ERRATA

MARTINS, Robson de Sousa. **Integração de Sistemas: Simplificando a Vida do Cidadão**. 2012. 348p. Trabalho de Conclusão de Curso (Pós-Graduação em Desenvolvimento de Aplicações Corporativas em Java/SOA) – Faculdade de Informática e Administração Paulista, São Paulo, 2012.

Página	Título / Subtítulo
335	**Apêndice B**: **Implantação na Nuvem** **Adição dos certificados digitais da cadeia confiável na JVM**

Os comandos corretos para adicionar os certificados da cadeia confiável na JVM são:

```
$ cd <java-home>/lib/security

$ sudo keytool -import -v -trustcacerts -file <path/icprmartins.cer>
-keystore cacerts -keypass changeit -storepass changeit -alias icprmartins

$ sudo keytool -import -v -trustcacerts -file <path/acrmartins.cer>
-keystore cacerts -keypass changeit -storepass changeit -alias acrmartins
```

APÊNDICE E – MIGRAÇÃO PARA OPENSHIFT / JBOSS AS7

MARTINS, Robson de Sousa. **Integração de Sistemas: Simplificando a Vida do Cidadão**. 2012. 348p. Trabalho de Conclusão de Curso (Pós-Graduação em Desenvolvimento de Aplicações Corporativas em Java/SOA) – Faculdade de Informática e Administração Paulista, São Paulo, 2012.

São Paulo

2013

LISTA DE ILUSTRAÇÕES

FIGURAS

SUMÁRIO

APÊNDICE E – MIGRAÇÃO PARA OPENSHIFT / JBOSS AS7

Após a expiração do prazo de experimentação gratuita do serviço *Amazon* EC2 (*Amazon Elastic Compute Cloud*), em junho de 2013, os servidores virtualizados que hospedavam a prova de conceito do trabalho referenciado foram desligados e a conta definitivamente cancelada.

Por esse motivo, optou-se por migrar os sistemas que compõem a prova de conceito para uma nova conta criada no serviço de nuvem *OpenShift*, da *RedHat*: <https://www.openshift.com> (acesso em 28 dez. 2013).

Este apêndice descreve as configurações e implementações necessárias para a realização da migração das aplicações e serviços que compõem a prova de conceito do trabalho original elaborado em 2012.

Características da conta gratuita do Serviço OpenShift

- Suporta até quatro instâncias (*applications*), podendo suportar cada uma dentre várias linguagens de programação e plataformas disponíveis, tais como: Java EE (JBoss AS), PHP, Python, Ruby, Perl, JavaScript;
- Suporta os bancos de dados: MySQL, PostgreSQL, MongoDB, além do H2 embarcado no servidor JBoss AS;
- Suporta configuração de *alias* (domínios próprios, via DNS) para cada uma das instâncias da conta;
- Suporta conexão via SSH ou GIT para realizar o *deploy* das aplicações;
- Suporta HTTPS (SSL), porém na conta gratuita (*OpenShift Online*) não é possível personalizar os certificados digitais das instâncias;
- É baseado em servidores rodando o sistema operacional *RedHat Enterprise Linux*;
- A plataforma Java EE disponibilizada na conta gratuita está baseada somente em JBoss AS 7 (JBoss 6 está disponível somente nas contas pagas).

Desafios da migração para o OpenShift

Como a prova de conceito foi desenvolvida para rodar em um servidor de aplicação JBoss 6.x sobre Java JRE/JVM 7, muitos detalhes de implementação se tornaram fontes de problemas durante a migração para o JBoss AS 7 fornecido pelo serviço *OpenShift*. Alguns deles:

- **Alteração da estrutura de diretórios e arquivos do servidor JBoss**: arquivos de configuração de *datasources* (fontes de dados), de *security domains* (propriedades de segurança JAAS), dentre outros, tiveram suas localizações e estruturas internas modificadas entre as versões 6 e 7 do JBoss.

- **Diferenças de implementação do servidor JBoss**: algumas funcionalidades, como módulos de *login* (JAAS), descritores de EJB's e *Web Services*, *framework* Java Server Faces (JSF), sofreram alterações de comportamento e configuração por parte das aplicações.

