Automático: Carga de archivos ISO - JobScheduler

**Contenido**

[Alcance 2](#_Toc39660536)

[Aplicativo 3](#_Toc39660537)

[Artefactos 3](#_Toc39660538)

[Aplicativo: 4](#_Toc39660539)

[Crear proyecto maven 5](#_Toc39660540)

[Crear clases 7](#_Toc39660541)

[BicCodeService 7](#_Toc39660542)

[BicCode 7](#_Toc39660543)

[Oauth2DTO 7](#_Toc39660544)

[Data 7](#_Toc39660545)

[BicCodeResponse 7](#_Toc39660546)

[BicCodeMapper: 7](#_Toc39660547)

[Compilar y subir el código al ambiente JobScheduler 9](#_Toc39660548)

[Generar la documetación del aplicativo 11](#_Toc39660549)

[Crear el Job para probar el aplicativo 15](#_Toc39660550)

[Crear el Job Chain 17](#_Toc39660551)

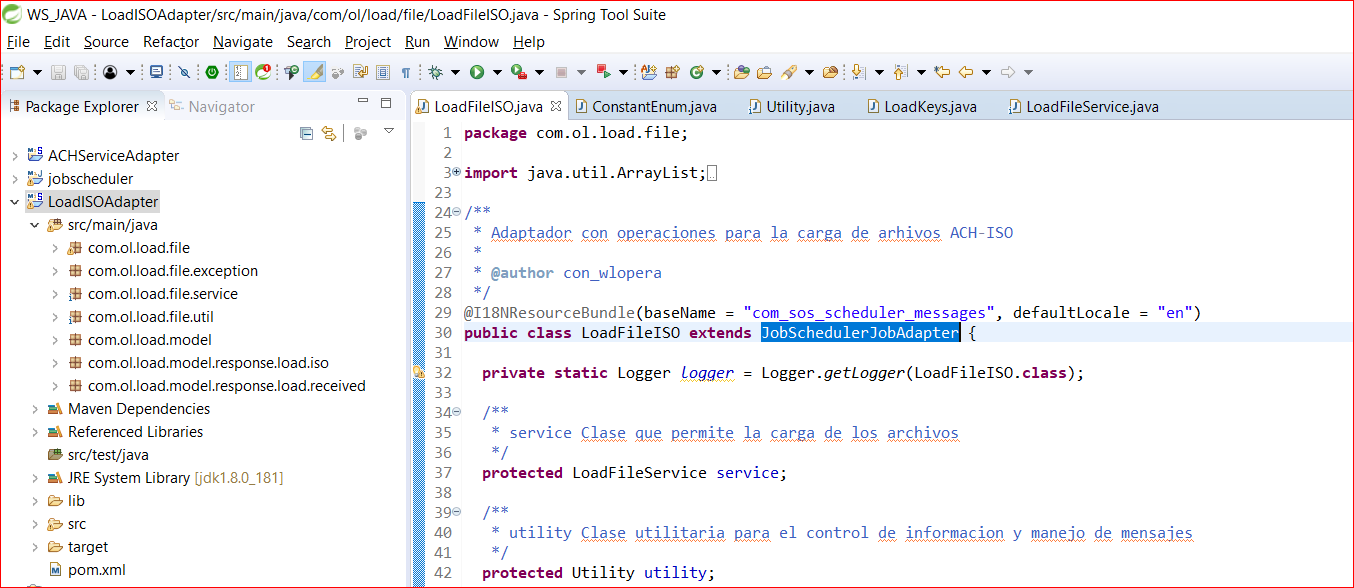
[Crear la orden (Order) 19](#_Toc39660552)

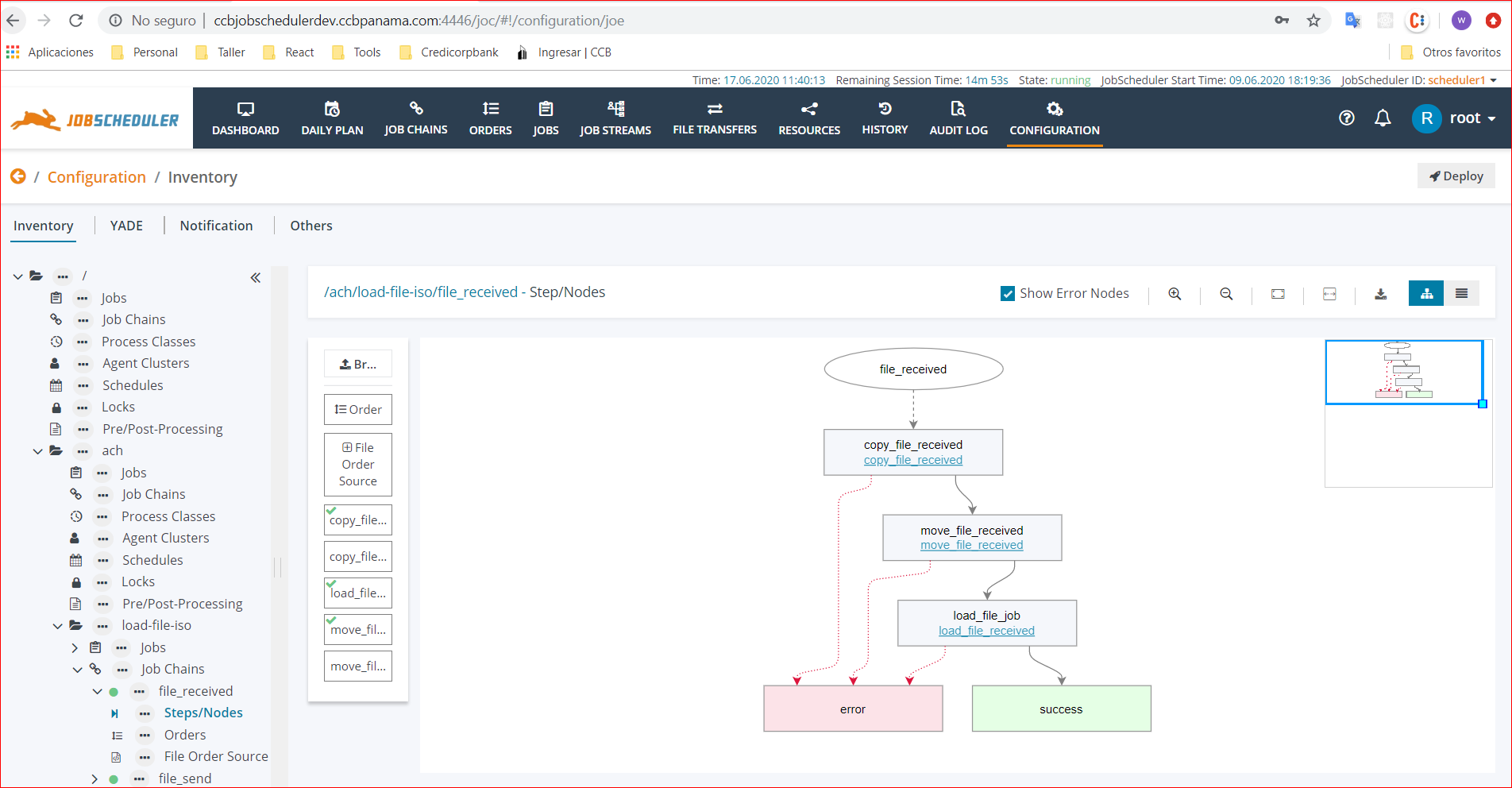
[Probar el Job 21](#_Toc39660553)

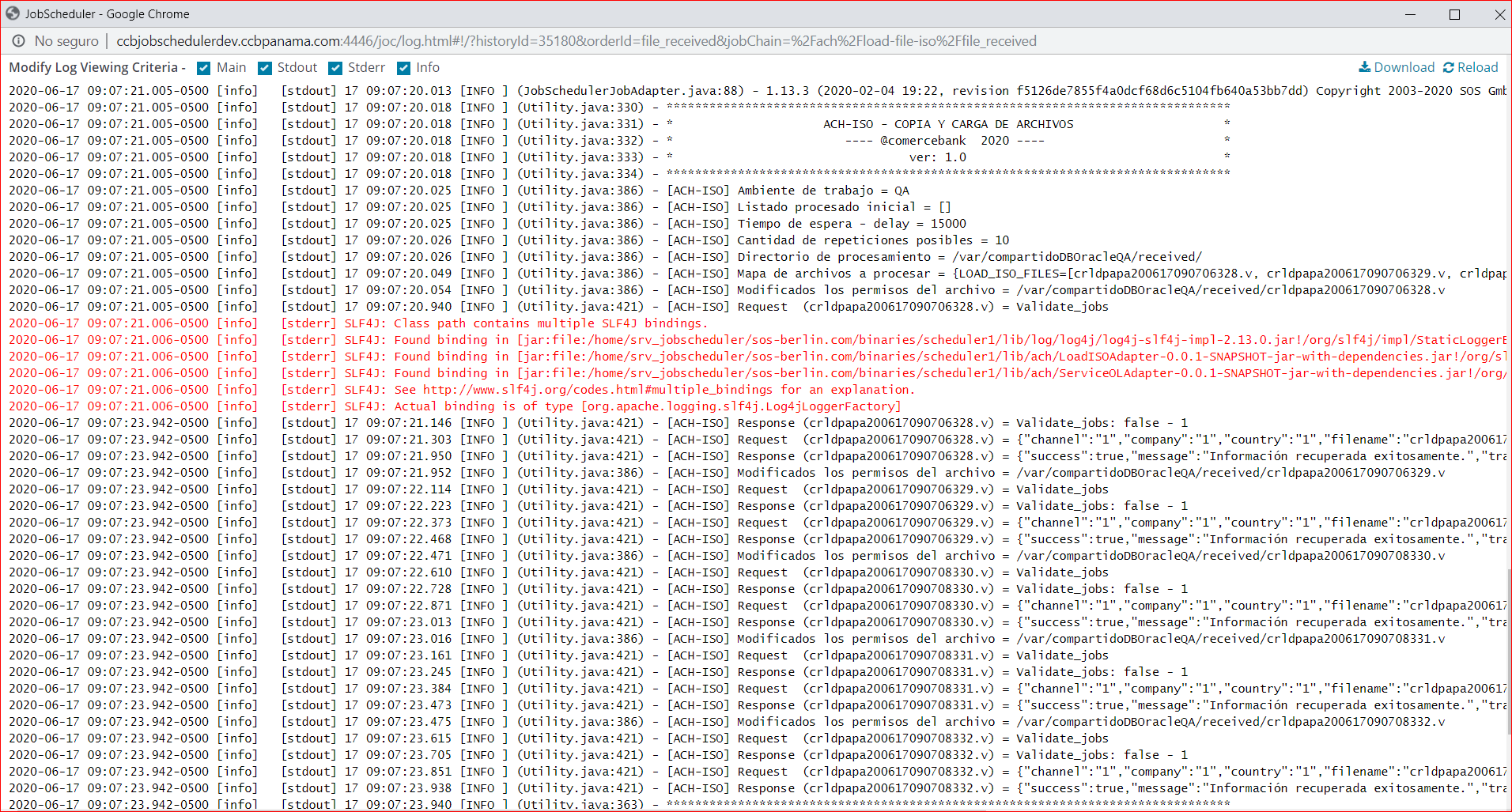
## Alcance

Este documento tiene por objetivo analizar cómo generar archivos de Java (.jar) para trabajos con el API de Java (engine-job-api) en un proyecto maven.

Para ello se crea un programa Java (.jar) que extiende de JobSchedulerJobAdapter, que permite la carga de archivos ACH-ISO. Uso de los servicios Open Legacy **LoadFileReceived** y **LoadIsoFiles** que genera la carga y procesamiento de archivos ISO enviadas y recibidas por Telered.

