OC PIZZA

SOLUTION TECHNIQUE D'UN SYSTÈME DE GESTION DE PIZZERIA

Dossier d'exploitation

Version 1

Auteur
Sif KESSI
Analyste-Programmeur

Table des matières

1 - Versions

Auteur	Date	Description	Version
Sif KESSI	15/05/21	Création du document	1

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier d’exploitation de l'application OCPizzaManager

Il contient les informations dont l'équipe d'exploitation a besoin pour assurer une exploitation en règle du système et pouvoir réagir de manière approprié lorsqu'un problème surgit.

2.2 - Références

Pour de plus amples informations, se référer :

1. **P8_OCPizza_02_dossier_de_conception_fonctionnelle :** Dossier de conception fonctionnelle de l'application
2. **P8_OCPizza_03_dossier_de_conception_technique** : Dossier de conception technique de l'application

3 - Pré-requis

3.1 - Système

L'application web a été développée en langage JavaEE avec l'IDE IntelliJ à l'aide du Framework Spring MVC, les dependances du projet sont gérées par Maven

3.1.1 - Serveur de Base de données

Serveur de base de données hébergeant le schéma OCPizzaDB : PostGreSQL

3.1.1.1 - Caractéristiques techniques

Avantages :

● outils riches et variés (interface graphique, bibliothèques, API …) ce qui en fait un SGBD-R de qualité, robuste et puissant.

● logiciel libre, c'est-à-dire gratuit et dont les sources sont disponibles.

Fonctionnement :

● architecture client/serveur :

- une partie serveur = une application (Postmaster concernant PostGreSQL) fonctionnant sur le serveur de bases de données, capable de traiter les requêtes des clients.
- une partie client devant être installée sur toutes les machines nécessitant d'accéder au serveur de base de données (un client peut éventuellement fonctionner sur le serveur lui-même)

● Les clients peuvent interroger le serveur de base de données à l'aide de requêtes SQL.

3.1.2 - Serveur Web

Serveur physique ou virtuel hébergeant l'application web. : Apache Tomcat 8.5.1

3.1.2.1 - Caractéristiques techniques

Tomcat implémente les spécifications des Servlets et des JSP de Sun Microsystems. Il inclut des outils pour la configuration et la gestion, mais peut également être configuré en éditant des fichiers de configuration XML. Comme Tomcat inclut un serveur HTTP interne, il est aussi considéré comme un serveur web

3.2 - Bases de données

La base de données et le schéma suivants doivent être accessibles et à jour :

- **Schéma OCPizzaManager - Base de données PostGreSQL :** version 9.6.2

3.3 - Web-services

Les web services suivants doivent être accessibles et à jour :

- **GoogleMap :** permettra de convertir une adresse postale en point geolocalisable pour les livreurs, l'utilisation de cette API nécessite une clé d'identification :

  'B>WF2FzyF>opguZQ7DqoJ(Uz'

- **Monetico (Credit Mutuel) :** serveur qui permet de valider un paiement bancaire sur site web ou terminal

- **Twilio API :** elle permettra de parametrer et envoyer des SMS aux clients pour les informer du statut de leur commande

4 - Procédure de déploiement

4.1 - Déploiement de l'application web

4.1.1 - Composition de l'application web

L'application OCPizzaManager est construite sous la forme d'une archive ZIP contenant les répertoires suivants:

- **Client** : contient les fichiers liés au package « **Interface Client** »
- **GestionCommande**: contient les fichiers liés au package « **Interface Gestion commande** »
- **Administration** : contient les fichier liés au package « **Interface Administration** »
- **Templates** : contient les fichiers HTML de l'application
- **Static** : Contient les fichiers .CSS et .JS de l'application
- **Docs** : contient la documentation de l'application
- **README.md** : fichier expliquant comment lancer l'application et son fonctionnement

4.1.2 - Variables d'environnement

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

Nom	Valeur	Obligatoire	Description
ENV	PRODUCTION	OUI	Indique à l'application qu'il faut utiliser la configuration de production et non celle de développement
GMAP_API_KEY	**'B>WF2FzyF>opguZQ7DqoJ(Uz'**	OUI	Stock la clé API nécessaire au bon fonctionnement du web service GoogleMap

4.1.3 - Configuration

Voici les différents types de fichiers de configuration necessaire à l'application :

4.1.3.1 - Fichier log4j.xml

fichier de configuration des logs

4.1.3.2 - Fichier application.properties

fichier de configuration de l'application

4.1.3.3 - Fichier pom.xml

fichier des dependances maven

4.1.4 - Deploiement

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 positionné sur la branche « master » du projet.

