par ex. toutes les 5 minutes) en cas de fort trafic sur l'application.

- Utilisation / Saturation de la **mémoire vive** : vérifie toutes les 15 minutes si l'utilisation de la mémoire vive ne dépasse pas les 70 %. Envoie une alerte dans le cas contraire.

  La fréquence de vérification peut être revue à la baisse (par ex. toutes les 5 minutes) en cas de fort trafic sur l'application.

- Stockage de la **base de données** : vérifie toutes les 30 minutes si le stockage de la base de données sur le disque du serveur ne dépasse pas les 70 %. Envoie une alerte dans le cas contraire.

  La fréquence de vérification peut être revue à la baisse.

7.2 - Supervision de l'application

En cas de panne du serveur HTTP (ici Gunicorn), Supervisor permet de vérifier son état et de le relancer automatiquement si nécessaire.

Le fichier de configuration de Supervisor se trouve dans **/etc/supervisor/conf.d/ocpizzapp-gunicorn.conf**

Si une modification est opérée dans ce fichier, il est alors nécessaire de mettre à jour la configuration de Supervisor en exécutant les commandes suivantes :

```
$ sudo supervisorctl reread

$ sudo supervisorctl update
```

Afin de vérifier que le processus tourne correctement après ces modifications, lancer la commande suivante :

```
$ sudo supervisorctl status
```

Si tout va bien, le retour de cette commande devrait afficher **« Running »**

8 - PROCÉDURE DE SAUVEGARDE ET RESTAURATION

8.1 - Sauvegarde et restauration de la base de données

Le processus de sauvegarde et de restauration de l'application passe essentiellement par la sauvegarde de la base de données de celle-ci. Heroku propose cette fonctionnalité.

- Pour sauvegarder la base de données, exécuter la commande ci-dessous :

```
$ heroku pg:backups:capture --app ocpizzapp
```

- Il est possible de lister les backups afin de restaurer la base à un instant t :

```
$ heroku pg:backups --app ocpizzapp
```

Ceci vous donnera une liste des sauvegardes effectuées avec un id pour chacune

- Pour restaurer la base de données, exécuter la commande suivante :

```
$ heroku pg:backups:restore <backup id> DATABASE_URL --app ocpizzapp
```

Pour plus d'informations sur les options de sauvegarde et restauration de l'application, vous pouvez consulter la documentation de Heroku : https://devcenter.heroku.com/articles/heroku-postgres-backups

9 - GLOSSAIRE

SGBD	Système de gestion de base de données
Django	Framework permettant la réalisation d'application web en Python
Gunicorn	Serveur web HTTP WSGI écrit en Python