Manual de configuración de Proyecto “bancard\_ms\_dispatcher”

(Entorno de desarrollo)

# Introducción

Este documento tiene como fin realizar la configuración en entorno local del proyecto Java Spring **bancard\_ms\_endpoints** tambiénconocido como **bancard\_ms\_dispatcher** específicamente en la entidad Bancard S.A. Existen datos proporcionados que son confidenciales. No divulgar, publicar o compartir éste documento.

# Entorno y recursos

Esta sección tiene como fin exponer los recursos que serán necesarios para poder configurar el proyecto Java Spring.

## Recursos de Software

|  |  |  |
| --- | --- | --- |
| **RECURSO** | **VERSIÓN** | **COMPROBACIÓN** |
| Sistema Operativo Ubuntu | 16.04 LTS | $ lsb\_release -a |
| Servidor Web Tomcat | 8.0 | $ sudo find / -name version.sh  $ cd /la/ruta/del/bin $ ./versión.sh |
| Maven Build Manager | 3.2.x o superior1 | $ mvn -v |
| Motor de Base de Datos Postgresql | 9.1 o superior | $ psql --version |
| Open JDK | 1.7.x | $ java -version |
| **Obs.:** los valores para versión en color rojo indican la versión obligatoria para que pueda funcionar | | |

## Recursos de red

|  |
| --- |
| **RECURSO** |
| Acceso a internet |
| Acceso ssh al servidor linux 10.100.19.117 |

## Otros recursos

|  |
| --- |
| **RECURSO** |
| Backup de base de datos de desarrollo en Postgresql (se explicarán los pasos más adelante) |
| JARs de microservicios |

## Variables

|  |  |
| --- | --- |
| **CLAVE** | **VALOR POSIBLE** |
| **JAVA\_HOME** | /usr/lib/jvm/java-1.7.0-openjdk-amd64 |
| **TOMCAT** | /opt/tomcat |

# Pasos para configuración

1. (*Paso Previo*) Crear una carpeta de proyectos, ingresar en el mismo y desde terminal clonar los proyectos.

$ git clone [git@10.100.19.117:/home/git/bancard/microservicios/bancard\_ms\_api.git](about:blank)

$ git clone [git@10.100.19.117:/home/git/bancard/microservicios/bancard\_ms\_endpoints.git](about:blank)

1. (*Paso Previo*) Antes de configurar el dispatcher instalaremos el microservicio API.

$ cd bancard\_ms\_api

$ mvn clean install

Aquí se instala el jar de la API en el repositorio del maven, normalmente alojado en ~/.m2/repository

1. (*Paso Previo*) Restauraremos un backup de la base de datos postgres.
   1. Hacemos el backup en el servidor de microservicios de desarrollo.

$ ssh usuario@servidor

$ pg\_dump bancard\_dev -U postgres -h localhost -F c > bancard\_dev\_2017.backup

* 1. Copiamos el archive **bancard\_dev\_2017.backup** a nuestro equipo y restauramos.

$ pg\_restore -h localhost -p 5432 -U postgres -d "bancard\_dev" -v "bancard\_dev\_2017.backup"

1. Crear la estructura de carpetas con sus respectivos archivos y permisos.

$ mkdir -p /opt/bancard/envs/development/

$ mkdir -p /opt/bancard/apache-tomcat/

$ mkdir -p /opt/bancard/plugins/

$ touch /opt/bancard/envs/development/logback.xml

$ touch /opt/bancard/envs/application.yml

$ touch /opt/bancard/apache-tomcat/logs/dispatcher.log

$ chmod 777 /opt/bancard/ -R

$ **TOMCAT**/bin/tomcatctl

1. Cargar configuración en el archive logback.xml

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<property name="LOGGER\_HOME" value="/opt/bancard/apache-tomcat/logs/" />

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<layout class="ch.qos.logback.classic.PatternLayout">

<Pattern>%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{2} - %msg%n</Pattern>

</layout>

</appender>

<appender name="FILE-AUDIT" class="ch.qos.logback.core.rolling.RollingFileAppender">

<file>${LOGGER\_HOME}/audit-file-dispatcher.log</file>

<encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">

<Pattern>%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{2} - %msg%n</Pattern>

</encoder>

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<!-- rollover daily -->

<fileNamePattern>${LOGGER\_HOME}/archived-dispatcher/audit-file-dispatcher.%d{yyyy-MM-dd}.%i.log</fileNamePattern>

<timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">

<maxFileSize>200MB</maxFileSize>

</timeBasedFileNamingAndTriggeringPolicy>

</rollingPolicy>

</appender>

<appender name="FILE-DISPATCHER" class="ch.qos.logback.core.rolling.RollingFileAppender">

<file>${LOGGER\_HOME}/dispatcher.log</file>

<encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">

<Pattern>%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{0} - %msg%n</Patte

rn>

</encoder>

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<!-- rollover daily -->

<fileNamePattern>${LOGGER\_HOME}/archived-dispatcher/dispatcher.%d{yyyy-MM-dd}.%i.log</fileNamePattern>

<timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">

<maxFileSize>100MB</maxFileSize>

</timeBasedFileNamingAndTriggeringPolicy>

</rollingPolicy>

</appender>

<!-- Send logs to both console and file audit -->

<logger name="py.com.bancard.dispatcher" level="INFO" additivity="false">

<appender-ref ref="FILE-AUDIT" />

<appender-ref ref="STDOUT" />

</logger>

<logger name="org.springframework" level="ERROR" />

<root level="INFO">

<appender-ref ref="FILE-DISPATCHER" />

</root>

</configuration>

1. Cargar configuración en el archivo application.yml

spring.jackson.serialization.INDENT\_OUTPUT: true

server.context-path: /bancard

logging.file: /opt/bancard/apache-tomcat/logs/dispatcher.log

logging.config: /opt/bancard/envs/development/logback.xml

ms.default.timeout: 5000

ms.default.max\_timeout: 10

ms.default.number\_of\_fails: 2

ms.executor.pool.size: 200

ms.dynamic.jar.location: /opt/bancard/plugins

liquibase:

changeLog: "classpath:db/changelog/db.changelog-master.xml"

#deshabilitado para todos los perfiles por el momento

security:

basic:

enabled: false

---

spring:

profiles: default

datasource:

platform: postgres

driverClassName: org.postgresql.Driver

url: "jdbc:postgresql://localhost:5432/bancard\_dev"

username: postgres

password: 12345

jpa:

database: POSTGRESQL

show-sql: true

hibernate.ddl-auto: none

database-platform: org.hibernate.dialect.PostgreSQL9Dialect

---

spring:

profiles: test

datasource:

platform: h2

driverClassName: org.h2.Driver

url: "jdbc:h2:mem:test;MODE=PostgreSQL;DB\_CLOSE\_DELAY=-1;DB\_CLOSE\_ON\_EXIT=FALSE"

# schema: classpath:/schema-h2.sql

jpa:

database: H2

show-sql: true

hibernate.ddl-auto: create-drop

database-platform: H2

1. Cargar configuración en archivo tomcatctl

#!/bin/bash

# description: Tomcat Start Stop Restart

# processname: tomcat

# chkconfig: 234 20 80

JAVA\_OPTS="-Dspring.config.location=file:/opt/bancard/envs/application.yml"

export JAVA\_OPTS

JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64

export JAVA\_HOME

PATH=$JAVA\_HOME/bin:$PATH

CATALINA\_OPTS="-Xms512m -Xmx1024m -XX:MaxPermSize=256m"

export CATALINA\_OPTS

export PATH

CATALINA\_HOME=/opt/tomcat

case $1 in

start)

sh $CATALINA\_HOME/bin/startup.sh

;;

stop)

sh $CATALINA\_HOME/bin/shutdown.sh

;;

restart)

sh $CATALINA\_HOME/bin/shutdown.sh

sh $CATALINA\_HOME/bin/startup.sh

;;

esac

exit 0

1. En la carpeta /opt/bancard/plugins cargar todos los jar de los microservicios, debería quedar de la siguiente manera:

-rw-r--r-- 1 6176913 jun 23 2016 bancard-aseguradora-tajy-jar-with-dependencies.jar

-rw-r--r-- 1 13305690 mar 8 2016 bancard-asur-jar-with-dependencies.jar

-rw-r--r-- 1 535084 ago 29 15:33 bancard-donaciones-jar-with-dependencies.jar

-rw-rw-r-- 1 13341343 dic 9 11:09 bancard-familiar-prestamo-jar-with-dependencies.jar

-rw-r--r-- 1 576387 nov 2 13:40 bancard-fielco-jar-with-dependencies.jar

-rw-r--r-- 1 13302880 dic 13 11:27 bancard-fpj-prestamos-jar-with-dependencies.jar

-rw-r--r-- 1 2247090 nov 23 10:19 bancard-futuro-sepelios-jar-with-dependencies.jar

-rw-r--r-- 1 13307186 dic 29 12:40 bancard-ips-alquileres-jar-with-dependencies.jar

-rw-r--r-- 1 13318368 dic 8 13:50 bancard-ips-aportes-jar-with-dependencies.jar

-rw-r--r-- 1 22320942 ene 4 12:51 bancard-jposclient-facturadores-jar-with-dependencies.jar

-rw-r--r-- 1 6177227 jun 2 2016 bancard-la-agricola-seguros-jar-with-dependencies.jar

-rw-r--r-- 1 6166876 jul 5 2016 bancard-mapfre-jar-with-dependencies.jar

-rw-r--r-- 1 558053 jul 26 16:31 bancard-megaloi-jar-with-dependencies.jar

-rw-r--r-- 1 9055073 dic 15 2015 bancard-ms-ande-0.0.3-SNAPSHOT.jar

-rw-r--r-- 1 25119036 dic 15 2015 bancard-ms-familiar-0.0.4-SNAPSHOT.jar

-rw-r--r-- 1 6173032 sep 27 10:16 bancard-ms-lhcompras-jar-with-dependencies.jar

-rw-r--r-- 1 6177854 jun 2 2016 bancard-nobleza-seguros-jar-with-dependencies.jar