- **Incapacidade de personalizar certificados digitais**: o *OpenShift* não permite o acesso direto aos diretórios de instalação do JRE, impossibilitando a adição de certificados digitais a cadeia confiável da JVM. Além disso, a conta gratuita do *OpenShift* não permite personalizar os certificados para uso com HTTPS / SSL.

Mediante essas dificuldades, uma nova implementação de todos os sistemas que compõem a prova de conceito foi realizada, com configurações específicas para implantação no servidor de aplicações JBoss AS 7 do *OpenShift*. A seguir estão descritas as características dessa nova implementação / configuração.

Implantação no Servidor de Aplicação

Como a conta gratuita do *OpenShift* suporta no máximo quatro instâncias, e nem mesmo oferece um servidor *Web Apache* exclusivo para ser configurado como *proxy* para as aplicações Java EE, decidiu-se implantar todos os sistemas e serviços (ICP Admin, SICid, Banco Seguro e Receita Nacional) numa única instância rodando JBoss AS 7.

Protocolo HTTPS / SSL

Devido às dificuldades expostas anteriormente, optou-se por utilizar somente o protocolo HTTP no acesso aos sistemas da prova de conceito, realizando-se unicamente o redirecionamento do domínio **`http://tcc.fiap.robsonmartins.com`** à instância correspondente do *OpenShift*. O uso do protocolo HTTPS / SSL foi abandonado nesta implementação.

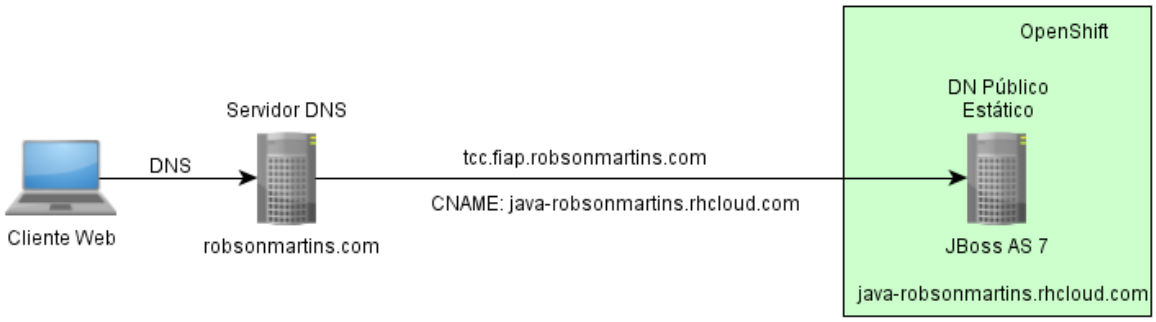

Figura 1 - Arquitetura de servidores DNS

Para desabilitar o protocolo SSL nos *web services*, foi necessária uma alteração nos arquivos: **`banco.ws.BancoSeguroService`** (linha 28) e **`sicid.ws.SICidService`** (linha 29):

```
//@WebContext(contextRoot = "<context>", urlPattern = "/<url>",
//            transportGuarantee = "CONFIDENTIAL", authMethod = "BASIC",
//            secureWSDLAccess = true)
@WebContext(contextRoot = "<context>", urlPattern = "/<url>",
       authMethod = "BASIC", secureWSDLAccess = true)
```

Configuração do Banco de Dados e dos Data Sources

No JBoss 7.x, o banco de dados embarcado oferecido é o H2 *Database Engine* <http://www.h2database.com> (acesso em 28 dez. 2013), e não mais o HyperSQL <http://hsqldb.org> (acesso em 28 dez. 2013), que era disponibilizado integrado com o JBoss 6.x. Isso obrigou a realização de uma nova configuração do arquivo **`persistence.xml`** no EAR das aplicações:

TccFiapEApp/EarContent/META-INF/persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">

  <persistence-unit name="sicid">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>java:/SICidDS</jta-data-source>
    <class>sicid.bean.CertificadoConfiavel</class>
    <class>sicid.bean.ConsumidorConfiavel</class>
    <class>sicid.bean.Usuario</class>
    <class>sicid.bean.Cidadao</class>
    <properties>
      <property name="hibernate.hbm2ddl.auto" value="update" />
            <property name="hibernate.dialect"
value="org.hibernate.dialect.H2Dialect" />
    </properties>
  </persistence-unit>