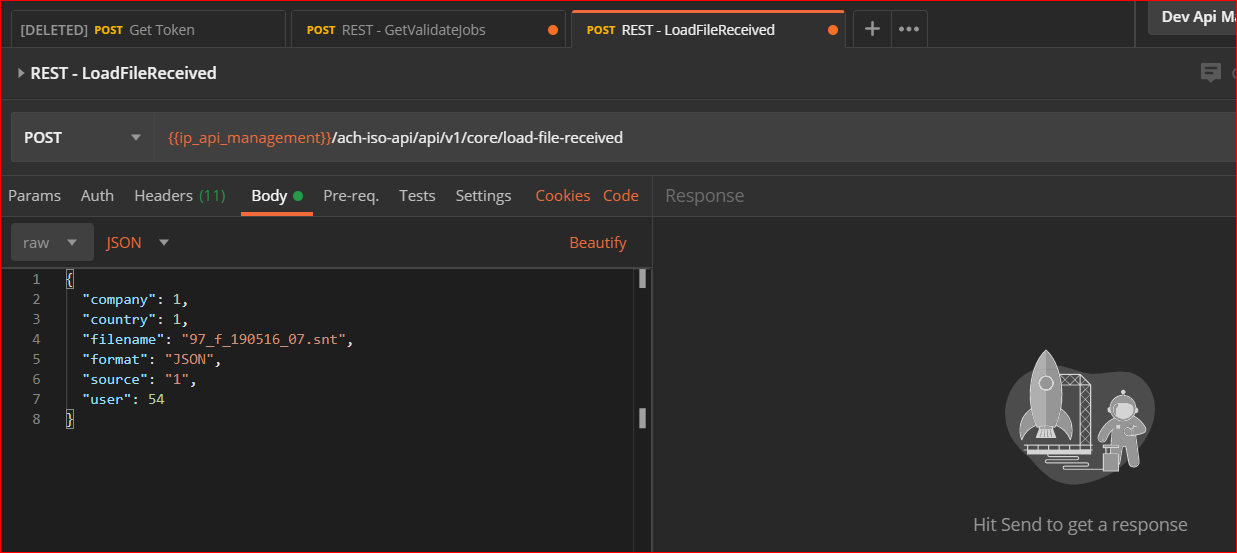






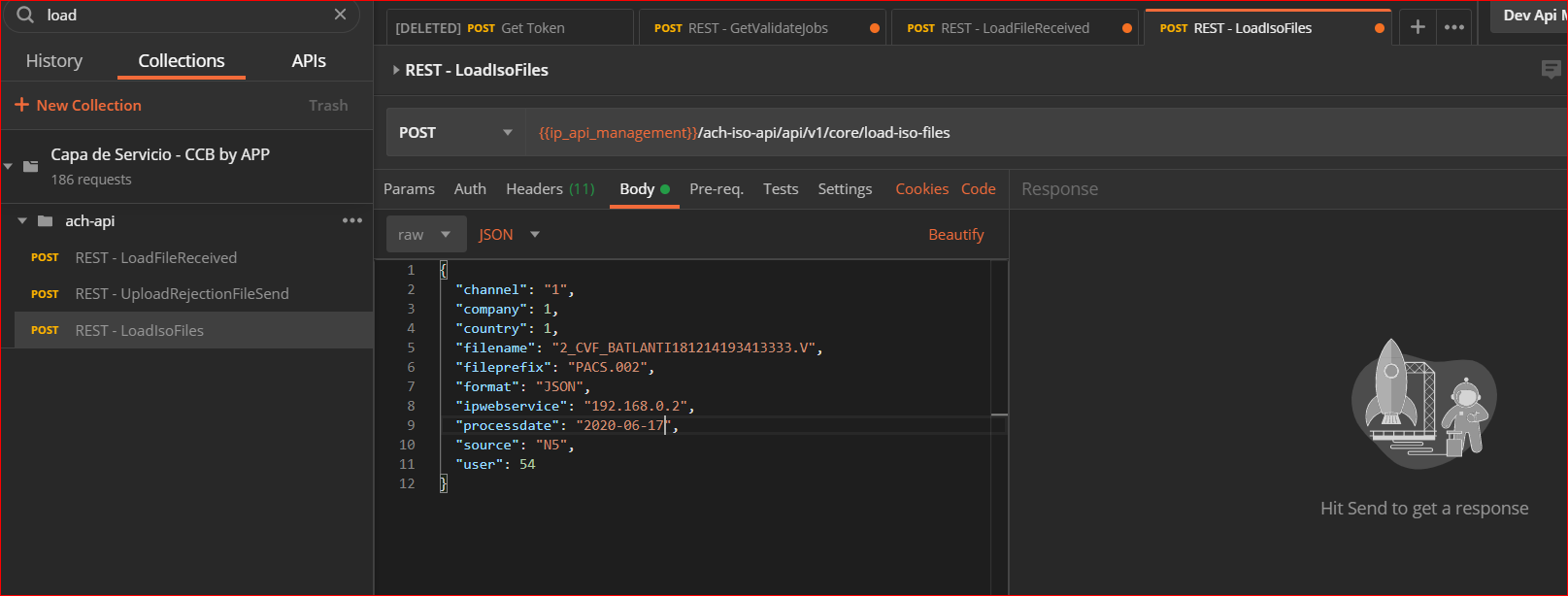
**Servicio loadFileReceived**

Carga de archivos ISO de crédito y débito. (008: Créditos y 003: Débitos)



**Servicio loadIsoFiles**

Cargar todos los archivos ISO que se reciban de Telered, por ejemplo ISO de confirmación PACS.002, tiene como parámetros el nombre del archivo y un prefijo que indica que tipo de archivo ISO es el que se quiere cargar. Por ejemplo PACS.002, PACS.004, CAMT.029, ACMT.022,PACS.008, CAMT.056, PACS.003,PACS.007, ACMT.023 esto nos servirá para buscar el directorio del archivo correspondiente en la tabla de parámetros y toda la lógica inmersa en cada tipo de archivo ISO, viene a tener la forma de un paquete orquestador que hará todo el proceso de carga y proceso



## Aplicativo

JobScheduler cuenta con JITL (Biblioteca de plantillas integradas de JobScheduler) que proporciona a los usuarios una forma de implementar rápidamente trabajos (Job) sin tener que comenzar a escribir sus propios trabajos desde cero. Estas plantillas permiten generar Jobs y da accesos a los parámetros requeridos para ese Job.

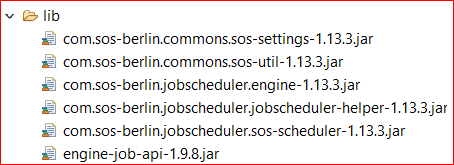
Todas las plantillas JITL se incluyen en la instalación de JobScheduler.  Están preinstalados en la carpeta *live/sos* del JobScheduler y, por lo tanto, pueden llamarse directamente desde el JOC.

**El objetivo de este documento es crear una guía que ayude a generar una clase Java que extienda de *JobSchedulerJobAdapter* y consuma Servicios Open Legacy para la carga de archivos ISO enviados y recibidos por Telered y ejecutar una orden que realiza el movimiento de archivos y su carga y procesamiento en Base de datos a través de tareas.**

## Artefactos

* com.sos-berlin.commons.sos-settings-1.13.3.jar
* com.sos-berlin.commons.sos-util-1.13.3.jar
* com.sos-berlin.jobscheduler.engine-1.13.3.jar
* com.sos-berlin.jobscheduler.jobscheduler-helper-1.13.3.jar
* com.sos-berlin.jobscheduler.sos-scheduler-1.13.3.jar
* engine-job-api-1.9.8.jar
* log4j-api-2.13.0.jar
* log4j-core-2.13.0.jar

Muchas de las librerías son tomadas de la instalación del JobSheduler:

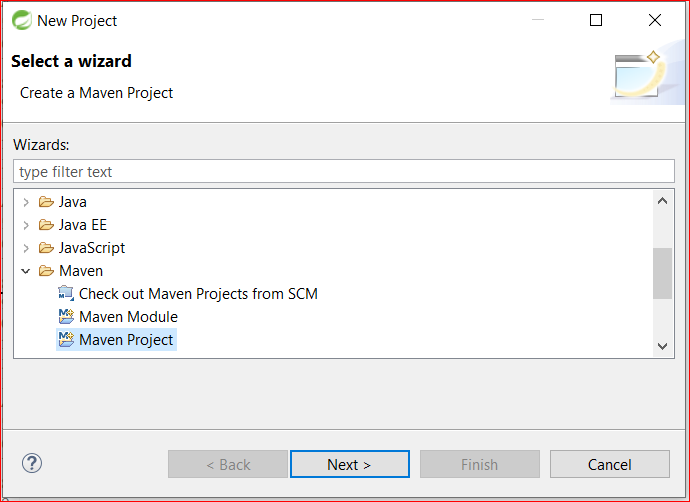


Las otras librerías agregar al pom.xml del maven

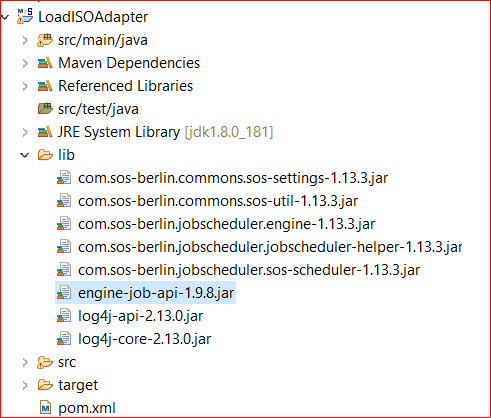
## Aplicativo:

Proyecto JobScheduler – Api Job – Java, para carga de archivos ISO a través de Servicios REST de Open Legacy. LoadFileReceived y LoadIsoFiles

Proyecto maven



* Agregar “*Source Folder*” => “src/main/java” : Código fuente
* Crear carpeta Lib con los .jar básicos requeridos o “agregarlos en el pom.xml”



* Agregar dependencias adicionales al pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.wlopera</groupId>

<artifactId>LoadISOAdapter</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LoadISOAdapter</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.26</version>

</dependency>

<dependency>

<groupId>org.codehaus.jackson</groupId>

<artifactId>jackson-mapper-asl</artifactId>

<version>1.9.13</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>3.0.2.RELEASE</version>

</dependency>

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.8.6</version>

</dependency>

<!-- https://mvnrepository.com/artifact/javax.xml/jaxp-api -->

<dependency>

<groupId>javax.xml</groupId>

<artifactId>jaxp-api</artifactId>

<version>1.4</version>

</dependency>

<!-- https://mvnrepository.com/artifact/javax.xml/jaxp-api -->

<dependency>

<groupId>javax.xml</groupId>

<artifactId>jaxp-api</artifactId>

<version>1.4</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- any other plugins -->

<plugin>

<artifactId>maven-assembly-plugin</artifactId>

<executions>

<execution>

<phase>package</phase>

<goals>

<goal>single</goal>

</goals>

</execution>

</executions>

<configuration>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

</configuration>

</plugin>

</plugins>

</build>

</project>

# Clases

LoadFileISO: Extiende del Adaptador de JobScheduler, clase principal donde están los métodos requeridos para interactuar con *JobScheduler* y los métodos requeridos con la lógica de consulta del *token* de conexión y métodos (servicios) que cargan y procesan los archivos ISO.

LoadFileService: Servicios expuestos.

* getOauthToken: Consulta el token de acceso a los servicios Open Legacy.
* hasFileUpload: Consulta si existe un proceso de carga actualmente en proceso.
* loadFileReceived: Carga archivos ISO de débito y crédito.
* loadIsoFile: Cargar de archivos ISO que se reciban de Telered.

Utility: Utilitario para el manejo de datos y mensajes.

ConstantEnum: Constantes Enum con valores constantes requeridos.

Loadkeys: Constantes relacionadas con la carga de archivos.

LoadMapper: Conversiones Objetos Java 🡸🡺 Json.

LoadFileException: Manejo de excepciones.

Oauth2DTO: Modelo Clase para manejo del token.

Post: Objetos de petición de las consultas a los servicios.

LoadResponseIso: Objeto de respuesta del servicio LoadIsoFiles

* Data Objeto de datos de la respuesta del servicio LoadIsoFiles
* Error Objeto de error de la respuesta del servicio LoadIsoFiles

LoadResponseReceived: Objeto de respuesta del servicio LoadFileReceived

* Data Objeto de datos de la respuesta del servicio LoadFileReceived
* Error Objeto de error de la respuesta del servicio LoadFileReceived

**Notas Importantes del código:**

spooler\_process(): Método llamado por el JobScheduler para procesar la orden. (Se puede, entre otras cosas, agregar lógica de control y validación de la orden, por ejemplo: crear o modificar variables de la orden)

spooler\_task(): Método con acceso a la tarea (Se puede, entre otras cosas, tener acceso a los parámetros de la orden : spooler\_task.order().params())

spooler\_task\_before(): Método que permite tener acceso al proceso antes de generarse la tarea

spooler\_task\_after(): Método que permite tener acceso al proceso después de generarse la tarea

Oauth2DTO getOauthToken(String environment): Método que permite obtener el token de conexión a los servicios dependiendo del ambiente Desarrollo, calidad o producción.

Utiliza las clases:

* RestTemplate: Cliente síncrono para realizar solicitudes HTTP, exponiendo una API de método de plantilla simple sobre bibliotecas de cliente HTTP subyacentes como JDK HttpURLConnection, Apache HttpComponents y otras.
* HttpHeaders: Una estructura de datos que representa encabezados de solicitud o respuesta HTTP, que asigna nombres de encabezado de cadena a una lista de valores de cadena.

RestTemplate restTemplate = new RestTemplate();

HttpHeaders headers = new HttpHeaders();

headers.setAccept(Collections.singletonList(MediaType.APPLICATION\_JSON));

headers.setAccept(Collections.singletonList(MediaType.APPLICATION\_JSON));

headers.add("Authorization", AUTHORIZATION);

HttpEntity<String> entity = new HttpEntity<String>("parameters", headers);

return restTemplate.exchange(serviceUrl + OAUTH2\_ENDPOINT, HttpMethod.POST, entity, Oauth2DTO.class).getBody();

loadFileReceived(String accessToken, String filename, String environment): Método que permite la carga de archivos ISO de débito y crédito.

Utiliza las clases:

* RestTemplate: Cliente síncrono para realizar solicitudes HTTP, exponiendo una API de método de plantilla simple sobre bibliotecas de cliente HTTP subyacentes como JDK HttpURLConnection, Apache HttpComponents y otras.
* HttpHeaders: Una estructura de datos que representa encabezados de solicitud o respuesta HTTP, que asigna nombres de encabezado de cadena a una lista de valores de cadena.
* Post: Clase con parámetros requeridos por la petición.

public LoadResponseReceived loadFileReceived(String accessToken, String filename, String enviroment)

throws LoadFileException {

try {

RestTemplate restTemplate = new RestTemplate();

HttpHeaders headers = new HttpHeaders();

headers.add(LoadKeys.TOKEN\_AUTHORIZATION, accessToken);

headers.setContentType(MediaType.APPLICATION\_JSON);

Post post = new Post("1", "1", filename, "1", "JSON", 54);

utility.printService(LoadKeys.KEY\_REQUEST, filename, LoadMapper.getPostJson(post));

HttpEntity<Post> request = new HttpEntity<>(post, headers);

String response = restTemplate

.exchange(getEndpointBase(enviroment) + ConstantEnum.ENDPOINT\_LOAD\_FILE\_RECEIVED.getValue(), HttpMethod.POST,

request, String.class).getBody();

return LoadMapper.getLoadResponseReceived(response);

} catch (Exception e) {

throw new LoadFileException(LoadKeys.ERROR\_CODE\_402, LoadKeys.MESSAGE\_CODE\_402, e.getMessage(),

e.getStackTrace());

}

}

LoadResponseIso(String accessToken, String filename, String environment): Método que permite la carga de archivos ISO enviados por Telered.