-rw-r--r-- 1 13305080 mar 8 2016 bancard-regional-seguros-jar-with-dependencies.jar

-rw-r--r-- 1 2718542 sep 30 16:06 bancard-rumbos-seguros-jar-with-dependencies.jar

-rw-r--r-- 1 555484 dic 29 08:53 bancard-santa-clara-jar-with-dependencies.jar

-rw-r--r-- 1 13329625 ene 3 12:48 bancard-tigo-gw-pagos-jar-with-dependencies.jar

-rw-r--r-- 1 63824 jun 29 2016 bancard-universidad-catolica-0.0.1-SNAPSHOT.jar

-rw-r--r-- 1 6767854 sep 2 12:32 bancard-zimple-jar-with-dependencies.jar

1. Reiniciamos el servicios del Tomcat

$ cd **TOMCAT**/bin/

$ sudo ./tomcatctl stop

$ sudo ./tomcatctl start

1. Debemos verificar si nuestro servicio del Tomcat está levantando las configuraciones, prestando especial atención al -Dspring.config.location=file:/opt/bancard/envs/application.yml

$ ps aux | grep application.yml

Observe: aquí debería mostrar un proceso corriendo.

1. A continuación es preciso ingresar a la carpeta **bancard\_ms\_endpoints**

$ cd /la/ruta/bancard\_ms\_endpoints

1. Luego ejecutar el comando de empaquetado y realizar el deploy en el servidor Tomcat

$ mvn clean package -DskipTests

$ sudo cp -v target/ bancard-ms-dispatcher-0.0.3-SNAPSHOT.war **TOMCAT**/webapps/bancard.war

Observe:

* La variable **TOMCAT** representa el path de la instalación del Tomcat, que puede varias conforme al método de instalación del mismo.
* El empaquetado al momento de hacer el deploy en el Tomcat debe llevar el nombre bancard.war

1. Controlar el comportamiento del despliegue en el log catalina.out

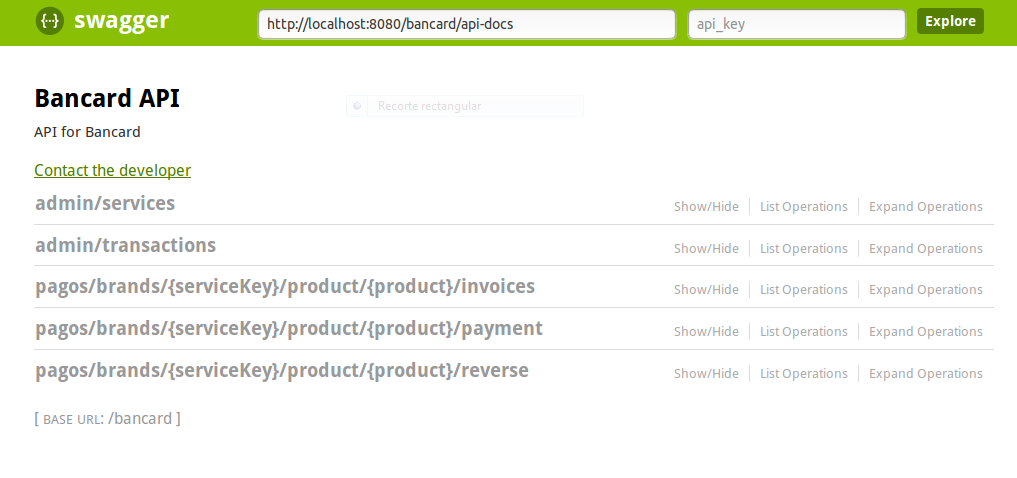
$ sudo tail -fn200 **TOMCAT**/logs/catalina.out

1. Verificar en su navegador web si el proyecto ha sido desplegado correctamente ingresando a la siguiente url:

[localhost:8080/bancard/swagger/index.html](http://localhost:8080/bancard/swagger/index.html)

Observe:

* El puerto del Tomcat puede varias dependiendo de su configuración local.
* Aquí debería desplegar una pantalla como se muestra en la imagen:



1. En el caso de que no se muestre la pantalla del Swagger, verificar los pasos anteriores revisando los logs y configuraciones.

## Paso para pruebas con test profile

$ mvn -Dspring.config.location=file:/opt/bancard/envs/application.yml -Dspring.profiles.active=test clean package

Observe: la configuración del perfil (profile) se encuentra en el application.yml

---

spring:

profiles: test

datasource:

platform: h2

driverClassName: org.h2.Driver

url: "jdbc:h2:mem:test;MODE=PostgreSQL;DB\_CLOSE\_DELAY=-1;DB\_CLOSE\_ON\_EXIT=FALSE"

# schema: classpath:/schema-h2.sql

jpa:

database: H2

show-sql: true

hibernate.ddl-auto: create-drop

database-platform: H2