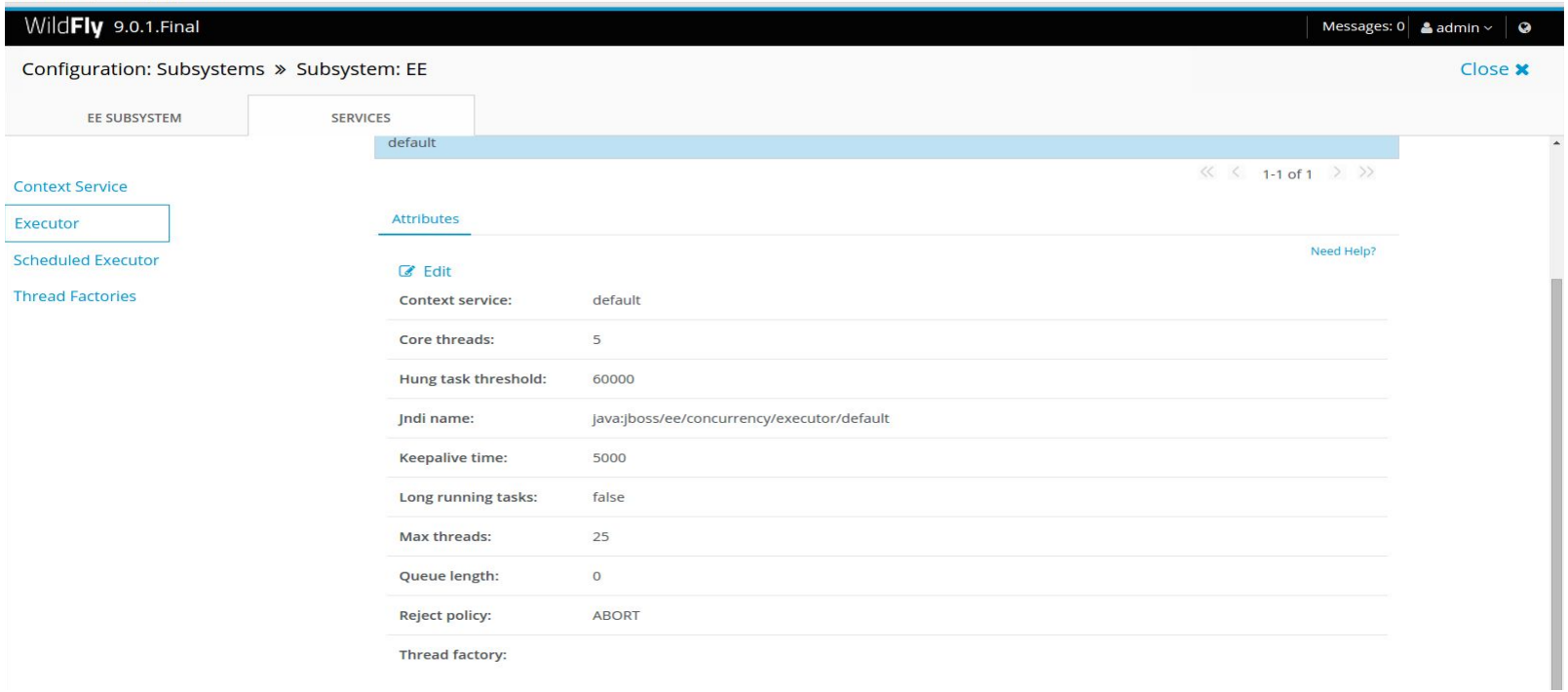
Estudo de caso Gate2all sob a arquitetura JEE

CDI

JMS 2.0

EJB 3

Maior controle no uso do poll de threads



Modelo de uso com JEE

GatewayServlet.java

gateway-roteador ▸ src/main/java ▸ br.com.ntk.servlet ▸ GatewayServlet ▸ getProcessor() : Processor

```
1  package br.com.ntk.servlet;
2
3  import java.util.concurrent.Executor;
13
14 public abstract class GatewayServlet extends HttpServlet {
15     private static final long serialVersionUID = -4294404416746373387L;
16
17     @Resource(lookup = "java:jboss/ee/concurrency/executor/default")
18     private Executor executor;
19
20     @Inject
21     private RequestProcessor requestProcessor;
22
23     /** @see Processor  **/
24     protected abstract Processor getProcessor();
25
26     /**
27      * Utilizado caso seja necessario obter parametros da request, para possiveis
28      * tratamento do metodo getProcessor()
29      **/
30     protected HttpServletRequest httpServletRequest;
31
32     public void doPost(final HttpServletRequest request, final HttpServletResponse response) {
33         request.getSession().invalidate();
34         final AsyncContext asyncContext = request.startAsync();
35         asyncContext.setTimeout(100 * 1000);
36         asyncContext.addListener(new CallbackAsyncListener());
37         httpServletRequest = request;
38         Processor processor = getProcessor();
39         processor.setSessionID(request.getSession().getId());
40         processor.setAsyncContext(asyncContext);
41         this.requestProcessor.setProcessor(processor);
42         executor.execute(this.requestProcessor);
43     }
44
45 }
```

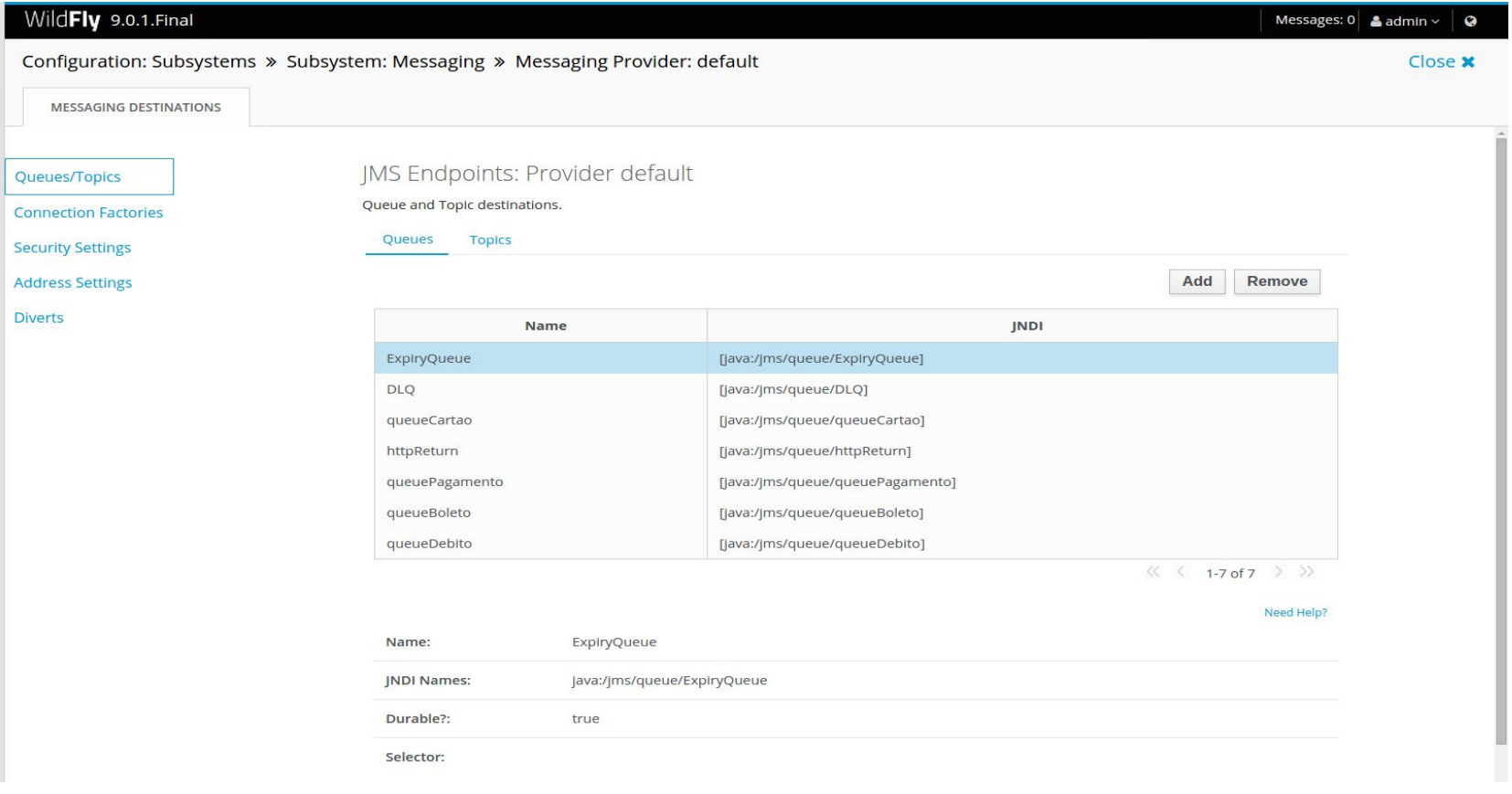
Modelo de uso atual

```java
@WebListener
public class InitializeServletListener implements ServletContextListener {

  public void contextInitialized(ServletContextEvent sce) {

    Executor executor = new ThreadPoolExecutor(20, 50, 5, TimeUnit.MINUTES,
        new LinkedBlockingDeque<Runnable>());

    sce.getServletContext().setAttribute("executor", executor);

        System.out.println("======== LoadSpringContext Iniciado ========");
        LoadSpringContext.getInstance().getContext();
    }

  public void contextDestroyed(ServletContextEvent sce) {

    DefaultMessageListenerContainer defaultMessageListenerContainer = (DefaultMessageListenerContainer)
          LoadSpringContext.getInstance().getContext().getBean("jmsContainer");
    //encerra as threads do JMS gerenciada pelo spring
    defaultMessageListenerContainer.shutdown();

    ProducerMessage producer = (ProducerMessage) LoadSpringContext.getInstance().getContext().getBean("producerMessage");
    //fecha a conexao e a sessao com o Hornetq
    producer.close();

    ThreadPoolExecutor executor = (ThreadPoolExecutor) sce.getServletContext().getAttribute("executor");
    //encerra o poll de threads do Executor, responsavel pelo gerenciamento assincrono das servlets
    executor.shutdown();
    sce.getServletContext().removeAttribute("executor");

  }

}
```

