OC Pizzéria

Gestion de commande

Dossier d'exploitation

Version 2.0

Auteur
Jean baptiste Servais
Developpeur

1 - VERSIONS

Auteur	Date	Description	Version
Jean baptiste Servais	03/04/19	Création du document	01/02/00

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier d'exploitation de l'application de l'application: gestion de commande de la pizzéria OCpizzéria.

L'objectif du document est d'un de constituer un dossier informatif sur les outils que nous allons utiliser. Que soit soit au niveau de la base de donnée (postgresql) mais aussi du serveur, du déploiement de l'application (sur heroku) et du framework (django).

2.2 - Références

Pour de plus amples informations, se référer :

1. DCT – 2.0: Dossier de conception technique de l'application
2. DCF – 2.0 Dossier de conception fonctionnelle de l'application

3 - Pré-requis

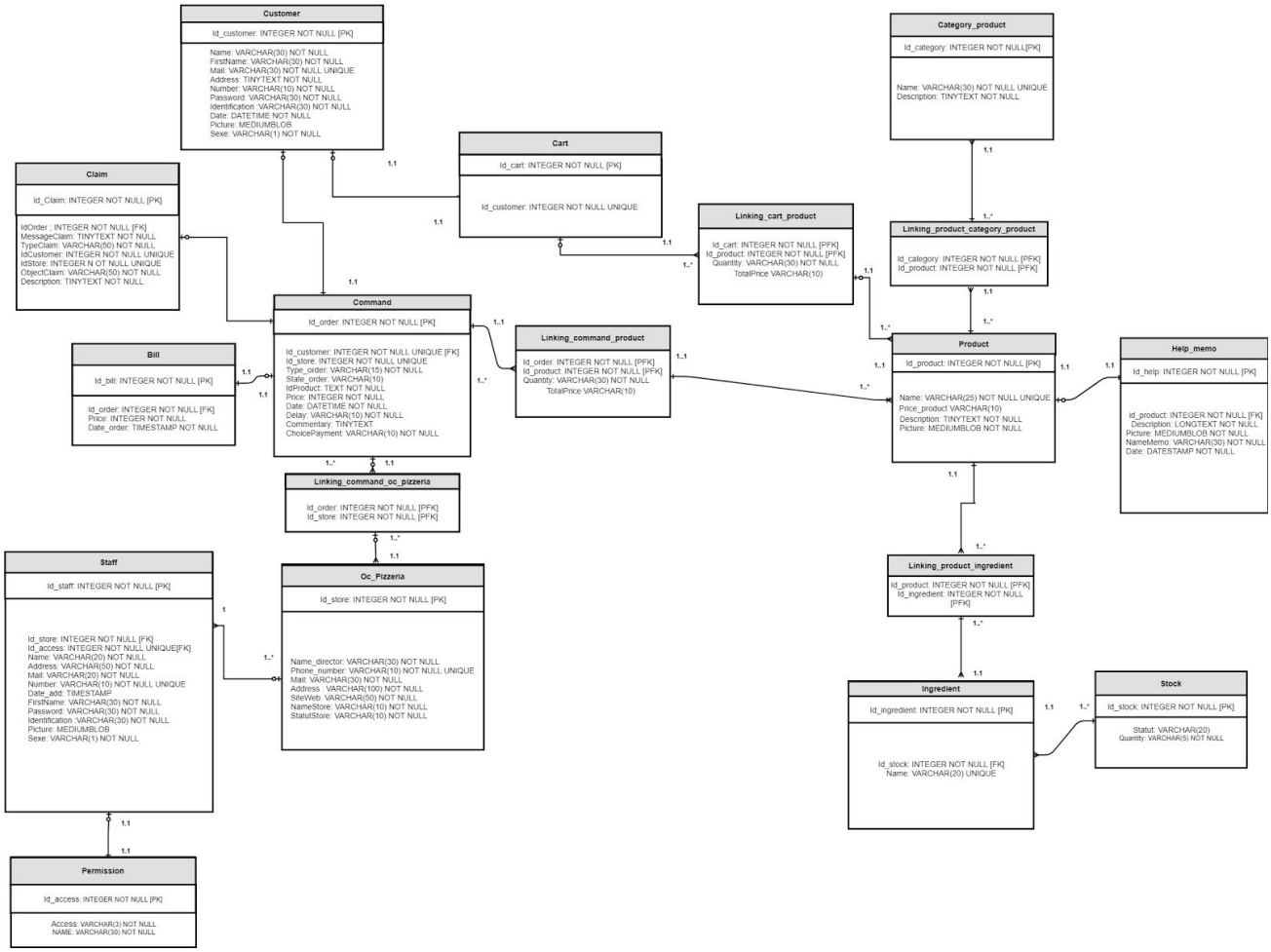
3.1 - Système

3.1.1 - Serveur de Base de données

Nous allons nous servir d'une base de donnée iWeb déployée avec un système de gestion de base de donnée (SGBD) de type PostgreSQL que nous déploieront sur Heroku.

