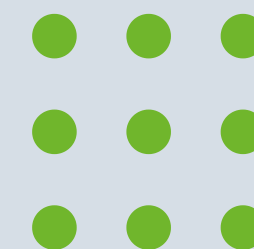


building your ideas

Implementación de Nueva
Recaudación

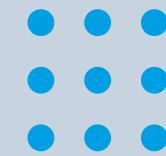




3

+

**Implementación de Nueva
+ Recaudación**





Objetivos

- Implementar Nueva Recaudación





Agenda

- Configuración Recaudación OTC
- Creación de COLAS MQ
- Creación de Proyecto Quakus
- Despliegue y Pruebas





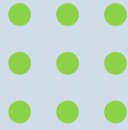
Configuración de Recaudación en OTC





OTC Web


OTC posee un componente de Administración Web que nos permite configurar una Recaudación nueva o modificar una existente. Los pasos son los siguientes:

1. Crear Punto Final
 2. Crear Convenio
 3. Crear Servicio
 4. Crear Flujo
 5. Crear Flujo Servicio Enriquecimiento
 6. Crear Servicio Canal
 7. Crear Tipo Identificador
 8. Crear Datos Adicionales (opcional)
- 



Script de Base de Datos

Otra manera de configurar una recaudación es por medio de un SCRIPT de base de datos, que genere la metadata requerida.





Crear XSLT de entrada

Para recaudaciones de Microservicios se debe hacer enriquecimiento del mensaje Core <http://www.bolivariano.com/mensaje/MensajeOTC> agregando los siguiente campos adicionales:

- e_prov_req (Cola de requerimiento proveedor)
- e_prov_resp (Cola de respuesta proveedor)
- e_tipo_proceso (tipo de proceso, Manual -M o Automatico - A)
- e_tipo_ejecucion (tipo de proceso, Orden de ejecución PB o BP)

El resto de valores se deben pasar como vienen del OTC-CORE





Crear XSLT de Salida

Para recaudaciones de Microservicios se debe hacer validar del mensaje Core
<http://www.bolivariano.com/mensaje/MensajeOTC>.








Creación de QUEUE en IBM MQ



Creación de QUEUE

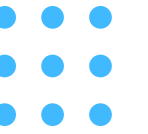
En IBM MQ se deben crear las colas configuradas en los campos:

- e_prov_req (Cola de requerimiento proveedor)
 - e_prov_resp (Cola de respuesta proveedor)
- 

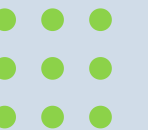


Creación de Proyecto Quarkus

Requisitos




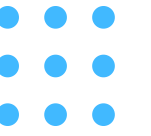
- Git
- Java™ JDK 11
- Un IDE para desarrollo en Java
- Maven 3.6.2 o posterior





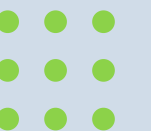
Creación de Proyecto

1. Para crear un proyecto Quarkus podemos usar el inicializador disponible en <https://code.quarkus.io/>
 - a. Group: `com.bolivariano.microservices.*` (depende del nombre de la recaudación, por ejemplo recdemo)
 - b. Artifact: `rec-*` (depende del nombre de la recaudación, por ejemplo rec-demo)
 - c. Los demás valores son por defecto
- 

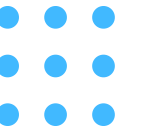


Modificar Proyecto

1. Agregar modelo de Datos: agregar las clases del paquete domain (lo puedes copiar del proyecto rec-pago u otra recaudación)
2. Agregar las siguientes dependencias maven.
3. Agregar parametros de configuración
4. Crear clases `ApplicationProperties` y correspondiente a patron `Request/Responde`

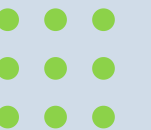


Dependencias Maven

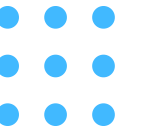


Core

```
<artifactId>quarkus-resteasy</artifactId>  
<artifactId>quarkus-resteasy-jackson</artifactId>  
<groupId>io.quarkus</groupId>  
<artifactId>lombok</artifactId><version>1.18.20</version>  
<artifactId>com.ibm.mq.allclient</artifactId><version>9.2.1.0</version>
```

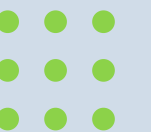


Dependencias Maven

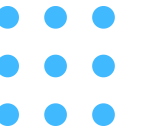


SOAP WEBSERVICE

```
<artifactId>quarkus-cxf</artifactId><version>1.1.0</version>
```

