Integrar Java con JobScheduler - API Jobs (engine-job-api)

**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)

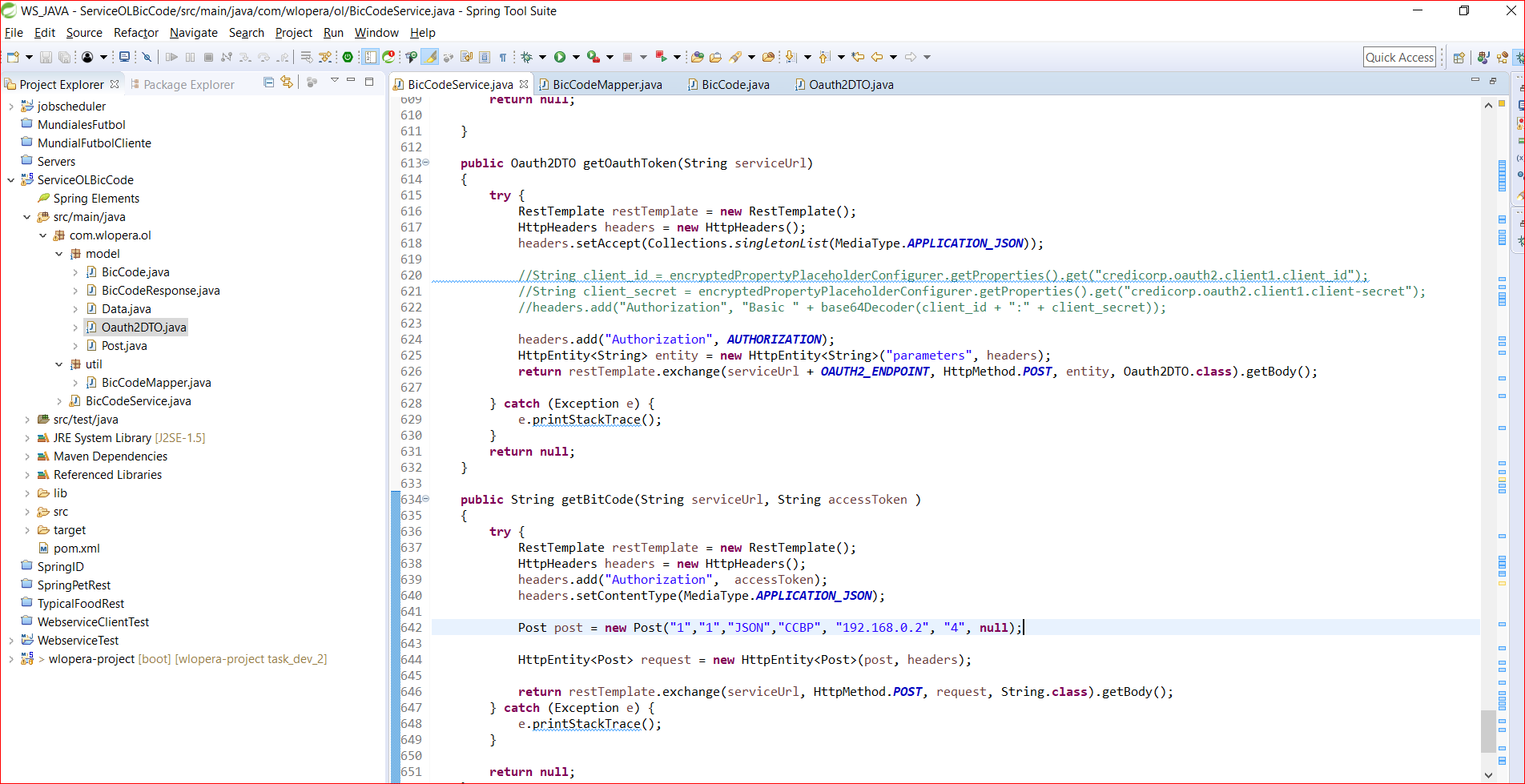
[Crear la orden (Order) 19](#_Toc39660552)

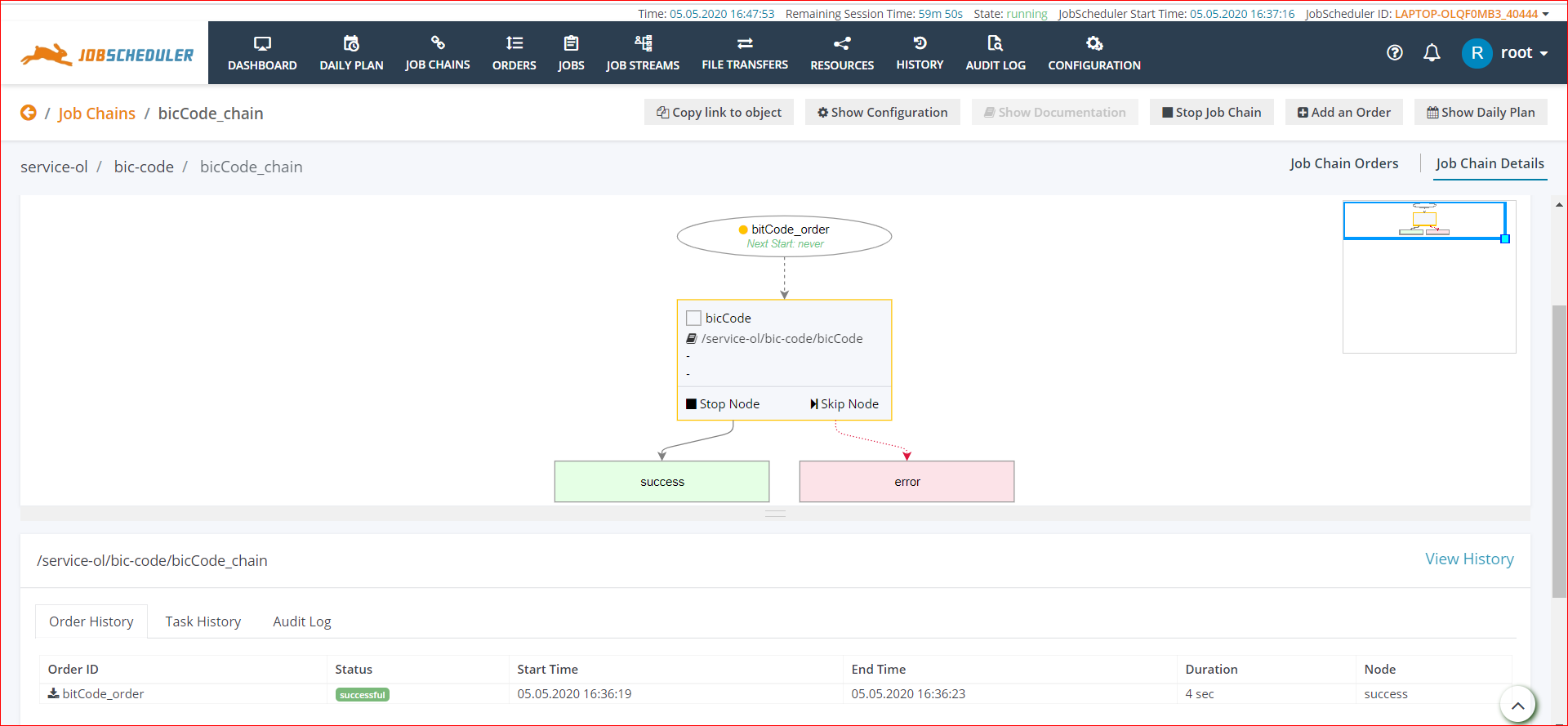
[Probar el Job 21](#_Toc39660553)

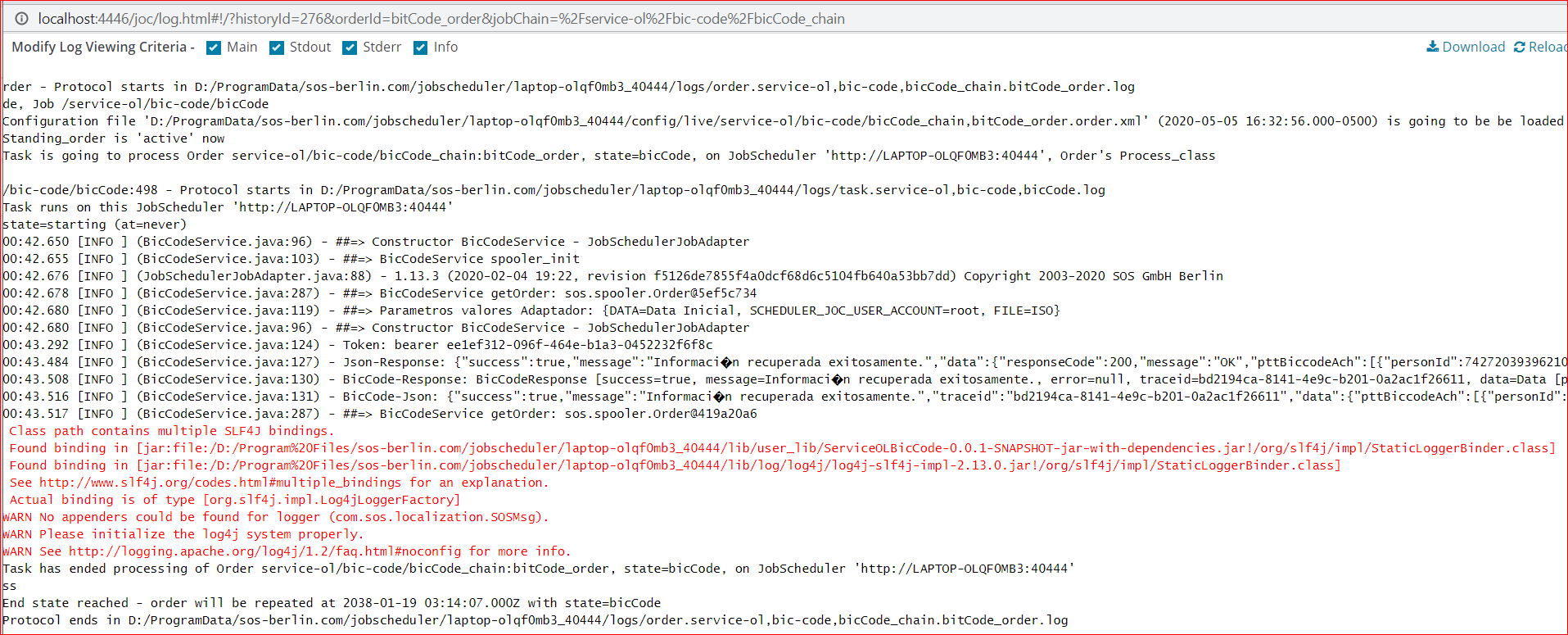
## Alcance

Este documento explica  cómo integrar el archivo JobScheduler .jar para trabajos con el API de Java (engine-job-api) en un proyecto maven.