  <persistence-unit name="icpadmin">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>java:/ICPAdminDS</jta-data-source>
    <class>icp.bean.Certificado</class>
    <class>icp.bean.Usuario</class>
    <properties>
      <property name="hibernate.hbm2ddl.auto" value="update" />
            <property name="hibernate.dialect"
value="org.hibernate.dialect.H2Dialect" />
    </properties>
  </persistence-unit>
```

```
  <persistence-unit name="bancoseguro">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>java:/BancoSeguroDS</jta-data-source>
    <class>banco.bean.ConsumidorConfiavel</class>
    <class>banco.bean.Usuario</class>
    <class>banco.bean.Conta</class>
    <class>banco.bean.Extrato</class>
    <properties>
      <property name="hibernate.hbm2ddl.auto" value="update" />
            <property name="hibernate.dialect"
value="org.hibernate.dialect.H2Dialect" />
    </properties>
  </persistence-unit>

  <persistence-unit name="receita">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>java:/ReceitaNacionalDS</jta-data-source>
    <class>receita.bean.Cidadao</class>
    <class>receita.bean.Tributo</class>
    <properties>
      <property name="hibernate.hbm2ddl.auto" value="update" />
            <property name="hibernate.dialect"
value="org.hibernate.dialect.H2Dialect" />
    </properties>
  </persistence-unit>

</persistence>
```

Além disso, no JBoss 7.x, os arquivos de *datasource* (**<jboss-server-home>/deploy/*-ds.xml**) não são mais válidos. Em seu lugar, toda a configuração de *datasources* passou a ser unificada dentro do arquivo **<jboss-home>/standalone/configuration/standalone.xml** (que no *OpenShift* está no caminho **app-root/runtime/repo/.openshift/config/standalone.xml**):

<jboss-home>/standalone/configuration/standalone.xml

(no OpenShift: app-root/runtime/repo/.openshift/config/standalone.xml)

```
<?xml version='1.0' encoding='UTF-8'?>
<server xmlns="urn:jboss:domain:1.1">
  <profile>
    <!-- ... outras configurações ... -->

    <subsystem xmlns="urn:jboss:domain:datasources:1.0">
      <datasources>

        <datasource jndi-name="java:/ICPAdminDS" pool-name="ICPAdminDS"
enabled="true" use-java-context="true">
          <connection-url>jdbc:h2:${jboss.server.data.dir}${/}icpadmin$
{/}icpadminDB;DB_CLOSE_DELAY=-1</connection-url>
          <driver>h2</driver>
          <security>
            <user-name>sa</user-name>
            <password>sa</password>
          </security>
        </datasource>

        <datasource jndi-name="java:/SICidDS" pool-name="SICidDS" enabled="true"
use-java-context="true">
          <connection-url>jdbc:h2:${jboss.server.data.dir}${/}tccfiap$
{/}sicidDB;DB_CLOSE_DELAY=-1</connection-url>
          <driver>h2</driver>
          <security>
            <user-name>sa</user-name>
            <password>sa</password>
          </security>
        </datasource>

        <datasource jndi-name="java:/BancoSeguroDS" pool-name="BancoSeguroDS"
enabled="true" use-java-context="true">
          <connection-url>jdbc:h2:${jboss.server.data.dir}${/}tccfiap$
{/}bancoseguroDB;DB_CLOSE_DELAY=-1</connection-url>
            <driver>h2</driver>
            <security>
              <user-name>sa</user-name>
              <password>sa</password>
            </security>
          </datasource>
```