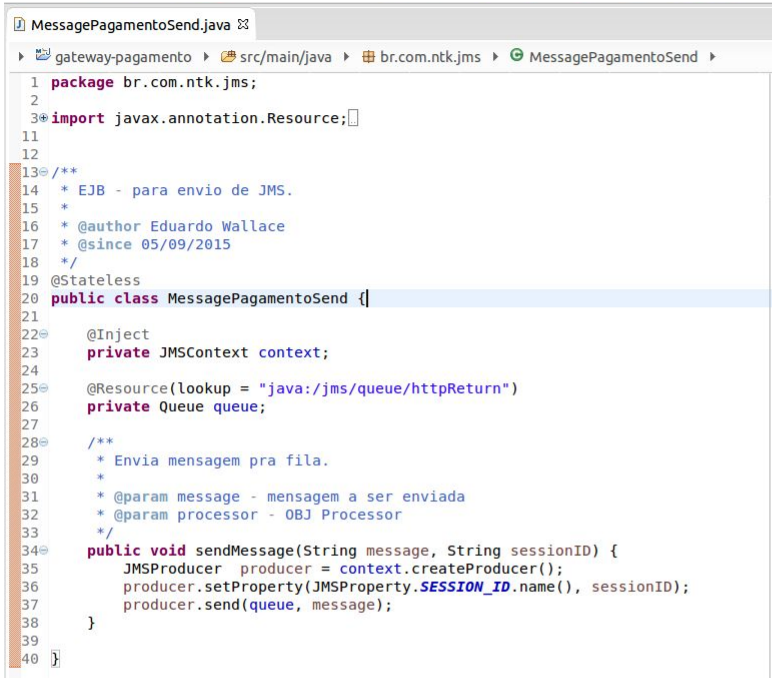
Modelo de uso atual

```java
public abstract class GatewayServlet extends HttpServlet {
  private static final long serialVersionUID = -4294404416746373387L;

  /** @see Processor  **/
  protected abstract Processor getProcessor();

  /**
   * Utilizado caso seja necessario obter parametros da request, para possiveis
   * tratamento do metodo getProcessor()
   **/
  protected HttpServletRequest httpServletRequest;

  public void doPost(final HttpServletRequest request, final HttpServletResponse response) {
    request.getSession().invalidate();

    final AsyncContext asyncContext = request.startAsync();
    asyncContext.setTimeout(100 * 1000);
    asyncContext.addListener(new CallbackAsyncListener());
    httpServletRequest = request;
    if(StringUtils.isBlank(request.getParameter("xml"))
        && !request.getServletPath().contains("notificacao") ) {
      StringBuilder sb = new StringBuilder();
      sb.append("<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>")
        .append("<erro>")
        .append("<codigo>412</codigo>")
        .append("<mensagem>XML nao informado</mensagem>")
        .append("</erro>");
      request.setAttribute("xml", sb.toString());
      asyncContext.complete();
      return;
    }
    Executor executor = (Executor) request.getServletContext().getAttribute("executor");
    Processor processor = getProcessor();
    processor.setSessionID(request.getSession().getId());
    processor.setAsyncContext(asyncContext);
    executor.execute(new RequestProcessor(processor));
  }

}
```

Facilidade ao usar JMS



Modelo de uso com JEE - envio



MessagePagamentoSend.java

gateway-pagamento ▸ src/main/java ▸ br.com.ntk.jms ▸ MessagePagamentoSend ▸

```java
package br.com.ntk.jms;

import javax.annotation.Resource;


/**
 * EJB - para envio de JMS.
 *
 * @author Eduardo Wallace
 * @since 05/09/2015
 */
@Stateless
public class MessagePagamentoSend {

    @Inject
    private JMSContext context;

    @Resource(lookup = "java:/jms/queue/httpReturn")
    private Queue queue;

    /**
     * Envia mensagem pra fila.
     *
     * @param message - mensagem a ser enviada
     * @param processor - OBJ Processor
     */
    public void sendMessage(String message, String sessionID) {
        JMSProducer  producer = context.createProducer();
        producer.setProperty(JMSProperty.SESSION_ID.name(), sessionID);
        producer.send(queue, message);
    }

}
```

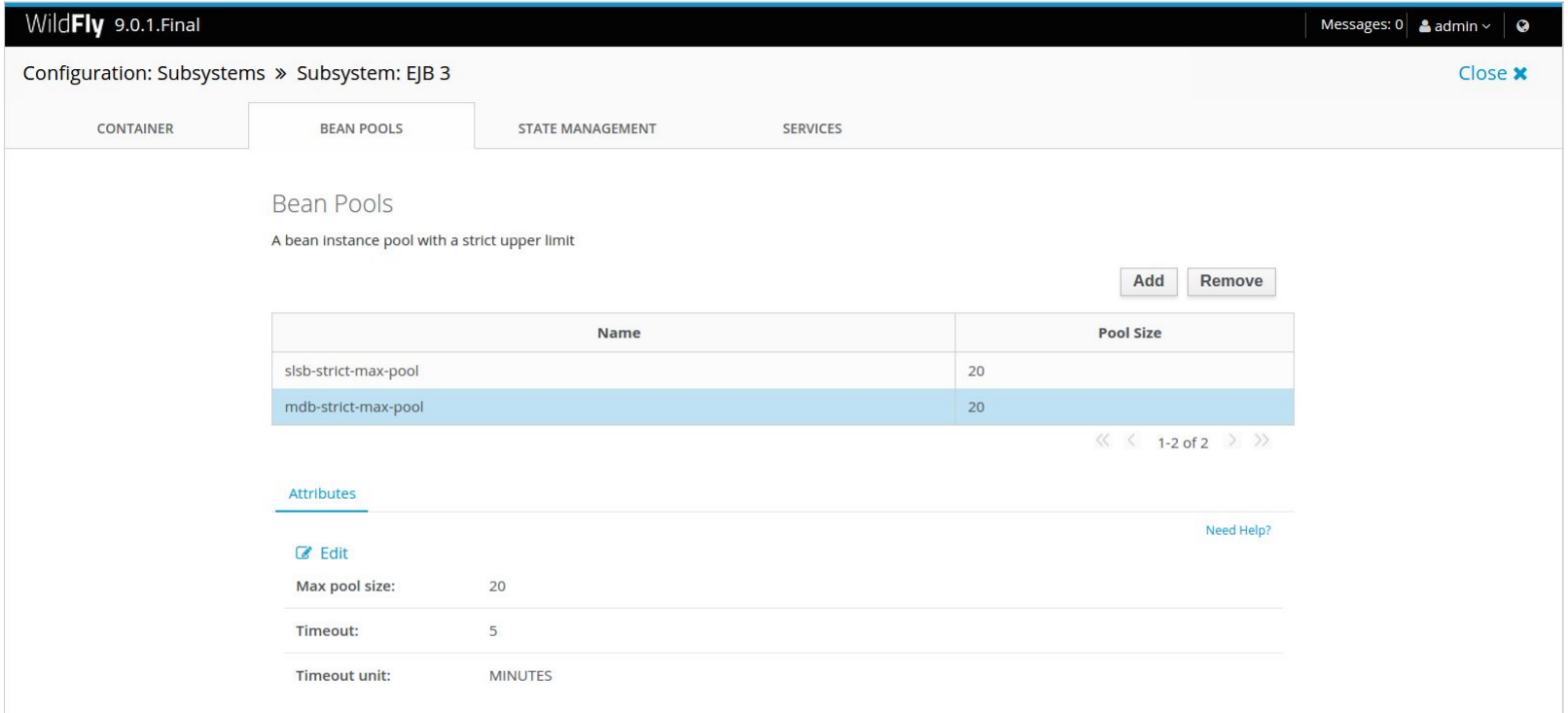
Modelo de uso atual

hornetq-jms.xml

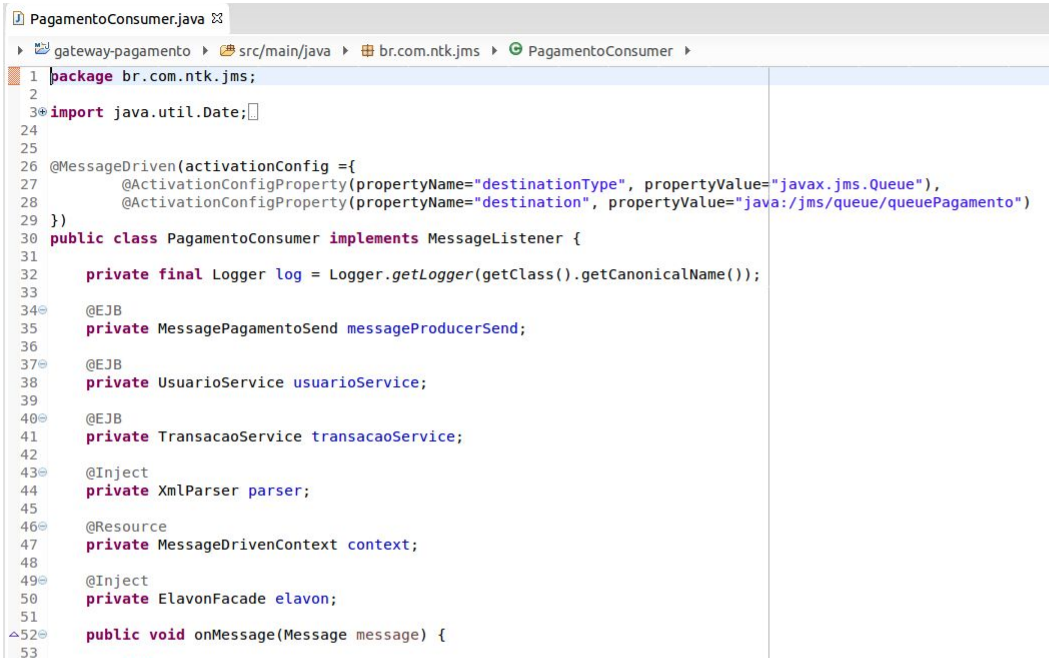
```
            <entry name="/queue/Router_1"/>
    </queue>
    <queue name="Pagamento_1">
            <entry name="/queue/Pagamento_1"/>
    </queue>
-->


    <queue name="httpReturn">
    <entry name="/queue/httpReturn"/>
    </queue>
<queue name="queueLog">
            <entry name="/queue/queueLog"/>
    </queue>
<queue name="queuePagamento">
    <entry name="/queue/queuePagamento"/>
</queue>
<queue name="queueCartao">
    <entry name="/queue/queueCartao"/>
</queue>
<queue name="queueBoleto">
    <entry name="/queue/queueBoleto"/>
</queue>
<queue name="queueDebito">
    <entry name="/queue/queueDebito"/>
</queue>
<queue name="queueSubAquirer">
    <entry name="/queue/queueSubAquirer"/>
</queue>
<queue name="queueCheque">
  <entry name="/queue/queueCheque"/>
</queue>
<queue name="queueAntifraude">
    <entry name="/queue/queueAntifraude"/>
</queue>
```

