OC Pizza
Système de gestion de Pizzérias

Dossier d'exploitation

Version 1.1

Auteur
Rémy VALLET
Développeur d'application{{Auteur_Role}}

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
Rémy VALLET	13/11/2020	Création du document – Rédaction Chapitres 2 & 3	1.0
Rémy VALLET	20/11/2020	Rédaction des chapitres 4 à 9	1.1

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier d'exploitation de l'application ***OC Pizza*** à l'attention des développeurs, à la maintenance applicative et à l'équipe technique.

L'objectif de ce document est de formaliser les instructions de déploiement et de maintenance du système d'information.

2.2 - Références

Pour de plus amples informations, se référer :

1. **Projet_8-Dossier_de_conception_technique** : Dossier de conception technique de l'application
2. **Projet_8-Dossier_de_conception_fonctionnelle** : Dossier de conception fonctionnelle de l'application
3. **Projet_8-PV_Livraison** : Procès-verbal de la livraison finale.

3 - PREREQUIS

3.1 - Système

L'application back-office, front-office ainsi que la BDD PostgreSQL est hébergé sur Heroku.

Elle est liée au nom de domaine 'oc-pizza.fr' enregistré sur Gandi (https://www.gandi.net/fr).

Le déploiement s'effectue avec l'interface Heroku CLI téléchargeable sur le site (https://devcenter.heroku.com/articles/deploying-spring-boot-apps-to-heroku).

3.1.1 - Serveur de Base de données

Le serveur de base de données hébergeant la base oc_pizza est PostgreSQL dans sa version 13.1, en raison du choix de l'hébergeur (Migration https://devcenter.heroku.com/articles/heroku-mysql).

3.1.2 - Serveur Web

Le serveur hébergeant l'application web est NGINX pour des raisons de performance. Un serveur web open-source léger et performant avec une faible utilisation de la mémoire avec un traitement asynchrones des requêtes et un équilibreur de charge.

3.2 - Bases de données

Les bases de données et schémas suivants doivent être accessibles et à jour :

- **BD oc_pizza :** version 1.0

3.3 - Web-services

Les web services suivants doivent être accessibles et à jour :

- **API de paiement Ingenico :** version 1.0 (https://www.ingenico.com/fr/omnicanal | https://epayments-api.developer-ingenico.com/)
- **API Google Maps** : (https://cloud.google.com/maps-platform/routes)

3.4 - Autres Ressources

Les micro-services de l'application doivent tous avoir été livrés sur la branche de release portant le nom de la version et le pom.xml parent doit contenir ces numéros de versions.

4 - PROCEDURE DE DEPLOIEMENT

4.1 - Déploiement de l'Application

4.1.1 - Déploiement du front-office

Utilisation du déploiement automatique sur Heroku. Sélectionner l'application Heroku-CLI et vérifier sur le menu déploiement que le déploiement est bien paramétré sur Github.

Déploiement automatique : sélectionner la branche master de l'application et activer le déploiement automatique.

Déploiement manuel : sélectionner déploiement manuel puis déployer la branche.

4.1.1.1 - Fichier package.json

- ✓ Vérifier que les dépendances 'dependencies' soient à jour par rapports aux DevDepedencies : particulièrement "typescript", "@angular/cli" et "@angular/compiler-cli" avec les versions souhaitées, ainsi que "engines" avec les versions de 'node' et 'npm' souhaitées.
- ✓ Ajouter la ligne "heroku-postbuild": "ng build --prod" sous les 'scripts'.
- ✓ Vérifier que la configuration de démarrage est bien sur node "start": "node server.js",

4.1.2 - Déploiement du back-office

Utilisation du déploiement automatique via git. Après avoir effectué le commit d'un service, lancer les commandes à partir de la branche master.

```
heroku login // Entrez les identifiant Heroku du projet
heroku create // Ajout de la git remote heroku
git push heroku master
```

4.1.2.1 - Fichier pom.xml

- ✓ Vérifier que le POM parent contient bien les informations de build pour la copie des dépendances (« maven-dependency-plugin »).

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-dependency-plugin</artifactId>
      <version>3.0.1</version>
      <executions>
        <execution>
          <id>copy-dependencies</id>
          <phase>package</phase>
          <goals><goal>copy-dependencies</goal></goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

Le serveur d'application doit être exécuté avec la variable d'environnement suivante définie au démarrage.

ENV	PROD
heroku config :set ENV= 'PROD'	

4.1.4 - Cloud-config

L'ensemble des fichiers de configurations nécessaires au lancement des modules est contenus dans ce micro-service.

4.1.4.1 - Fichiers gateway-bmo.yml, ms-user.yml, ms-orga.yml, ms-product.yml, ms-order.yml, ms-transaction.yml

Contiennent le paramétrage de l'application pour l'enregistrement auprès du serveur Eureka et l'accès à la base de données.