Utiliza las clases:

* RestTemplate: Cliente síncrono para realizar solicitudes HTTP, exponiendo una API de método de plantilla simple sobre bibliotecas de cliente HTTP subyacentes como JDK HttpURLConnection, Apache HttpComponents y otras.
* HttpHeaders: Una estructura de datos que representa encabezados de solicitud o respuesta HTTP, que asigna nombres de encabezado de cadena a una lista de valores de cadena.
* Post: Clase con parámetros requeridos por la petición.

public LoadResponseIso loadIsoFile(String accessToken, String filename, String enviroment) throws LoadFileException {

try {

RestTemplate restTemplate = new RestTemplate();

HttpHeaders headers = new HttpHeaders();

headers.add(LoadKeys.TOKEN\_AUTHORIZATION, accessToken);

headers.setContentType(MediaType.APPLICATION\_JSON);

DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");

Post post = new Post("1", "1", "1", filename, null, "N5", "JSON", "192.168.0.2", formatter.format(new Date()),

54);

utility.printService(LoadKeys.KEY\_REQUEST, filename, LoadMapper.getPostJson(post));

HttpEntity<Post> request = new HttpEntity<>(post, headers);

String response = restTemplate

.exchange(getEndpointBase(enviroment) + ConstantEnum.ENDPOINT\_LOAD\_ISO\_FILES.getValue(), HttpMethod.POST,

request, String.class).getBody();

return LoadMapper.getLoadResponseIso(response);

} catch (Exception e) {

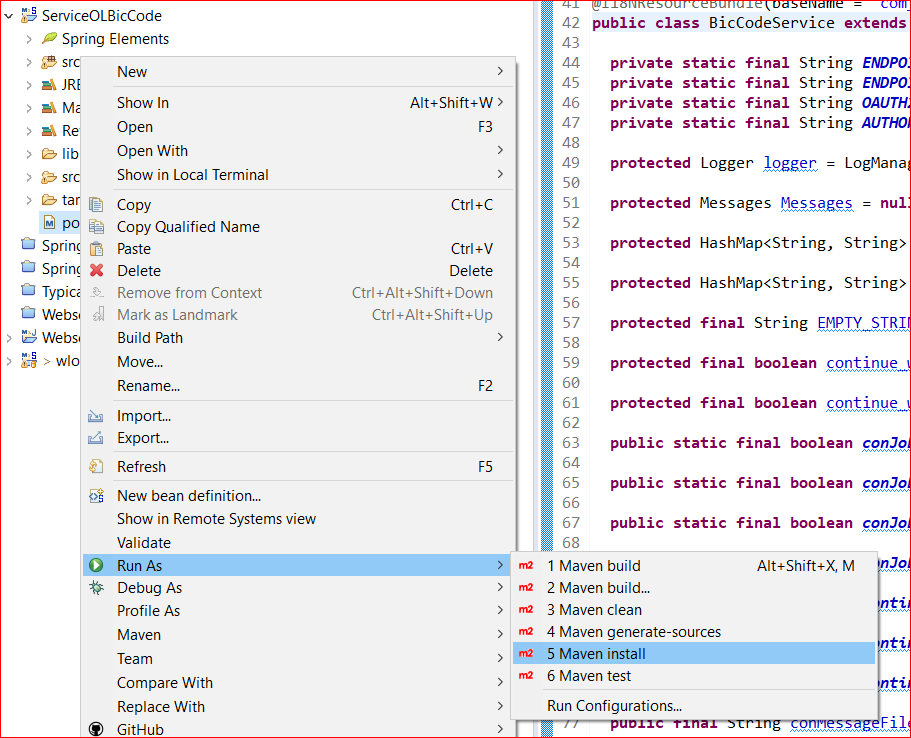
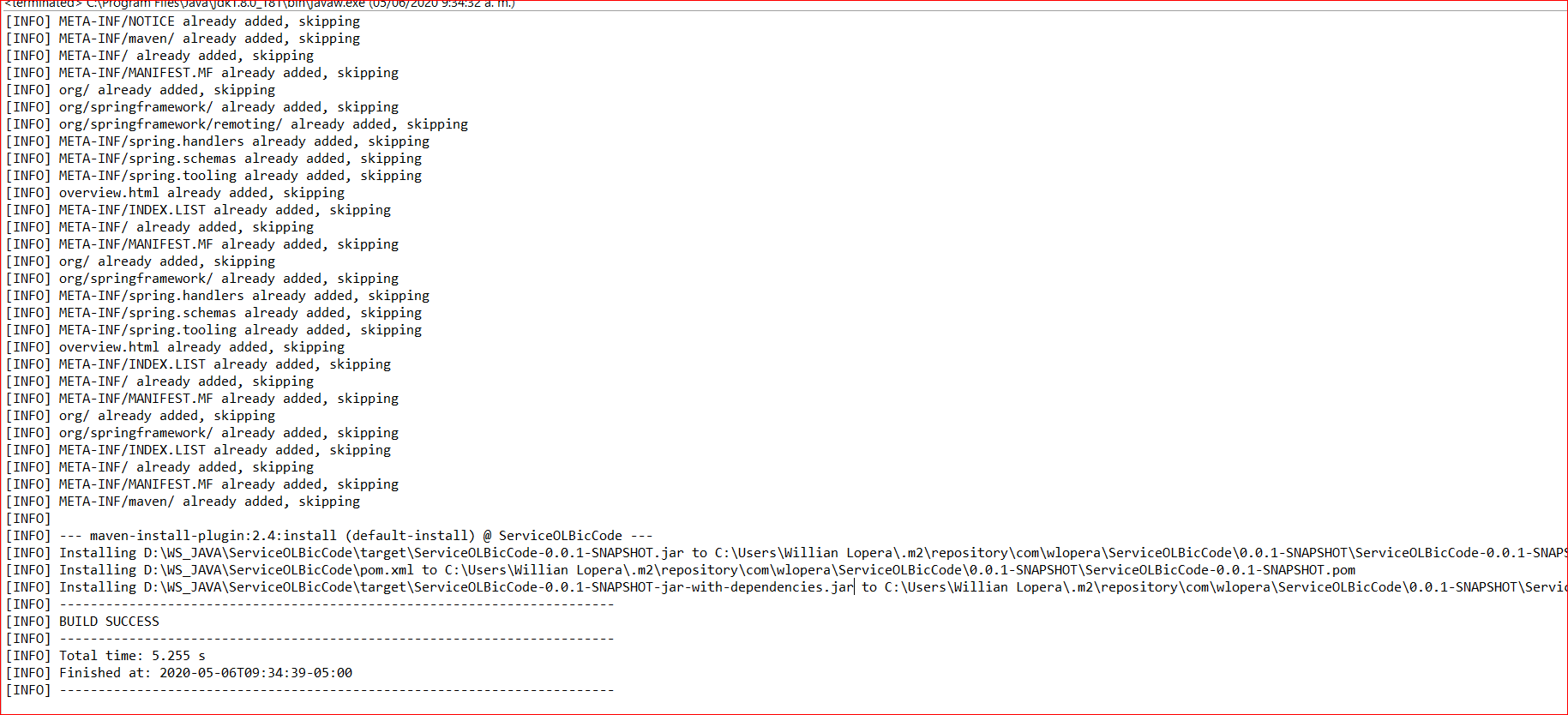
throw new LoadFileException(LoadKeys.MESSAGE\_CODE\_403, LoadKeys.MESSAGE\_CODE\_403, e.getMessage(),

e.getStackTrace());

}

}

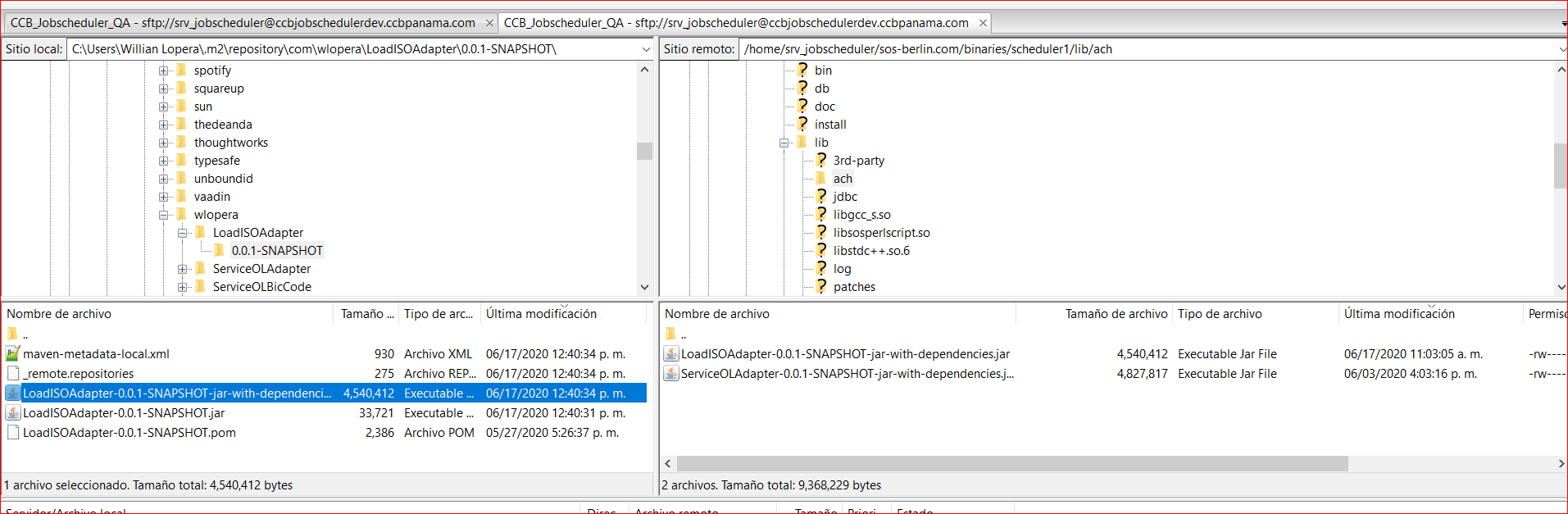
# Compilar y subir el código al ambiente JobScheduler

* Copiar el LoadISOAdapter-0.0.1-SNAPSHOT-jar-with-dependencies.jar con dependencias generado en la carpeta del usuario (repositorio maven) a la carpeta de librerías del JobScheduler

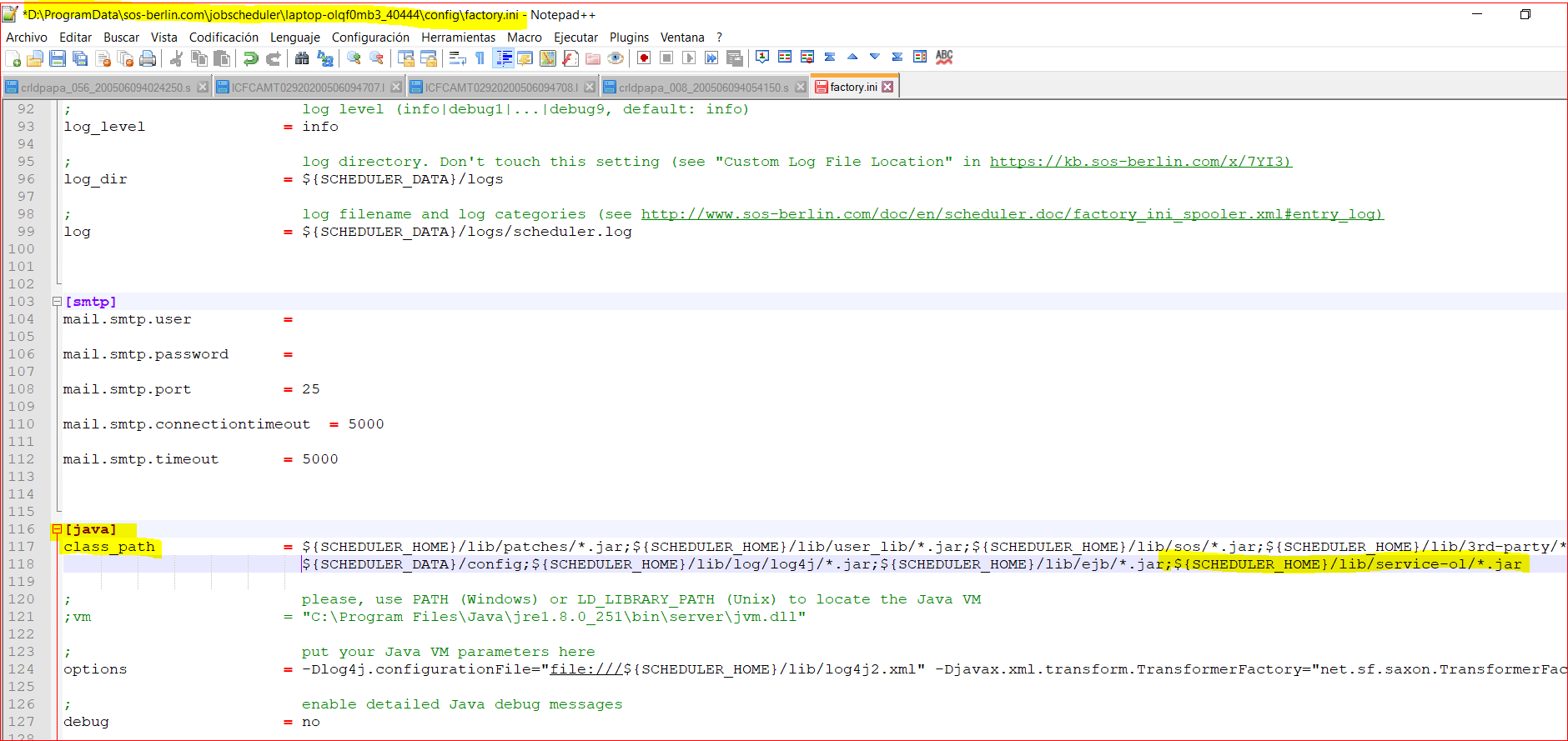
C:\Users\Willian Lopera\.m2\repository\com\wlopera\LoadISOAdapter\0.0.1-SNAPSHOT\=>

/home/srv\_jobscheduler/sos-berlin.com/binaries/scheduler1/lib/ach



* Agregar al classpath de Java en el archivo Factory.ini la ruta donde se copió el archivo .jar. Al subir el servicio del JobScheduler se tiene acceso al .jar copiado.

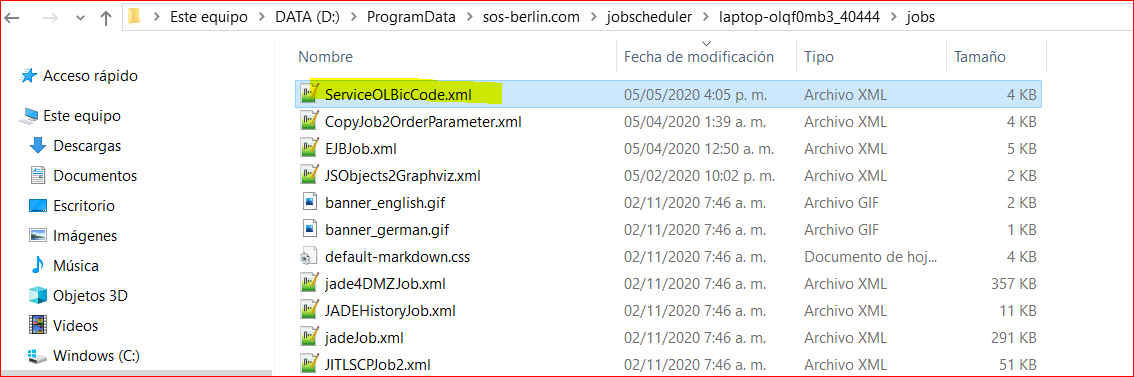
D:\ProgramData\sos-berlin.com\jobscheduler\laptop-olqf0mb3\_40444\config\ factory.ini



# Generar la documentación del aplicativo

Generar un archivo .xml con la documentación del proyecto o métodos expuestos en el aplicativo. Este archivo contiene diferentes Tags que permiten documentar las diversas partes que componen la documentación del aplicativo (Descripción, recursos, configuración, parámetros y documentación)

D:\ProgramData\sos-berlin.com\jobscheduler\laptop-olqf0mb3\_40444\jobs\ServiceOLBicCode.xml



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

<?xml-stylesheet type="text/xsl" href="scheduler\_job\_documentation\_v1.1.xsl"?>

<description xmlns="http://www.sos-berlin.com/schema/scheduler\_job\_documentation\_v1.1"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.sos-berlin.com/schema/scheduler\_job\_documentation\_v1.1 http://www.sos-berlin.com/schema/scheduler\_job\_documentation\_v1.1.xsd">

