IT Consulting & Development

OC Pizza

OC PIZZA

Système informatique de gestion des pizzerias

Dossier d'exploitation

Version 1.0
06/2021

Buglioni Thomas

Index

1.Présentation générale

Mise en place d'un nouveau système informatique pour l'ensemble des pizzerias du groupe.

Contexte

« OC Pizza » est un jeune groupe de pizzeria en plein essor. Créé par Franck et Lola, le groupe est spécialisé dans les pizzas livrées ou à emporter. Il compte déjà 5 points de vente et prévoit d'en ouvrir au moins 3 de plus d'ici 6 mois.

Le système informatique actuel ne correspond plus aux besoins du groupe car il ne permet pas une gestion centralisée de toutes les pizzerias.

De plus, il est très difficile pour les responsables de suivre ce qui se passe dans les points de ventes. Enfin, les livreurs ne peuvent pas indiquer « en live » que la livraison est effectuée.

Besoins exprimés par le client

● être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation ;

● suivre en temps réel les commandes passées, en préparation et en livraison ;

● suivre en temps réel le stock d'ingrédients restants pour savoir quelles pizzas peuvent encore être réalisées ;

● proposer un site Internet pour que les clients puissent :

- o passer leurs commandes, en plus de la prise de commande par téléphone ou sur place ;
- o payer en ligne leur commande s'ils le souhaitent – sinon, ils paieront directement à la livraison ;
- o modifier ou annuler leur commande tant que celle-ci n'a pas été préparée.

● proposer un aide-mémoire aux pizzaiolos indiquant la recette de chaque pizza

Date de livraison du système informatique :

Pour l'ouverture des 3 nouvelles pizzerias, dans 6 mois.

2.Presentation du document

Le présent document constitue le dossier d'exploitation de l'application de gestion de pizza, pour l'entreprise OC PIZZA

L'objectif de ce document et de rassembler les elements sur la gestion de l'application en production via un server sur « DIGITAL OCEAN »

3.Caractéristiques techniques

Hébergement : Digital ocean

Serveur (config min):

- 1 vCPU
- Memory : 1 GB
- SSD 25 GB
- Transfer 1TB

4.Procedure de déploiement

DROPLET

Sur DIGITAL OCEAN, un Droplet est un serveur. Il est sélectionné par l'acheteur et peut avoir plusieurs configurations possibles

Creation d'un DROPLET sur le site de DIGITAL OCEAN:

- ubuntu
- Utilisation de la configuration minimale (respect des caractéristiques techniques, voir plus haut section 3)
- Connexion sécurisée via SSH

Variable d'environnement

Variable 1 (obligatoire)
Fonction : Variable pour se connecter à settings de Django

Emplacement : sudo vi /etc/supervisor/conf.d/pizza_app-gunicorn.conf

Installation : uniquement après installation de NGINX/GUNICORN/SUPERVISOR

Valeur : DJANGO_SETTINGS_MODULE='pizza_app.settings.production'

User

L'acces au server en tant que « root » est à éviter par l'absence de mot de passe ainsi un super utilisateur est crée avec les même droits que root pour palier à ce problème.

Creation d'un user « sudo » et suppression de l'accès root

johnsmith@178.62.117.192

Mdp : johnsmith

Installation des dependences du projet

```
1. sudo apt-get update
2. sudo apt-get install python3-pip python3-dev libpq-dev
   postgresql postgresql-
3. git clone votredepot.git
4. sudo apt install virtualenv
5. virtualenv env -p python3
6. source env/bin/activate
7. pip install -r disquaire/requirements.txt
```

Creation de la base de donnée

Creation d'une base de donnée avec PostgreSQL, à partir de requirements.txt.

Ajout d'une database et d'un user associé.

```
sudo -u postgres psql postgres=# CREATE DATABASE disquaire;
postgres=# CREATE USER pizza_app_user WITH PASSWORD
'0+0=LaTeteàT0t0';
```

Une dernière modification est nécessaire avant de poursuivre.