Modelo de uso atual - envio

```java
@Component
public class MessageProducerSend {

  @Autowired
  private ConnectionFactory connectionFactoryHornetQ;

  private Connection connection;
  private MessageProducer producer;
  private Session session;

  /**
   * Setter ConnectionFactory bean TransacaoDebitoService
   * @param connectionFactory injection container spring
   */
  public void setConnectionFactoryHornetQ(
      ConnectionFactory connectionFactoryHornetQ) {
    this.connectionFactoryHornetQ = connectionFactoryHornetQ;
  }

  private void prepareSender() throws JMSException {
    if (connection == null) {
      connection = connectionFactoryHornetQ.createConnection();
    }
    if (session == null) {
      session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
    }
    if (producer == null) {
      producer = session.createProducer(null);
    }
  }

  private void setMessageProperties(Message message, Map<String, Object> parameters) throws JMSException {
    //MORE CODE
  }

  public void createAndSend(String message, Map<String, Object> parameters, String queue) throws JMSException {
    prepareSender();
    TextMessage textMessage = session.createTextMessage();
    setMessageProperties(textMessage, parameters);
    textMessage.setText(message);
    if (StringUtils.isEmpty(queue)) {
      throw new JMSException("nao preciso informar o nome da fila");
    }
    producer.send(session.createQueue(queue), textMessage);
  }

}
```


WildFly 9.0.1.Final
Messages: 0
admin
Configuration: Subsystems » Subsystem: EJB 3
Close
CONTAINER
BEAN POOLS
STATE MANAGEMENT
SERVICES
Bean Pools
A bean instance pool with a strict upper limit
Add
Remove
Name
Pool Size
slsb-strict-max-pool
20
mdb-strict-max-pool
20
1-2 of 2
Attributes
Need Help?
Edit
Max pool size:
20
Timeout:
5
Timeout unit:
MINUTES

Modelo de uso com JEE - consumo

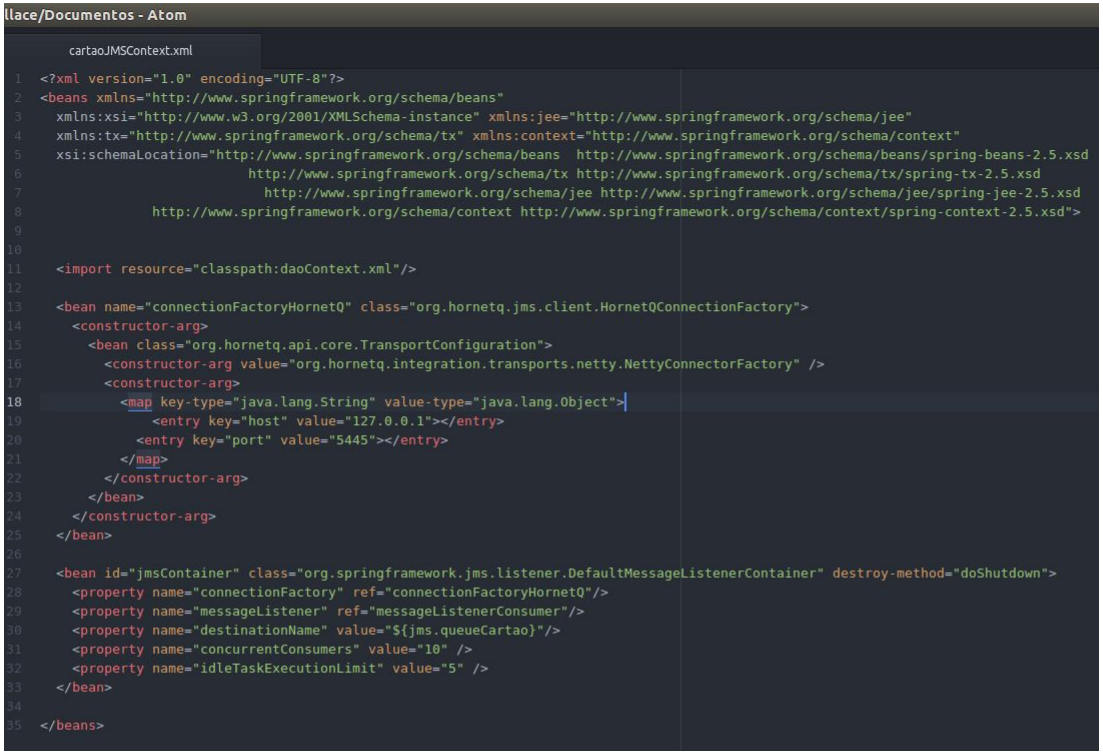


PagamentoConsumer.java

gateway-pagamento ▸ src/main/java ▸ br.com.ntk.jms ▸ PagamentoConsumer ▸

```
1  package br.com.ntk.jms;
2
3  import java.util.Date;
24
25
26 @MessageDriven(activationConfig ={
27         @ActivationConfigProperty(propertyName="destinationType", propertyValue="javax.jms.Queue"),
28         @ActivationConfigProperty(propertyName="destination", propertyValue="java:/jms/queue/queuePagamento")
29 })
30 public class PagamentoConsumer implements MessageListener {
31
32     private final Logger log = Logger.getLogger(getClass().getCanonicalName());
33
34     @EJB
35     private MessagePagamentoSend messageProducerSend;
36
37     @EJB
38     private UsuarioService usuarioService;
39
40     @EJB
41     private TransacaoService transacaoService;
42
43     @Inject
44     private XmlParser parser;
45
46     @Resource
47     private MessageDrivenContext context;
48
49     @Inject
50     private ElavonFacade elavon;
51
52     public void onMessage(Message message) {
53
```

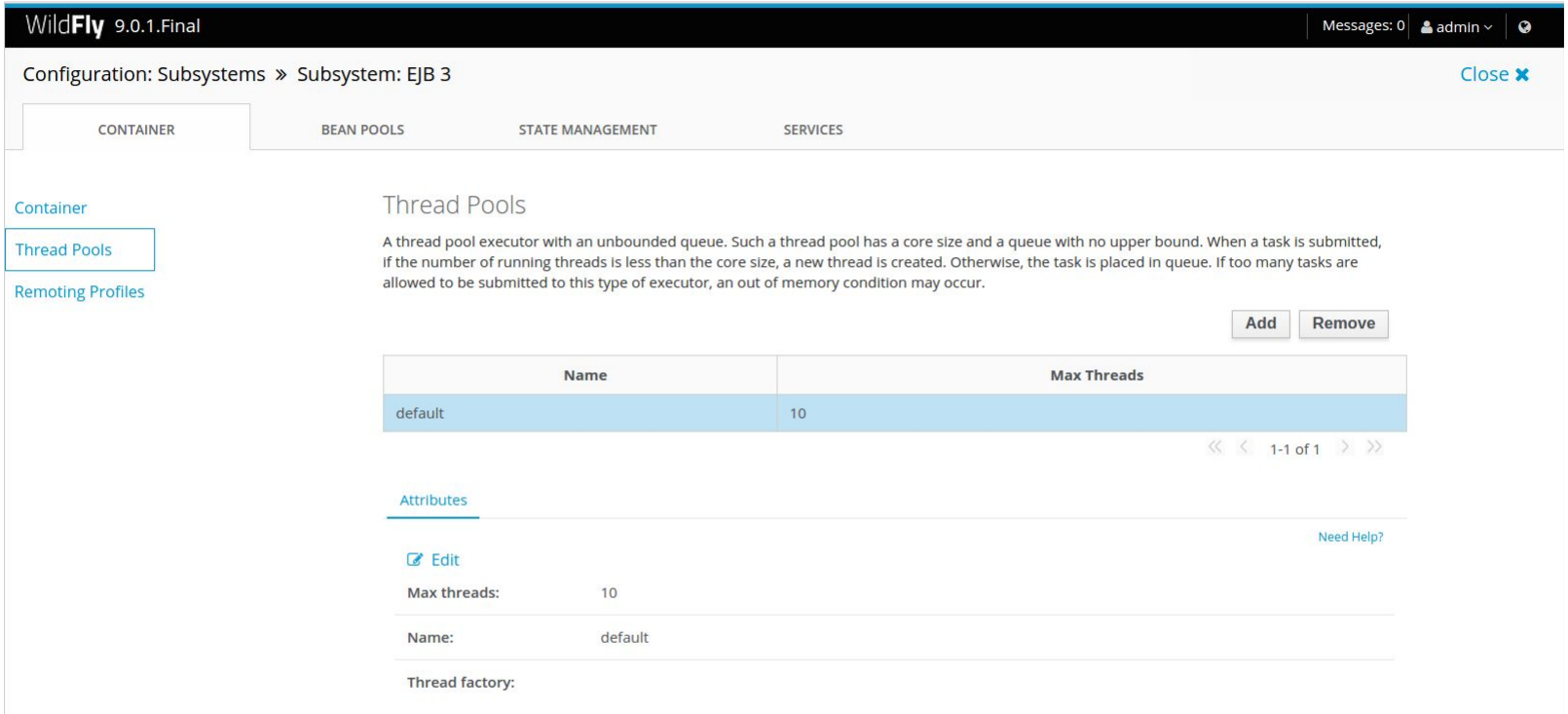
Modelo de uso atual



Modelo de uso atual - consumo

```
@Component
public class MessageListenerConsumer implements MessageListener {

  private static Logger LOGGER = LogManager.getLogger(MessageListenerConsumer.class);

  @Autowired
  TransacaoCartaoService transacaoCartaoService;

  @Autowired
  MessageProducerSend producer;

  public void onMessage(Message message) {
```

Facilidade em usar Factory



Modelo de uso com JEE

CallbackContext.java

gateway-roteador ▸ src/main/java ▸ br.com.ntk.listener ▸ CallbackContext ▸ init() : void

```java
/**
 * EJB Singleton, utilizado para controlar o armazenamento e remocao dos Callbacks.
 *
 * @author Eduardo Wallace
 * @since 05/09/2015
 */
@Startup
@Singleton
@ConcurrencyManagement(ConcurrencyManagementType.CONTAINER)
public class CallbackContext {

    private final Logger log = Logger.getLogger(getClass().getCanonicalName());

    private ConcurrentHashMap<String, AsyncContext> mapRequests;

    @PostConstruct
    private void init() {
        log.info("Singleton EJB CallbackContext - Iniciado!");
        mapRequests = new ConcurrentHashMap<String, AsyncContext>();
    }

    /** Armazena o  AsyncContext **/
    public void put(String sessionID, AsyncContext asyncContext) {
        mapRequests.put(sessionID, asyncContext);
    }

    /** Remove o AsyncContext **/
    public void remove(String sessionID) {
        mapRequests.remove(sessionID);
    }

    /**
     * Conclui o processamento da transacao do gateway chamando o Callback
     *
     * @param xml - contem os dados da transacao
     * @param sessionID - id da sessao
     */
    @Lock(LockType.WRITE)
    public void complete(String xml, String sessionID) {
        if (mapRequests.containsKey(sessionID)) {
            AsyncContext asyncContext = (AsyncContext) mapRequests.get(sessionID);
            mapRequests.remove(sessionID);
            asyncContext.getRequest().setAttribute("xml", xml);
            asyncContext.complete();
        } else {
            log.warning("Timeout " + xml);
        }
    }

}
```

