|  |
| --- |
| **OC PIZZA**  **Projet 8 : Documentez votre système de gestion de pizzeria**  Dossier d'exploitation  Version 1.0 |
| **Auteur**  AUBRUN Éric  *Analyste-programmeur* |

Table des matières

1 - Versions 4

2 - Introduction 5

2.1 - Objet du document 5

2.2 - Références 5

3 - Prérequis 6

3.1 - Système 6

3.1.1 - Serveur de base de données 6

3.1.2 - Serveur Web 6

3.1.3 - Serveur Batch et API 6

3.2 - Bases de données 6

3.3 - Web-services 7

3.3.1 - PAYMILL, API de paiement en ligne intégrant des bibliothèques Java 7

3.3.2 - JDBC (Java DataBase Connectivity) API d’accès aux bases de données relationnelles 7

3.3.3 - SMS API d’envoi de SMS en java via un SDK 7

3.3.4 - JMM : API de géolocalisation « Java MaxMind » 7

3.3.5 - JAAS (Java Authentication and Authorization Service) : API de gestion de l’authentification. 7

3.3.6 - JavaMail API pour la gestion de courrier électronique. 8

4 - Procédure de déploiement 10

4.1 - Déploiement des Batches 10

4.1.1 - Artefacts 10

4.1.2 - Variables d'environnement 10

4.1.3 - Configuration 10

4.1.3.1 - Vérifications 11

4.2 - Déploiement de l'Application Web 12

4.2.1 - Artefacts 12

4.2.2 - Environnement de l’application web 12

4.2.2.1 - Variables d’environnement 12

4.2.3 - Répertoire de configuration applicatif 12

4.2.3.1 - Fichier xxx.yyy 12

4.2.4 - DataSources 12

4.2.5 - Ressources 13

4.2.6 - Vérifications 13

5 - Procédure de démarrage / arrêt 14

5.1 - Base de données 14

5.2 - Batches 14

5.3 - Application web 14

6 - Procédure de mise à jour 15

6.1 - Base de données 15

6.2 - Batches 15

6.3 - Application web 15

7 - Supervision/Monitoring 16

7.1 - Supervision de l’application web 16

8 - Procédure de sauvegarde et restauration 17

9 - Glossaire 18

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Auteur | Date | Description | Version |
| Eric AUBRUN | 10/05/2021 | Création du document | 1.0 |
|  |  |  |  |

# Introduction

## 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.

## 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

# Prérequis

## Système

### 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.

### Serveur Web

Le serveur est hébergé́ 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.

### Serveur Batch et API

Le serveur de batch et de l’API 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.

## Bases de données

La base de données doit être accessible lors du chargement du web service. La base de données 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.

## Web-services

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

### PAYMILL, API de paiement en ligne intégrant des bibliothèques Java

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

### 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. \* ;).

### 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.

### 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.

### JAAS (Java Authentication and Authorization Service) : API de gestion de l’authentification.

Pour la mise en place de cette API, nous suggérons le recours à l’excellent « Guide to the Java Authentication and Authorization Service (JAAS) » :

« *When using JAAS in an application, several APIs are involved:*

* *CallbackHandler: Used for gathering user credentials and optionally provided when creating the LoginContext*
* *Configuration: Responsible for loading LoginModule implementations and can be optionally provided at LoginContext creation*
* *LoginModule: Effectively used for authenticating users*

*We'll use the default implementation for the Configuration API and provide our own implementations for the CallbackHandler and the LoginModule APIs*. »[[1]](#footnote-1)

### JavaMail API pour la gestion de courrier électronique.

Il faut créer une classe qui permette le chargement des bibliothèques nécessaires ainsi que la création de divers objets nécessaire à l’implémentation de l’API :

import java.util.Properties;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

public class SendEmail {

public static void main(String[] args) {

// Recipient's email ID needs to be mentioned.

String to = "destinationemail@gmail.com";

// Sender's email ID needs to be mentioned

String from = "fromemail@gmail.com";

final String username = "manishaspatil";//change accordingly

final String password = "\*\*\*\*\*\*";//change accordingly

// Assuming you are sending email through relay.jangosmtp.net

String host = "relay.jangosmtp.net";

Properties props = new Properties();

props.put("mail.smtp.auth", "true");

props.put("mail.smtp.starttls.enable", "true");

props.put("mail.smtp.host", host);

props.put("mail.smtp.port", "25");

// Get the Session object.

Session session = Session.getInstance(props,

new javax.mail.Authenticator() {

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(username, password);

}

});[[2]](#footnote-2)

A la suite de quoi, nous devons instancier les objets précédemment créés afin de constituer le pattern de l’API. Nous pouvons, si nous le souhaitons, injecter cette API dans un batch dans le but d’automatiser l’envoie de mail.

# Procédure de déploiement

## Déploiement des Batches

### Artefacts

Les batches de l’application Xxx 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

Extraire l'archive **XXX.zip** dans le répertoire :

**/xxx/yyy**

Positionner les droits d'exécution sur les scripts SH de lancement des batches.

### 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.

### 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.

#### Vérifications

Afin de vérifier le bon déploiement du système, un batch de test permet de tester le bon fonctionnement de la base de données ainsi que du service web et de l’application web. Ces procédures de test peuvent être initiées à l’aide de Maven. Ils sont automatiquement exécutés dans le cadre du packaging du war.

## Déploiement de l'Application Web

### Artefacts

zip

### Environnement de l’application web

#### Variables d’environnement

Le serveur d'application JOnAS doit être exécuté avec la variable d'environnement suivante définie au démarrage. Elle est nécessaire afin de récupérer le répertoire contenant les fichiers de configuration de l'application :

**-Dcom.ocpizza.apps.conf=$home\_application\_conf\_directory**

INFO : il ne faut pas mettre de « / » à la fin de la valeur de la variable et ne pas utiliser d'espace dans le chemin.

### Répertoire de configuration applicatif

Le répertoire de configuration applicatif doit être créé sur le système de fichier et définit de la façon suivante :

**$home\_application\_conf\_directory/applicationX**

… fichiers de configuration… :

* ...

#### Fichier xxx.yyy

...

### DataSources

Les accès aux bases de données doivent se configurer à l'aide des fichiers…

Le fichier de drivers **postgresql (postgresql-9.2.x.)** doit être déposé dans le répertoire :

**$home\_server/lib/ext**

...

### Ressources

...

### Vérifications

Afin de vérifier le bon déploiement de l’application, faire ceci…

# Procédure de démarrage / arrêt

## Base de données

## Batches

## Application web

# Procédure de mise à jour

## Base de données

## Batches

## Application web

# Supervision/Monitoring

## Supervision de l’application web

Afin de tester que l’application web est toujours fonctionnelles, faire ceci…

# Procédure de sauvegarde et restauration

# Glossaire

|  |  |
| --- | --- |
|  |  |
|  |  |

1. Source : <https://www.baeldung.com/java-authentication-authorization-service> [↑](#footnote-ref-1)
2. Source : <https://www.tutorialspoint.com/javamail_api/javamail_api_quick_guide.htm> [↑](#footnote-ref-2)