3.2 - Bases de données

Voici le schéma de la base de donnée. La base de donnée est à jour :



3.3 - Web-services

Nous utiliseront différents web services qui sont accessibles et à jour :

- Paypal pour le paiement en ligne,
- Bootstrap (thème de template payant) pour le design de l'application.

4 - Procédure de déploiement

4.1 - Déploiement de l'application

4.1.1 - Les éléments composant l'application

Afin de pouvoir déployer notre application sur heroku il faut au préalable:

- construire un fichier .gitignore.txt car lors du push tous les fichiers inscrient dans ce fichier ne seront pas pris en compte.
- Un fichier Procfile sans extension contenant web: “gunicorn plateforme.wsgi --log-file -”.
- Un fichier requirements.txt contenant toutes les librairies qui ont été utilisé dans l'ensemble de l'application (pour cela il suffit de faire un pip freeze > requirements.txt).
- Un runtime.txt contenant la version de python que nous utilisons ici ca sera la version 3-7-1 (cela va renseigner à heroku le langage qui est utilisé).
- Un plan de test au format .txt (assurant ainsi le bon fonctionnement des pages web, des fonctions utilisées et de nos requètes api).
- Un dossier staticfiles (copie du fichier static) contenant toutes les images, css, js, thème bootstrap ...
- Un dossier templates contenant nos templates (nos pages web).
- Le dossier contenant l'application Pizzéria (celui contenant le fichier setting_production.py pour la production et settings pour le developpement)
- Le dossier contenant l'application Users.
- Le dossier contenant l'application Cart.
- Le dossier contenant l'application Claim.
- Le dossier contenant l'application Order.
- Le dossier contenant l'application Store.
- Le dossier contenant l'application Stock.
- Le jeu de donnée postgresql pour la database.

4.1.2 - Variables d'environnement

Voici les variables d'environnement de l'application Gestion de commade OcPizzéria :

Nom	Obligatoire	Description
Settings	oui	Pour signaler que nous sommes en developpement et non en production car en production nous nous servons de setting_Production.py
Secret_Key	Oui	Secret_key Django nécéssaire aux hashes
Environement virtuel	Oui	Sécurise l'environement du developpeur et du developpement avec virtualenv ou venv.
Database_url	Oui	Url menant a la base de donnée
Pip	Oui	Qui nous permet d'installer les librairies nécéssaires pour l'application.

4.1.3 - Configuration

Voici les différents fichiers de configuration :

- **Procfile** : Contient : "gunicorn plateforme.wsgi --log-file -". Grace à ce fichier Heroku utilise le serveur web gunicorne.
- **requirements.txt** : contient toutes les librairies que nous avons utilisées. Ce fichier dit à Heroku quoi installer mais renseinge aussi les autres membres de l'équipe de quels sont les librairies utilisées.
- **setting_prod.py**: settings pour la production. Signal à Heroku la nouvelle database, le nouvelle host ect.
    - contient INSTALLED_APPS
    - MIDDLEWAR
    - ROOT_URLCONF
    - TEMPLTAES
    - WSGI_APPLICATION
    - DATABASES
    - AUTH_PASSWORD_VALIDATORS
    - LANGAGE
    - TIME_ZONE
    - STATIC_ROOT
    - STATIC_URL

A noter qu'il suffit d'importer le fichier de developpement settings.py à l'importer et à apporter

les modifications dans settings_prod.py.

N'oubliez pas de mettre à debug = True → mode developpement, False production.

4.1.3.1 - Déploiement

Le déploiement sur Heroku nécéssite: *les éléments composant l'application web* en 4.1.1.

Heroku propose différentes manières de déployer l'application. La première est avec heroku CLI (à installer) et l'autre avec Github (GIT). Nous avons choisis de prendre celle avec heroku CLI bien sur nous sommes ouvert à l'aute proposition.

Pour cela il faut:

installer Heroku CLI https://devcenter.heroku.com/articles/heroku-cli afin de pouvoir interragir avec le cmd,

puis de se login a partir du cmd (ctrl+r → cmd) après s'etre inscrit à heroku sur https//:signup.heroku.com.

Une fois loggin activer votre environement virtuel.

Clonner votre dossier avec heroku ' git:clone -a application_name' (toujours sur le cmd);

positionnez vous dans votre application cd application_name.

On ajoute ! git add .,

on commit ! git commit -am "make it better" et on push ! git push heroku master.

Instructions issuent d'Heroku

4.1.3.2 - Création de l'application

Pour pouvoir faire notre application sous Django il faut par exemple:

se diriger vers le cmd ctrl + r → cmd entrée.

Créez un environement virtuel avec pip (vérifiez que vous l'avez installer lors du téléchargement de python sinon réinstaller ou modifier python en cochant pip).

Installer virtualenv avec pip install virtualenv.