```
          <datasource jndi-name="java:/ReceitaNacionalDS" pool-
name="ReceitaNacionalDS" enabled="true" use-java-context="true">
            <connection-url>jdbc:h2:${jboss.server.data.dir}${/}tccfiap$
{/}receitaDB;DB_CLOSE_DELAY=-1</connection-url>
            <driver>h2</driver>
            <security>
              <user-name>sa</user-name>
              <password>sa</password>
            </security>
          </datasource>
          <!-- ... outros datasources ... -->
        </datasources>
        <!-- ... outras configurações do subsystem ... -->
      </subsystem>
      <!-- ... outros subsystems ... -->
  </profile>
</server>
```

Outro problema apresentado na migração do banco de dados foi o tamanho das colunas do tipo `VARCHAR`: durante a criação das tabelas pelo *Hibernate*, as propriedades tipo `String` sem especificação de tamanho (que seria feita através do *annotation* `@Column`) são automaticamente mapeadas como `VARCHAR(255)` no banco de dados H2, desrespeitando o modelo de dados projetado para alguns dos sistemas da prova de conceito (os campos sem especificação de tamanho devem ser teoricamente ilimitados, capazes de armazenar uma quantidade não-previsível de dados).

Por esse motivo, ajustes foram realizados manualmente nas estruturas das tabelas, após sua criação, através de um cliente SQL do banco de dados H2.

Configuração de Segurança do JAAS

No JBoss 7.x, o arquivo de configuração do JAAS (`<jboss-server-home>/conf/login-config.xml`) não é mais válido. Em seu lugar, toda a configuração de *security domains* e módulos JAAS passou a ser unificada dentro do arquivo `<jboss-home>/standalone/configuration/standalone.xml` (que no *OpenShift* está no caminho `app-root/runtime/repo/.openshift/config/standalone.xml`):

<jboss-home>/standalone/configuration/standalone.xml

(no OpenShift: app-root/runtime/repo/.openshift/config/standalone.xml)

```
<?xml version='1.0' encoding='UTF-8'?>
<server xmlns="urn:jboss:domain:1.1">
  <profile>
    <!-- ... outras configurações ... -->
    <subsystem xmlns="urn:jboss:domain:security:1.1">
      <security-domains>
        <security-domain name="SICidService" cache-type="default">
          <authentication>
            <login-module
code="com.robsonmartins.fiap.tcc.sicid.jaas.SICidDBLoginModule" flag="required">
              <module-option name="dsJndiName" value="java:/SICidDS"/>
              <module-option name="principalsQuery" value="SELECT id as
PrincipalID from ConsumidorConfiavel WHERE dname=?"/>
              <module-option name="rolesQuery" value="SELECT role as Role, 'Roles'
from ConsumidorConfiavel WHERE id=?"/>

              <module-option name="fullPrincipalsQuery" value="SELECT * FROM
ConsumidorConfiavel"/>
              <module-option name="principalIdForEmpty" value="wsuser"/>
              <module-option name="roleForEmpty" value="wsuser"/>
              <module-option name="sicid.disable" value="true"/>
            </login-module>
          </authentication>
        </security-domain>

        <security-domain name="SICidWeb" cache-type="default">
          <authentication>
            <login-module
code="com.robsonmartins.fiap.tcc.sicid.jaas.SICidDBLoginModule" flag="required">
              <module-option name="dsJndiName" value="java:/SICidDS"/>
              <module-option name="principalsQuery" value="SELECT username as
PrincipalID FROM Usuario WHERE dname=?"/>
              <module-option name="rolesQuery" value="SELECT role as Role, 'Roles'
FROM Usuario WHERE username=?"/>
              <module-option name="fullPrincipalsQuery" value="SELECT * FROM
Usuario"/>
              <module-option name="principalIdForEmpty" value="admin"/>
              <module-option name="roleForEmpty" value="admin"/>
              <module-option name="sicid.clientProps"
value="sicidweb.properties"/>
            </login-module>
          </authentication>
        </security-domain>
```

```
        <security-domain name="ICPAdmin" cache-type="default">
          <authentication>
            <login-module
code="com.robsonmartins.fiap.tcc.sicid.jaas.SICidDBLoginModule" flag="required">
              <module-option name="dsJndiName" value="java:/ICPAdminDS"/>
              <module-option name="principalsQuery" value="SELECT username as
PrincipalID FROM Usuario WHERE dname=?"/>
              <module-option name="rolesQuery" value="SELECT role as Role, 'Roles'
FROM Usuario WHERE username=?"/>
              <module-option name="fullPrincipalsQuery" value="SELECT * FROM
Usuario"/>
              <module-option name="principalIdForEmpty" value="admin"/>
              <module-option name="roleForEmpty" value="admin"/>
              <module-option name="sicid.clientProps"
value="icpadmin.properties"/>
            </login-module>
          </authentication>
        </security-domain>