Modelo de uso atual

StartCartao.java

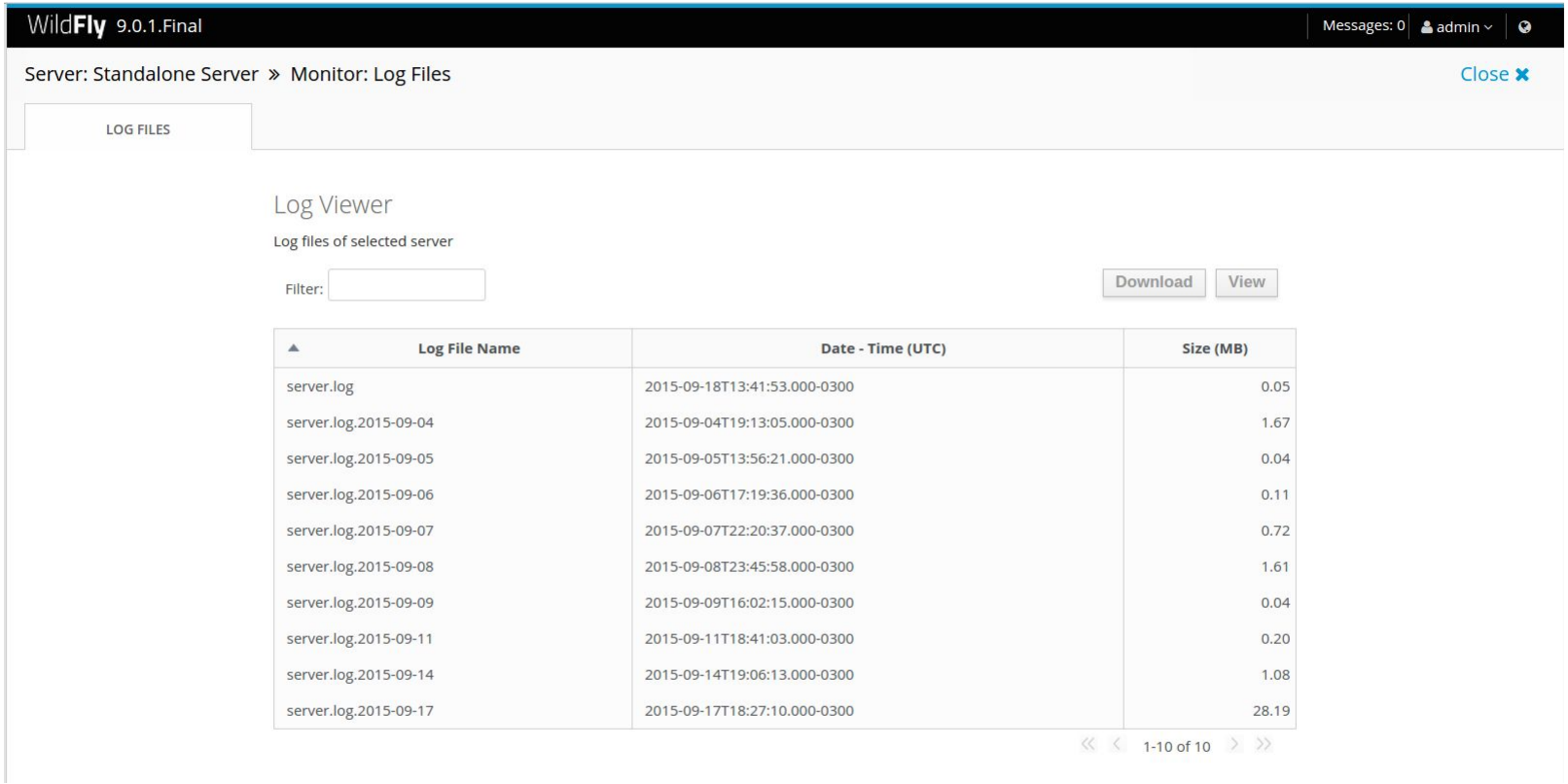
```java
package br.com.gateway.cartao.util;



/**
 *
 * @author Luis Requejo
 * @author NTK Solutions
 *
 */
public class StartCartao {

  public static void main(String[] args) {
    LoadSpringContext.getInstance().getContext();
    try {
      XmlParse.getInstance();
    } catch (Exception e) {
      System.out.println("Erro na criacao dos contextos JAXB");
      System.exit(0);
    }
    System.out.println("Modulo Cartao Iniciado");
  }
}

```

Modelo de uso atual

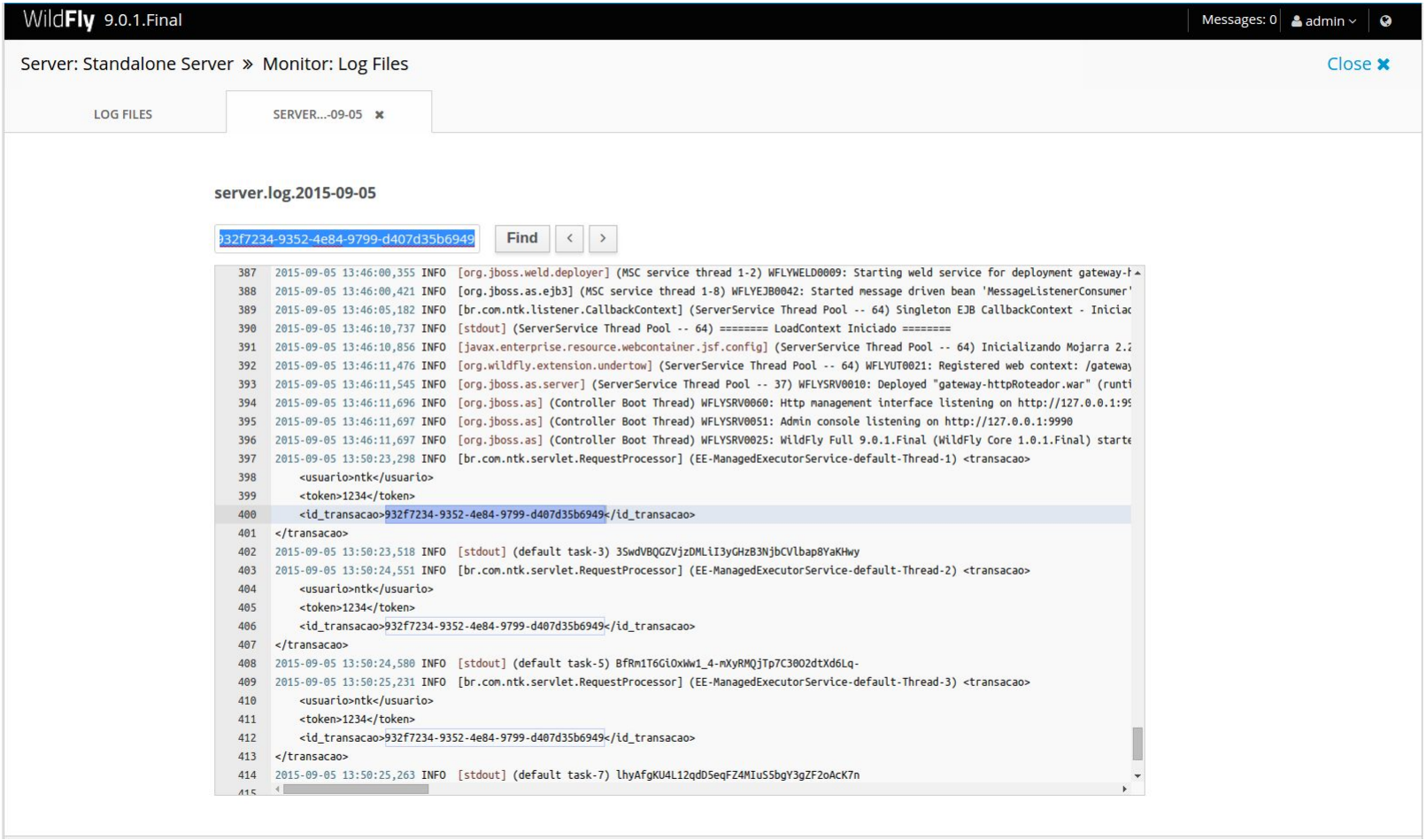
LoadSpringContext.java

```java
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LoadSpringContext {

  private static LoadSpringContext springContext = null;
  private static final ConcurrentHashMap<String, AsyncContext> mapRequests;

  private final ApplicationContext context;

  private LoadSpringContext() {
    context = new ClassPathXmlApplicationContext("jmsContext.xml");
  }

  public static LoadSpringContext getInstance(){
    if (springContext == null) {
      springContext = new LoadSpringContext();
    }
    return springContext;
  }

  public ApplicationContext getContext() {
    return context;
  }

  static{
    mapRequests = new ConcurrentHashMap<String, AsyncContext>();
  }

  public ConcurrentHashMap<String, AsyncContext> getMapRequests() {
    return mapRequests;
  }

  /**
   * Conclui o processamento da transacao do gateway
   *
   * @param xml - contem os dados da transacao
   * @param sessionID - id da sessao
   */
  public void complete(String xml, String sessionID) {
    if(mapRequests.containsKey(sessionID)){
      AsyncContext asyncContext = (AsyncContext) mapRequests.get(sessionID);
      synchronized (asyncContext) {
        mapRequests.remove(sessionID);
        asyncContext.getRequest().setAttribute("xml", xml);
        asyncContext.complete();
      }
    }else{
      System.out.println("time out "+xml);
    }
  }
```