<job xmlns:xhtml="http://www.w3.org/1999/xhtml"

name="ServiceOLBicCode"

title="Servicio Open Legacy Código BIC"

order="no"

tasks="1">

<script language="java"

java\_class="com.wlopera.ol.BicCodeService"

resource="1">

</script>

</job>

<releases xmlns:xi="http://www.w3.org/2001/XInclude"

xmlns:xhtml="http://www.w3.org/1999/xhtml">

<release id="1.0" created="2020-05-05" modified="2020-05-05">

<title>version 1.0</title>

<author name="William Lopera" email="william.lopera@pranical.com"/>

</release>

</releases>

<resources xmlns:xhtml="http://www.w3.org/1999/xhtml">

<file os="all" type="java" file="com.wlopera.ol.BicCodeService" id="1">

<note language="de">

<div xmlns="http://www.w3.org/1999/xhtml">Administración código BIC </div>

</note>

<note language="en">

<div xmlns="http://www.w3.org/1999/xhtml">Administración código BIC</div>

</note>

</file>

</resources>

<configuration>

<params id="job\_parameter">

<param xmlns:xhtml="http://www.w3.org/1999/xhtml"

name="DATA"

default\_value=""

required="false">

<note language="de">

<div xmlns="http://www.w3.org/1999/xhtml">

Valor del campo DATA, solo como prueba

</div>

</note>

<note language="en">

<div xmlns="http://www.w3.org/1999/xhtml">

Valor del campo DATA, solo como prueba

</div>

</note>

</param>

<param xmlns:xhtml="http://www.w3.org/1999/xhtml"

name="FILE"

default\_value=""

required="true">

<note language="de">

<div xmlns="http://www.w3.org/1999/xhtml">

Valor del campo FILE, Documento ISO

</div>

</note>

<note language="en">

<div xmlns="http://www.w3.org/1999/xhtml">

Valor del campo FILE, Documento ISO

</div>

</note>

</param>

</params>

</configuration>

<documentation xmlns:xhtml="http://www.w3.org/1999/xhtml" language="de">

<div xmlns="http://www.w3.org/1999/xhtml">

Servicio Open Legay para administración de Códigos BIC

<br/>

<br/>

Estas funciones permiten:

<br/>

<br/>

<ul>

<li>Consultar los Códigos BIC actuales</li>

<li>Crear códigos BIC</li>

<li>Modificar códigos BIC</li>

<li>Eliminar códigos BIC </li>

</ul>

Uso de PL para Consulta, creación, modificación y borado de codigos BIC

</div>

</documentation>

<documentation xmlns:xhtml="http://www.w3.org/1999/xhtml" language="en">

<div xmlns="http://www.w3.org/1999/xhtml">

Servicio Open Legay para administración de Códigos BIC

<br/>

<br/>

Estas funciones permiten:

<br/>

<br/>

<ul>

<li>Consultar los Códigos BIC actuales</li>

<li>Crear códigos BIC</li>

<li>Modificar códigos BIC</li>

<li>Eliminar códigos BIC </li>

</ul>

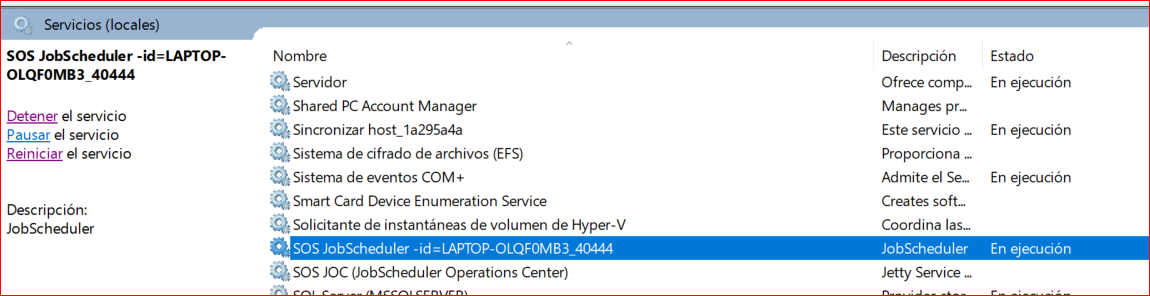
Uso de PL para Consulta, creación, modificación y borado de codigos BIC

</div>

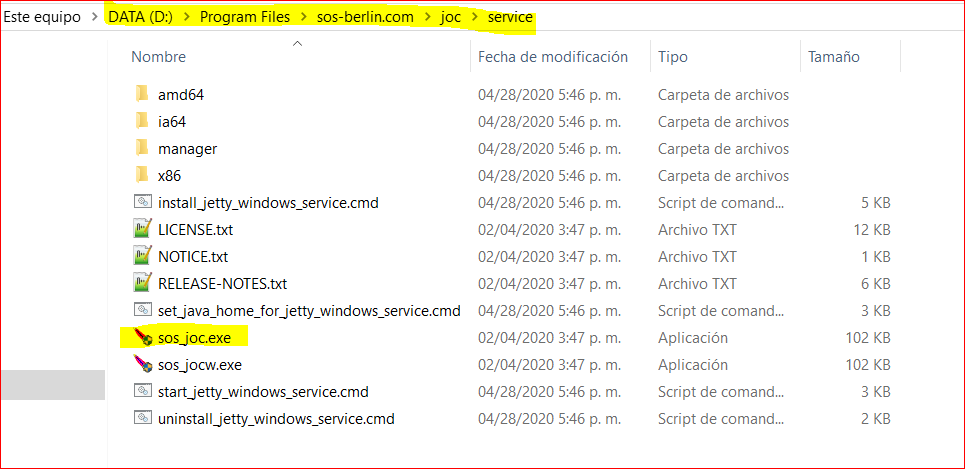
</documentation>

</description>

* Conexión al ambiente JobScheduler:
  + Levantar Servicio Jobscheduler



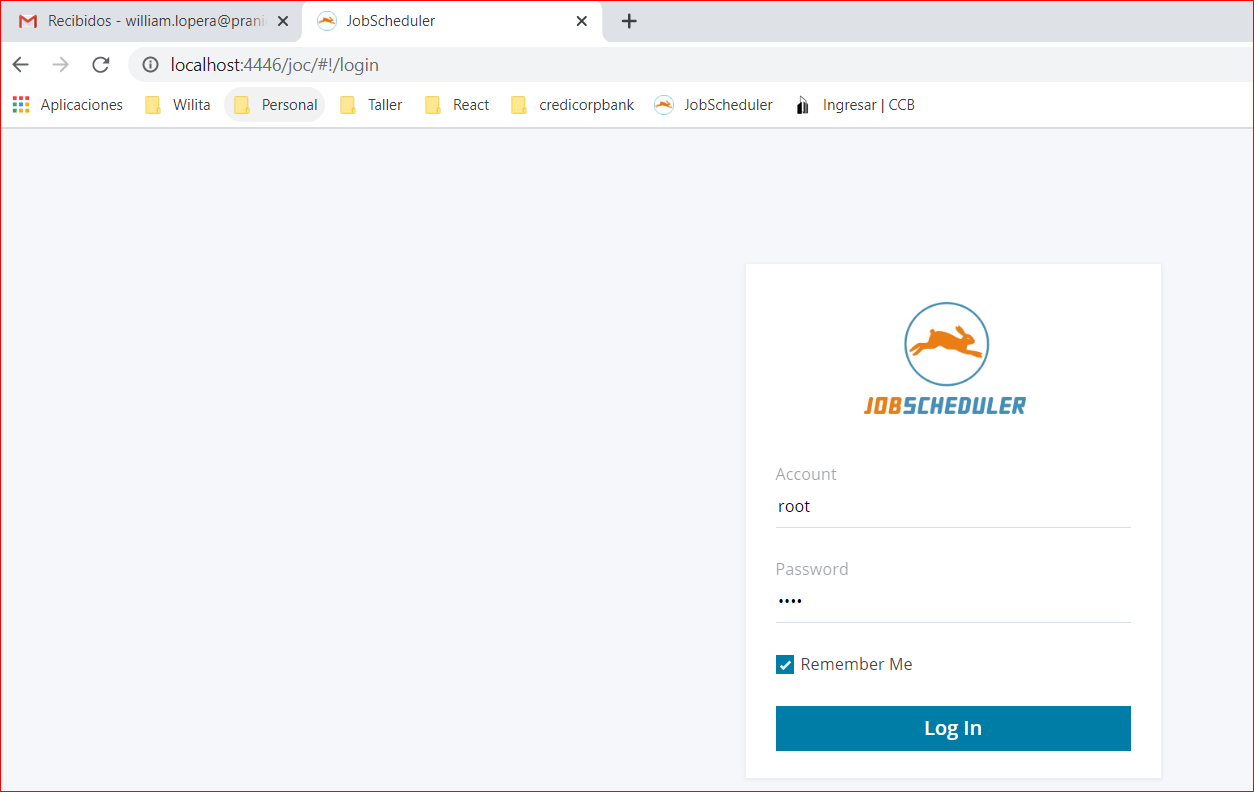
* Levantar el JOC - Jetty (en caso de error en el servicio, levantar por consola)



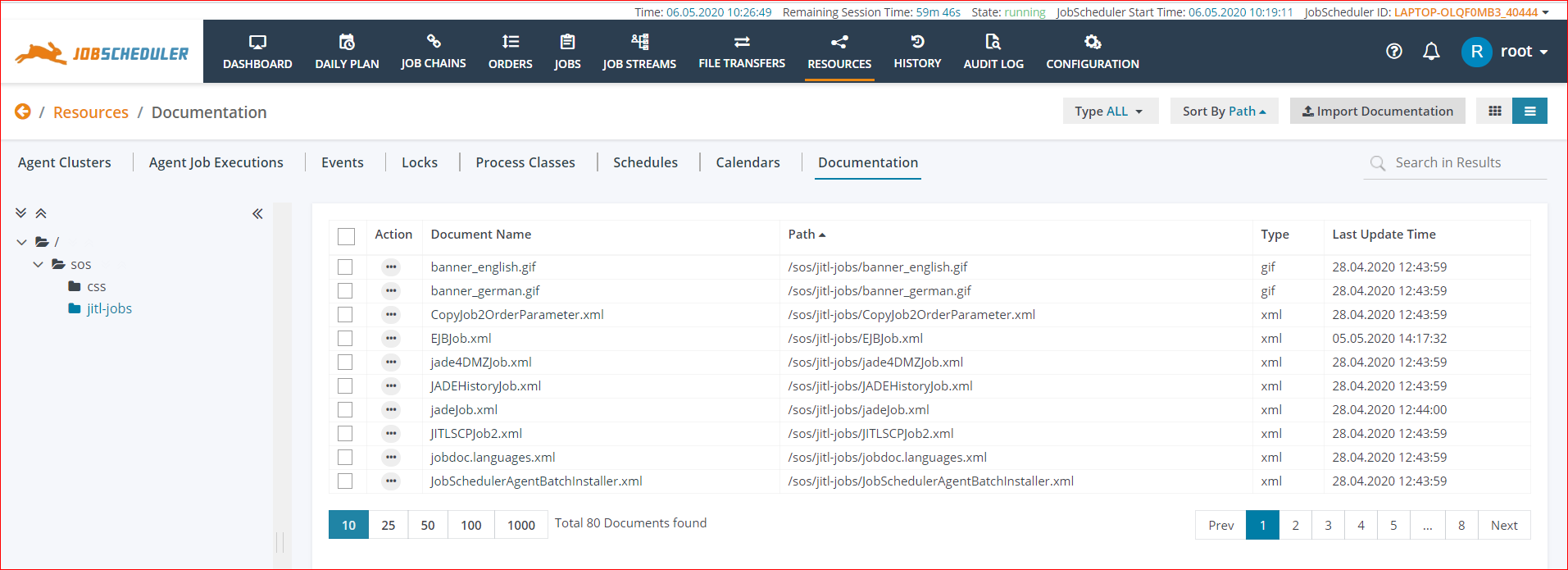
* Conectarse al ambiente JobScheduler

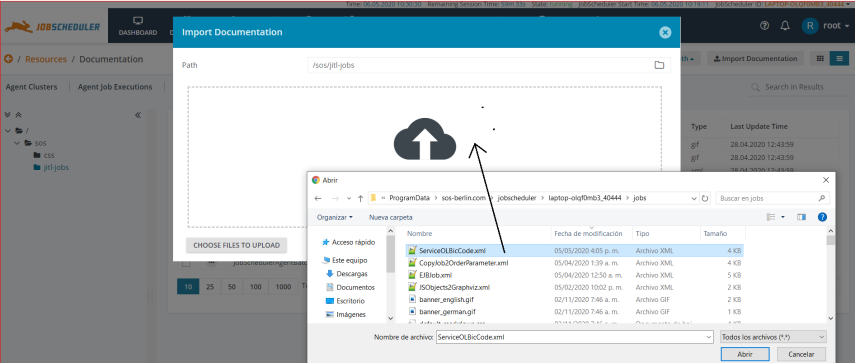
url: <http://localhost:4446/joc/#!/login>

user/password: root/root

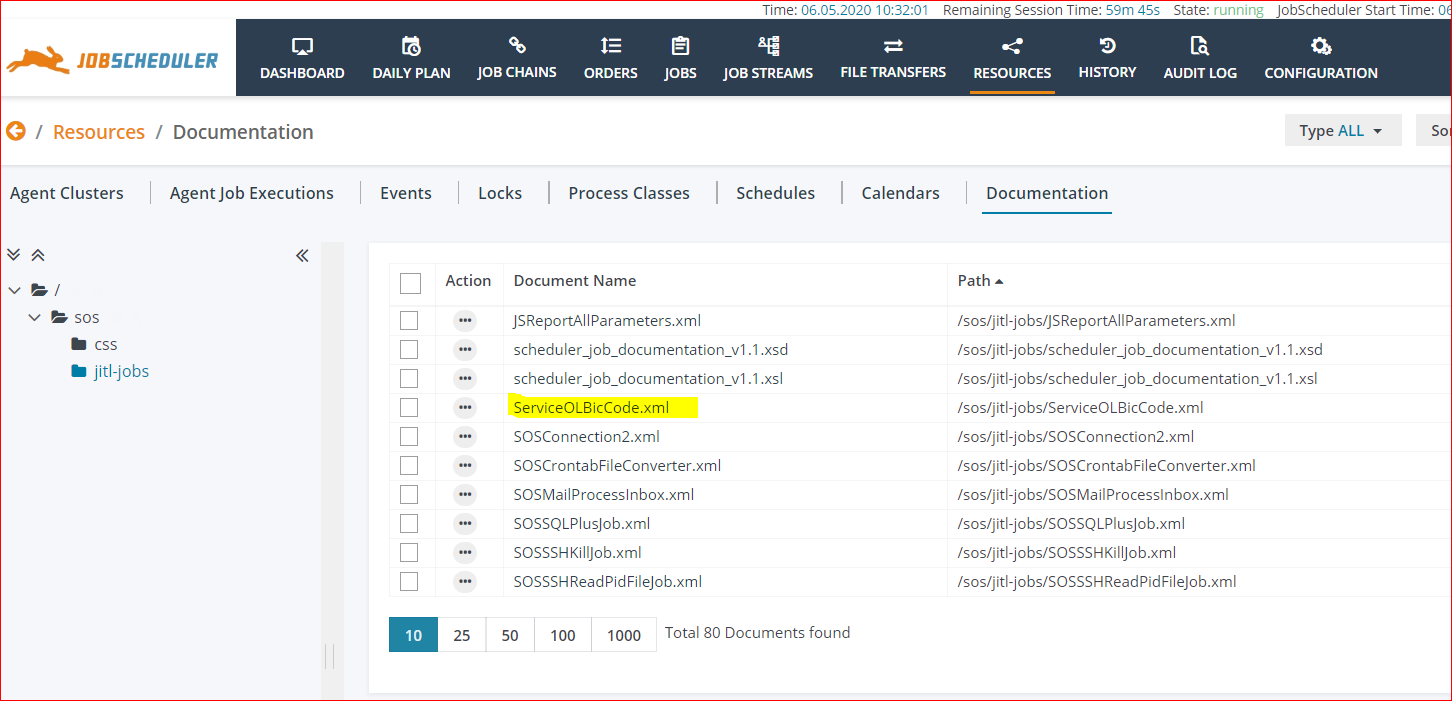


* Agregar el documento al JobScheduler: Importar el documento .xml creado en el Import Documentation del menú Resources/Documentacion