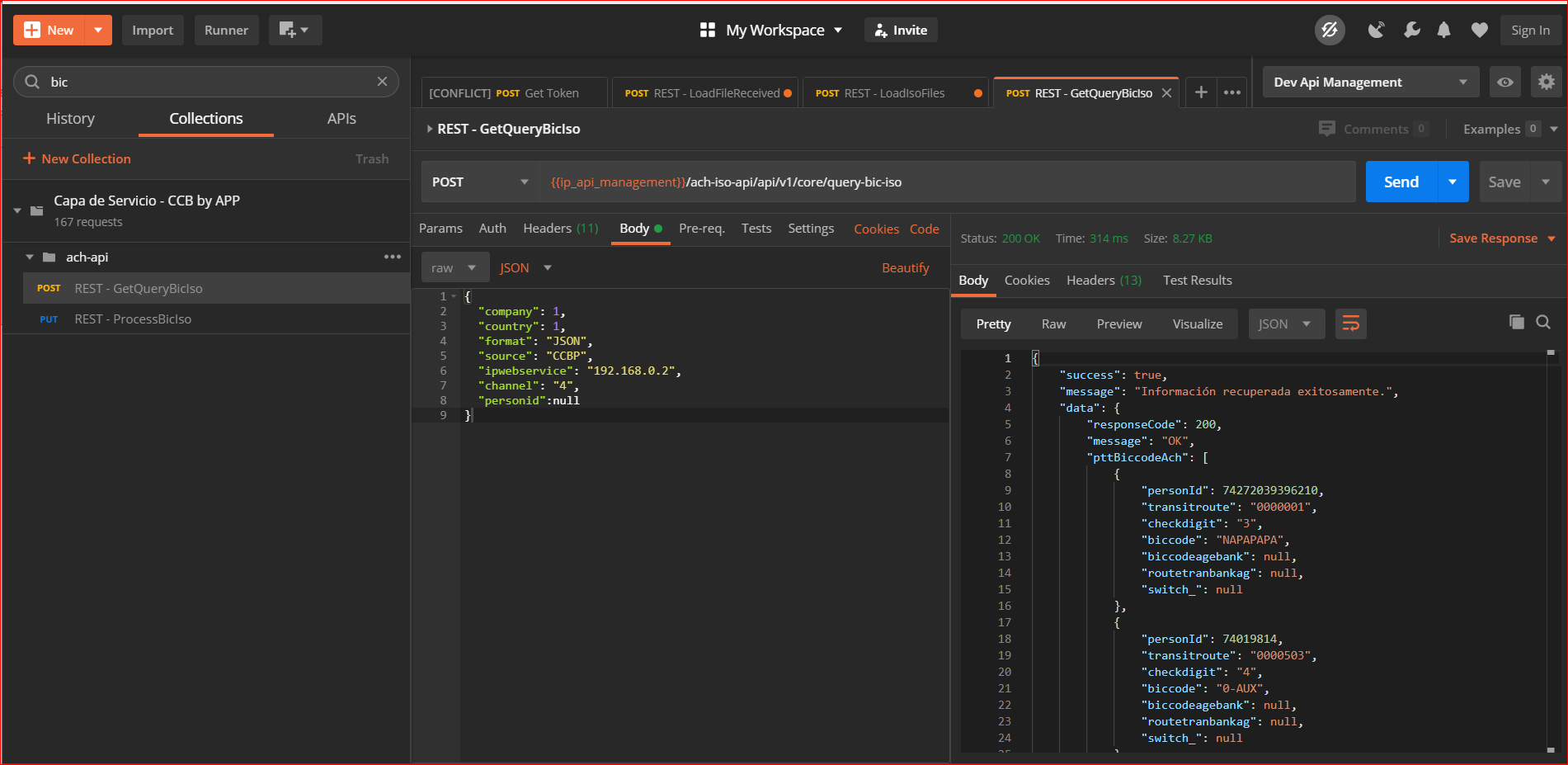
Además, este documento proporciona un pequeño ejemplo para crear una clase de trabajo API de Java y cómo usarlo en un trabajo JobScheduler.







Servicio de consultas Códigos BIC.



## 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.

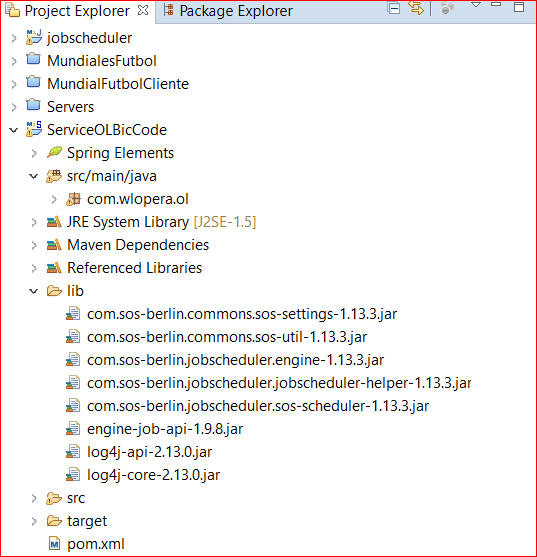
Todas las plantillas JITL se incluyen en la instalación de JobScheduler.  Están preinstalados en la carpeta *live/sos* del JobScheduler y, por lo tanto, pueden llamarse directamente desde el JOC.

**El objetivo de este documento es explicar cómo generar una clase Java que extienda de *JobSchedulerJobAdapter* y consumir un Servicio Open Legacy que consulta los códigos BIC y ejecutar una orden que consuma una tarea con ese servicio.**

## 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 tomadas de la instalación del JobSheduler:

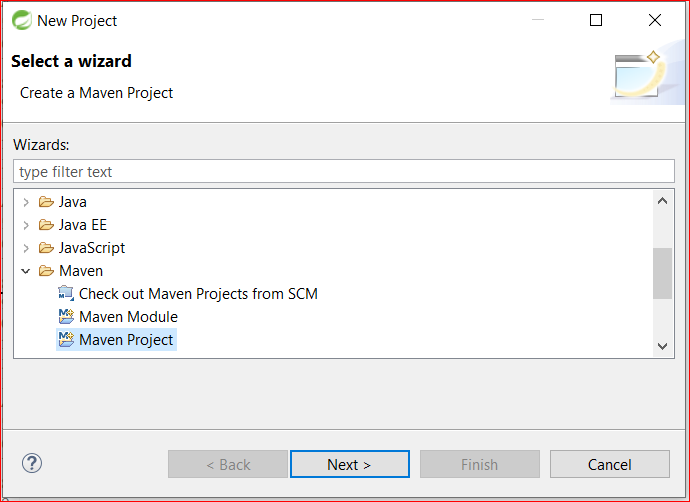


Se deben agregar al pom.xml de maven

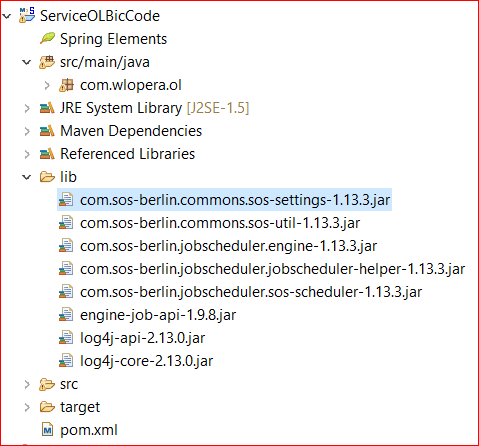
## Aplicativo:

Proyecto JobScheduler – Api Job – Java, para consumo de un Servicio REST de Open Legacy. Consulta de Códigos BIC.

# Crear 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

<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>

</dependencies>

* Agregar Plugin para generar .jar con las dependencias de clases requeridas por el aplicativo (pom.xml)

<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>

# Crear clases

BicCodeService: 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 del método que consulta los códigos BIC. (Esta clase debe depurarse)

BicCode: Parámetros de los códigos BIC

Oauth2DTO: Parámetros del token

Data: Parámetros con parte de la respuesta de los códigos BIC

BicCodeResponse: Respuesta del servicio de consulta de los códigos BIC

BicCodeMapper: Mapeo de datos Json <=> Objeto BicCodeResponse

**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 serviceUrl): Método que permite obtener el token de conexión a los servicios.

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();

getBitCode(String serviceUrl, String accessToken): Método que permite obtener los códigos BIC a partir del endpoint, el token de conexión y los parámetros de la petición.

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.

RestTemplate restTemplate = new RestTemplate();

HttpHeaders headers = new HttpHeaders();

headers.add("Authorization", accessToken);

headers.setContentType(MediaType.APPLICATION\_JSON);

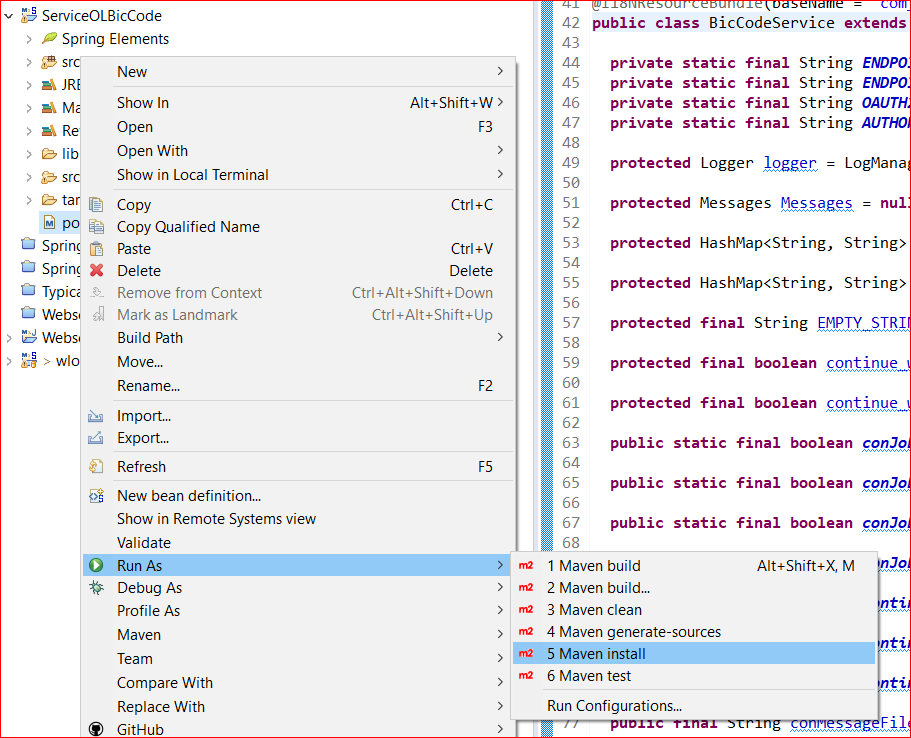
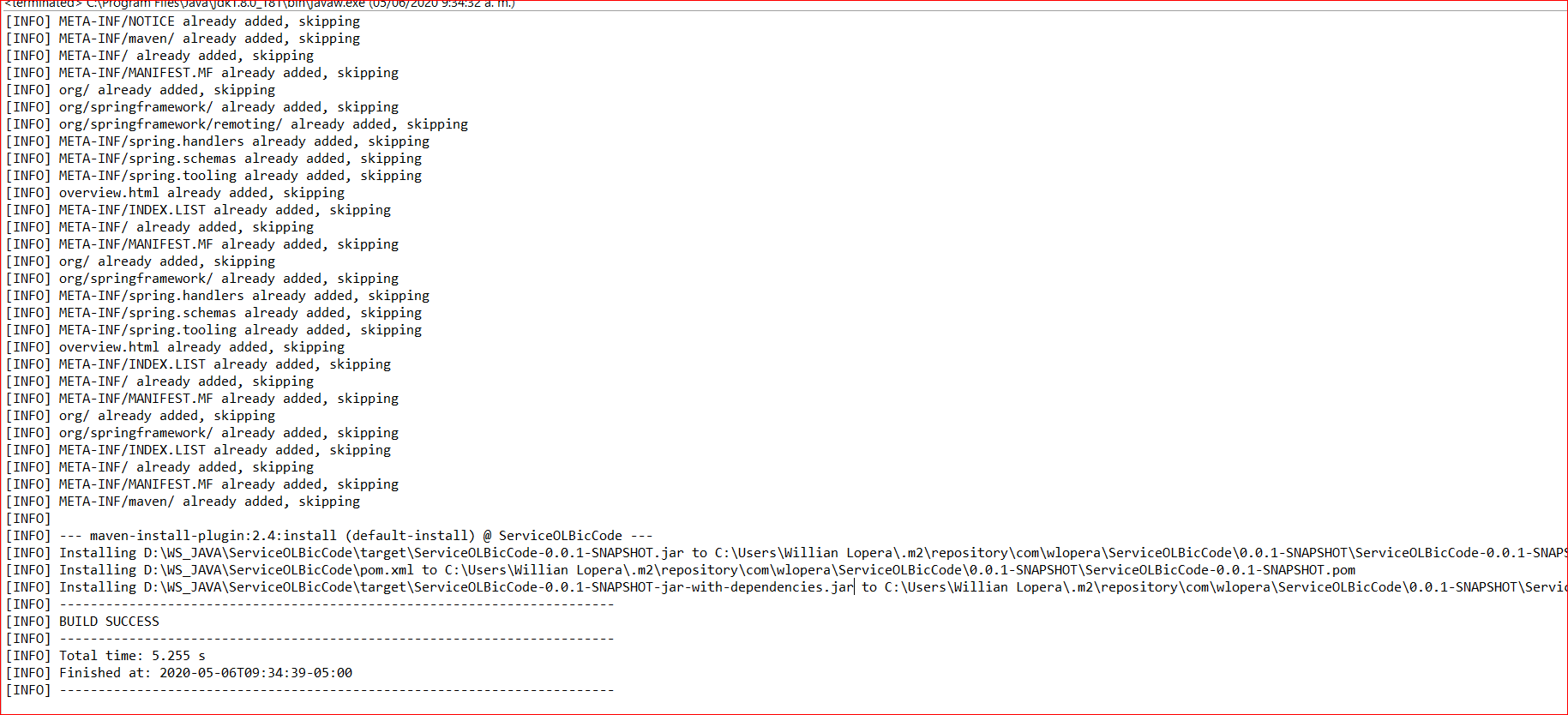
Post post = new Post("1","1","JSON","CCBP", "192.168.0.2", "4", null);