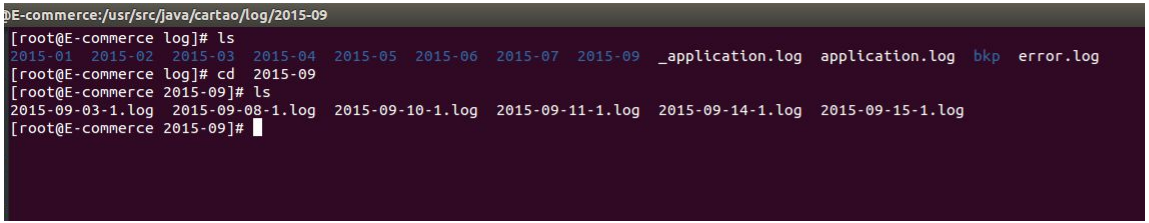
Gerenciamento dos Logs



Gerenciamento dos Logs



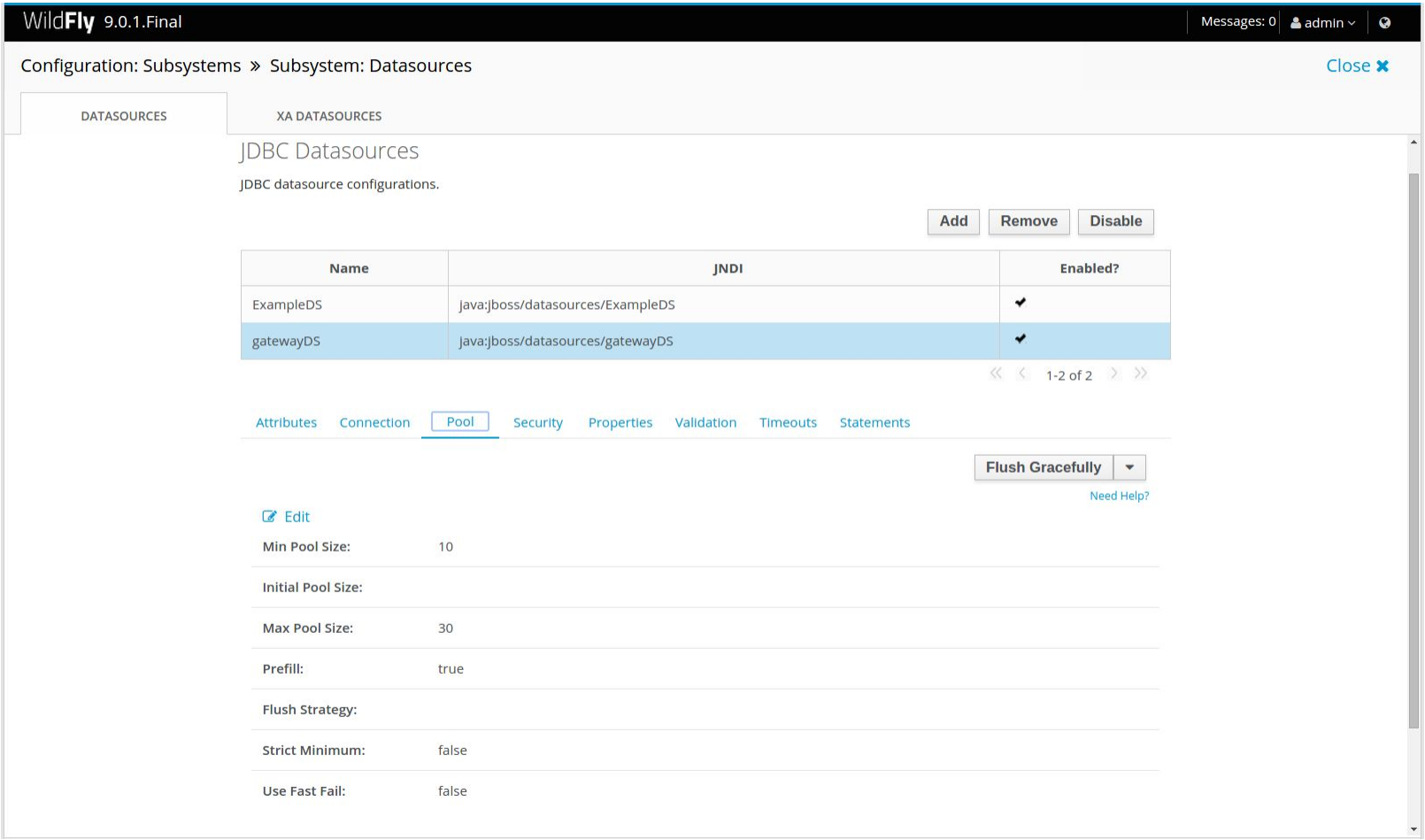
Modelo atual dos Logs



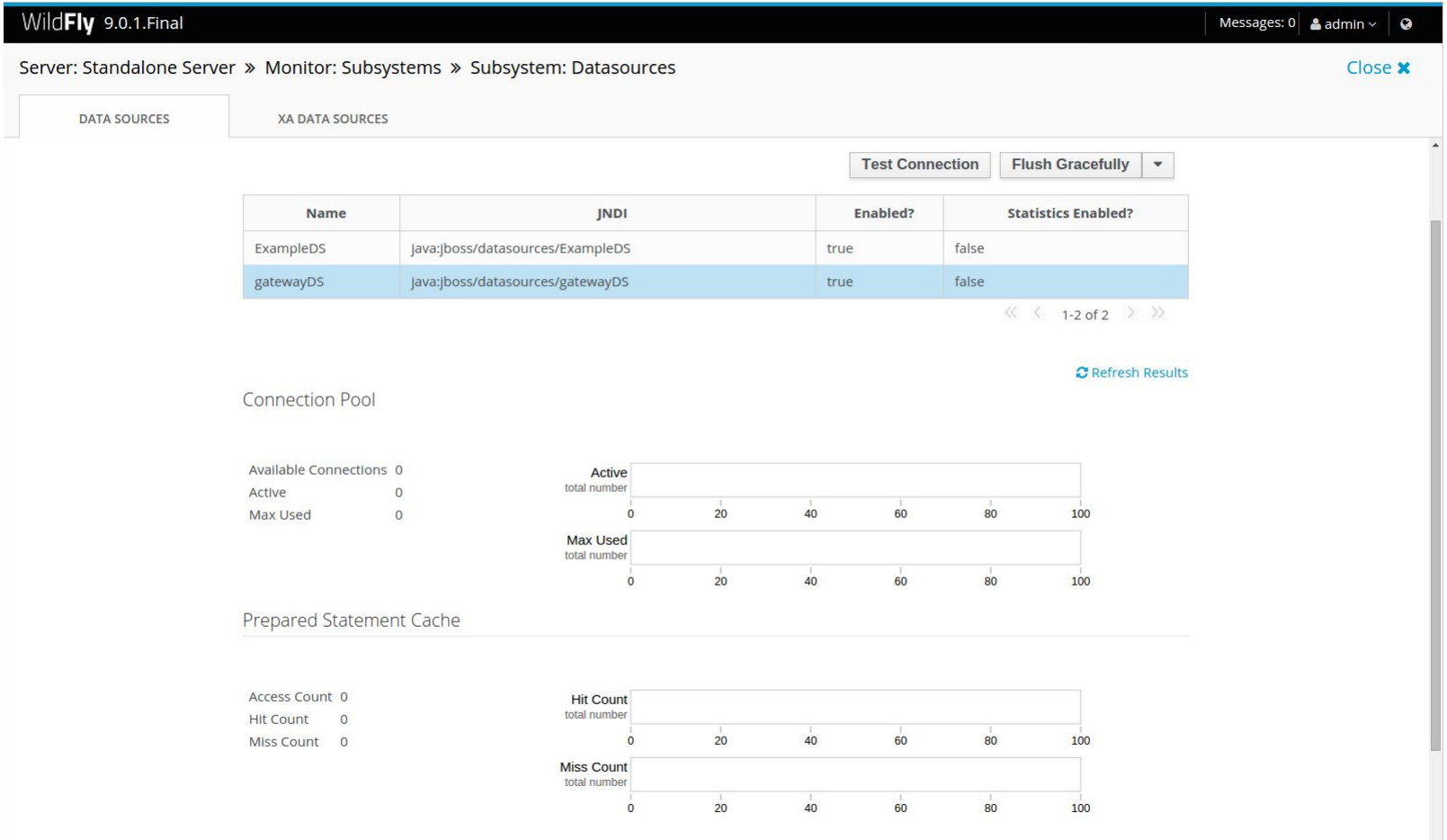
Modelo atual dos Logs

```
18/09/2015 13:46:00.006 | DEBUG | b.c.n.u.EmailUtil:50   | Destinatarios: [eduardo.silva@ntk-consult.com.br]
com.amazonaws.AmazonServiceException: 1 validation error detected: Value null at 'source' failed to satisfy constraint: Member must not be null (Service: AmazonSimpleEmailService; Status Code: 400; Error Cod
e: ValidationError; Request ID: c0ee89c4-5e24-11e5-be3e-8b07a1f570e6)
        at com.amazonaws.http.AmazonHttpClient.handleErrorResponse(AmazonHttpClient.java:1160)
        at com.amazonaws.http.AmazonHttpClient.executeOneRequest(AmazonHttpClient.java:748)
        at com.amazonaws.http.AmazonHttpClient.executeHelper(AmazonHttpClient.java:467)
        at com.amazonaws.http.AmazonHttpClient.execute(AmazonHttpClient.java:302)
        at com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient.invoke(AmazonSimpleEmailServiceClient.java:1731)
        at com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient.sendEmail(AmazonSimpleEmailServiceClient.java:1461)
        at br.com.ntk.util.EmailUtil.semAnexo(EmailUtil.java:78)
        at br.com.ntk.schedule.ReenvioQuartzJob.execute(ReenvioQuartzJob.java:36)
        at org.quartz.core.JobRunShell.run(JobRunShell.java:202)
        at org.quartz.simpl.SimpleThreadPool$WorkerThread.run(SimpleThreadPool.java:573)
18/09/2015 13:46:01.429 | ERROR | b.c.n.u.EmailUtil:90   | ERRO AO ENVIAR O EMAIL: 1 validation error detected: Value null at 'source' failed to satisfy constraint: Member must not be null (Service: AmazonSi
mpleEmailService; Status Code: 400; Error Code: ValidationError; Request ID: c0ee89c4-5e24-11e5-be3e-8b07a1f570e6)
18/09/2015 13:46:01.431 | DEBUG | b.c.n.d.EmailDAO:48   | ID_Mensagem: null atualizado para email  1068 com sucesso!
18/09/2015 13:46:01.433 | DEBUG | b.c.n.d.EmailDAO:60   | Destinatario: eduardo.silva@ntk-consult.com.br Status: 1 atualizado!
18/09/2015 13:48:00.003 | DEBUG | b.c.n.s.ReenvioQuartzJob:24   | JobExecution
18/09/2015 13:48:00.004 | DEBUG | b.c.n.s.ReenvioQuartzJob:31   | Email para retentativa encontrado: br.com.ntk.model.Email@bfa683
18/09/2015 13:48:00.004 | DEBUG | b.c.n.u.EmailUtil:49   | Remetente: null
18/09/2015 13:48:00.004 | DEBUG | b.c.n.u.EmailUtil:50   | Destinatarios: [eduardo.silva@ntk-consult.com.br]
com.amazonaws.AmazonServiceException: 1 validation error detected: Value null at 'source' failed to satisfy constraint: Member must not be null (Service: AmazonSimpleEmailService; Status Code: 400; Error Cod
e: ValidationError; Request ID: 08430920-5e25-11e5-b998-fd4074f4c6a0)
        at com.amazonaws.http.AmazonHttpClient.handleErrorResponse(AmazonHttpClient.java:1160)
        at com.amazonaws.http.AmazonHttpClient.executeOneRequest(AmazonHttpClient.java:748)
        at com.amazonaws.http.AmazonHttpClient.executeHelper(AmazonHttpClient.java:467)
        at com.amazonaws.http.AmazonHttpClient.execute(AmazonHttpClient.java:302)
        at com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient.invoke(AmazonSimpleEmailServiceClient.java:1731)
        at com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient.sendEmail(AmazonSimpleEmailServiceClient.java:1461)
        at br.com.ntk.util.EmailUtil.semAnexo(EmailUtil.java:78)
        at br.com.ntk.schedule.ReenvioQuartzJob.execute(ReenvioQuartzJob.java:36)
        at org.quartz.core.JobRunShell.run(JobRunShell.java:202)
        at org.quartz.simpl.SimpleThreadPool$WorkerThread.run(SimpleThreadPool.java:573)
18/09/2015 13:48:01.097 | ERROR | b.c.n.u.EmailUtil:90   | ERRO AO ENVIAR O EMAIL: 1 validation error detected: Value null at 'source' failed to satisfy constraint: Member must not be null (Service: AmazonSi
mpleEmailService; Status Code: 400; Error Code: ValidationError; Request ID: 08430920-5e25-11e5-b998-fd4074f4c6a0)
18/09/2015 13:48:01.099 | DEBUG | b.c.n.d.EmailDAO:48   | ID_Mensagem: null atualizado para email  1068 com sucesso!
18/09/2015 13:48:01.101 | DEBUG | b.c.n.d.EmailDAO:60   | Destinatario: eduardo.silva@ntk-consult.com.br Status: 1 atualizado!
18/09/2015 13:50:00.005 | DEBUG | b.c.n.s.ReenvioQuartzJob:24   | JobExecution
18/09/2015 13:50:00.005 | DEBUG | b.c.n.s.ReenvioQuartzJob:31   | Email para retentativa encontrado: br.com.ntk.model.Email@bfa683
18/09/2015 13:50:00.005 | DEBUG | b.c.n.u.EmailUtil:49   | Remetente: null
18/09/2015 13:50:00.006 | DEBUG | b.c.n.u.EmailUtil:50   | Destinatarios: [eduardo.silva@ntk-consult.com.br]
com.amazonaws.AmazonServiceException: 1 validation error detected: Value null at 'source' failed to satisfy constraint: Member must not be null (Service: AmazonSimpleEmailService; Status Code: 400; Error Cod
e: ValidationError; Request ID: 5001e3b6-5e25-11e5-bcec-1ba2018ca6d8)
        at com.amazonaws.http.AmazonHttpClient.handleErrorResponse(AmazonHttpClient.java:1160)
        at com.amazonaws.http.AmazonHttpClient.executeOneRequest(AmazonHttpClient.java:748)
        at com.amazonaws.http.AmazonHttpClient.executeHelper(AmazonHttpClient.java:467)
        at com.amazonaws.http.AmazonHttpClient.execute(AmazonHttpClient.java:302)
        at com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient.invoke(AmazonSimpleEmailServiceClient.java:1731)
        at com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient.sendEmail(AmazonSimpleEmailServiceClient.java:1461)
        at br.com.ntk.util.EmailUtil.semAnexo(EmailUtil.java:78)
        at br.com.ntk.schedule.ReenvioQuartzJob.execute(ReenvioQuartzJob.java:36)
        at org.quartz.core.JobRunShell.run(JobRunShell.java:202)
        at org.quartz.simpl.SimpleThreadPool$WorkerThread.run(SimpleThreadPool.java:573)
18/09/2015 13:50:01.485 | ERROR | b.c.n.u.EmailUtil:90   | ERRO AO ENVIAR O EMAIL: 1 validation error detected: Value null at 'source' failed to satisfy constraint: Member must not be null (Service: AmazonSi
mpleEmailService; Status Code: 400; Error Code: ValidationError; Request ID: 5001e3b6-5e25-11e5-bcec-1ba2018ca6d8)
18/09/2015 13:50:01.487 | DEBUG | b.c.n.d.EmailDAO:48   | ID_Mensagem: null atualizado para email  1068 com sucesso!
18/09/2015 13:50:01.489 | DEBUG | b.c.n.d.EmailDAO:60   | Destinatario: eduardo.silva@ntk-consult.com.br Status: 1 atualizado!
```