        <security-domain name="BancoSeguro" cache-type="default">
          <authentication>
            <login-module
code="com.robsonmartins.fiap.tcc.sicid.jaas.SICidDBLoginModule" flag="required">
              <module-option name="dsJndiName" value="java:/BancoSeguroDS"/>
              <module-option name="principalsQuery" value="SELECT cpf as
PrincipalID FROM Usuario WHERE dname=?"/>
              <module-option name="rolesQuery" value="SELECT role as Role, 'Roles'
FROM Usuario WHERE cpf=?"/>
              <module-option name="fullPrincipalsQuery" value="SELECT * FROM
Usuario"/>
              <module-option name="principalIdForEmpty" value="gerente"/>
              <module-option name="roleForEmpty" value="gerente"/>
              <module-option name="sicid.clientProps"
value="bancoseguro.properties"/>
            </login-module>
          </authentication>
        </security-domain>
```

```
        <security-domain name="BancoSeguroService" cache-type="default">
          <authentication>
            <login-module
code="com.robsonmartins.fiap.tcc.sicid.jaas.SICidDBLoginModule" flag="required">
              <module-option name="dsJndiName" value="java:/BancoSeguroDS"/>
              <module-option name="principalsQuery" value="SELECT id as
PrincipalID from ConsumidorConfiavel WHERE dname=?"/>
              <module-option name="rolesQuery" value="SELECT role as Role, 'Roles'
from ConsumidorConfiavel WHERE id=?"/>
              <module-option name="fullPrincipalsQuery" value="SELECT * FROM
ConsumidorConfiavel"/>
              <module-option name="principalIdForEmpty" value="wsuser"/>
              <module-option name="roleForEmpty" value="wsuser"/>
              <module-option name="sicid.clientProps"
value="bancoseguro.properties"/>
            </login-module>
          </authentication>
        </security-domain>

        <security-domain name="ReceitaNacional" cache-type="default">
          <authentication>
            <login-module
code="com.robsonmartins.fiap.tcc.sicid.jaas.SICidDBLoginModule" flag="required">
              <module-option name="dsJndiName" value="java:/ReceitaNacionalDS"/>
              <module-option name="principalsQuery" value="SELECT ric as
PrincipalID FROM Cidadao WHERE dname=?"/>
              <module-option name="rolesQuery" value="SELECT role as Role, 'Roles'
FROM Cidadao WHERE ric=?"/>
              <module-option name="fullPrincipalsQuery" value="SELECT * FROM
Cidadao WHERE role='governo'"/>
              <module-option name="principalIdForEmpty" value="governo"/>
              <module-option name="roleForEmpty" value="governo"/>
              <module-option name="sicid.clientProps" value="receita.properties"/>
            </login-module>
          </authentication>
        </security-domain>
        <!-- ... outros security domains ... -->
      </security-domains>
    </subsystem>
    <!-- ... outros subsystems ... -->
  </profile>
</server>
```

Outro problema apresentado no funcionamento do JAAS no JBoss 7.x foi a maneira como o *username* é tratado como um **`Principal`** dentro de uma sessão de aplicação. No JBoss 6.x, era armazenado o certificado digital (codificado em Base64) utilizado no *login*, como sendo o **`UserPrincipal`** autenticado na sessão. Então, uma chamada a **`FacesContext.getCurrentInstance()`** **`.getExternalContext().getRequest().getUserPrincipal().getName()`** dentro de uma aplicação, era capaz de retornar o certificado digital do usuário autenticado.

No JBoss 7.x, esse comportamento foi alterado: o *username* é tratado como o resultado da *query SQL* realizada na tabela **`"Principals"`** (em uma autenticação baseada em *LoginModule* por banco de dados). Então, a chamada a **`getUserPrincipal().getName()`** dentro de uma aplicação, passou a retornar o *username* gravado no banco de dados do *login*, e não mais o certificado digital do usuário.

Isso impactou o projeto de todas as aplicações e serviços, tornando necessária uma alteração de implementação do *LoginModule* do SICid:

SICidLoginModule

(com.robsonmartins.fiap.tcc.sicid.jaas.SICidDBLoginModule)

```
public class SICidDBLoginModule extends DatabaseServerLoginModule {
  //...
  private Principal identity;
  private Principal usernameIdentity;

  //...
  @Override
  public boolean login() throws LoginException {

    //...
    this.roles = getRoleSets();
    //...