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

return restTemplate.exchange(serviceUrl, HttpMethod.POST, request, String.class).getBody();

Nota: personid = null (Retorna todos los códigos BIC)

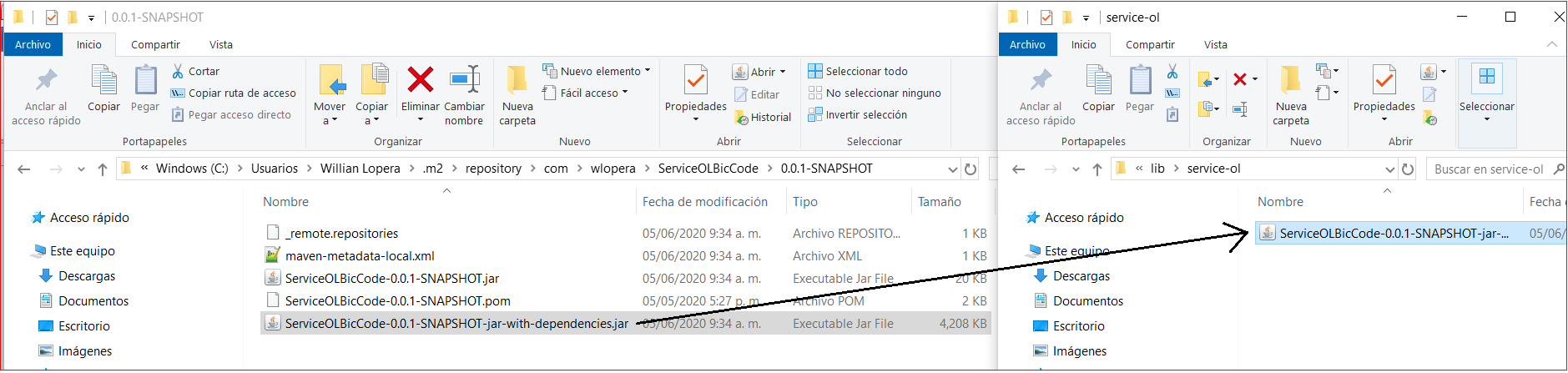
# Compilar y subir el código al ambiente JobScheduler

* Copiar el archivo.jar con dependencias generado en la carpeta del usuario (repositorio maven) a la carpeta de librerías del JobScheduler

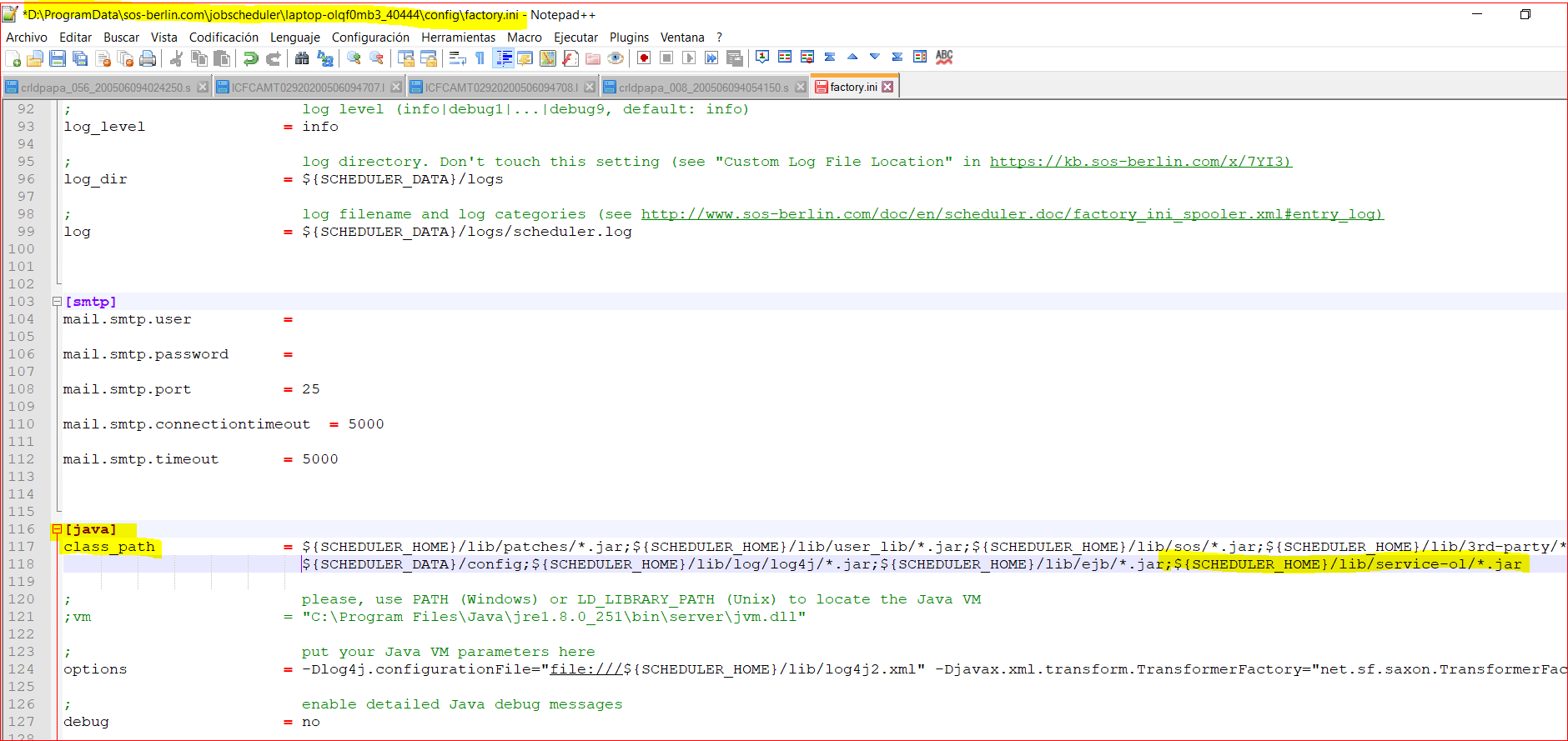
C:\Users\Willian Lopera\.m2\repository\com\wlopera\ServiceOLBicCode\0.0.1-SNAPSHOT =>

D:\Program Files\sos-berlin.com\jobscheduler\laptop-olqf0mb3\_40444\lib\service-ol



* 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.

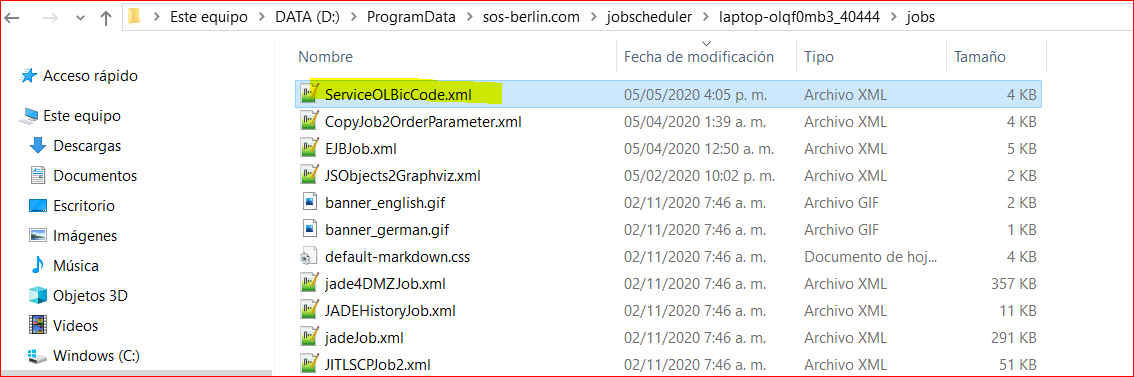
D:\ProgramData\sos-berlin.com\jobscheduler\laptop-olqf0mb3\_40444\config\ factory.ini



# Generar la documetación 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>

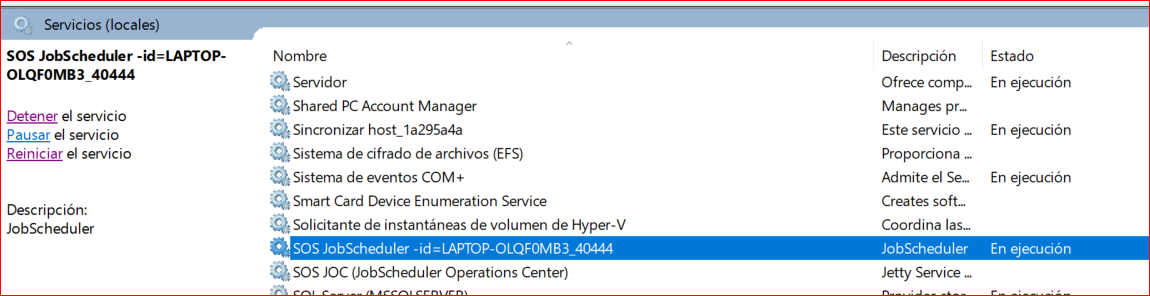
Uso de PL para Consulta, creación, modificación y borado de codigos BIC