Gerenciamento Datasource



Gerenciamento Datasource



Gerenciamento Datasource

persistence.xml

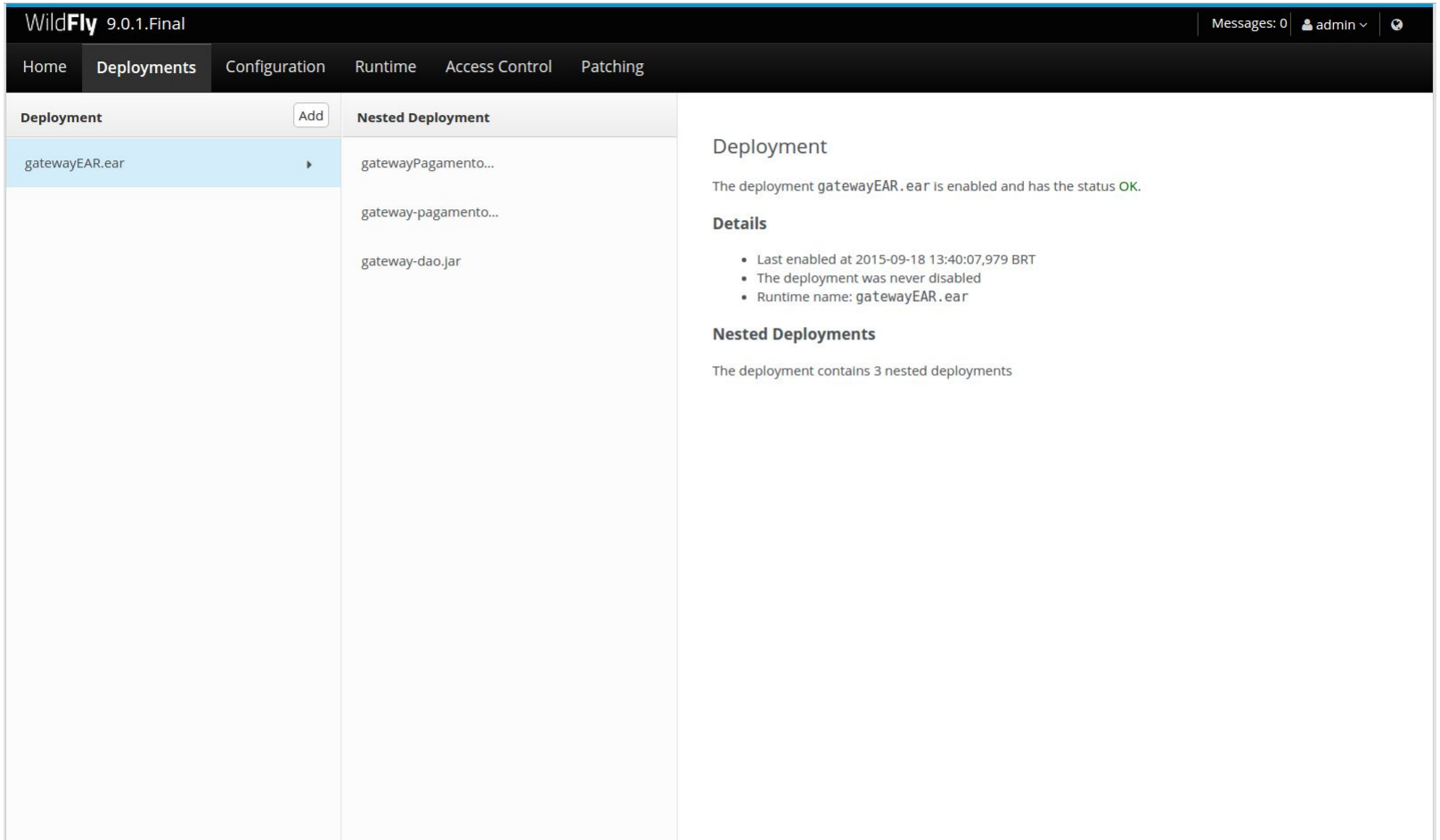
```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1"
    xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="gatewayPU" transaction-type="JTA">

        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <jta-data-source>java:jboss/datasources/gatewayDS</jta-data-source>

         <class>br.com.gateway.entity.Usuario</class>
         <class>br.com.gateway.entity.Configuracao</class>
         <class>br.com.ntk.model.Transacao</class>

        <properties>
            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5InnoDBDialect" />
            <property name="hibernate.hbm2ddl.auto" value="update" /> <!-- create-drop -->
            <property name="hibernate.connection.charSet" value="UTF-8" />
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.format_sql" value="true" />
        </properties>

    </persistence-unit>
</persistence>
```