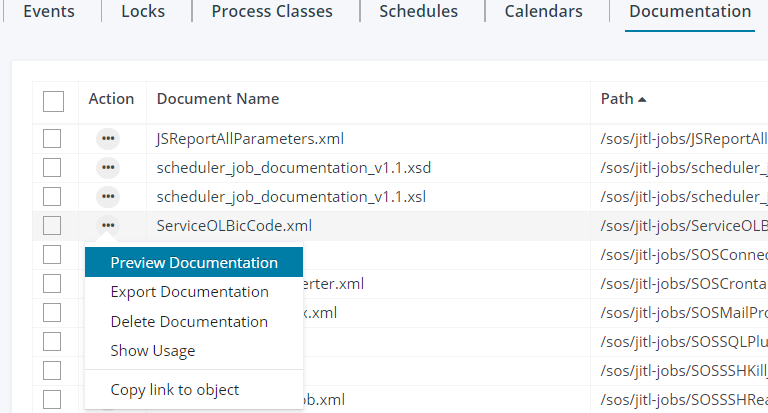


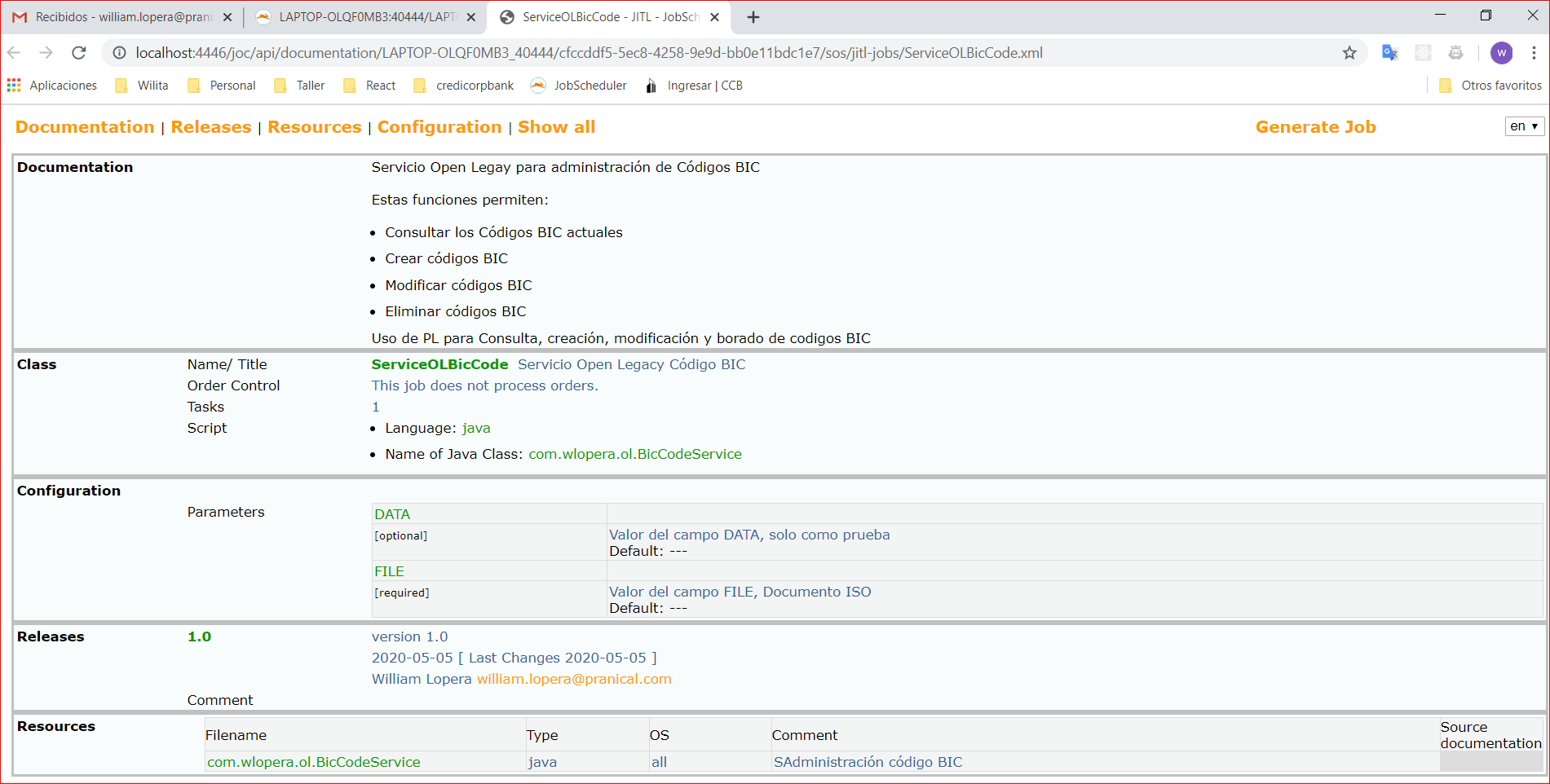


* Se agrega el archivo a la documentación



* Pre visualizar el documento

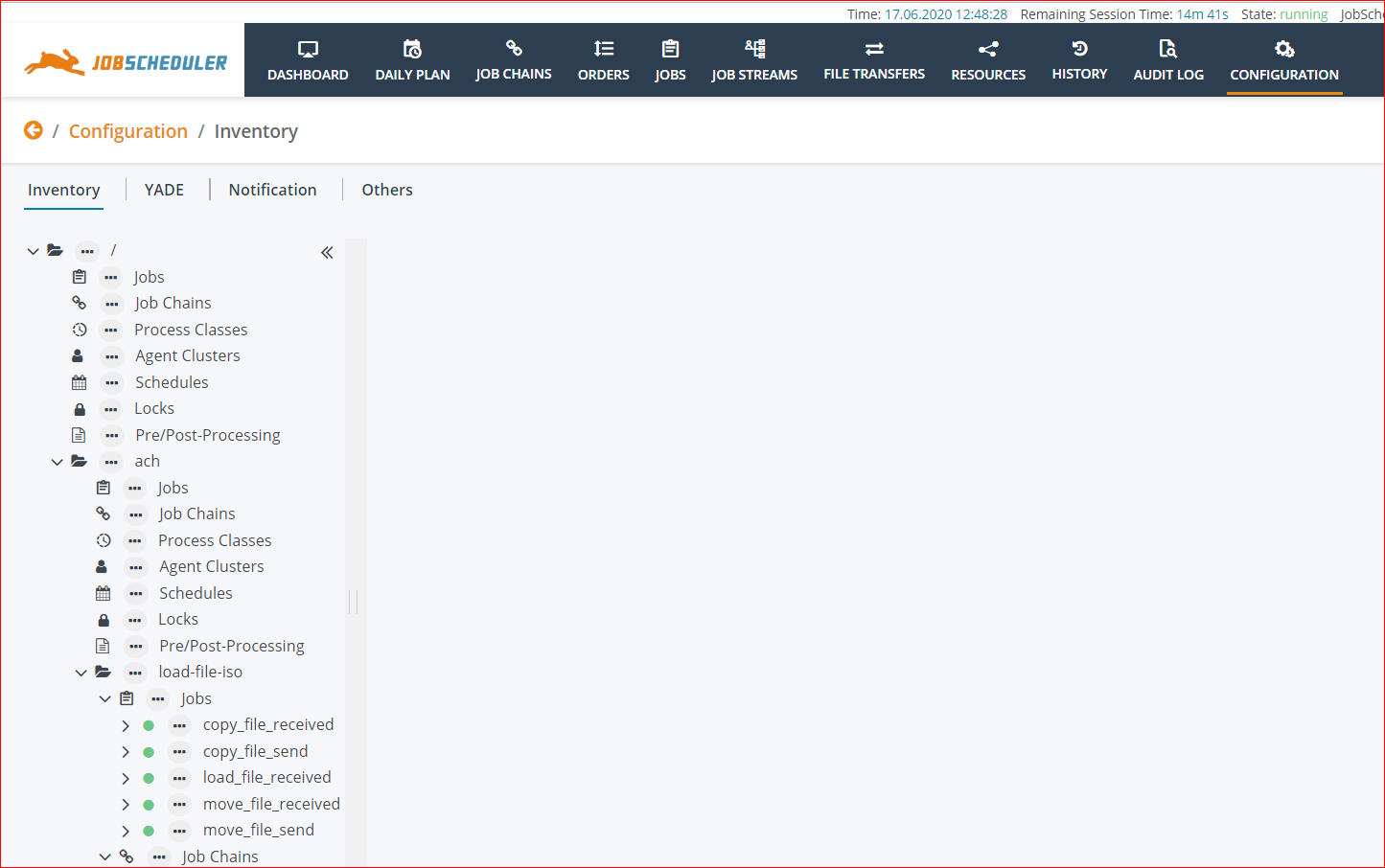




# Jobs, Job-chains y orders para probar el aplicativo

* Crear la carpeta de trabajo donde estarán nuestros Jobs, Jobs-chains y Orders

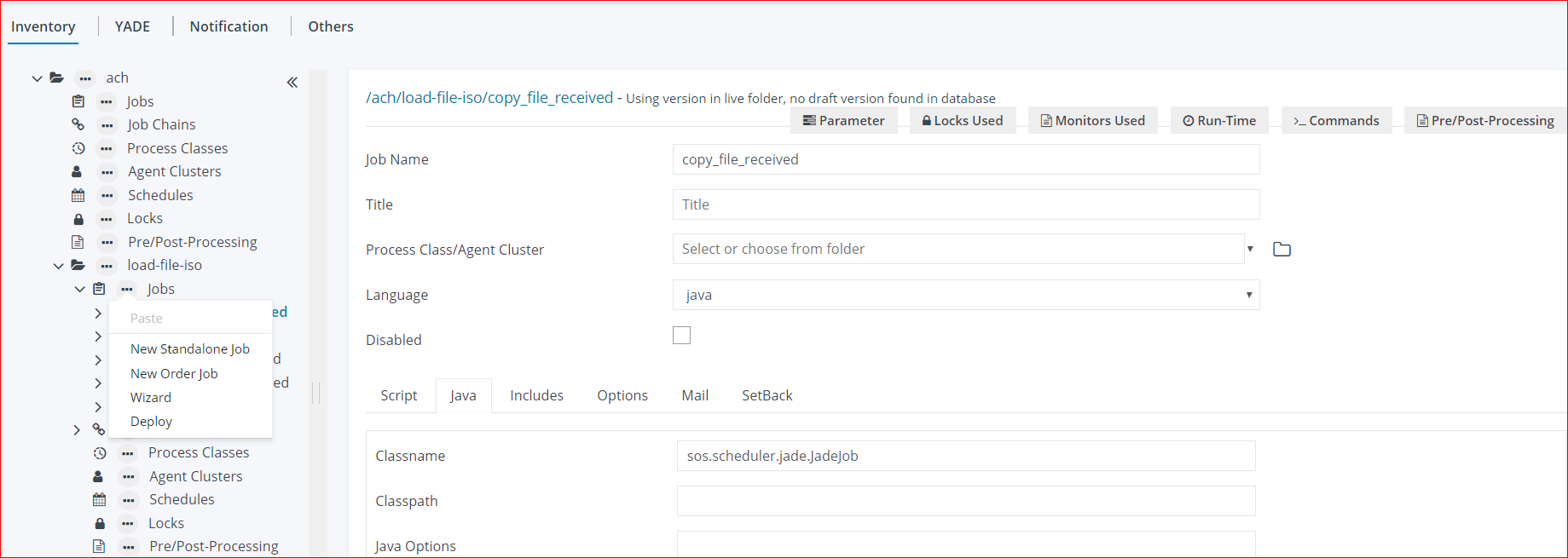
D:\ProgramData\sos-berlin.com\jobscheduler\laptop-olqf0mb3\_40444\config\live\ach\load-file-iso



# Crear Jobs

* Crear un nuevo Job: New Order job

En la opción de menú *Configuration.* Buscar la carpeta *ach*



* Agregar los datos del Job
  + Job Name: copy\_file\_received (Identificador: obligatorio)
  + Title: Copy File Received (Titulo del Job: opcional)
  + Language: java (Lenguaje que procesa el Job: obligatorio)
  + Classname: sos.scheduler.jade.Jadejob (Clase que copia archivos: obligatorio)

# Jobs Generados:

# copy\_file\_received.job.xml: Realiza la copia de archivos recibidos de Telered (source\_dir) a la carpeta de respaldo recibido\_ccb (target\_dir)

<?xml version="1.0" encoding="ISO-8859-1" ?>

<job order="yes" stop\_on\_error="false">

  <params>

    <param name="operation" value="copy"/>

  <!-- QA -->

    <param name="source\_dir" value="/var/CompartidoISOACH/salida/"/>

    <param name="target\_dir" value="/var/CompartidoISOACH/recibido\_ccb/[date:yyyy-MM-dd]"/>

  <!-- DEV

    <param name="source\_dir" value="D:\JobScheduler\data\CompartidoISOACH\salida\"/>

    <param name="target\_dir" value="D:\JobScheduler\data\CompartidoISOACH\recibido\_ccb\[date:yyyy-MM-dd]"/>