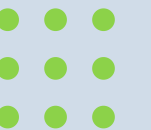


Dependencias Maven

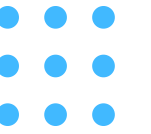


SYBASE

```
<artifactId>jconn4d</artifactId><version>1.2</version>
```

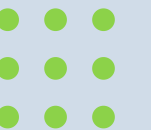


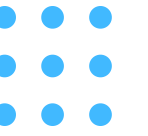
Dependencias Maven



ISO 8583

```
<artifactId>jpos</artifactId><version>2.1.7</version>
```





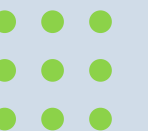
Parametros de Configuración

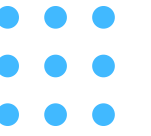
SOAP WEBSERVICES

quarkus.cxf.client."**NOMBRE SERVICIOS**".service-interface
quarkus.cxf.client."**NOMBRE SERVICIOS**.client-endpoint-url

NOMBRE PROYECTO.provider.connTimeout

NOMBRE PROYECTO.provider.connRequestTimeout



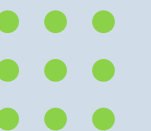


Parametros de Configuración

MQ

NOMBRE PROYECTO.mq.host
NOMBRE PROYECTO.mq.port
NOMBRE PROYECTO.mq.channel
NOMBRE PROYECTO.mq.queueManager
NOMBRE PROYECTO.mq.requestQueue
NOMBRE PROYECTO.mq.responseQueue

NOMBRE PROYECTO.mq.poolJms
NOMBRE PROYECTO.mq.delayReconnect
NOMBRE PROYECTO.mq.timeToLive
NOMBRE PROYECTO.mq.characterSet





Parámetros de Configuración

JPOS

NOMBRE PROYECTO.tcp.host

NOMBRE PROYECTO.tcp.port

NOMBRE PROYECTO.tcp.timeout



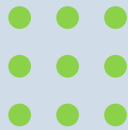


ApplicationProperties

Crear clase ApplicationProperties y agregar interfaces internas que corresponde a las propiedades definidas.

Anotar la interfaz con las siguiente propiedades:

```
@StaticInitSafe
@Configuration(prefix = "NOMBRE PROYECTO", namingStrategy =
NamingStrategy.VERBATIM)
@Named("applicationConfig")
```





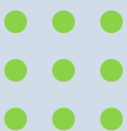
MQConnectionFactoryData

@Qualifier

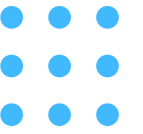
@Retention(RetentionPolicy.RUNTIME)

@Target({TYPE, METHOD, FIELD, PARAMETER})

```
public @interface MQConnectionFactoryData {  
}
```



MQConfiguration

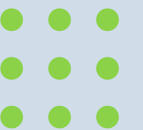


```
@ApplicationScoped
public class MQConfiguration {

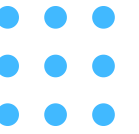
    @Inject
    ApplicationProperties applicationProperties;

    @Produces
    @MQConnectionFactoryData
    public MQConnectionFactory getConnectionFactory() throws JMSEException {
        MQQueueConnectionFactory connectionFactory = new MQQueueConnectionFactory();
        connectionFactory.setHostName(applicationProperties.mq().host());
        connectionFactory.setQueueManager(applicationProperties.mq().queueManager());
        connectionFactory.setPort(applicationProperties.mq().port());
        connectionFactory.setChannel(applicationProperties.mq().channel());
        connectionFactory.setTransportType(CommonConstants.WMQ_CM_CLIENT);
        connectionFactory.setIntProperty(CommonConstants.WMQ_CLIENT_RECONNECT_OPTIONS,
            CommonConstants.WMQ_CLIENT_RECONNECT);

        return connectionFactory;
    }
}
```



MessageListener



```
ApplicationScoped  
public class MessageListener implements Runnable {
```

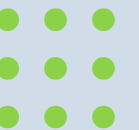
```
    @Inject  
    Logger log;
```

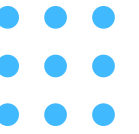
```
    @Inject  
    @MQConnectionFactoryData  
    ConnectionFactory connectionFactory;
```

```
    @Inject  
    ApplicationProperties applicationProperties;
```

```
    @Inject  
    ProcessorService processorService;
```