</div>

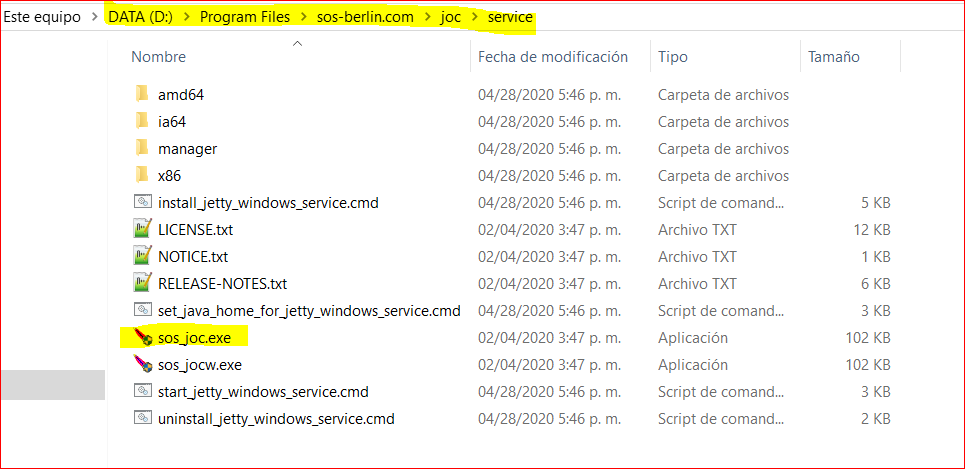
</documentation>

</description>

* Conexión al ambiente JobScheduler:
  + Levantar Servicio Jobscheduler



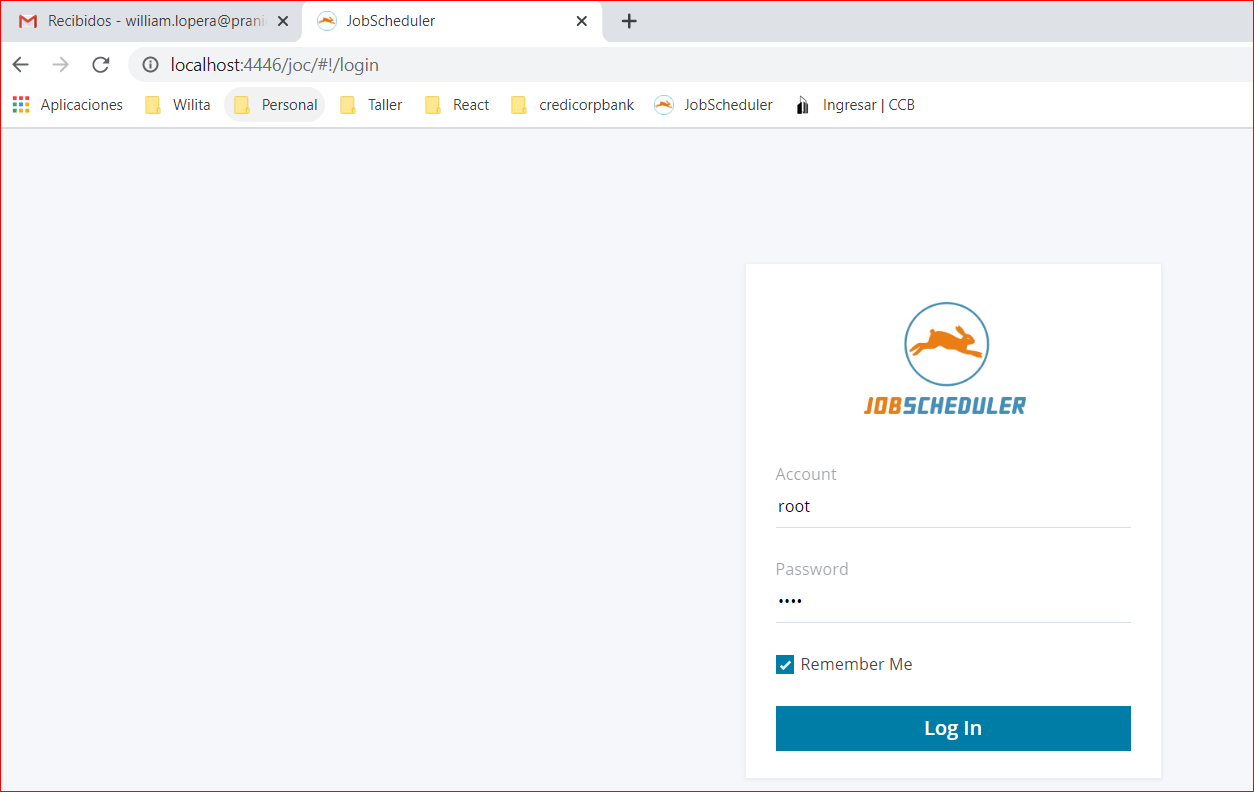
* Levantar el JOC - Jetty (en caso de error en el servicio, levantar por consola)



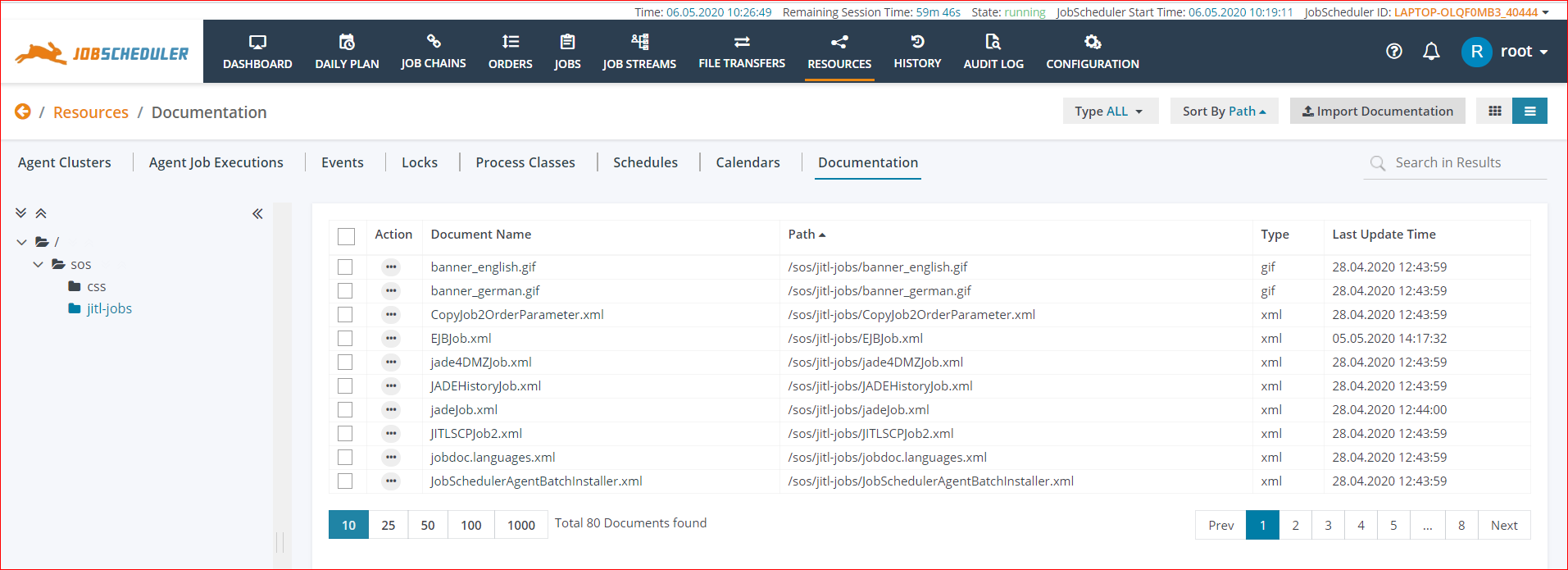
* Conectarse al ambiente JobScheduler

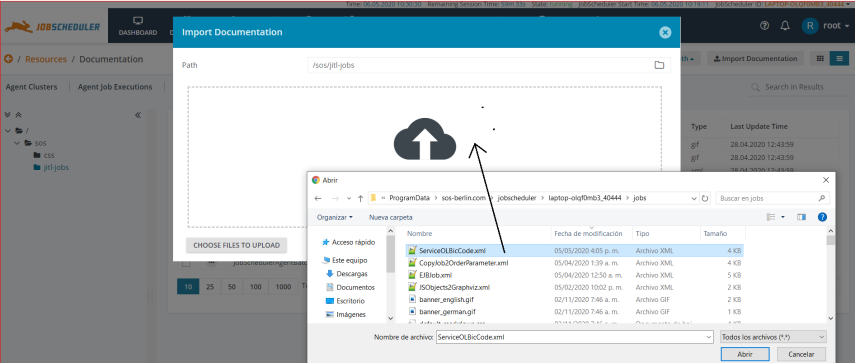
url: <http://localhost:4446/joc/#!/login>

user/password: root/root

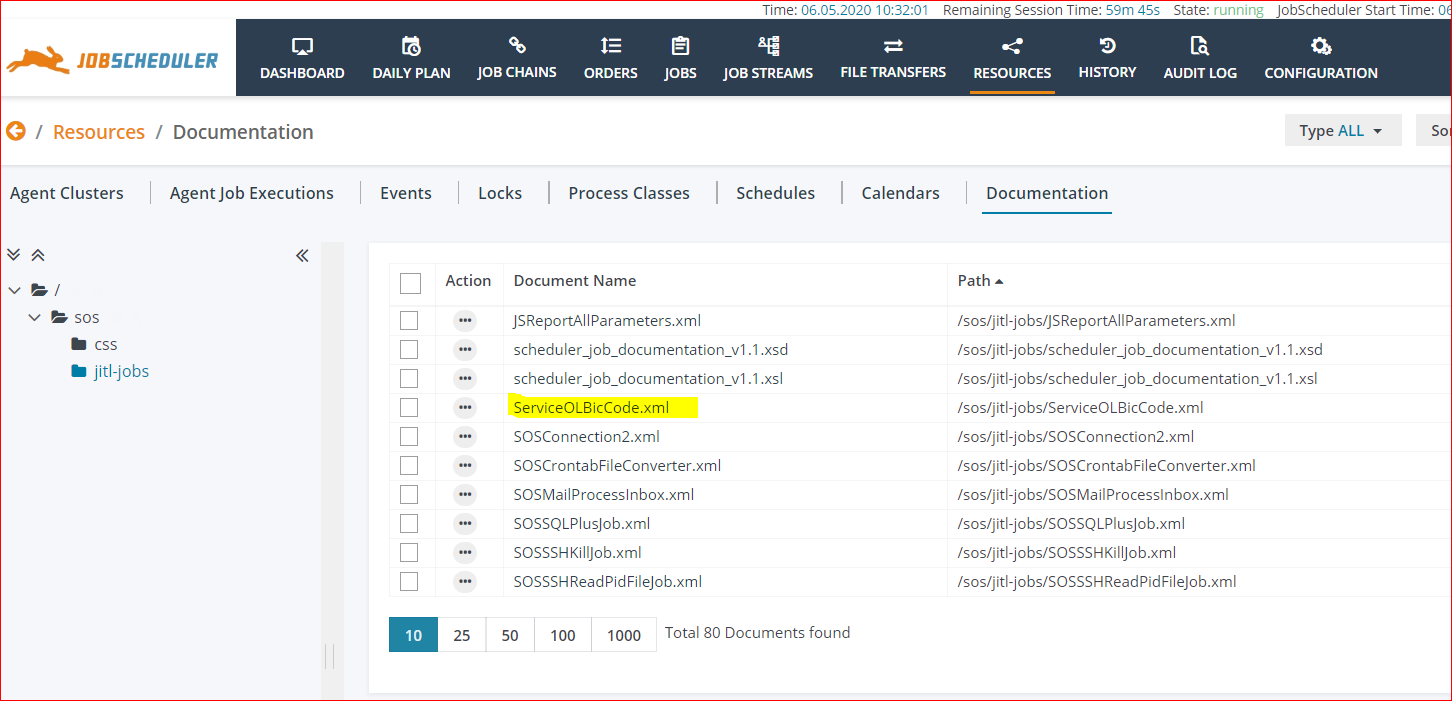


* Agregar el documento al JobScheduler: Importar el documento .xml creado en el Import Documentation del menú Resources/Documentacion