  -->

    <param name="passive\_mode" value="true"/>

  </params>

  <script language="java" java\_class="sos.scheduler.jade.JadeJob"/>

  <run\_time/>

</job>

# Nota: Realiza un respaldo de los archivos copiados en un directorio por fechas

# copy\_file\_send.job.xml: Realiza la copia de archivos a enviar a Telered (target\_dir) a la carpeta de respaldo enviado\_ccb (source\_dir)

<?xml version="1.0" encoding="ISO-8859-1" ?>

<job order="yes" stop\_on\_error="false">

  <params>

    <param name="operation" value="copy"/>

  <!-- QA -->

    <param name="source\_dir" value="/var/compartidoDBOracleQA/send/"/>

    <param name="target\_dir" value="/var/compartidoDBOracleQA/enviado\_ccb/[date:yyyy-MM-dd]"/>

  <!-- DEV

    <param name="source\_dir" value="D:\JobScheduler\data\compartidoDBOracleQA\send\"/>

    <param name="target\_dir" value="D:\JobScheduler\data\compartidoDBOracleQA\enviado\_ccb\[date:yyyy-MM-dd]"/>

  -->

    <param name="passive\_mode" value="true"/>

  </params>

  <script language="java" java\_class="sos.scheduler.jade.JadeJob"/>

  <run\_time/>

</job>

# Nota: Realiza un respaldo de los archivos copiados en un directorio por fechas

# move\_file\_received.job.xml: Realiza el movimiento de archivos recibidos de Telered (source\_dir) a la carpeta de procesamiento en base de datos (target\_dir). [ salida 🡺 received]

<?xml version="1.0" encoding="ISO-8859-1" ?>

<job order="yes" stop\_on\_error="false">

  <params>

    <param name="operation" value="move"/>

  <!-- QA -->

    <param name="source\_dir" value="/var/CompartidoISOACH/salida/"/>

    <param name="target\_dir" value="/var/compartidoDBOracleQA/received/"/>

  <!-- DEV

    <param name="source\_dir" value="D:\JobScheduler\data\CompartidoISOACH\salida\"/>

    <param name="target\_dir" value="D:\JobScheduler\data\compartidoDBOracleQA\received\"/>

  -->

    <param name="passive\_mode" value="true"/>

    <param name="force\_files" value="false"/>

  </params>

  <script language="java" java\_class="sos.scheduler.jade.JadeJob"/>

  <monitor name="process0" ordering="0">

    <script language="java:javascript"><![CDATA[

      function spooler\_process\_before(){

          var file\_path = spooler\_task.order.params.value('ftp\_result\_filenames');

          spooler\_task.order.params.set\_var( 'file\_path', file\_path );

        return true;

      }

      ]]></script>

  </monitor>

  <run\_time/>

</job>

# Nota: Se agrega a la orden la variable *file\_path* con los nombres de los archivos movidos para luego enviar a cargar los mismos.

# move\_file\_send.job.xml: Realiza el movimiento de archivos a enviar a Telered (target\_dir) desde la carpeta de base de datos (source\_dir). [send 🡺 entrada]

<?xml version="1.0" encoding="ISO-8859-1" ?>

<job order="yes" stop\_on\_error="false">

  <params>

    <param name="operation" value="move"/>

  <!-- QA -->

    <param name="source\_dir" value="/var/compartidoDBOracleQA/send/"/>

    <param name="target\_dir" value="/var/CompartidoISOACH/entrada/"/>

  <!-- DEV

    <param name="source\_dir" value="D:\JobScheduler\data\compartidoDBOracleQA\send\"/>

    <param name="target\_dir" value="D:\JobScheduler\data\CompartidoISOACH\entrada\"/>

  -->

    <param name="passive\_mode" value="true"/>

  </params>

  <script language="java" java\_class="sos.scheduler.jade.JadeJob"/>

  <monitor name="process0" ordering="0">

    <script language="java:javascript"><![CDATA[

      function spooler\_process\_before(){

          var file\_path = spooler\_task.order.params.value('ftp\_result\_filenames');

          spooler\_task.order.params.set\_var( 'file\_path', file\_path );

        return true;

      }

      ]]></script>

  </monitor>

  <run\_time/>

</job>

# Nota: Se agrega a la orden la variable *file\_path* con los nombres de los archivos movidos por si se requiere algún procesamiento de los mismos. (No es necesario)

# load\_file\_received.job.xml: Realiza el movimiento de archivos

<?xml version="1.0" encoding="ISO-8859-1" ?>

<job order="yes" stop\_on\_error="false">

<params>

  <!-- QA -->

    <param name="target\_dir" value="/var/compartidoDBOracleQA/received/"/>

  <!-- DEV

    <param name="target\_dir" value="D:\JobScheduler\data\compartidoDBOracleQA\received\"/>

  -->

  </params>

  <script language="java" java\_class="com.ol.load.file.LoadFileISO"/>

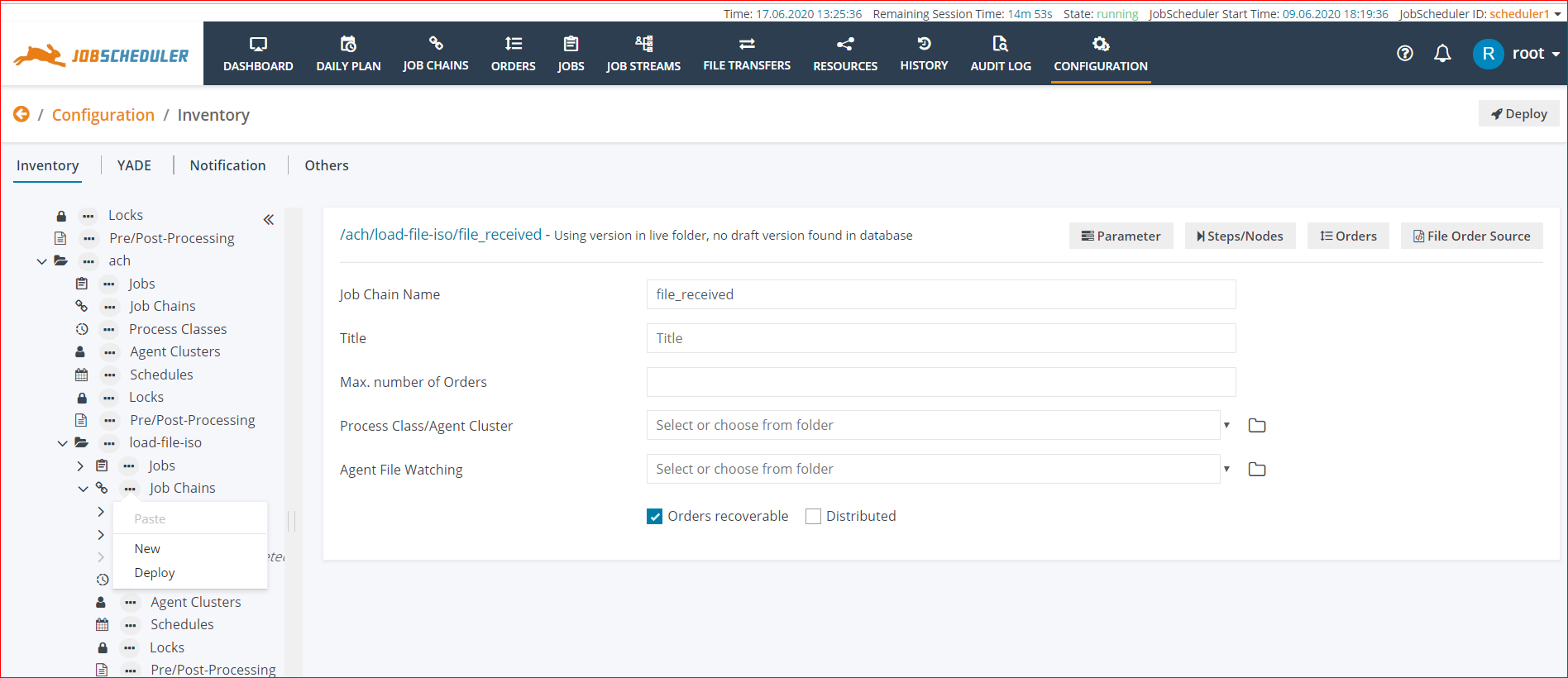
  <run\_time/>

</job>

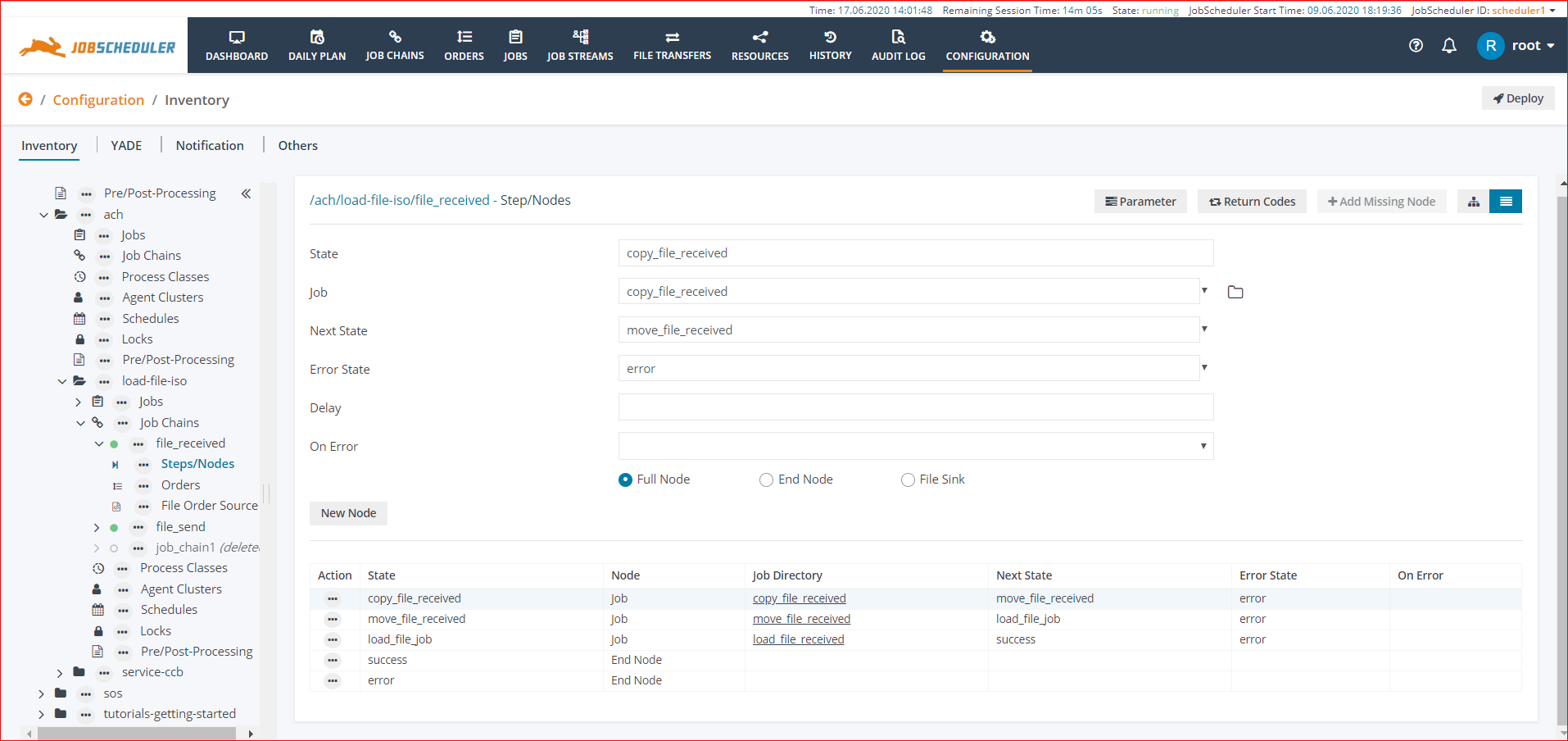
# Nota: Utiliza la clase creada: com.ol.lod.file.LoadFileISO

# Crear Job Chains

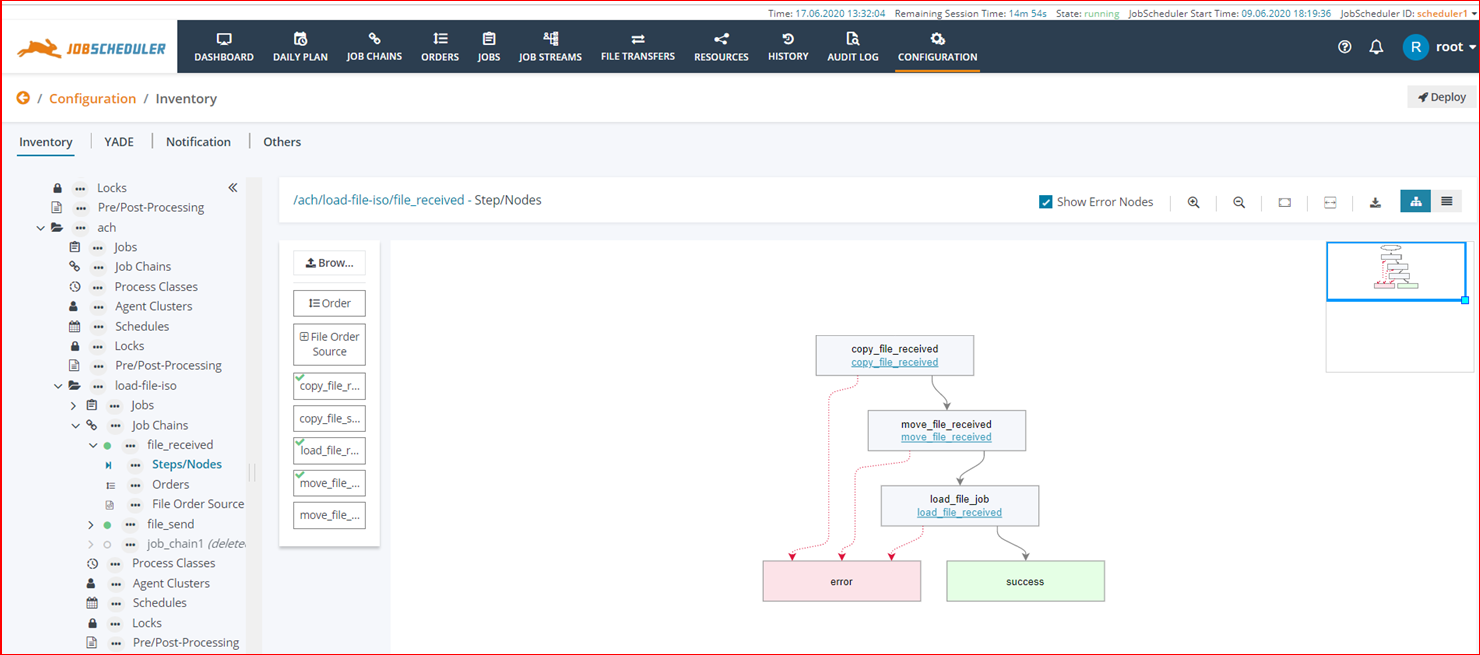
* Crear un nuevo Job Chain



* Agregar los datos del Job Chain
  + Job Chain Name : file\_ received (Identificador: oblilgatorio)
  + Title: file\_received (Titulo del Job Chain: opcional)
  + Otros parámetros opcionales
* Crear los pasos (nodos) del Job Chain
  + Agregando los nodos por formulario:
    - Nodo: copy\_file\_received
    - Nodo: move\_file\_received
    - Nodo: load\_file\_received
    - Nodo: Crear success
    - Nodo: Crear error
    - Nodo: fileReceived
  + Colocar la relación entre los nodos (éxito/fracaso)



* + O agregar los nodos de manera gráfica



# Job Chains Generados:

* file\_received.job\_chain.xml:
* Realiza el respaldo de los archivos
* Realiza el movimiento de los archivos a la carpeta de procesamiento
* Realiza la carga y procesamiento de los archivos

<?xml version="1.0" encoding="ISO-8859-1" ?>

<job\_chain>

  <job\_chain\_node job="copy\_file\_received" next\_state="move\_file\_received" error\_state="error" state="copy\_file\_received"/>

  <job\_chain\_node job="move\_file\_received" next\_state="load\_file\_job" error\_state="error" state="move\_file\_received"/>

  <job\_chain\_node job="load\_file\_received" next\_state="success" error\_state="error" state="load\_file\_job"/>

  <job\_chain\_node state="success"/>

  <job\_chain\_node state="error"/>

</job\_chain>

* file\_send.job\_chain.xml
* Realiza el respaldo de los archivos
* Realiza el movimiento de los archivos a la carpeta de envío a Telered

<?xml version="1.0" encoding="ISO-8859-1" ?

