OC PIZZA

Dossier d'exploitation

Version 1.1

Auteur
AUBRUN Éric
Analyste-programmeur

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
Eric AUBRUN	10/05/2021	Création du document	1.0
Eric AUBRUN	23/05/2021	Finalisation du document	1.1

2 - INTRODUCTION

2.1 - Objet du document

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

Ce document décrit les spécifications techniques du déploiement de cette application.

2.2 - Références

Pour de plus amples informations, se référer :

1. **PDOCPizza_01_fonctionnelle.pdf – 1.0:** Le Dossier de conception technique de l'application
2. **PDOCPizza_02_technique.pdf – 1.0** : Le Dossier de conception technique de l'application
3. **PDOCPizza_04_livraison.pdf – 1.0:** Le PV de livraison finale

3 - PREREQUIS

3.1 - Système

3.1.1 - Serveur de base de données

Le serveur de base de données est basé sur PostgreSQL 12.5. Il est hébergé sur un serveur de type linux Debian.

3.1.2 - Serveur Web

Le serveur est hébergé chez le fournisseur de l'application. Il héberge, la base de données, le service web.

L'ensemble des images (jpg) et des différentes ressources utilisées par le client sont hébergés sur ce serveur.

Ces services sont hébergés sur des serveurs Apache Tomcat.

3.1.3 - Serveur Batch et Web Services

Le serveur de batch et du Web Service accueille l'application permettant de commander automatiquement les produits et ingrédients. Il lisse sur chaque point de vente les différents niveaux de stock permettant la poursuite de l'activité sans interruption.

3.2 - Bases de données

La base de données doit être accessible lors du chargement du web service, elle est intitulée OCPizza.

Paramètres de connexion à la BDD :

url=jdbc:postgresql://localhost:5432/OCPizza

username=postgres

password=postgres

Si vous modifiez ces paramètres, vous devez modifier le fichier application.properties du

service web.

3.3 - Web-services

Le fonctionnement de l'application nécessite que les web-services suivants soit activés et opérationnels :

3.3.1 - PAYMILL, API de paiement en ligne intégrant des bibliothèques Java

Un pont JavaScript va permettre de récupérer les informations de paiement de manière sécurisée.

3.3.2 - JDBC (Java DataBase Connectivity) API d'accès aux bases de données relationnelles

Toutes les classes de JDBC sont dans le package java.sql. Il est donc nécessaire de l'importer dans tous les programmes devant l'utiliser (import java.sql. * ;).

3.3.3 - SMS API d'envoi de SMS en java via un SDK

Cette API est constituée de 3 méthodes : une méthode de réception de SMS, une méthode de création de SMS et une méthode batch d'envoi de SMS.

3.3.4 - JMM : API de géolocalisation « Java MaxMind »

Un import dans le POM parent ainsi qu'un chargement dans le Main de l'application permet d'avoir accès à la géolocalisation.

3.3.5 - JAAS (Java Authentication and Authorization Service) : API de gestion de l'authentification.

Cette API donne accès aux deux fonctionnalités d'authentification et d'autorisation de manière sécurisée.

3.3.6 - JavaMail API pour la gestion de courrier électronique.

Cette API permet de déployer la fonctionnalité d'envoi de mail aux clients de sorte que ceux-ci puissent avoir accès aux niveaux d'avancée de leurs commandes. Nous pouvons, si nous le souhaitons, injecter cette API dans un batch dans le but d'automatiser l'envoie de mail.

4 - PROCEDURE DE DEPLOIEMENT

4.1 - Déploiement des Batches

4.1.1 - Artefacts

Les batches de l'application OCPizza sont construits sous la forme d'une archive ZIP contenant les répertoires :

- **bin** : les scripts SH de lancement des différents batches et les fichiers jar
- **conf** : les fichiers de configuration

Il est nécessaire d'extraire l'archive **OCPizza.zip** et de positionner les droits d'exécution sur les scripts SH de lancement des batches.