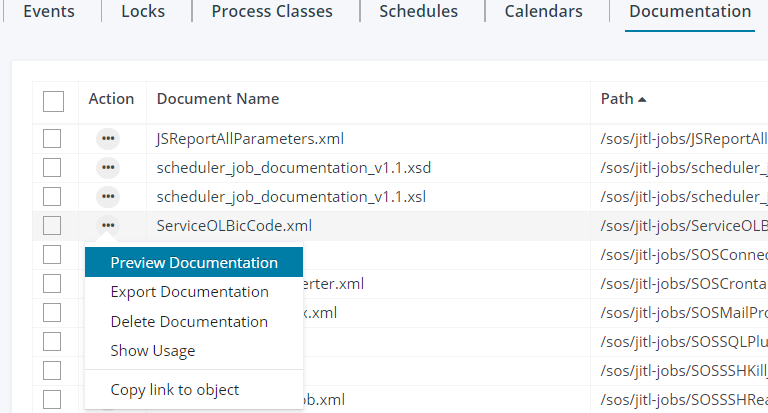


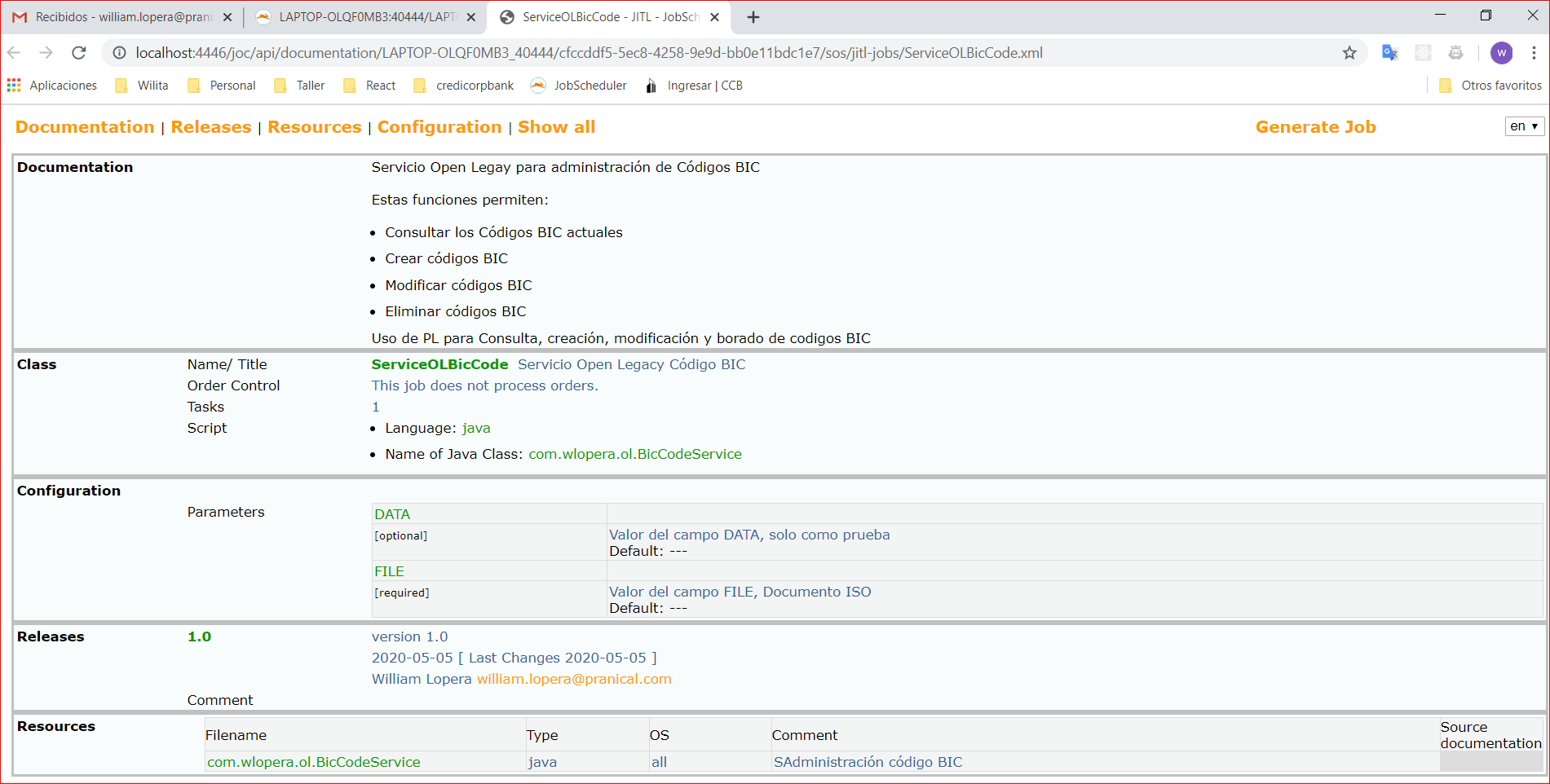


* Se agrega el archivo a la documentación



* Pre visualizar el documento





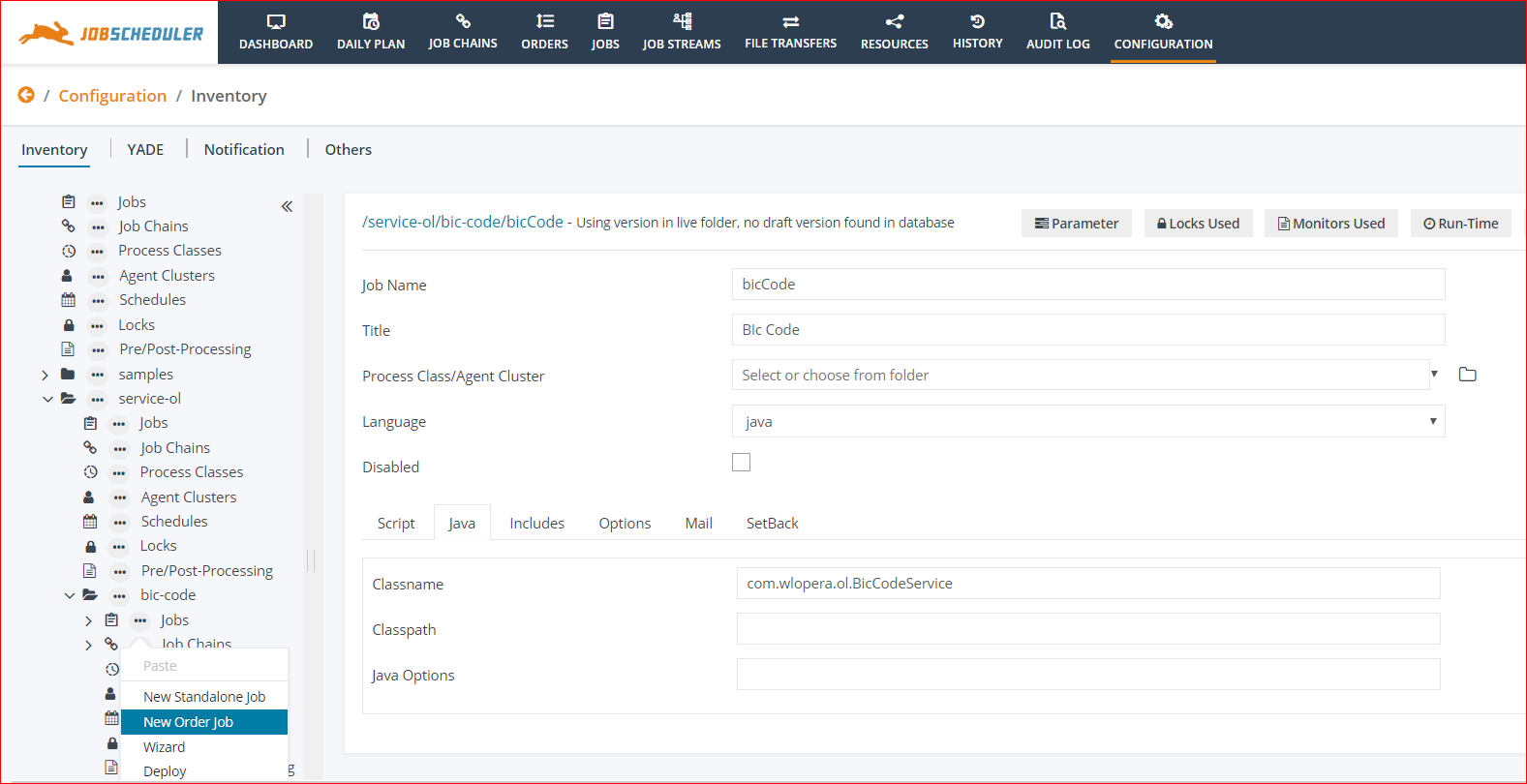
# Crear el Job para probar el aplicativo

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

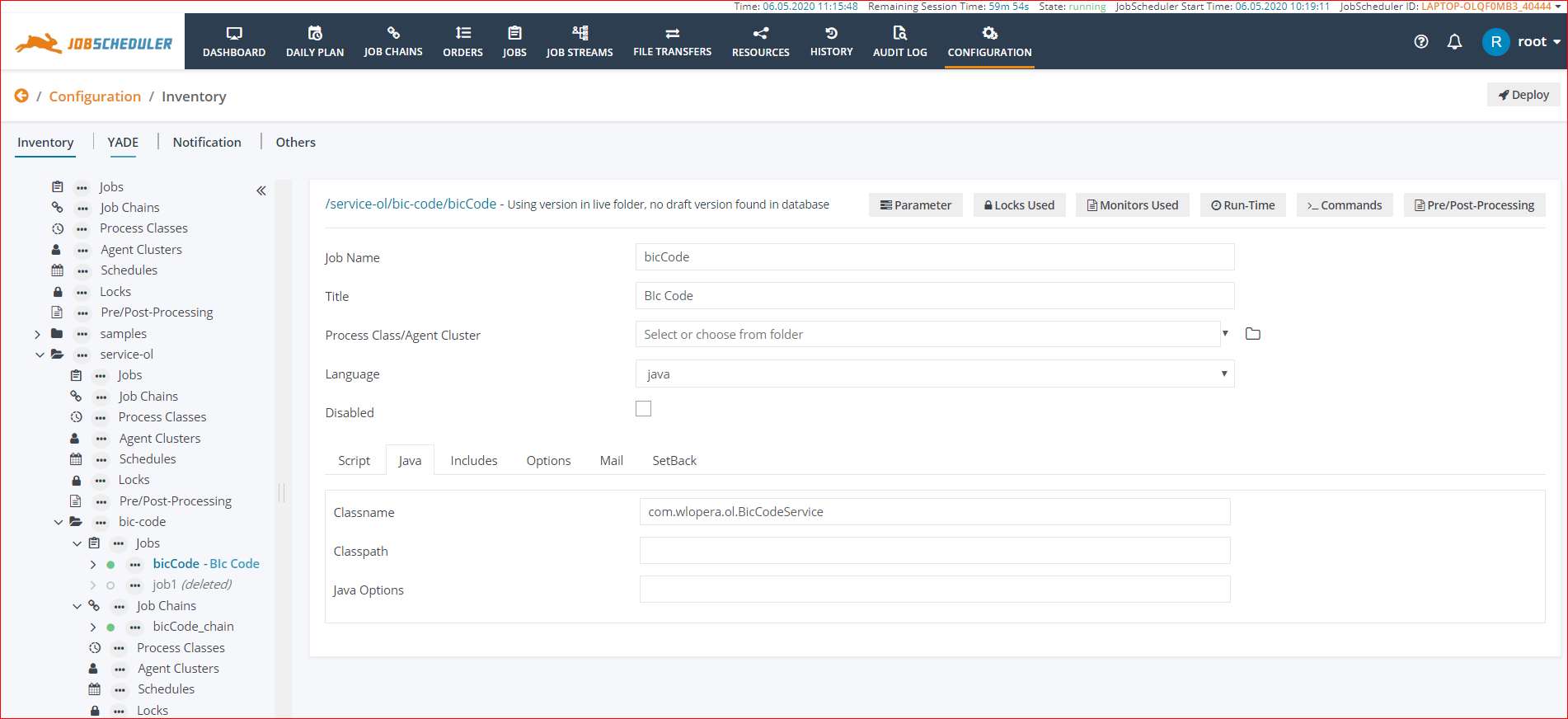
D:\ProgramData\sos-berlin.com\jobscheduler\laptop-olqf0mb3\_40444\config\live\service-ol\bic-code

* Crear un nuevo Job: New Order job

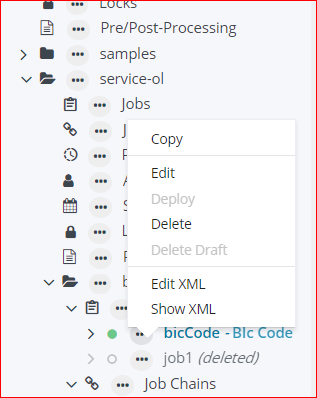
En la opción de menú *Configuration.* Buscar la carpeta *service-ol*



* Agregar los datos del nuevo Job
  + Job Name: bitCode (Identificador)
  + Title: Bic Code (Titulo del Job)
  + Language: java (Language que procesa el Job)
  + Classname: com.wlopera.ol.BicCodeService (Clase que procesa el Job)



* Editar el archivo.xml a generar para agregar la referencia a la documentación

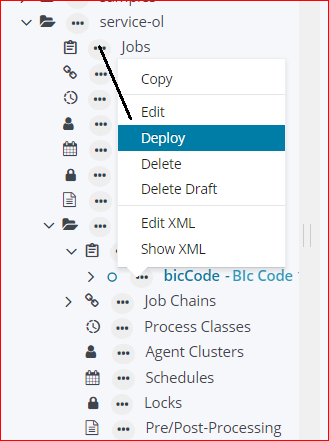


* Incluir el nombre de la documentación generada previamente en el paso 8 (jobs/ServiceOLBicCode.xml)