<job\_chain>

  <job\_chain\_node job="copy\_file\_send" next\_state="move\_file\_send" error\_state="error" state="copy\_file\_send"/>

  <job\_chain\_node job="move\_file\_send" next\_state="success" error\_state="error" state="move\_file\_send"/>

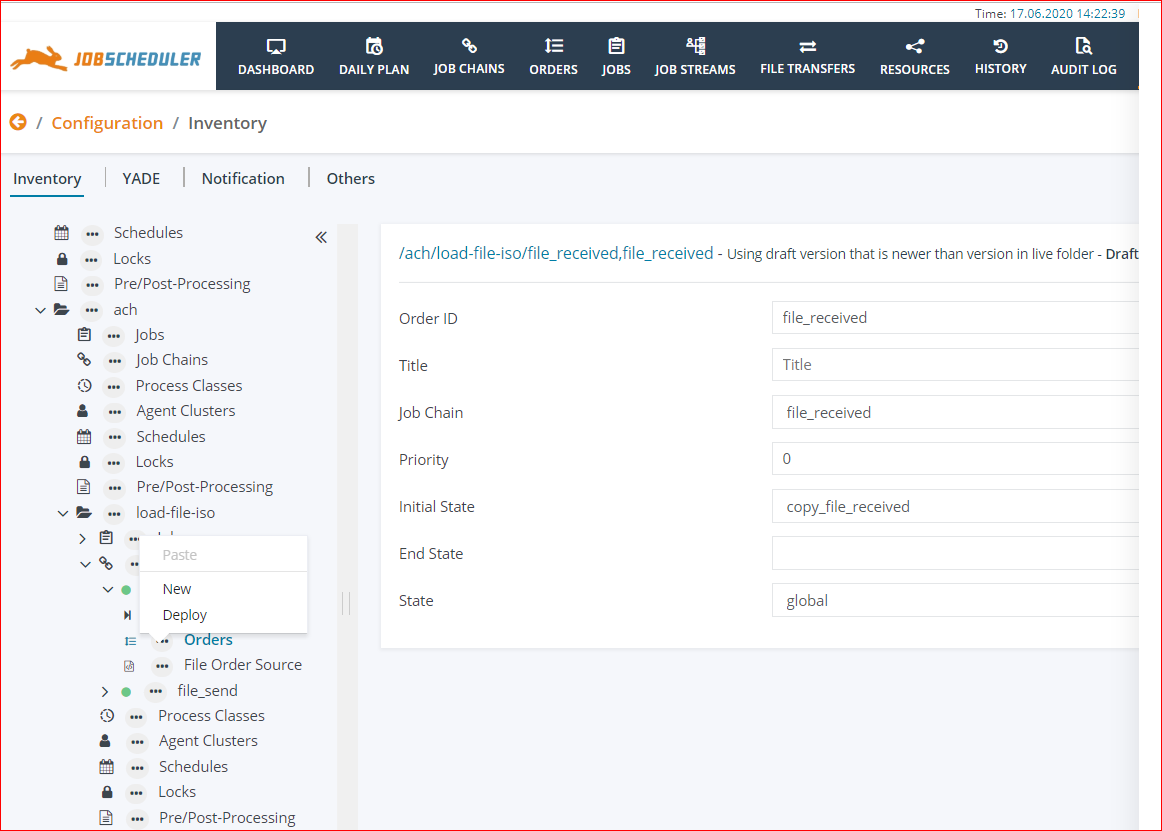
  <job\_chain\_node state="success"/>

  <job\_chain\_node state="error"/>

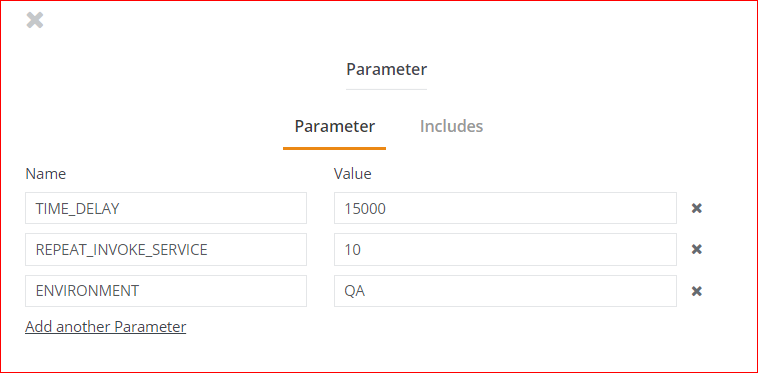
</job\_chain>

# Crear la orden (Order)

* Crear nueva Order.



* Agregar los datos
  + Order Id: filed\_received (identificador único)
  + Title: Titulo (opcional)
  + Job chain: file\_received (Identificador del Job Chain a ejecutarse en esta orden)
  + Inicial State: copy\_file\_received (Nodo inicla del Job\_chain)
* Agregar parámetros:
  + TIME\_DELAY : Tiempo de espera entre cargas de archivos
  + REPEAT\_INVOKE\_SERVICE: Número de intentos de procesamiento para cargar el archivo
  + ENVIRONMENT: Ambiente de Trabajo (DEV/QA/PROD)



# Orders Generadas:

* file\_received,file\_received.order.xml:

<?xml version="1.0" encoding="ISO-8859-1" ?>

<order>

  <params>

    <param name="TIME\_DELAY" value="15000"/>

    <param name="REPEAT\_INVOKE\_SERVICE" value="10"/>

    <param name="ENVIROMENT" value="QA"/>

  </params>

  <run\_time/>

</order>

* file\_send,file\_send.order.xml

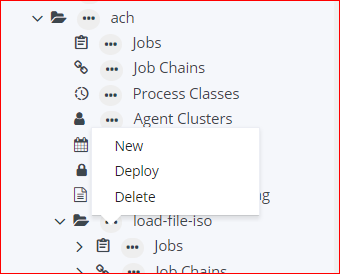
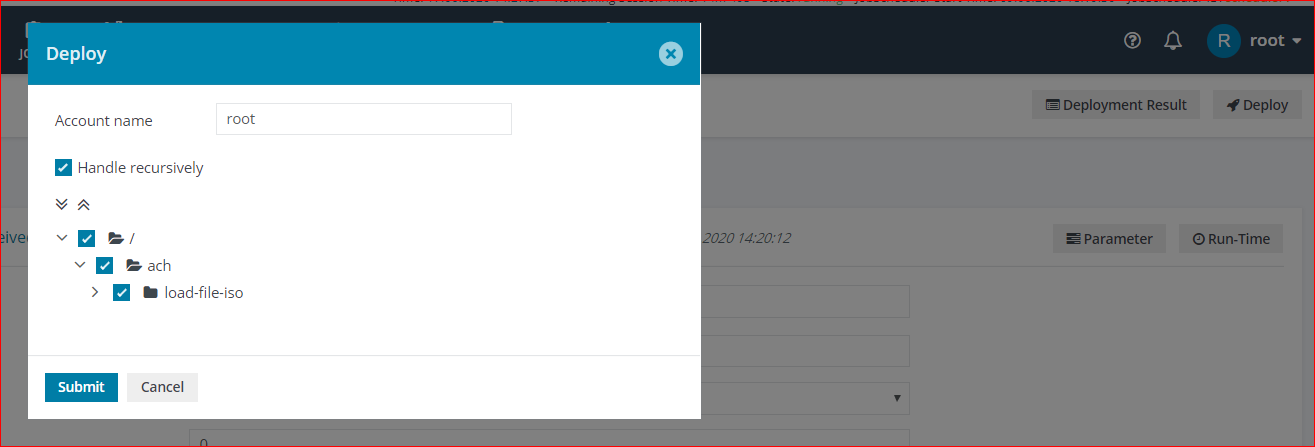
<?xml version="1.0" encoding="ISO-8859-1" ?>