4.1.4.1 - Création de l'application :

Une fois « Heroku CLI » installé, il faut se rendre à la racine du projet de l'application et entrer la commande suivante :

$ heroku create ocpizzamanager

4.1.4.2 - Configuration des variables d'environnement :

Les variables d'environnement se définissent via la ligne de commande de la façon suivante:

$ heroku config:set ENV='PRODUCTION'

$ heroku config:set SECRET_KEY='B>WF2FzyF>opguZQ7DqoJ(Uz'

Pour vérifier que les variables ont bien été définies, utilisez la commande suivante:

$ 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

4.1.5 - Vérifications

Une fois le déploiement terminé, les lignes suivantes doivent apparaitre dans le terminal :

remote : https://ocpizza.fr/ deployed to Heroku

remote : Verifying deploy... done.

Pour s'assurer du bon déploiement de l'application, rendez-vous à l'adresse www.ocpizza.fr afin de vérifier que celle-ci soit bien mise en ligne.

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 « Heroku CLI ».

4.2.1 – Connection à la base de données

Pour que Heroku crée la base de données, lancer la commande suivante :

$ heroku addons:create heroku-postgresql

4.2.2 - Vérifications

Afin de vérifier la création de la base de données, on lance la commande :

$ heroku pg

La vérification de l'état de la base de données peut se faire directement via le site d'Heroku dans la rubrique « Ressources » du projet.

Il faut ensuite se rendre dans la partie « Add-ons » puis cliquer sur « Heroku PostGreSQL ».

5 - Procédure de démarrage / arrêt

5.1 - Base de données

Afin de démarrer la base de données, lancer la commande suivante :

$ pg_ctl -D data -l journaltrace start

La commande *pg_ctl* est dédiée au contrôle du serveur PostgreSQL et donc au démarrage du serveur postgres.

L'option *-D* permet de préciser le répertoire contenant les données et les fichiers de configuration.

L'option *-l* permet de rediriger les messages vers un fichier de traces et l'option *start* permet de préciser l'action à effectuer.

Les options *start, stop, restart et reload* permettent respectivement de démarrer, d'arrêter, de redémarrer et de recharger le serveur.

5.2 - Application web

Afin de démarrer l'application, lancer la commande suivante :

$ heroku run

Pour arreter l'application, lancer la commande suivante :

$ heroku ps:scale web=0

6 - Procédure de mise à jour

La mise à jour de l'applicaton necessite de la passer en maintenance. Lors de la mise en maintenance, les utilisateurs n'auront plus accès au site.

Pour la mise en maintenance de l'application, lancer la commande suivante :
$ heroku maintenance:on

L'application peut être réactivée en lancant la commande suivante :
$ heroku maintenance:off

On peut verifier l'état de maintenance de l'application en lancant la commande :
$ heroku maintenance

7 - Supervision/Monitoring

On peut surveiller que l'application fonctionne bien depuis Heroku. L'application propose le service "Metrics". Depuis le tableau de bord d'Heroku on pourra surveiller les charges du serveur et le trafic.

Nous recommandons de mettre en place une surveillance sur ces 3 critères minimum :

- Rapprochement de la limite de stockage de la base de données
- Saturation de la memoire vive
- Saturation du processeur

Si une erreur ou un dysfonctionnement surviennent, on peut vérifier les logs du serveur en lançant la commande suivante :

$ heroku logs

8 - Procédure de sauvegarde et restauration

8.1 - Sauvergarder une base de données

Pour effectuer une sauvegarde de la base, on peut lancer ces deux commandes sur Heroku:

$ heroku pg:backups:capture --app ocpizzamanager

Cette commande permet de figer et de sauvegarder la base à un instant T. Puis pour la télécharger et la conserver, il faut effectuer la commande suivante:

$ heroku pg:backups:download

Afin de mettre en place un processus de sauvegarde efficace, on paramètre une tâche redondante à l'aide de la commande suivante :

$ heroku pg:backups:schedule: DATABASE_URL --at '08:00 France/Paris' --app ocpizzamanager

Pour vérifier que notre tâche est bien prise en compte, on lance la commande suivante :

$ heroku pg:backups:schedules --app ocpizzamanager

On obtient alors la liste de nos tâches en cours. Pour afficher la liste de nos sauvegardes, on lance la commande suivante :

$ heroku pg:backups --app ocpizzamanager

8.2 - Récupérer une base de données

Pour effectuer une récupération de la base de données, on lance la commande suivante :

$ heroku pg:backups:restore DATABASE_URL --app ocpizzamanager

8.3 - Importer une base de données

Nous devons d'abord compresser la base de donnée dans le bon format à l'aide de la commande suivante :

$ PGPASSWORD=*motdepasse* pg_dump -Fc --no-acl --no-owner -h localhost -U *utilisateur nomdelabasededonnées* > *nomdufichier*.dump

Ensuite il faut créer une URL signée (avec Amazon S3) comme suit:

$ aws s3 presign s3://*monadresse/monobjet*

Et enfin importer la base de donnée dans Heroku:

$ heroku pg:backups:restore '*monurlsignée*' DATABASE_URL