Presione Submit una vez modificado el xml

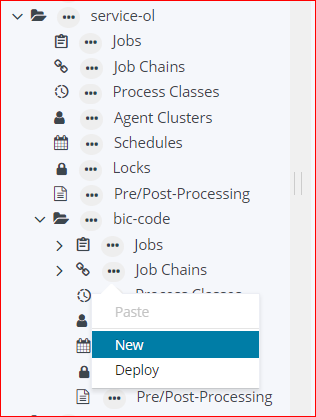


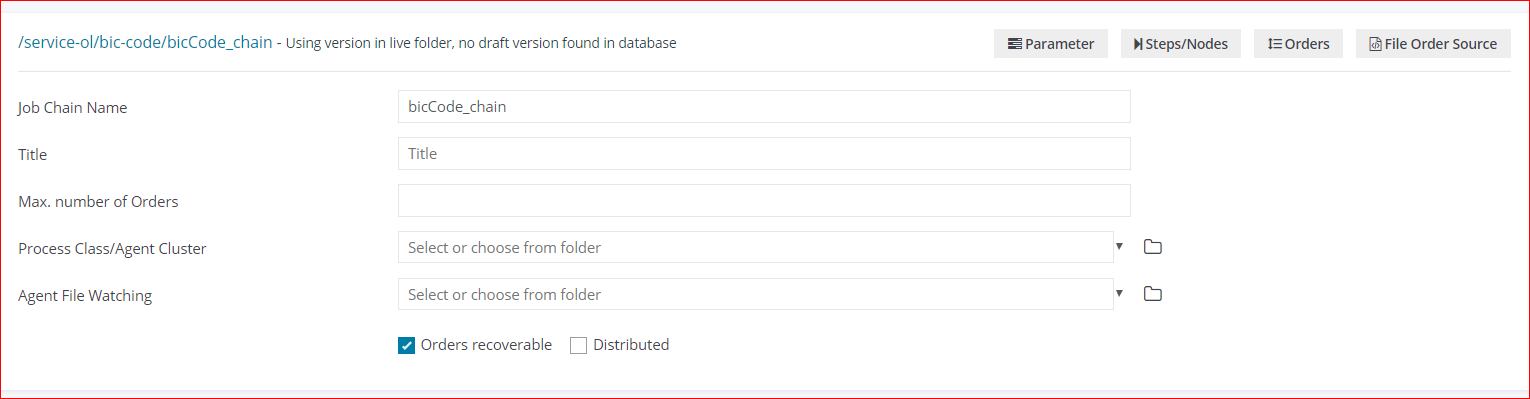
* Salvar (Deploy) los cambios.



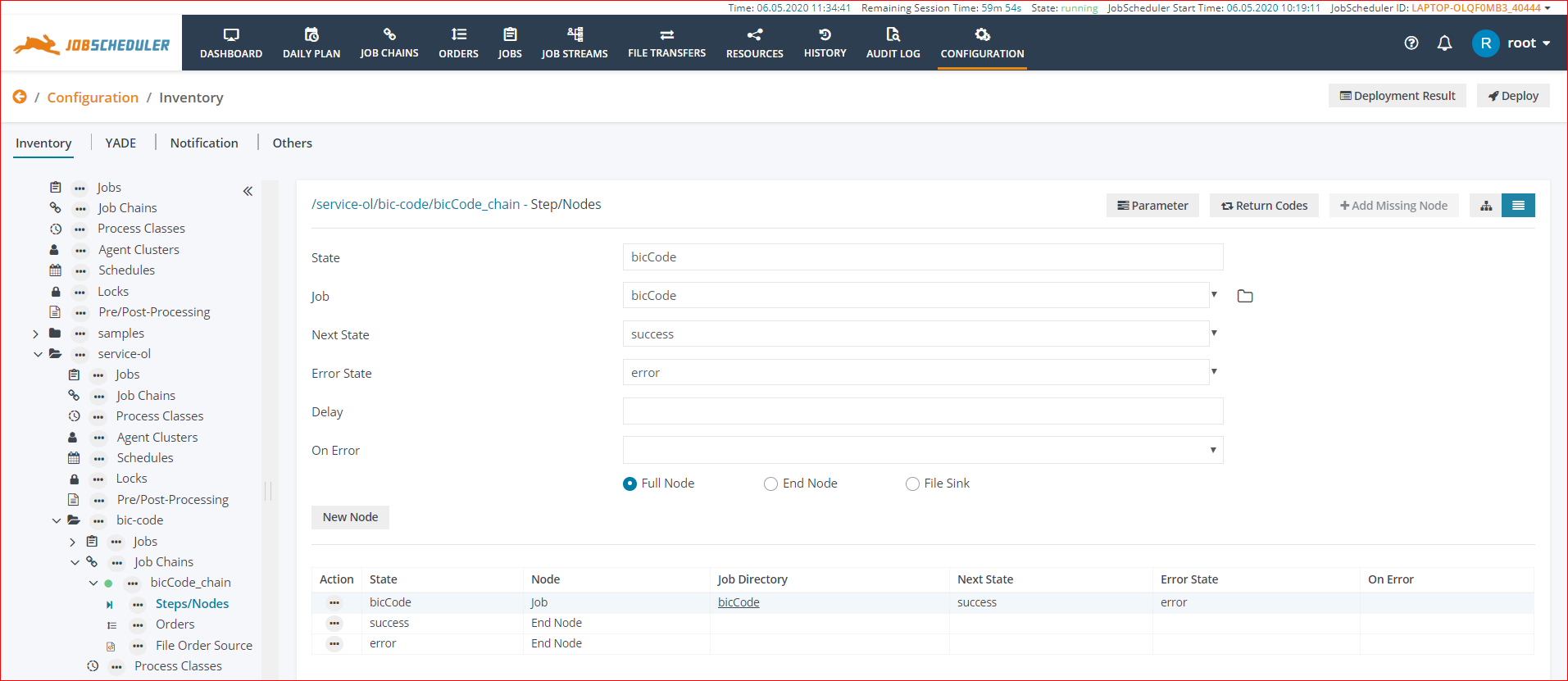
# Crear el Job Chain

* Job Chain name: bitCode\_chain

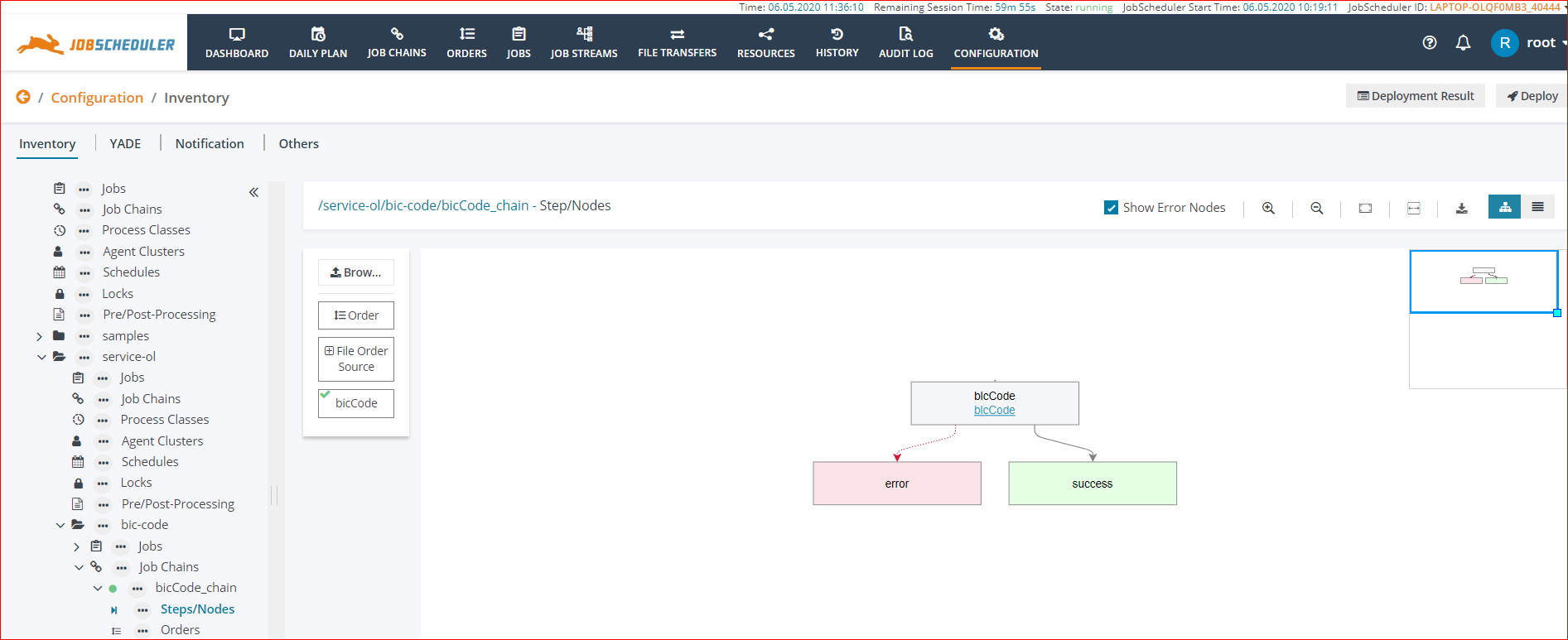




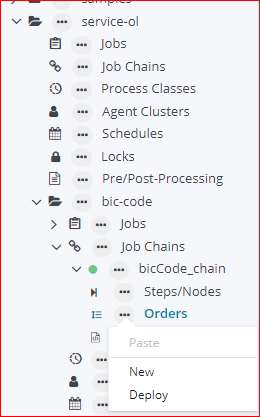
* Agregar los pasos (nodos) a seguir por el Job Chain
  + Agregando los nodos por formulario:
    - Nodo: Crear success
    - Nodo: Crear error
    - Nodo: bicCode
  + Colocar la relación entre los nodos (éxito/fracaso)



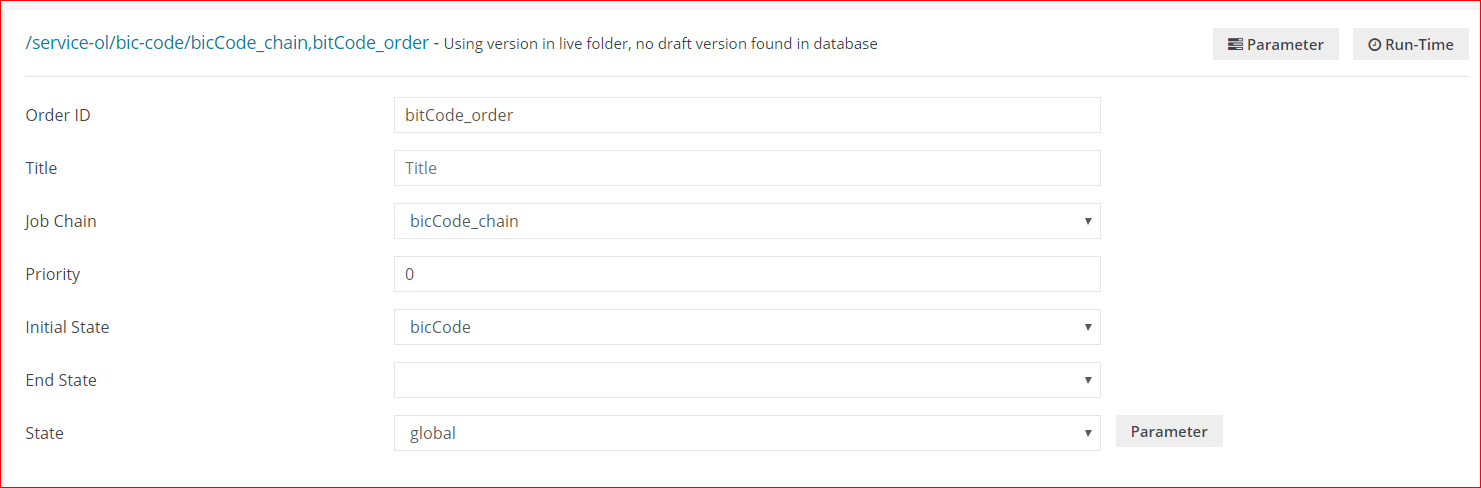
* + O agregando los nodos de manera gráfica