La documentation officielle de Django conseille de changer certains paramètres de la base de données afin d améliorer la performance des requêtes

```
postgres=# ALTER ROLE pizza_app_user SET client_encoding TO
'utf8';
postgres=# ALTER ROLE pizza_app_user SET
default_transaction_isolation TO 'read committed'; postgres=#
ALTER ROLE pizza_app_user SET timezone TO 'Europe/Paris';
postgres=# GRANT ALL PRIVILEGES ON DATABASE pizza_app TO
pizza_app_user; \q
```

Creation du fichier settings django, « production »

Afin de simplifier l'exploitation des elements confidentiels de la configuration de l'application, settings est élaboré en 2 niveau :

- le niveau 1 correspond à une configuration local de développement
- Le niveau 2 vient effacer certains elements du niveau 1 pour l'adapter au contexte (travis, production, etc)

Il est important de preciser que le niveau 1 est sur GITHUB mais PAS LE NIVEAU 2 (donc il doit être nommé dans gitignore pour ne pas être écraser.

```
mkdir pizza_app/settings
touch pizza_app/settings/production.py
```

Configuration du fichier production.py

```
from . import *
SECRET_KEY = '-~aO;| F;rE[??/w^zcumh(9'
DEBUG = False
ALLOWED_HOSTS = ['178.62.117.192']

DATABASES = { 'default': { 'ENGINE':
'django.db.backends.postgresql', # on utilise l'adaptateur
postgresql 'NAME': ‘pizza_app', 'USER': ‘pizza_app_user',
'PASSWORD': '0+0=LaTeteàT0t0', 'HOST': '', 'PORT': '5432', } }
```

Configuration de Nginx (server)

Utilisation de Nginx comme server web

```
sudo apt-get install nginx
```

1) créer un nouveau fichier dans sites-available ;
2) ajouter un lien symbolique dans sites-enabled grâce à la commande ln. (Le chemin doit partir de la racine du système et non du répertoire courant !)

```
sudo touch sites-available/pizza_app
sudo ln -s /etc/nginx/sites-available/pizza_app /etc/nginx/sites-
enabled
```

3) ouvrir `etc/nginx/sites-available/disquaire`

```
server { listen 80; server_name 178.62.117.192; root /home/
pizza_app/disquaire/; location /static { alias /home/johnsmith/
pizza_app/pizza_app/staticfiles/; } location / { proxy_set_header
```

```
Host $http_host; proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for; proxy_redirect off; proxy_pass http://
127.0.0.1:8000; } }
```

Configuration de Gunicorn (server)

Utilisation de Gunicorn comme server web complémentaire avec Nginx

L'installation et la configuration sont automatiquement faites par le projet django (requirements.txt + fichier wsgi)

L'exploitation de Gunicorn n'est pas directement faite par le developper, Supervisor sera en charge de le (re)démarrer si nécessaire.

Installer les elements propres à l'application

Afin de simplifier l'utilisations de fichiers statiques (images, fichier js, html, css, etc, logo, etc…) il est préférable de les regrouper dans un endroit unique.
Django peut gérer cette tact via une commande spécifique à faire sur le server si il y a un changement dans les fichiers statiques

Récupérer les fichier statics

```
./pizza_app/manage.py collectstatic
```

Faire les migrations de la base de donnée

```
./pizza_app/manage.py migrate
```

Creer un super utilisateur

```
./pizza_app/manage.py createsuperuser
```

Redemarer Nginx

```
sudo service nginx reload
```

Configuration de supervisor

Utilisation de supervisor faire la gestion du site et son redémarrage si nécessaire.

```
sudo apt-get install supervisor
sudo vi /etc/supervisor/conf.d/pizza_app-gunicorn.conf
```

```
[program:pizza_app-gunicorn]
command = /home/johnsmith/env/bin/gunicorn
disquaire_project.wsgi:application
user = johnsmith
directory = /home/johnsmith/pizza_app
autostart = true
autorestart = true
environment =
DJANGO_SETTINGS_MODULE='pizza_app.settings.production'
```

Relancer le site avec les mises a jour de supervisor

```
sudo supervisorctl reread
sudo supervisorctl update
sudo supervisorctl status
```

5.Procedure de démarrage / arrêt

La procedure de démarrage est en 2 étapes :

1. Une concernant le server Nginx et la gestion l'adresse IP/fichiers statics
2. La seconde concernant l'activation de l'application par Gunicorn géré par Supervisor