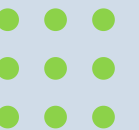
```
    private ExecutorService scheduler;  
}
```

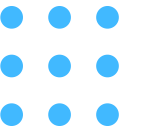




MessageListener

```
void onStart(@Observes StartupEvent event) {  
    int maxThread = applicationProperties.mq().poolJms().intValue();  
    scheduler = Executors.newFixedThreadPool(maxThread);  
    for (int i = 0; i < maxThread; i++) {  
        scheduler.submit(this);  
    }  
}  
  
void onStop(@Observes ShutdownEvent event) {  
    scheduler.shutdown();  
}  
}
```





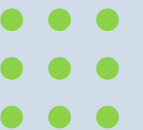
MessageListener

```
@Override
public void run() {
    log.info("Inicia thread listener");
    log.info("Config --> " + connectionFactory);
    log.info("Cola mq --> " + applicationProperties.mq().requestQueue());

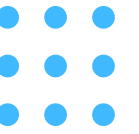
    JMSConsumer consumer = null;
    try (JMSContext context = connectionFactory.createContext(Session.AUTO_ACKNOWLEDGE)) {
        Queue queue = context.createQueue(applicationProperties.mq().requestQueue());
        consumer = context.createConsumer(queue);
        while (true) {
            Message message = consumer.receive();
            log.info("JMS --> " + message);
            if (Objects.isNull(message)) return;

            String correlationId = message.getJMSCorrelationID();
            String mensajeJMS = message.getBody(String.class);

            log.infof("%s Mensaje recibido: %s ", correlationId, mensajeJMS);
            processorService.processMessage(mensajeJMS, correlationId);
        }
    } catch (Exception e) {
        log.error("Error en consumo de servicio JMS -->", e);
    } finally {
        if (consumer != null) {
            consumer.close();
        }
    }
    Runnable runnableTask = () -> {
        try {
            log.info("THREAD DELAY --> " + applicationProperties.mq().delayReconnect());
            TimeUnit.MILLISECONDS.sleep(applicationProperties.mq().delayReconnect());
        } catch (InterruptedException e) {
            log.error("Error en setear DELAY RECONNECT --> ", e);
            Thread.currentThread().interrupt();
        }
    };
    new Thread(runnableTask).start();
    scheduler.submit(this);
}
}
```



MessageProducer



```
@ApplicationScoped
public class MessageProducer {

    @Inject
    Logger log;

    @MQConnectionFactoryData
    ConnectionFactory mqConnectionFactory;

    @Inject
    ApplicationProperties applicationProperties;

    public void sendMessage(String mensajeStr, final String correlationId){
        log.infof("Enviando mensaje de respuesta a la cola --> %s", applicationProperties.mq().responseQueue());
        log.infof("Mensaje --> %s", mensajeStr);
        try {final JMSContext context = mqConnectionFactory.createContext(Session.AUTO_ACKNOWLEDGE)} {
            final JMSProducer producer = context.createProducer();
            final Destination destination = context.createQueue(applicationProperties.mq().responseQueue());

            TextMessage messageJMS = context.createTextMessage();
            messageJMS.setJMSCorrelationID(correlationId);
            messageJMS.setJMSMessageID(correlationId);

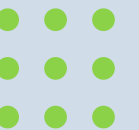
            producer.setTimeToLive(applicationProperties.mq().timeToLive());
            messageJMS.setIntProperty(JMS_IBM_MQMD_EXPIRY, applicationProperties.mq().timeToLive().intValue());
            messageJMS.setJMSExpiration(applicationProperties.mq().timeToLive());

            messageJMS.setIntProperty(JMS_IBM_CHARACTER_SET, applicationProperties.mq().characterSet());
            messageJMS.setJMSDeliveryMode(DeliveryMode.NON_PERSISTENT);
            messageJMS.setJMSPriority(Message.DEFAULT_PRIORITY);
            messageJMS.setJMSTimestamp(System.nanoTime());

            messageJMS.setText(mensajeStr);

            log.infof("MensajeJMS --> %s", messageJMS);
            log.infof("CorrelationId --> %s", correlationId);

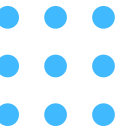
            producer.send(destination, messageJMS);
        } catch (Exception e) {
            log.error("Error en producir de servicio JMS", e);
            throw new IllegalArgumentException("Error al consultar deuda ANT: " + e.getMessage());
        }
    }
}
```