Modelo atual Datasource

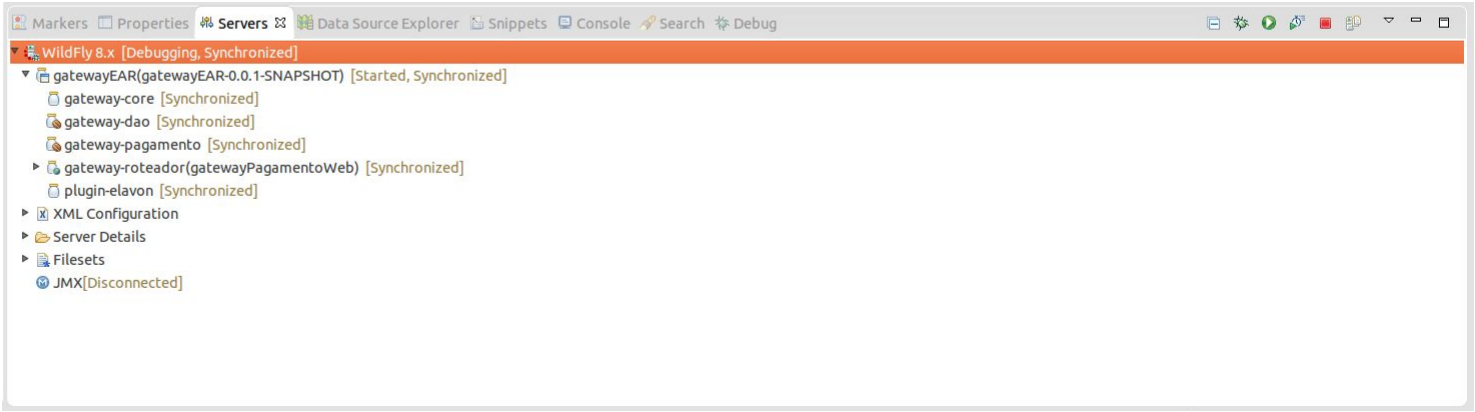
daoContext.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:jee="http://www.springframework.org/schema/jee"
  xmlns:tx="http://www.springframework.org/schema/tx" xmlns:context="http://www.springframework.org/schema/context"
  xmlns:util="http://www.springframework.org/schema/util"
  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
                      http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
                        http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-2.5.xsd
            http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-2.5.xsd
            http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-3.0.xsd">

  <util:properties id="applicationProps" location="file:${propConfig}" />
  <context:property-placeholder properties-ref="applicationProps" />

    <!--Gerenciador de Transacoes -->
    <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager" >
      <property name="dataSource" ref="dataSource"/>
    </bean>

    <!-- Habilita o comportamento transacional onde houver a anotação @Transacional -->
  <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <property name="driverClassName" value="${jdbc.driverClassName}"/>
        <property name="url" value="${jdbc.url}"/>
        <property name="username" value="${jdbc.username}"/>
        <property name="password" value="${jdbc.password}"/>
        <property name="validationQuery" value="SELECT 1"/>
        <property name="initialSize" value="5" />
    <property name="maxActive" value="10" />
    </bean>

    <tx:annotation-driven transaction-manager="transactionManager" />

  <!-- Dar Suport a Injecao de Dependencia do Bean via Annotations -->
  <context:annotation-config />

  <context:component-scan base-package="br.com.gateway" annotation-config="true" />