---

Nginx

Nginx permet d'utiliser entre autre les fichiers static. Ils doivent être ajoutés dans un dossier spécifique via « collectstatic » associé à la commande manage.py

```
source env/bin/activate
./manage.py collectstatic
```

Pour activer la connexion (lien entre l'ip public et l'ip local) où la désactiver.

```
sudo service nginx start
sudo service nginx stop
```

Supervisor

Gunicorn permet de lancer ou relancer l'application :

```
sudo supervisorctl start pizza_app-gunicorn
sudo supervisorctl stop pizza_app-gunicorn
```

6.Procédure de mise à jour

La procedure de mise à jour ce fait en 2 étapes:

1. Recuperation de l'application via github
2. Relancement de l'application (Gunicorn)

---

1. Recuperation de l'application via github

Pour récupérer la mise à jour de l'application il suffit de faire un pull:

```
git pull origin main
```

---

2. Recuperation des fichiers static (facultatif)

Si un fichier static a été modifié ou ajouté il doit être rendu accessible à Nginx via

```
source env/bin/activate
./manage.py collectstatic
```

Et en cas de mise à jour de la configuration de Nginx:

```
sudo service nginx reload
```

---

3. Relancement de l'application (Gunicorn)

Le relancement de Gunicorn se fait via Supervisor. Celui-ci va vérifier les modifications:

```
sudo supervisorctl reread
sudo supervisorctl update
```

7.Procedure de sauvegarde et restauration

Enfin la sauvegarde de la database est géré par une tache cron :

Creation de la sauvegarde

La sauvegarde est automatisé 1 fois par semaine via cron et le script suivant

Le script bash est configuré ici:

```
/home/thomasbuglioni/cron-job/save_data_db.sh
```

Il correspond à ceci

```
pg_dump nom_base > fichier de sortie
```

Un fichier de log est associé dans le meme dossier que le script cron afin de connaitre la date de la dernière sauvegarde

---

Recuperation de la sauvegarde

La recuperation doit être utilisée uniquement en dernier recours et doit être sur une database vide et correspondante (nom)

```
psql nombase < /home/thomasbuglioni/save_data_dbsavedata.pgsql
```

Configuration de Travis CI

Travis CI est en charge de l'intégration continue est de la verification des tests (développement), il est lancé à 2 niveaux :

- sur le site de travis en créant un compte et en y associant le projet
- Dans le projet, en ajoutant un fichier « .travis.yml »

Config du fichier :

```
dist: bionic
language: python
python:
  - "3.8"

addons:
  chrome: stable
install:
  - pip install pipenv
  - pipenv install --dev
before_script:
  # ajouter tchappui-webdrivers dans requirements.txt
  - install-webdrivers --path webdrivers
env:
  global:
    - SECRET_KEY=azertyuiopqsdfghjklmwxcvbn
    - DB_NAME=travis_ci_test
    - DB_USER=travis
    - DB_PASSWORD=
    - DB_HOST=127.0.0.1
    - DB_PORT=5432
services:
  - postgresql
script:
  - python manage.py test
```

Configuration de Sentry

Sentry se charge de la gestion des logs de la production :

il est lancé à 2 niveaux :

- sur le site de Sentry en créant un compte et en y associant le projet
- Installer sentry:

```
pip install --upgrade sentry-sdk
```

- Dans le projet, dans le fichier settings.py liée à la production ajouter:

```
import sentry_sdk
from sentry_sdk.integrations.django import DjangoIntegration

sentry_sdk.init(
    dsn="https://
7be3036738a942aabe046ba3340a0ee3@o727179.ingest.sentry.io/
5784131",
    integrations=[DjangoIntegration()],

    # Set traces_sample_rate to 1.0 to capture 100%
    # of transactions for performance monitoring.
    # We recommend adjusting this value in production,
    traces_sample_rate=1.0,

    # If you wish to associate users to errors (assuming you are
using
    # django.contrib.auth) you may enable sending PII data.
    send_default_pii=True,

    # By default the SDK will try to use the SENTRY_RELEASE
    # environment variable, or infer a git commit
    # SHA as release, however you may want to set
    # something more human-readable.
    # release="myapp@1.0.0",
```