Puis taper virtualenv nom_app (ici nous créons notre environement virtuel).

Activez l'environement virtuel !

Cd Script (dans notre environement virtuel) activate cd la racine,

installez Django ! Pip install django.

Créez votre projet avec django-admin startproject nom_application puis configurez vos urls et faites un petit coup de manage,py makemigrations, puis manage.py migrate.

Créez un superuser avec manage.py createsuperuser puis faites vos applications avec manage,py startapp nom_application.

4.1.4 - Vérifications

Afin de vérifier le bon déploiement de l'application allez sur Heroku et cliquez sur openapp. Si la page affiche un message comme log-rail tapez dans le cmd. Cela vous décris les erreurs. Réparez les ! S'il n'y a pas de bug l'application apparait. Attention au fichier Procfile, P majuscule, sans extension. Installer bel et bien gunicorn.

Sinon pour une première visualisation lors de votre git push heroku master regardez attentivement les différentes lignes. Cela peut etre une première indication des erreurs. A la fin normalement il devrait y avoir écrit **remote: Verifying deploy...done.** Cela est un bon signe pour un bon déploiement.

4.2 - Déploiement de la base de donnée

Direction heroku ! Il faut aller dans Ressource depuis votre dashboard depuis votre application. Dans les Adds-ons installer Heroku Postgresql. Afin de pouvoir vérifier vos données vous pouvez vous servir de psycopg2 (librairie python, pip install psycopg2)

ou vous servir de pgAdmin4 (installez le depuis internet https://www.postgresql.com, n'oubliez pas le mot de passe, sinon servez vous du root avec psql (set PATH=%PATH%; (copie de l'url du dossier BIN) postgresql -u root -p, taper le mot de passe.

Sinon servez vous de pgadmin4 configurez un serveur et rentrez les informations* issus de Heroku (cd prochain paragraphe)).

*En prenant les informations depuis : cliquez sur heroku Postgresql → réglages → voir les références et prendre les informations: hote, base de donnée, utilisateur, port, et mot de passe.

Pour Django en production allez dans vos settings et dans DATABASE mettez DATABASE['default'] = dj_database_url.config().

Sinon pour le mode developpement faites une base de donnée locale.

Il va falloir faire une migration avec heroku pour les databases et l'orm de django.

Faites depuis le cmd manage.pyheroku shell.

Vérifier les migrations avec heroku run python manage.py makemigrations. Puis migrate. Afin de voir les migrations allez dans le dossier de l'application, ouvrez le dossier migrations, normalement pour la première migrations le fichier s'apelle 001.xxx.py

Créez un superutilisateur avec heroku run python manage.py createsuperuser.

4.2.1 - Vérification

Sur Heroku allez dans ressource cliquez sur heroku postgresql (dans les Addons-on) une nouvelle page s'ouvre et vous pouvez vérifier l'état de la database. Encore une fois vous pouvez visulaliser votre database avec psycopg2 ou pgadmin4.

5 - Procédure de démarrage / arrêt

5.1 - Maintenance HEROKU

Depuis heroku nous pouvons mettre le site en maintenance.

Direction heroku, settings tout en bas vous pouvez mettre la maintenance en mode ON. Par défault elle est en off.

5.2 - Démarerrage/arret avec heroku CLI

Allez dans votre cmd. Tapez heroku run pour activé l'application.

Pour la désactiver tapez **heroku ps:scale web=0.**

Pour la réactiver tapez **heroku ps:scale web=1.**

6 - Procédure de mise à jour

Afin de pouvoir faire une mise à jour il faut mettre le site en maintenance.

6.1 - Heroku CLI maintenance

Vous pouvez faire votre maintenance directement sur heroku cité plus haut en 5.1 ou avec client en tapant les lignes suivantes:

heroku maintenance:on

ou heroku maintenance:off

7 - SUPERVISION/MONITORIN

Heroku dispose d'outils de monitoring. Allez sur heroku dans Metrics. Cepdant ajouter des métrics coute 7$. Pour de plus amples informations direction:

”https//:devcenter.heroku.com/articles/metrics”

8 - Procédure de sauvegarde et restauration

Afin de pouvoir sauvegarder et faire des restaurations vous pouvez allez sur heroku, dans activité. Vous pouvez revenir à une sauvegarde antérieure en cliquand sur revenir ici.

Sinon vous pouvez, avec le cmd sauvegarder la base de donnée avec:

heroku pg:backups:capture –app nom_application

Afin de pouvoir restaurer la base de donnée faites:

heroku pg:backups:restore<nom_sauvegarde> DATABASE_URL –app non_application

9 - GLOSSAIRE

Python	Langage de programmation
Base de donnée	Permet de pouvoir stocker des informations
SGBD	Système de gestion de base de donnée
Iweb	Base de donnée hébergée par le web
Django	framework
framework	Infrastructure logicielle
Heroku	Plateforme de déploiement