4.1.2 - Variables d'environnement

Ces applications sont développées à l'aide du langage java. Voici les variables d'environnement reconnues par les batches de l'application OCPizza :

Nom	Obligatoire	Description
JAVA_HOME	non	Répertoire racine de l'installation de l'application

Ces services sont développés à l'aide de java 1.8. Votre version de java doit donc être au minimum la version 1.8.

4.1.3 - Configuration

Voici les différents fichiers de configuration :

- **Log4j.xml :** fichier de configuration des logs pour l'application Angular
- **application.properties :** fichier de configuration contenant notamment l'adresse du web service.

4.1.3.1 - Vérifications

Un batch teste le fonctionnement de la base de données, du service web et de l'application web. Ces tests sont initiés à l'aide de Maven.

4.2 - Déploiement de l'Application Web

4.2.1 - Artefacts

L'application est « zippée » dans deux fichiers : Le service web (OCPizzaWeb.zip) et le client web (Angular.zip).

L'application pourra être packagée à l'aide de Maven ou à l'aide de npm pour Angular. L'application est versionnée sur GitHub et est disponible à l'adresse suivante : https://github.com/RicoBSJ/Projet_8_OC_DA_Java_J2EE.

La commande « mvn clean package » permet de packager le web service.

4.2.2 - DataSources

Le fichier application.properties configure l'accès aux bases de données.

4.2.3 - Vérifications

Lors de la compilation de l'application par Maven, des tests sont exécutés. Les résultats de ces tests sont publiés dans un rapport.

5 - PROCEDURE DE DEMARRAGE / ARRET

Le démarrage de l'application s'exécute de la manière suivante :

⇒ Démarrage de la base de données

⇒ Chargement des données

⇒ Démarrage du webservice

⇒ Démarrage des batch de test

⇒ Démarrage de l'application web

Les étapes d'arrêt de l'application sont les suivantes :

⇒ Arrêt de l'application web

⇒ Arrêt du webservice

⇒ Dump de la base de données

⇒ Arrêt de la base de données

La commande d'arrêt de Tomcat est la suivante : « ./shutdown.sh » (/usr/lib/apache-tomcat/bin/shutdown.sh).

5.1 - Base de données

Le fichier data.sql permet de charger, au lancement de l'application back-end, un jeu de données.

5.2 - Batches

Enregistré dans le dossier des librairies externe du client web, celui-ci exécute le batch.

5.3 - Application web

L'application web peut être initialisée dès lors qu'elle est déployée sur le serveur web, sous la forme d'un fichier jar.

6 - PROCEDURE DE MISE A JOUR

La tranche horaire de 00h00 à 06h00 sera utilisée pour appliquer les patchs de mise à jour de l'application.

Toute évolution ou rajout d'une nouvelle fonctionnalité devra faire l'objet d'un nouveau cahier des charges.

Préalablement à toute évolution, les anciens fichiers jar seront enregistrés dans le dossier /oldJar.

Une fois les nouveaux fichiers jar enregistrés dans le dossier de TomCat, l'application pourra être redémarrée selon la procédure précédente.

6.1 - Base de données

La base de données peut être restaurée en cas de nécessité, opération pouvant être facturée, mais toute évolution devra elle aussi figurer dans un nouveau projet.

6.2 - Batches

Les batchs pourront évoluer en fonction des besoins du client, ils feront partie, à ce titre, d'une nouvelle transaction.

La mise à jour du batch génèrera un nouveau fichier jar et un nouveau fichier sh.

6.3 - Application web

Pour toute version upgradée de l'application, le nouveau fichier jar sera directement chargé sur le serveur Tomcat lors des phases de mise à jour.

7 - SUPERVISION/MONITORING

7.1 - Supervision de l'application web

Dans le but de tester l'application web, vous pouvez vous rendre à l'adresse suivante : https://OCPizza.fr

Pour le cas où l'application ne serait pas accessible, nous vous invitons à nous adresser un mail à Maintenance@RicoDevFSProject.fr

En exécutant la commande mvn test, vous pouvez tester l'application dans son ensemble.

8 - PROCEDURE DE SAUVEGARDE ET RESTAURATION

La base de données est quotidiennement sauvegardée à 23h00.

Toute restauration du système fera l'objet d'une facturation.