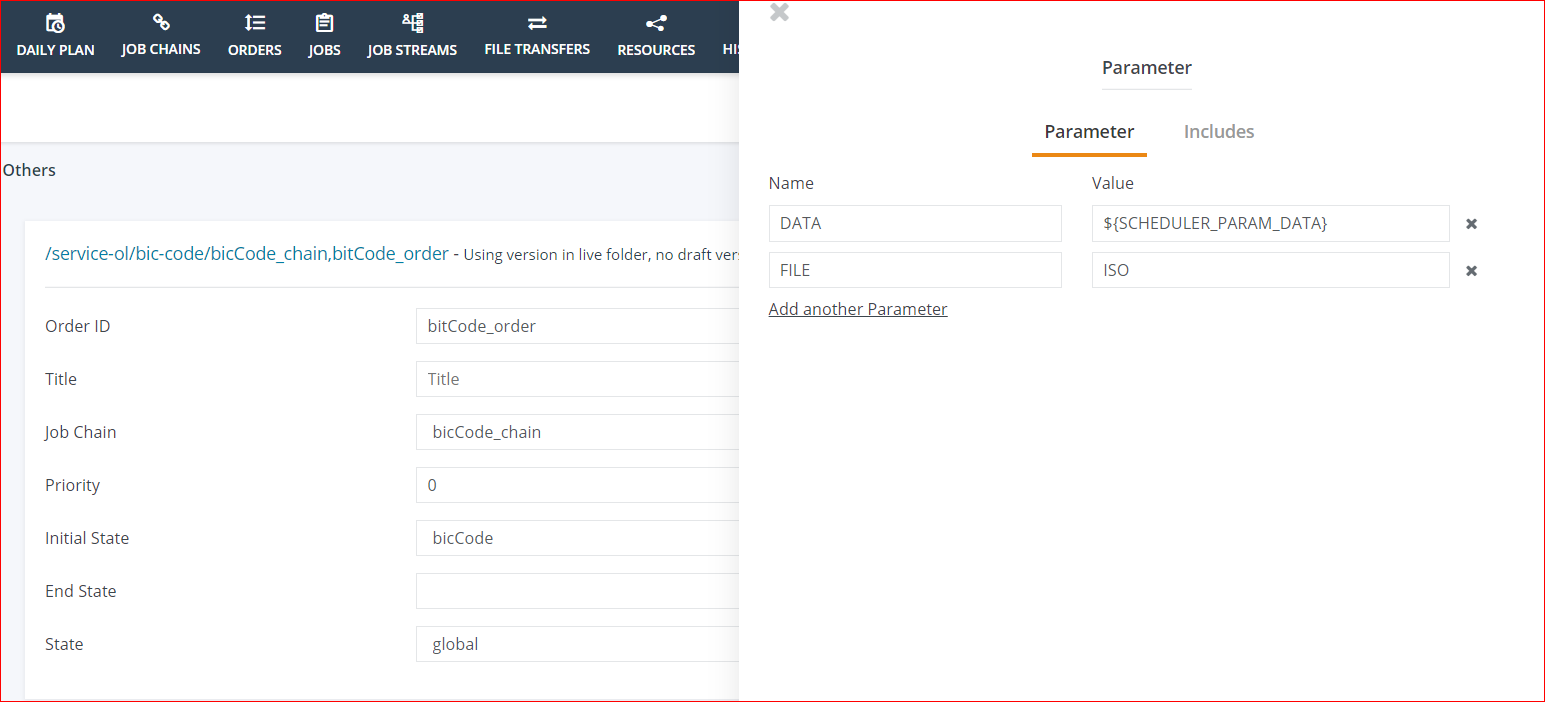
# Crear la orden (Order)



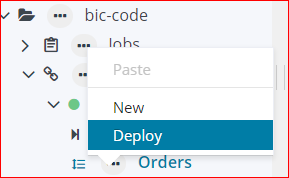
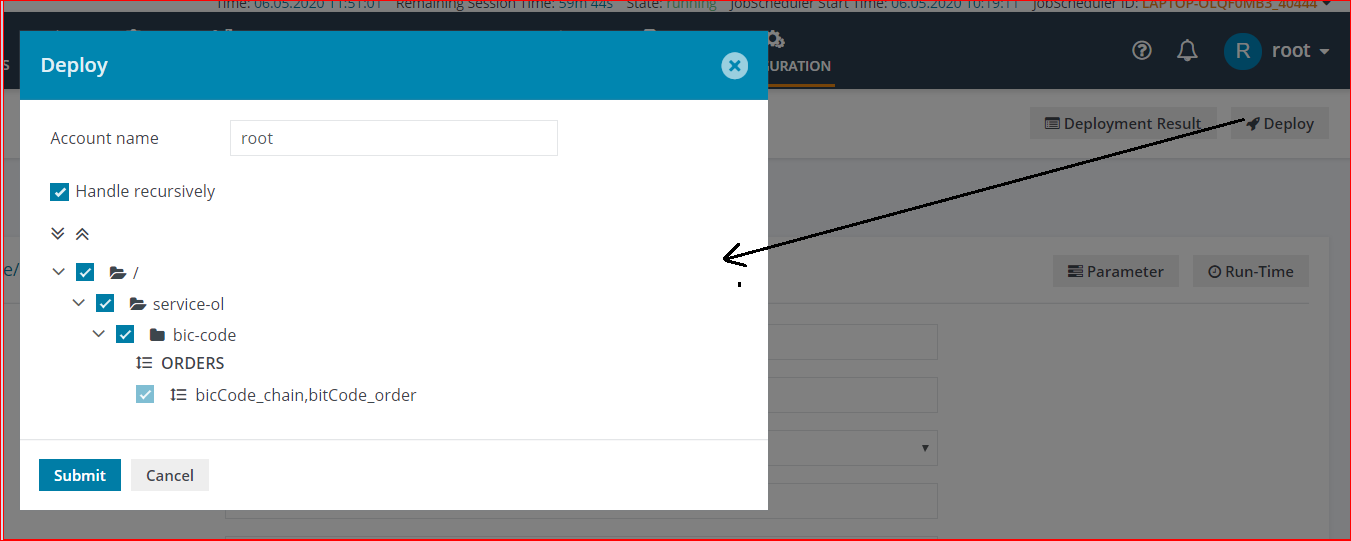
* Agregar los datos
  + Order Id: bitCode\_order (identificador único)
  + Title: Titulo (opcional
  + Job chain: bitCode\_chain (Identificador del Job Chain a ejecutarse en esta orden)
  + Inicial State: bitCode (Nodo inicla del Job\_chain)



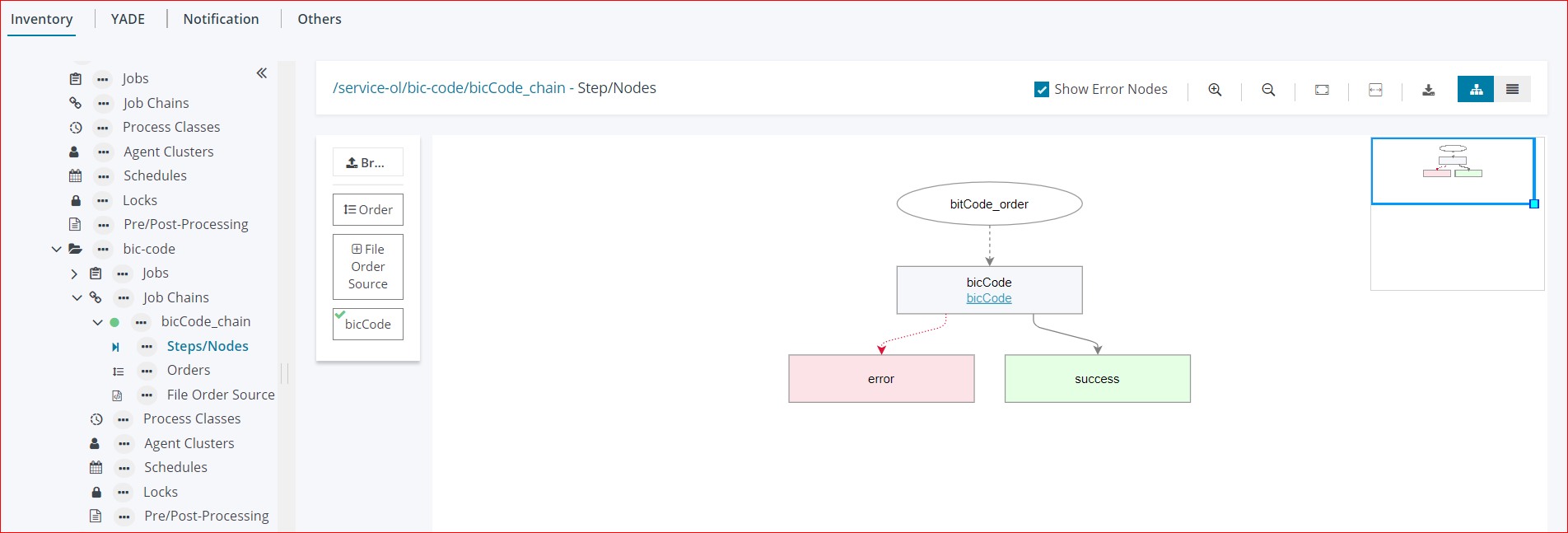
* Agregar parámetros:
  + Consultar de una variable : DATA = ${SCHEDULER\_PARAM\_DATA}
  + Valor inicial: File = ISO



