|  |
| --- |
| **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

3.1.2.1 - Caractéristiques techniques 6

3.1.3 - Serveur de Batch 6

3.1.4 - Serveur de Fichiers 6

3.2 - Bases de données 7

3.3 - Web-services 7

3.3.1 - PAYMILL, API de paiement en ligne intégrant des bibliothèques Java, voici la procédure d’activation : 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 8

3.3.4 - JMM : API de géolocalisation « Java MaxMind » 8

3.3.5 - JAAS (Java Authentication and Authorization Service) : API de gestion de l’authentification. 8

3.3.6 - JavaMail API pour la gestion de courrier électronique. 9

4 - Procédure de déploiement 11

4.1 - Déploiement des Batches 11

4.1.1 - Artefacts 11

4.1.2 - Variables d'environnement 11

4.1.3 - Configuration 11

4.1.3.1 - Fichier xxx.yyy 12

4.1.3.2 - Fichier zzz.ttt 12

4.1.3.3 - Fichier ... 12

4.1.4 - Ressources 12

4.1.5 - Vérifications 12

4.2 - Déploiement de l'Application Web 13

4.2.1 - Artefacts 13

4.2.2 - Environnement de l’application web 13

4.2.2.1 - Variables d’environnement 13

4.2.3 - Répertoire de configuration applicatif 13

4.2.3.1 - Fichier xxx.yyy 13

4.2.4 - DataSources 13

4.2.5 - Ressources 14

4.2.6 - Vérifications 14

5 - Procédure de démarrage / arrêt 15

5.1 - Base de données 15

5.2 - Batches 15

5.3 - Application web 15

6 - Procédure de mise à jour 16

6.1 - Base de données 16

6.2 - Batches 16

6.3 - Application web 16

7 - Supervision/Monitoring 17

7.1 - Supervision de l’application web 17

8 - Procédure de sauvegarde et restauration 18

9 - Glossaire 19

# 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 fondé sur PostgreSQL. La version utilisée est PostgreSQL 12.5. Le serveur est hébergé́ sur un serveur linux Debian.

Les différentes tables et champs de la base de données ont été définis dans le cadre du projet et du document DDF. Toute évolution de cette architecture pourra faire l’objet d’un projet d’évolution de la BDD.

Les évolutions de version ne sont pas prises en compte.

### Serveur Web

Le serveur est hébergé́ chez le fournisseur de l’application. Il héberge, la base de données, le service web et le web service.

#### Caractéristiques techniques

Le serveur web et le web service sont hébergés sur un serveur. La communication entre eux est protégée par système de security header.

Ces services sont hébergés sur des serveurs Apache Tomcat.

### Serveur de Batch

Le serveur de batch recueille un batch permettant de commander automatiquement les produits dont le stock minimum n’est pas atteint. Il permet d’assurer à chaque magasin un niveau de stock suffisant.

### Serveur de Fichiers

L’ensemble des images (jpeg ou png) et des ressources (vidéos par exemple) utilisées par le client sont hébergés sur ce serveur. Par exemple les images des pizzas sont stockées sur ce serveur.

## Bases de données

La base de données doit être accessibles lors du chargement du web service. La base de données est intitulée pizzeria.

Paramètres de connexion à la BDD :

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

username=postgres

password=Toysrusse

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, voici la procédure d’activation :

1. « *Créez votre propre tunnel de commande ou page de paiement.*
2. *Intégrez notre pont JavaScript afin de récupérer les informations de paiement de manière sécurisée.*
3. *Intégrez PAYMILL à votre serveur soit en utilisant une de nos bibliothèques, soit en appelant notre API directement*. »[[1]](#footnote-1)

### 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

Exemple de requête :

**public class ExempleClientHttpAPI {**

**private static final String URL = "https://api.smsmode.com/http/1.6/";**

**private static final String PATH\_SEND\_SMS = "sendSMS.do";**

**private static final String PATH\_SEND\_SMS\_BATCH = "sendSMSBatch.do";**

**private static final String ERROR\_FILE = "The specified file does not exist";[[2]](#footnote-2)**

Les 4 variables initialisées au début de ma requête vont permettre de créer des méthodes : une méthode de réception de SMS, une méthode de création de SMS, une méthode batch d’envoi de SMS et une méthode convertToString.

### JMM : API de géolocalisation « Java MaxMind »

L’import de la dépendance suivante dans le POM de l’application permet d’avoir accès aux bibliothèques de l’API :

<dependency>

<groupId>com.maxmind.geoip</groupId>

<artifactId>geoip-api</artifactId>

<version>1.3.1</version>

</dependency>

Il est également nécessaire d’importer les bases de données de pays et villes, propres à l’API Maxmind :

public class MyConstants {

// Country Data.

public static final String DATABASE\_COUNTRY\_PATH = "F:/GeoLite2/GeoLite2-Country.mmdb";

// City Data.

public static final String DATABASE\_CITY\_PATH = "F:/GeoLite2/GeoLite2-City.mmdb";

………

Nous devons ensuite créer une classe comportant un Main afin de charger ces bases de données.

### 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*. »[[3]](#footnote-3)

### 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);

}

});[[4]](#footnote-4)

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
* **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

Voici les variables d'environnement reconnues par les batches de l’application XXX :

|  |  |  |
| --- | --- | --- |
| Nom | Obligatoire | Description |
| XXX\_HOME | non | Répertoire racine de l’installation de l’application |
|  |  |  |

Définissez les variables d’environnement nécessaires comme ceci...

...

### Configuration

Voici les différents fichiers de configuration :

* **xxx.yyy** : fichier de configuration des logs
* **zzz.ttt** : fichier de configuration de l'application...
* **aaa** : fichier de configuration de la ressources XXX
* ...

#### Fichier xxx.yyy

…

#### Fichier zzz.ttt

...

#### Fichier ...

### Ressources

...

### Vérifications

Afin de vérifier le bon déploiement des batches, faire ceci…

## Déploiement de l'Application Web

### Artefacts

...

### 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.paymill.com/fr/integration/web-api/> [↑](#footnote-ref-1)
2. Source : <https://www.smsmode.com/api-sms/?gclid=CjwKCAjw-e2EBhAhEiwAJI5jgyDX9n_jJPfnAWTmniulY1QTY0i_8gT3wQK9awi2BHk4sYfQQarvtRoCRHQQAvD_BwE#1518520975385-9f96526b-74cf> [↑](#footnote-ref-2)
3. Source : <https://www.baeldung.com/java-authentication-authorization-service> [↑](#footnote-ref-3)
4. Source : <https://www.tutorialspoint.com/javamail_api/javamail_api_quick_guide.htm> [↑](#footnote-ref-4)