4.1.4.2 - Fichier orchestrator-job.yml

Contient les paramètres relatifs à la gestion des batch pour l'activation et la programmation des jobs planifiés (envoie d'email, génération de rapports…)

```
orchestrator-job:
  schedulers:
    mail-job-properties:
      job-name: DailyReportByPizzeria
      enabled: true
      cron-expression: "0 0 08 ? * MON,TUE,WED,THU,FRI *"
```

4.1.5 - DataSources

Les accès aux bases de données doivent se configurer à l'aide des fichiers de propriétés ‘application.yml’ externalisés.

```
spring:
  datasource:
    url: ${JDBC_DATABASE_URL}
    username: ${JDBC_DATABASE_USERNAME}
    password: ${JDBC_DATABASE_PASSWORD}
```

4.1.6 - Vérifications

Afin de vérifier le bon déploiement de l’application, la ligne suivante doit apparaitre dans le terminal

```
Remote : http://www.oc-pizza.fr/ deployed to heroku
Remote: Verifying deploy… done.
```

5 - PROCEDURE DE DEMARRAGE / ARRET

5.1 - Batches

Les batches peuvent être stoppé et arrêtés en paramétrant le batch à ‘enabled : true/false’ sur cloud-config puis en redémarrant le service orchestrator-job.

5.2 - Application et micro-services

Les micro-services peuvent être stoppé et redémarrer sur le serveur avec les commandes :

```
stop_service NomDuService
restart_service NomDuService
```

L’application peut être mis en mode maintenance dans la rubrique paramètres (‘settings’> ‘Maintenance Mode’ > ‘ON’) ou via la console Heroku-CLI avec les commandes suivantes.

```
/* Maintenance */
heroku maintenance:on
heroku maintenance:off

/* Arrêt et redémarrage */
heroku ps :scale web=0
heroku run
```

Les micro-services doivent être redémarrés dans l’ordre suivant :

- Cloud-config
- Eureka-server
- Les micro-services : ms-user, ms-orga, ms-product, ms-order et ms-transaction
- Les gateway et jobs : gateway-bmo et orchestrator-job

6 - PROCEDURE DE MISE A JOUR

La mise à jour d'un micro-service ou de l'application web nécessite de passer en mode maintenance l'application.

Il suffit de déployer sur git la branche master pour lancer le déploiement sur le serveur heroku.

Une fois le déploiement effectué, redémarrer le service et vérifiez les logs de démarrage.

6.1 - Ajustement de la charge serveur : Scaling & load balancing

Heroku utilise des Dynos, container applicative, permettant la scalabilité du serveur. L'équilibrage de charge (load balancing) s'effectue entre les dynos déployées.

- ✓ De façon horizontale (scale out) par l'ajout de dynos supplémentaires pour supporter la monté en charge des appels http concurrentiels.
  - ❖ Soit en ajustement auto-scaling, réglage que nous utilisons pour nos applications et qui répond à notre besoin d'ajuster la charge pendant les pics de trafics et de commandes aux heures d'ouverture des pizzerias.
  - ❖ Soit manuellement si le trafic est prévisible et continue sur la journée en ajoutant par exemple deux dynos supplémentaires avec la commande

    heroku ps:scale worker=2

- ✓ De façon verticale (scale up) par l'augmentation de la puissance des dynos (RAM, CPU, Puissance de calcul).

6.2 - Accès aux LOGS

L'accès aux logs s'effectue via Heroku CLI avec la commande : heroku logs –tail

Les logs centralisés agrègent les logs des applicatifs, du système (infrastructure heroku), de l'API Heroku (tel que la maintenance, le deploiement…) et des services additionnels heroku (add-ons).

Pour filtrer, il faut utiliser les paramètres mis à disposition. Par exemple pour avoir les logs en temps réel (--tail) de l'application pizzaApp (--source pizzaApp) sur la dynos n°1 (--dyno worker.1) :

heroku logs --source app --dyno worker.1 --tail

7 - SUPERVISION/MONITORING

7.1 - Supervision de l'application web

Afin de tester que l'application web est toujours fonctionnelle, nous mettons en place des tests automatisés qui sont lancés après chaque livraison.

Cette batterie de tests couvre toutes les parties fonctionnelles de l'IHM et simule une activité.

Les tests automatisés sont mis à jour en parallèle des évolutions de l'application. Ils fournissent un rapport généré qui indique précisément les fonctionnalités qui ne se comportent plus comme prévue.

- Tests de bout en bout (E2E) : ensembles de tests allant de l'identification de l'utilisateur jusqu'à la livraison ou au retrait.

Les informations sont disponibles dans la documentation Robot Framework à l'adresse suivante : https://robotframework.org/#documentation

Le monitoring du serveur est effectué via les outils fournis par l'hébergeur dans la rubrique Monitoring et Metrics avec notamment des surveillances sur l'usage Mémoire et CPU et de saturation de BDD.

Les informations sont disponibles dans la documentation Heroku à l'adresse suivante :

https://devcenter.heroku.com/categories/reference

8 - PROCEDURE DE SAUVEGARDE ET RESTAURATION

La base de données est sauvegardée de matière automatisée via le module Heroku PGBackups.

Installation du module à partir de la console Heroku-CLI

```
heroku addons:add pgbackups
```

Lancement manuel d'une sauvegarde :

Soit directement avec le nom d'application (se réfère à la variable d'application DATABASE_URL) soit en spécifiant le nom de la BDD.

```
heroku pgbackups:capture –app PizzaApp
heroku pg:backups:capture HEROKU_POSTGRESQL_OCPIZZA
```

Vérification/Historique des sauvegardes

```
heroku pgbackups
```

Restauration d'une sauvegarde

```
heroku pgbackups:restore DATABASE #refDB
```