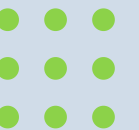


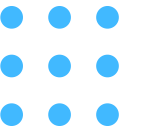
Despliegue en Openshift

Requisitos



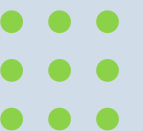
- Debe estar creado el proyecto en Openshift y tener acceso
- El proyecto debe estar subido al Gitlab del Banco

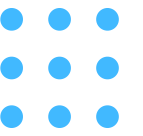




Despliegue

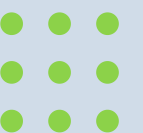
1. Login: ``oc login https://api.ocpmsbbdevqa.bolivariano.fin.ec:6443 -u cromeror -p *****``
2. Usar proyecto ``oc project bb-dev-medpag``
3. Añadir credenciales de gitlab: ``oc create secret generic git-secret --from-literal="username=ocp-reader" --from-literal="password=op3nsh1ft`` [NO NECESARIO, EL PROYECTO YA LO TIENE CREADO]
4. Crear config map: ``oc create cm tarjetacorporativa-application-properties --from-file="application.properties" --dry-run -o yaml | oc apply -f -``

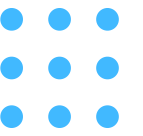




Despliegue

5. Crear Identity JKS: ``oc create secret generic identity-jks --from-file=identity.jks=identity.jk`` (NO NECESARIO, EL PROYECTO YA LO TIENE CREADO)
6. Crear Trustore JKS: ``oc create secret generic truststore-jks --from-file=truststore.jks=truststore.jk`` (NO NECESARIO, EL PROYECTO YA LO TIENE CREADO)
7. Crear aplicación: ``oc new-app --as-deployment-config --build-env MAVEN_MIRROR_URL=https://nexus.apps.ocpmsbbdevqa.bolivariano.fin.ec/repository/maven-public --build-env MAVEN_OPTS=-Dmaven.wagon.http.ssl.insecure=true --name tarjetacorporativa -i bb-images/bb-openjdk-11 -e TZ=America/Guayaquil -e LOGGING_LEVEL_ROOT=INFO --source-secret=git-secret https://gitlab.apps.ocpmsbbdevqa.bolivariano.fin.ec/activos/atc/tarjetacorporativa#dev``



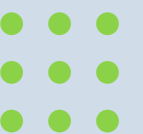


Despliegue

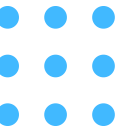
8. Crear volumen para properties: ``oc set volume dc/tarjetacorporativa --add --name=application-properties --type configmap --configmap-name=tarjetacorporativa-application-properties --mount-path=/deployments/config/application.properties --sub-path=application.properties``

9. Crear volumen para identity.jks ``oc set volume dc/tarjetacorporativa --add --name=identity-jks --type secret --secret-name=identity-jks --mount-path=/secret/identity.jks --sub-path=identity.jks``

10. Crear volumen para truststore.jks: ``oc set volume dc/tarjetacorporativa --add --name=truststore-jks --type secret --secret-name=truststore-jks --mount-path=/secret/truststore.jks --sub-path=truststore.jks``



Despliegue



11. Crear ruta: ``oc create route passthrough --service tarjetacorporativa --port 8443``
12. Establecer recursos: ``oc set resources dc/tarjetacorporativa --limits=cpu=2,memory=1Gi --requests=cpu=1,memory=512Mi``
13. Añadir host alises en en caso de ser necesario en el deployment config (template.spec.hostAliases):

```
```yaml
```

```
hostAliases:
```

```
- ip: 172.16.11.16
```

```
 hostnames:
```

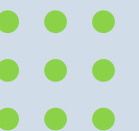
```
 - jms.bolivariano.fin.ec
```

```
- ip: 172.16.30.209
```

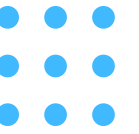
```
 hostnames:
```

```
 - frameworkdes.bbframework.com
```

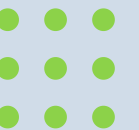
```
```
```



Pruebas



- Pruebas desde SOAP UI para servicio WEB
- Pruebas desde MQ para Cobis
- Pruebas desde Postman para servicio REST
- Pruebas integrales desde canal





¡Gracias!

Búscanos como:

Visita nuestra web:

Correo:

Teléfono:

@gizlosoftware

www.gizlocorp.com

info@gizlocorp.com

+1(207)900-8663