  <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="dataSource" />
  </bean>

</beans>
```

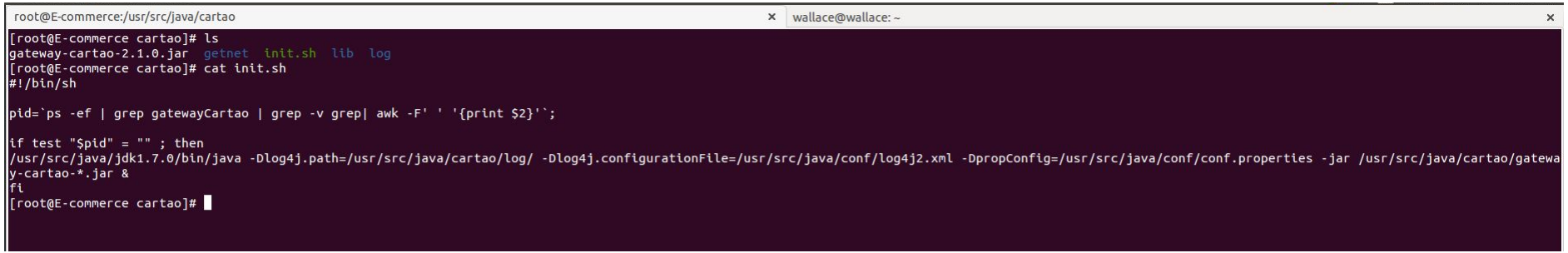
Facilidade no Deploy



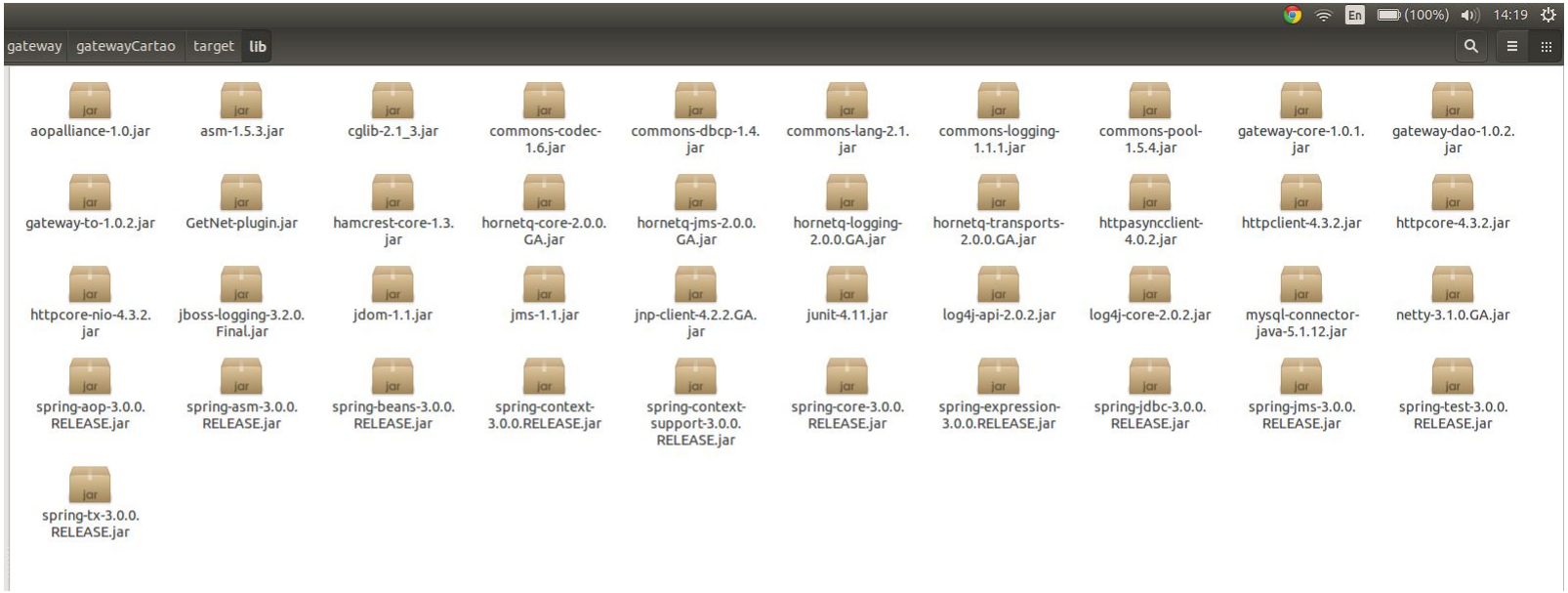
Facilidade no Deploy



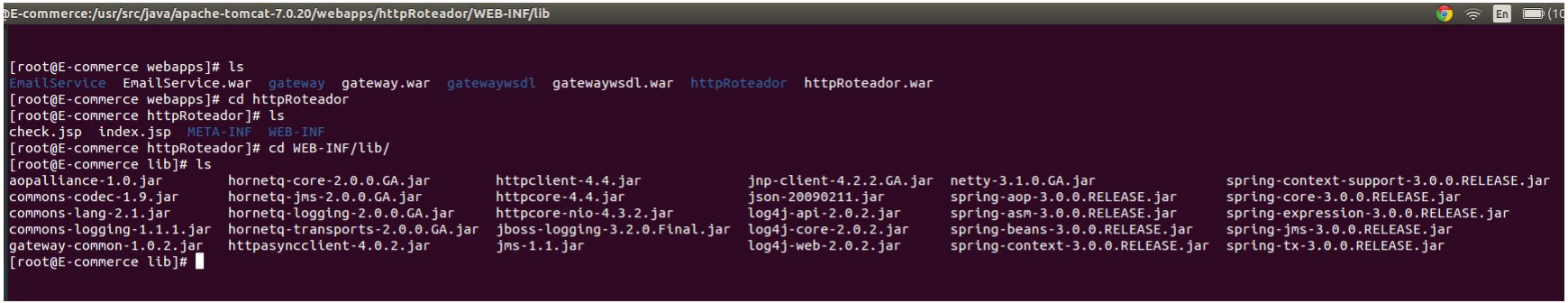
Modelo atual para cada modulo - Deploy



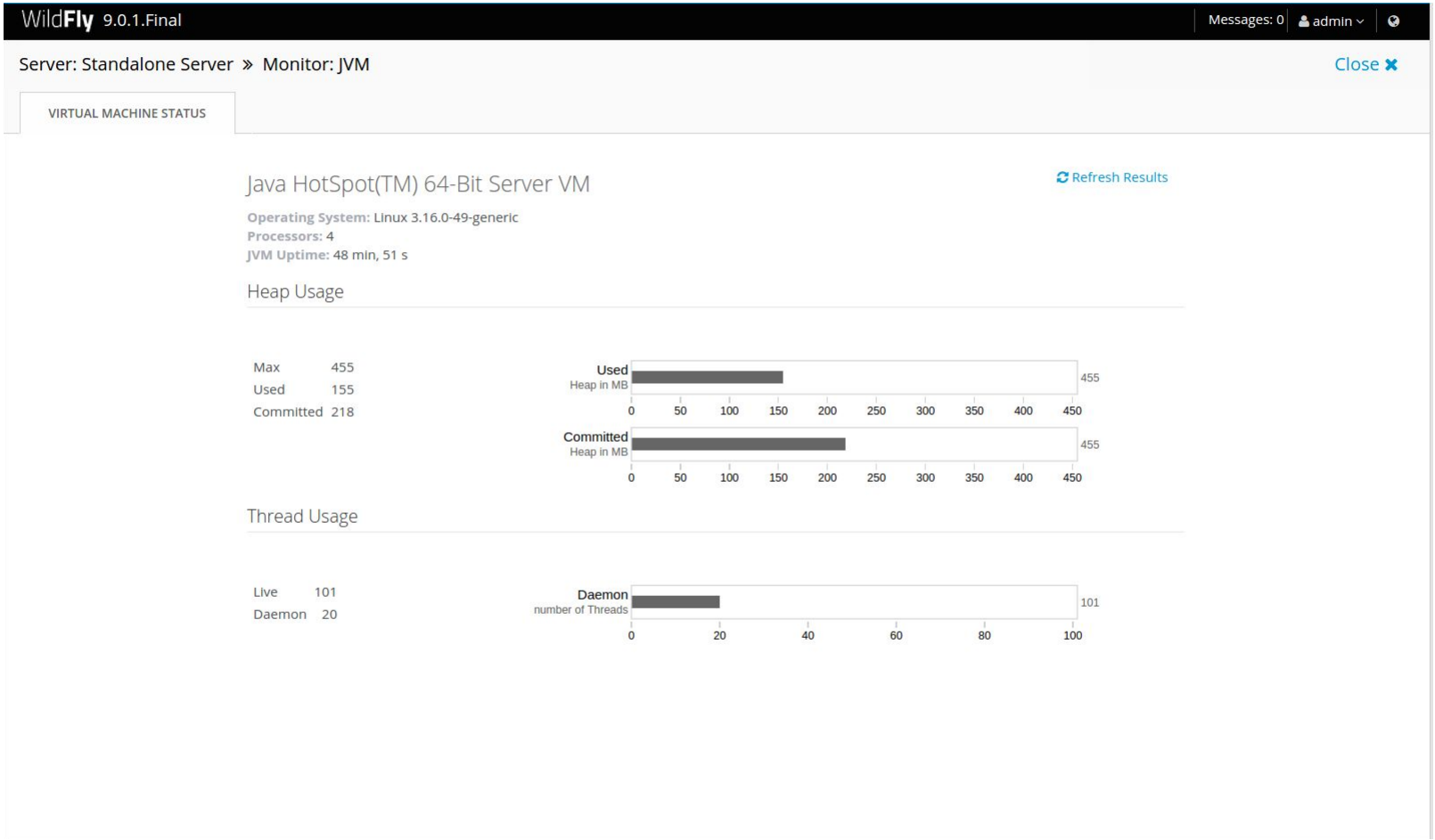
Modelo atual para cada modulo - Deploy

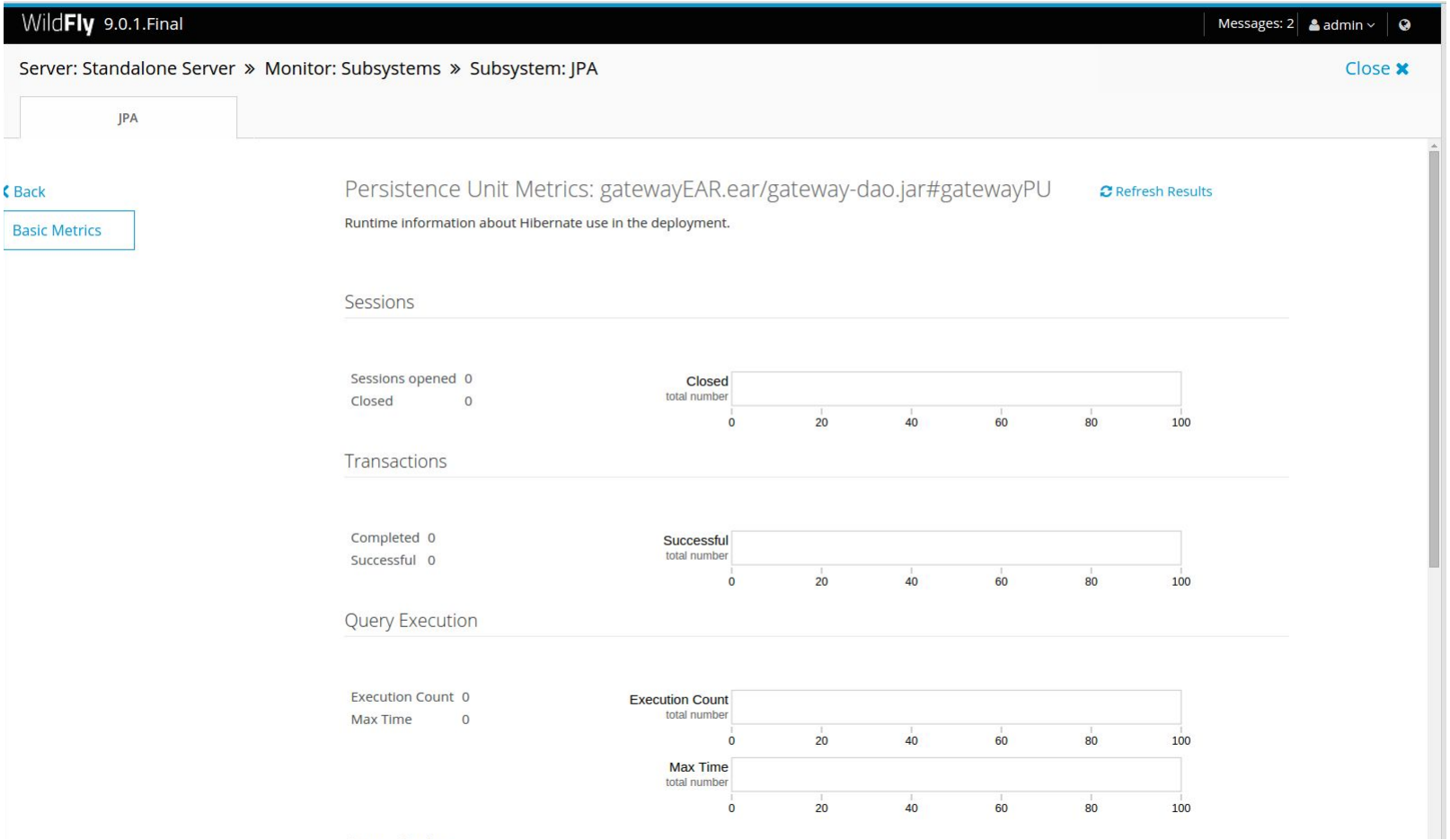


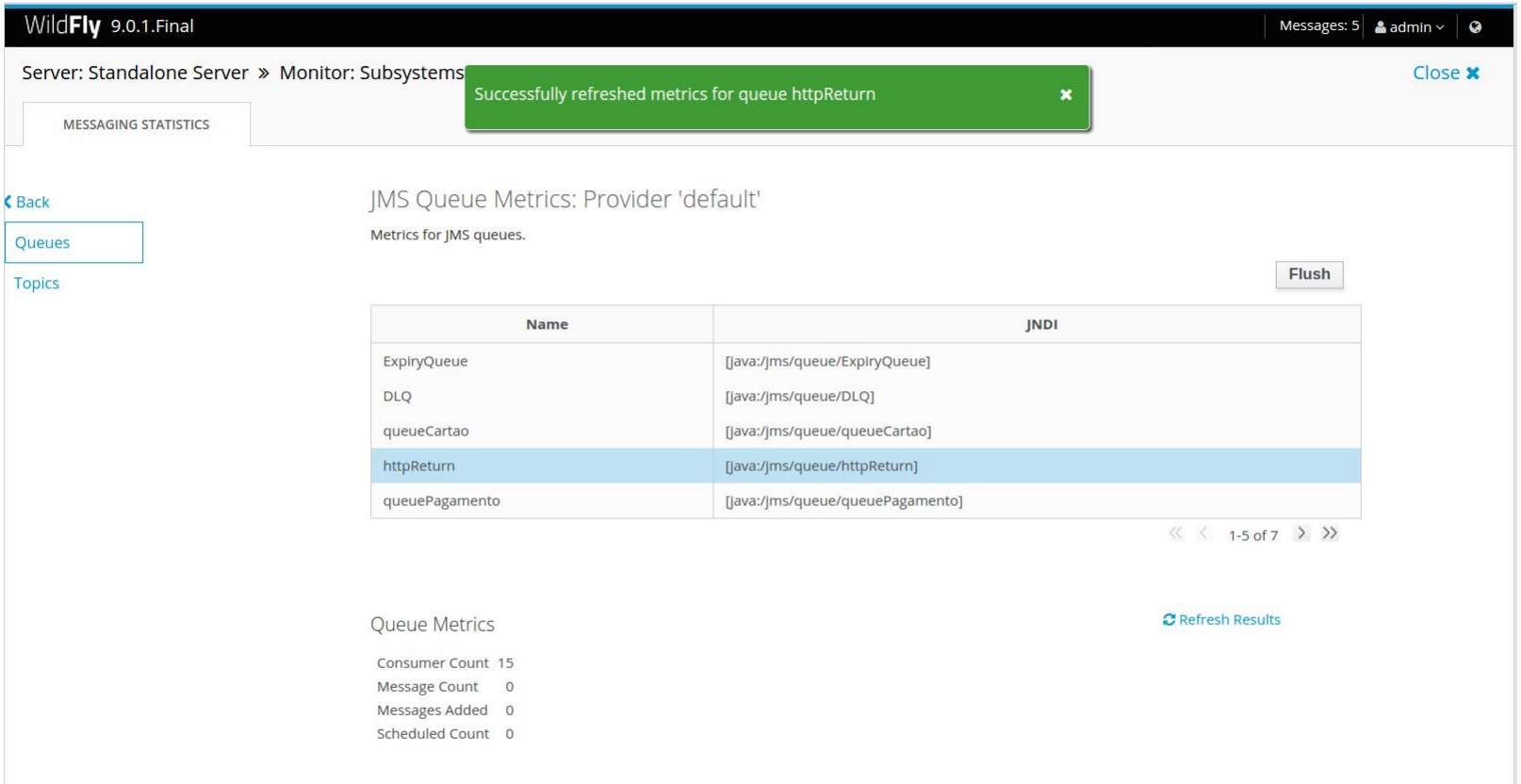
Modelo atual para cada modulo - Deploy



Monitoria da JVM




WildFly 9.0.1.Final
Messages: 2
admin
Server: Standalone Server » Monitor: Subsystems » Subsystem: JPA
Close
JPA
Back
Basic Metrics
Persistence Unit Metrics: gatewayEAR.ear/gateway-dao.jar#gatewayPU
Refresh Results
Runtime information about Hibernate use in the deployment.
Sessions
Sessions opened 0
Closed 0
Closed
total number
0 20 40 60 80 100
Transactions
Completed 0
Successful 0
Successful
total number
0 20 40 60 80 100
Query Execution
Execution Count 0
Max Time 0
Execution Count
total number
0 20 40 60 80 100
Max Time
total number
0 20 40 60 80 100


WildFly 9.0.1.Final
Messages: 5
admin
Server: Standalone Server » Monitor: Subsystems
Close
Successfully refreshed metrics for queue httpReturn
MESSAGING STATISTICS
Back
Queues
Topics
JMS Queue Metrics: Provider 'default'
Metrics for JMS queues.
Flush
Name
JNDI
ExpiryQueue
[java:/jms/queue/ExpiryQueue]
DLQ
[java:/jms/queue/DLQ]
queueCartao
[java:/jms/queue/queueCartao]
httpReturn
[java:/jms/queue/httpReturn]
queuePagamento
[java:/jms/queue/queuePagamento]
1-5 of 7
Queue Metrics
Refresh Results
Consumer Count 15
Message Count 0
Messages Added 0
Scheduled Count 0