* Salvar o Deploy los cambios

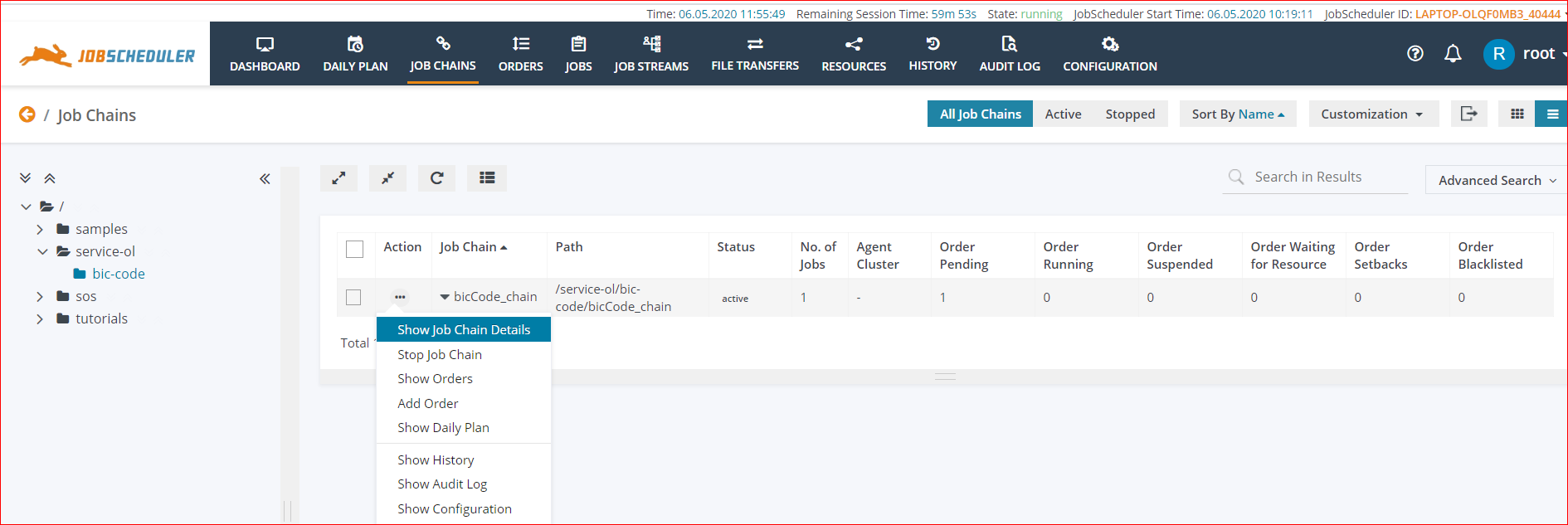
* Despliegue final de los Steps/node



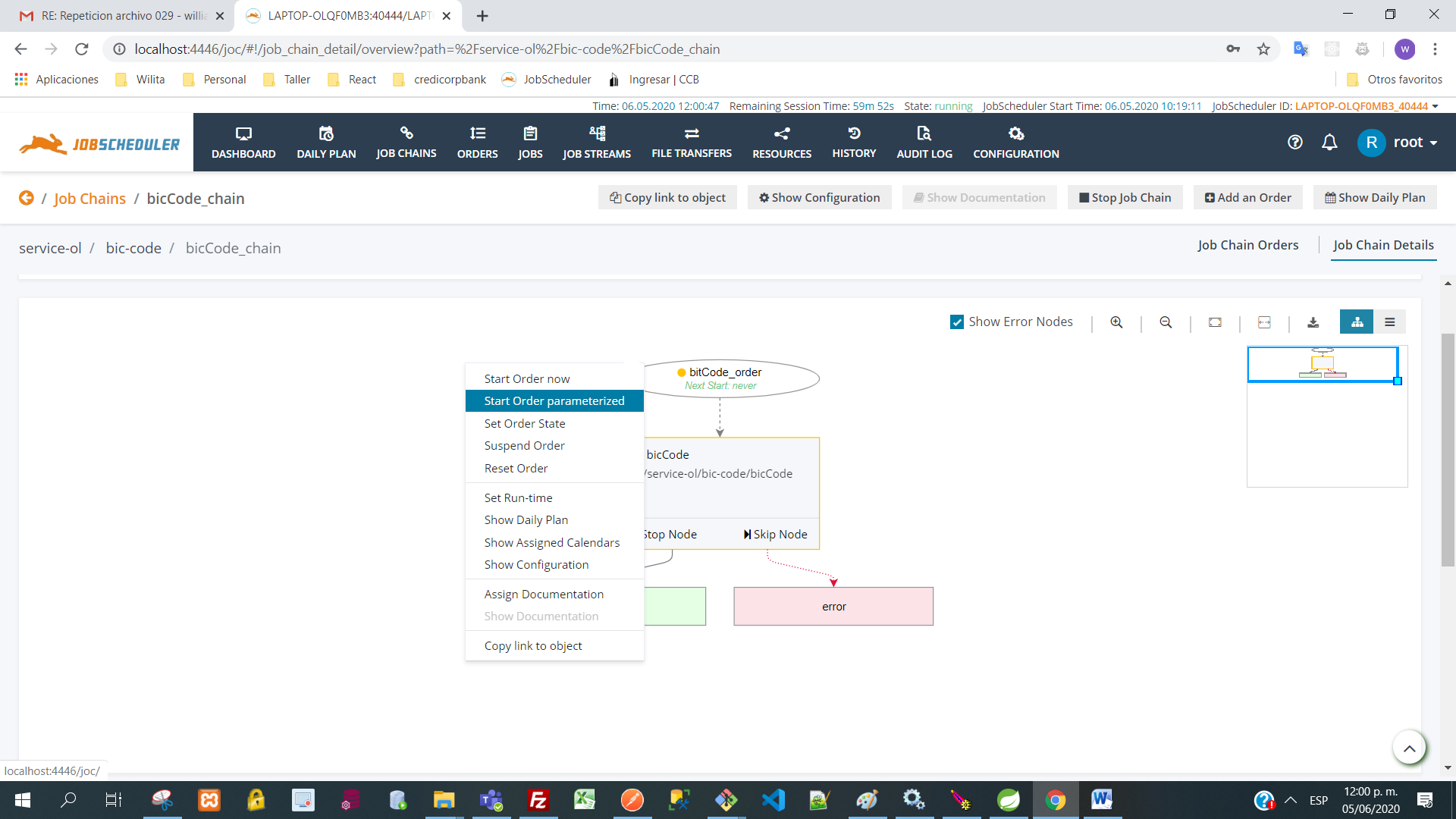
Nota: Reiniciar servidores si no se ven los cambios

# Probar el Job

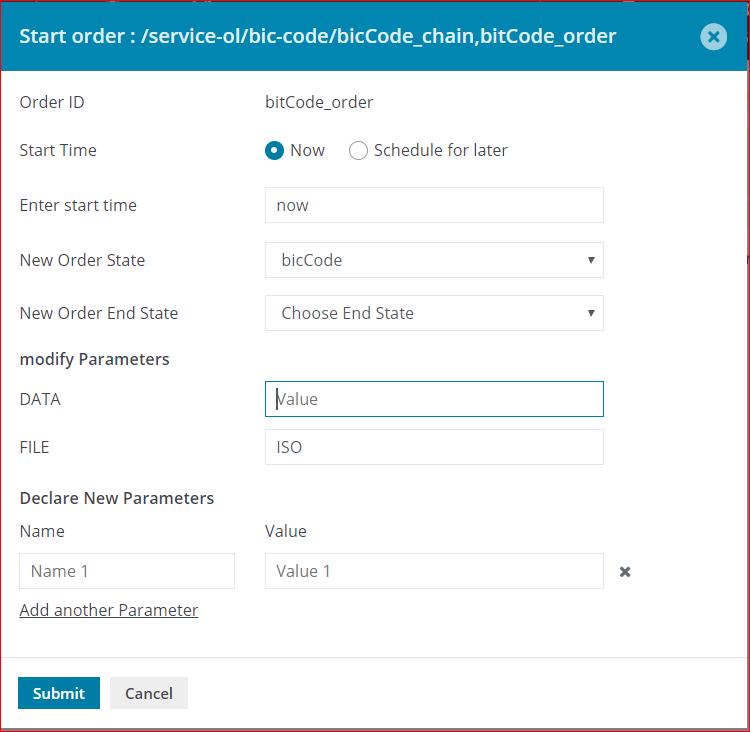
* En la opción de menú Job Chains. Buscar la carpeta service-ol/bic-code. Mostrar el Show Job Chain Details



* Comenzar na orden parametrizada

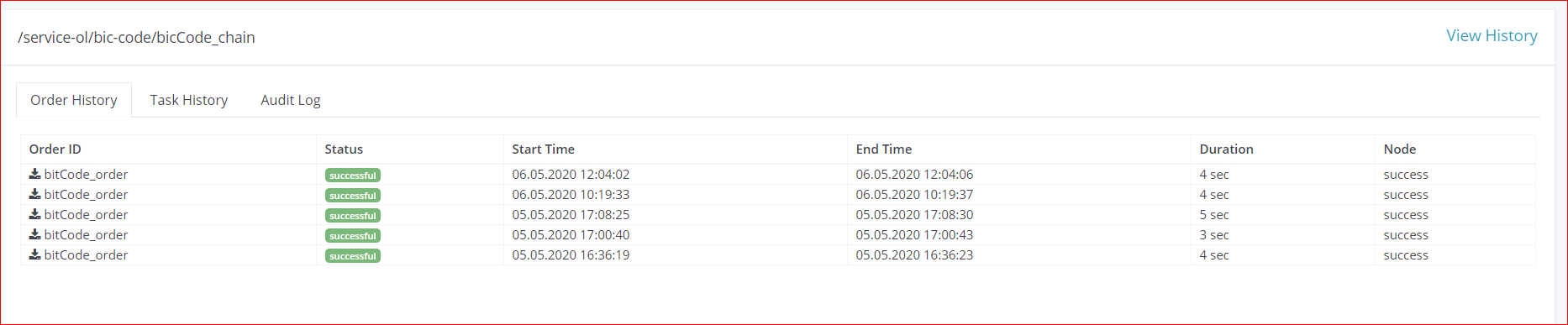


* Se despliega la vista que inicia la orden. Agregar parámetros y presione Submit para enviar

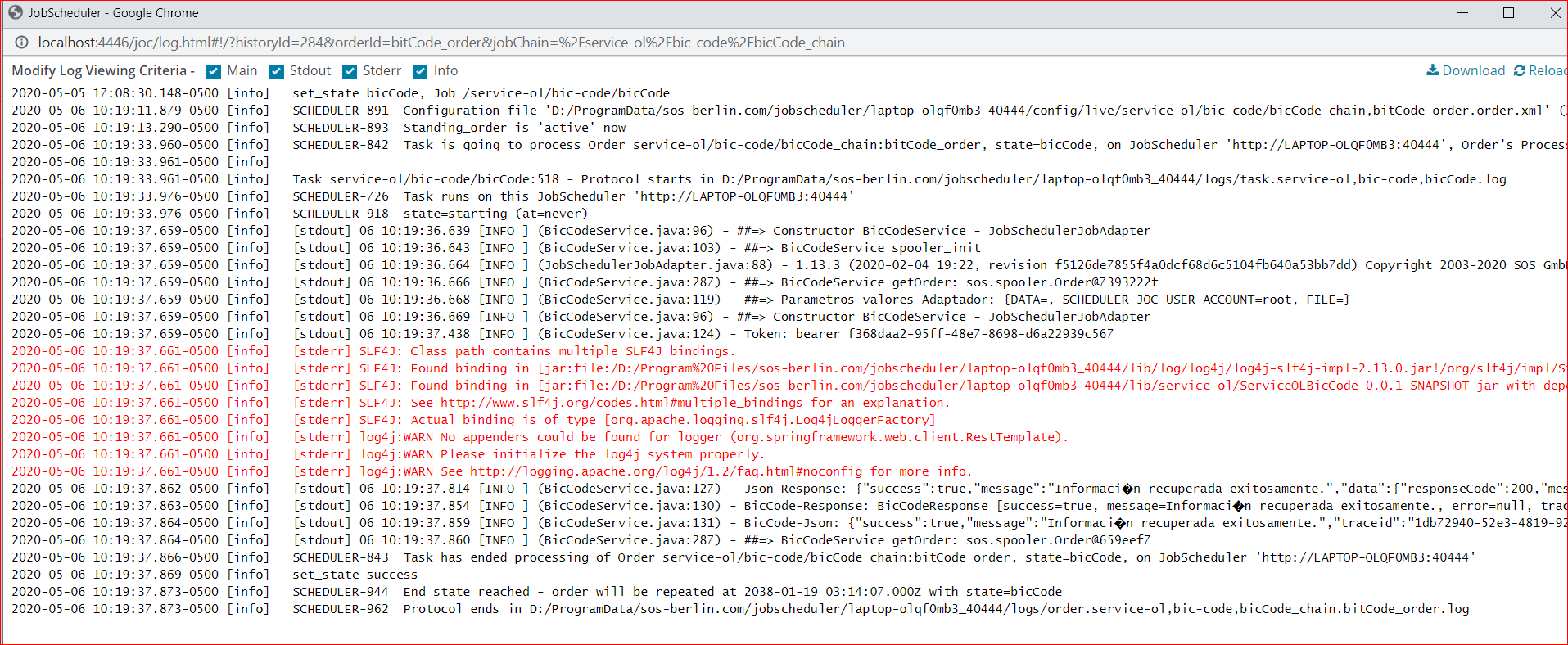


Nota: Observar parámetros que incluye la orden.

* Corre la orden



* Click en Order ID a consultar

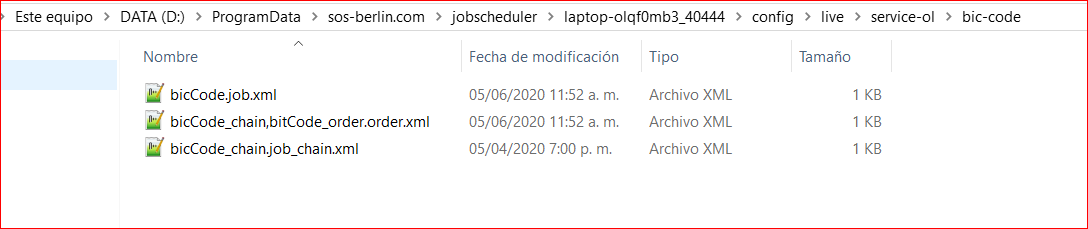


* Copiando y formateando la data de respuesta del servicio BIC Code



# 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>