<order>

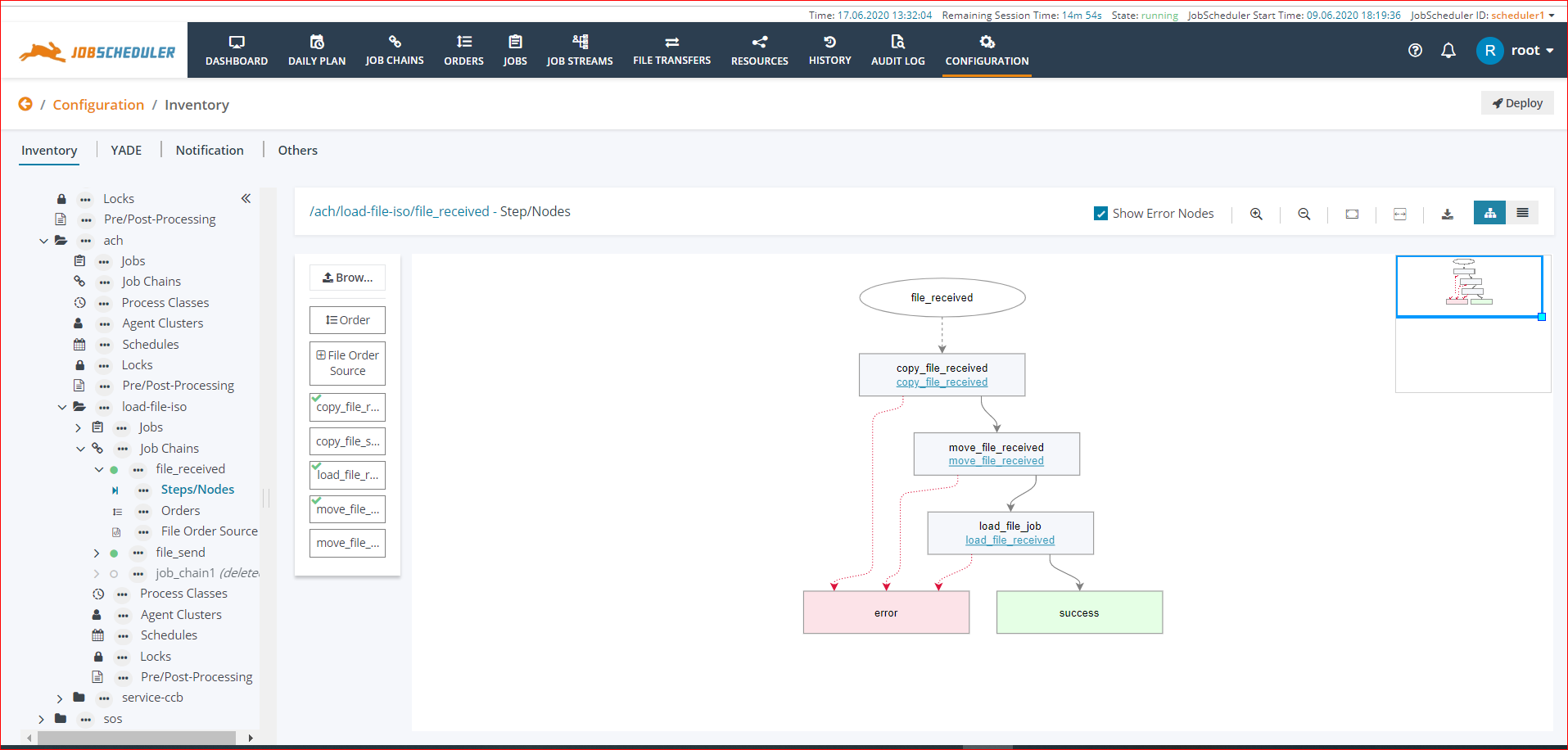
  <run\_time/>

</order>

# Salvar o Deploy los cambios

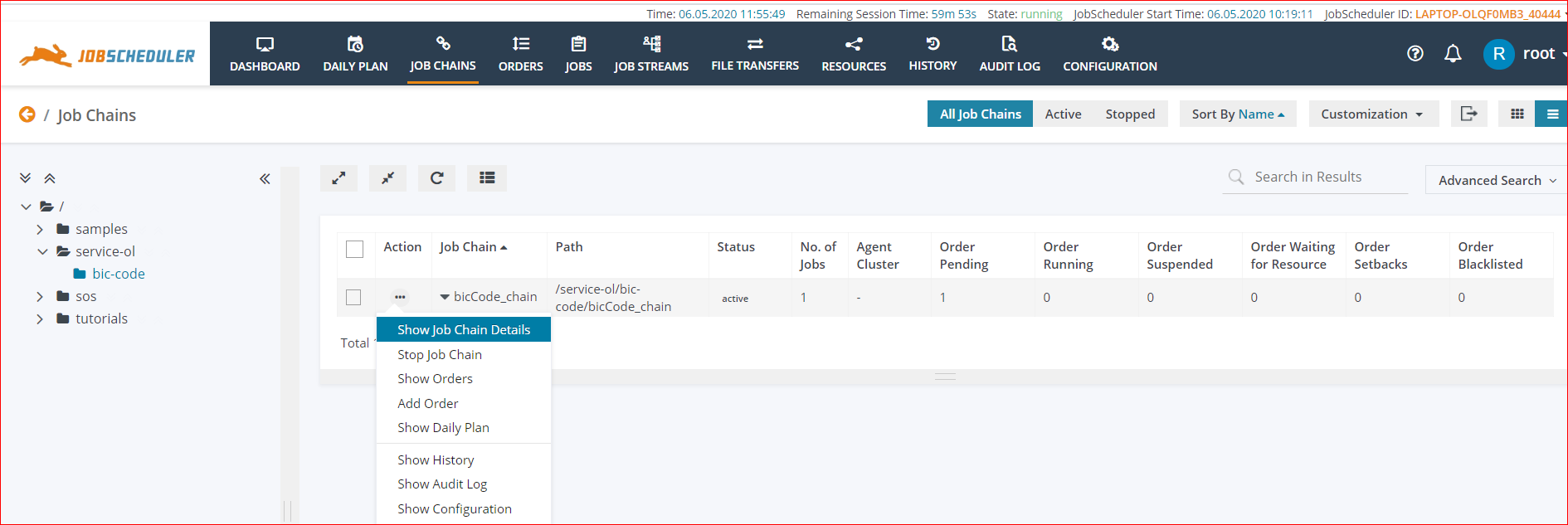
* Despliegue final de los Steps/node



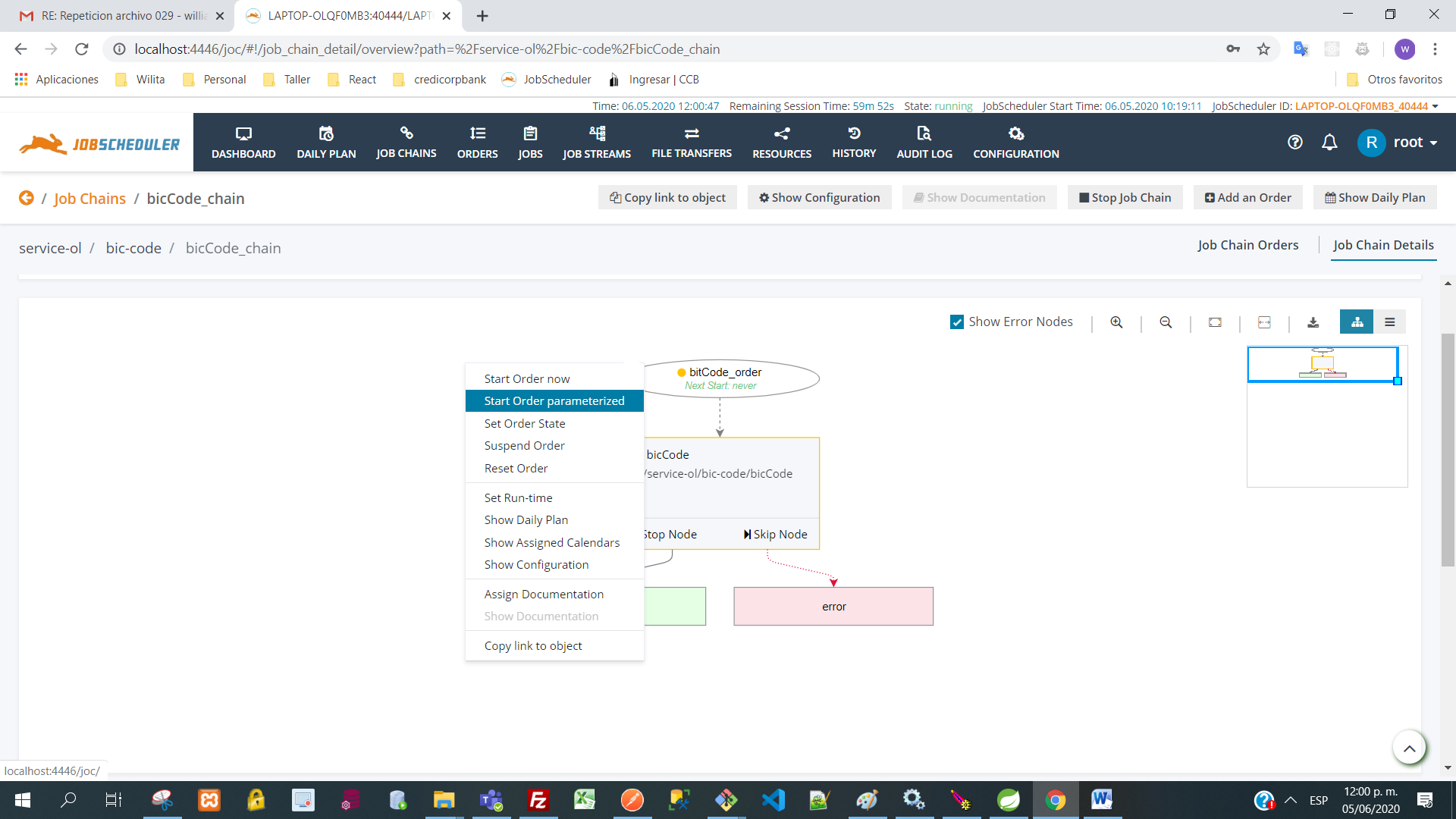
Nota: Reiniciar servidores si no se ven los cambios

# Probar el Job

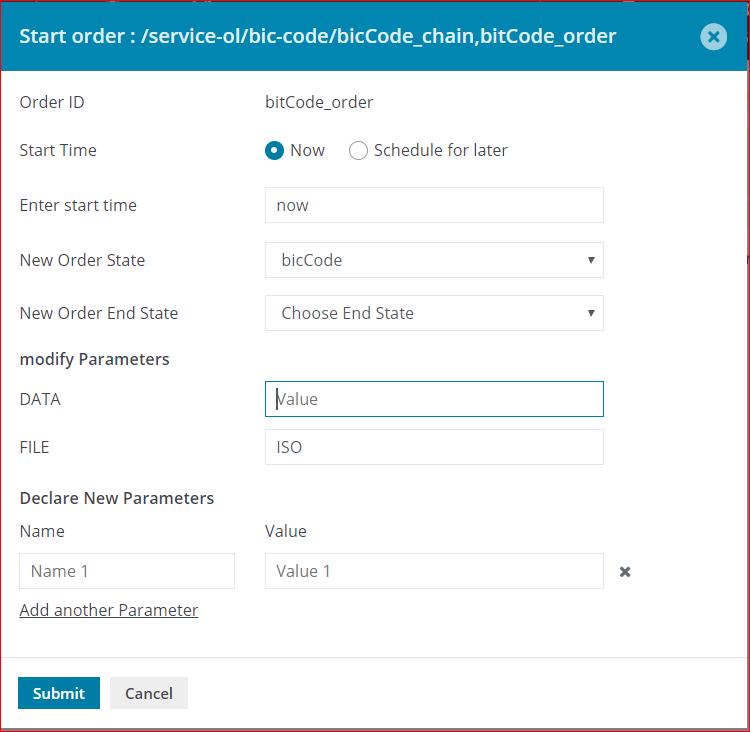
* En la opción de menú Job Chains. Buscar la carpeta service-ol/bic-code. Mostrar el Show Job Chain Details



* Comenzar na orden parametrizada

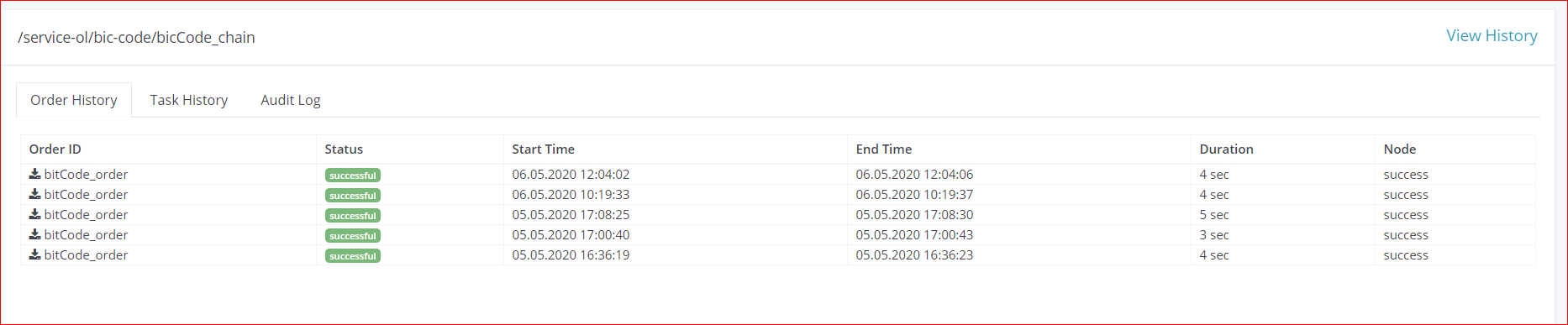


* Se despliega la vista que inicia la orden. Agregar parámetros y presione Submit para enviar

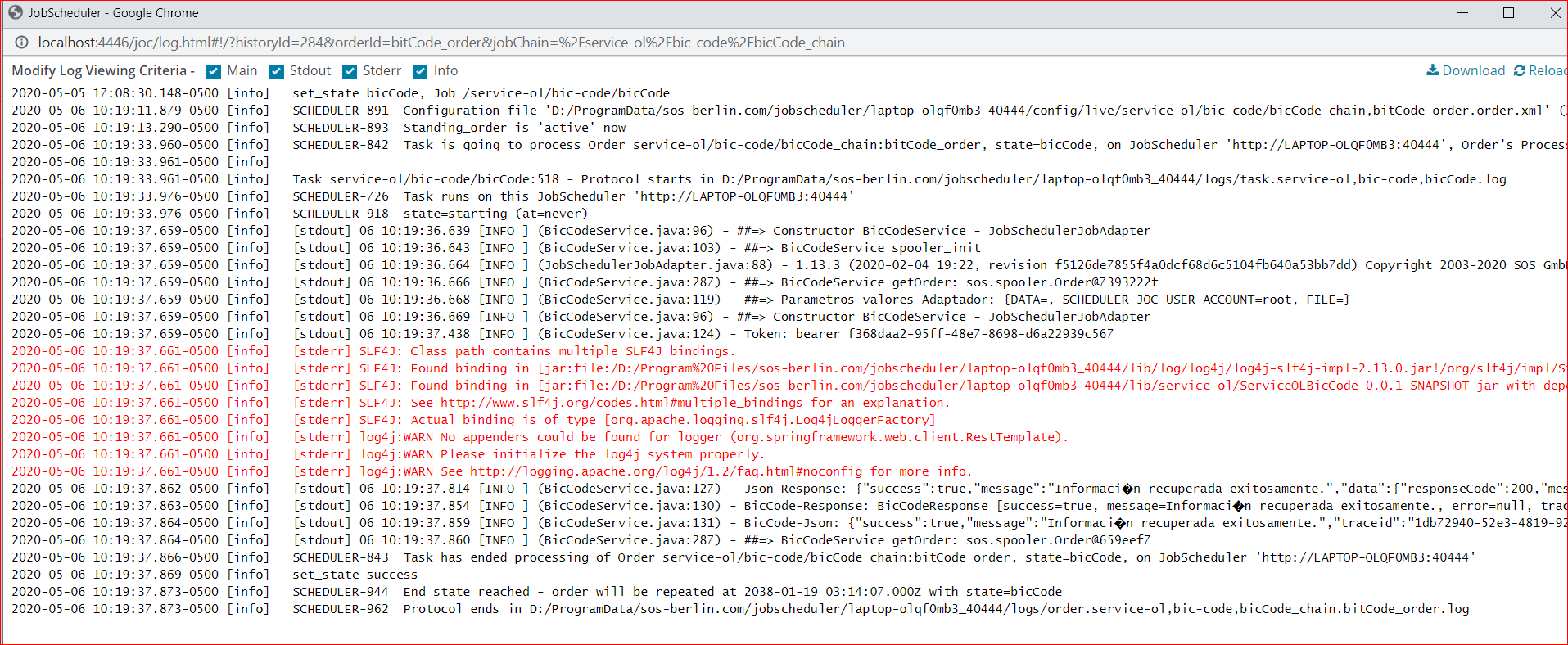


Nota: Observar parámetros que incluye la orden.

* Corre la orden



* Click en Order ID a consultar

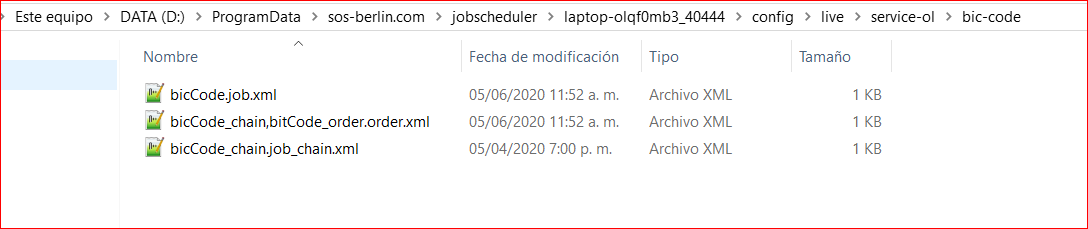


* Copiando y formateando la data de respuesta del servicio BIC Code



# Crear archivos de trabajo manualmente

Los archivos Job, Job\_chain y Order generados por el JobScheduler, en este ejemplo, se pueden generar manualmente desde un editor y luego llamarlos desde el JobScheduler automáticamente.



**bicCode.job.xml**

<?xml version="1.0" encoding="ISO-8859-1" ?>

<job title="BIc Code" order="yes" stop\_on\_error="false">

<description>

<include file="jobs/ServiceOLBicCode.xml"/>

</description>

<script language="java" java\_class="com.wlopera.ol.BicCodeService"/>

<monitor name="configuration\_monitor" ordering="0">

<script language="java" java\_class="com.sos.jitl.jobchainnodeparameter.monitor.JobchainNodeSubstituteMonitor"/>

</monitor>

<run\_time/>

</job>

**bicCode\_chain.job\_chain.xml**

<?xml version="1.0" encoding="ISO-8859-1" ?>

<job\_chain>

<job\_chain\_node job="bicCode" next\_state="success" error\_state="error" state="bicCode"/>

<job\_chain\_node state="success"/>

<job\_chain\_node state="error"/>

</job\_chain>

**bicCode\_chain,bitCode\_order.order.xml**

<?xml version="1.0" encoding="ISO-8859-1" ?>

<order priority="0" state="bicCode">

<params>

<param name="DATA" value="${SCHEDULER\_PARAM\_DATA}"/>

<param name="FILE" value="ISO"/>

</params>

<run\_time/>

</order>