    // substitui username pelo certificado no identity primario
    this.usernameIdentity = this.identity;
    try {
      this.identity = createIdentity(name); // name -> certificado
    } catch (Exception e) {
      if (trace) {
        log.trace(String.format("Failed to create principal: %s",name), e);
      }
```

```
      LoginException exception =
        new LoginException(
          String.format("Failed to create principal: %s",name));
      exception.initCause(e);
      throw exception;
    }
    return true;
    //...
  }

  //...

  @Override
  public boolean commit() throws LoginException {
    Set<Principal> principals = this.subject.getPrincipals();
    if (this.identity != null) {
      principals.add(this.identity);
    }
    if (this.usernameIdentity != null) {
      principals.add(this.usernameIdentity);
    }
    //...
  }
  @Override
  public boolean abort() throws LoginException {
    //...
    this.identity         = null;
    this.usernameIdentity = null;
    return true;
  }
  //...

  @Override
  public boolean logout() throws LoginException {
    Set<Principal> principals = this.subject.getPrincipals();
    if (this.identity != null) {
      principals.remove(this.identity);
    }
    if (this.usernameIdentity != null) {
      principals.remove(this.usernameIdentity);
    }
    return true;
  }
  //...
}
```

Implementação da Applet de Login (SICid Applet)

A partir da versão 45 do Java 7 (Java 7u45), tornou-se obrigatória a especificação de alguns atributos de segurança nas aplicações Java para *browsers* (*applets*). As *applets* que não possuem esses atributos, são executadas somente após as confirmações de segurança efetuadas pelo usuário, segundo a documentação (FAQ) do Java 7u45 <http://www.java.com/pt_BR/download/faq/release_changes.xml> (acesso em 28 dez. 2013). Em futuras versões do Java, as *applets* sem os atributos de segurança não serão mais executadas.

Desta forma, uma alteração foi necessária na implementação da *Applet* de *Login* utilizada nas aplicações web da prova de conceito: a inclusão dos atributos de segurança num arquivo MANIFEST, criado ao empacotar a *applet* em um arquivo JAR, através do *script* ANT **`build.xml`**:

SICidApplet

Script ANT: build.xml

```
<?xml version="1.0"?>
<project name="SICidApplet">
  <target name="generatejar">
    <jar destfile="dist/SICidApplet.jar">
      <fileset dir="bin" />
      <manifest>
        <attribute name="Permissions" value="all-permissions" />
        <attribute name="Codebase"
                   value="tcc.fiap.robsonmartins.com
                          java-robsonmartins.rhcloud.com localhost 127.0.0.1" />
        <attribute name="Caller-Allowable-Codebase" value="*" />
        <attribute name="Trusted-Only" value="false" />
        <attribute name="Trusted-Library" value="false" />
        <attribute name="Application-Library-Allowable-Codebase" value="*" />
        <attribute name="Main-Class" value="SICidApplet" />
        <attribute name="Application-Name"
                   value="Serviço de Identificação do Cidadão" />
        <attribute name="Implementation-Title" value="SICidApplet" />
        <attribute name="Implementation-Version" value="1.0" />
        <attribute name="Built-By" value="Robson Martins" />
        <attribute name="Implementation-Vendor" value="Robson Martins" />
        <attribute name="Application-Vendor" value="Robson Martins" />
      </manifest>
    </jar>
  </target>
```

```
  <target name="signjar">
    <signjar destDir="dist/signed"
        keystore="keystore/sicid.keystore"
        alias="sicid" storepass="sicid" keypass="sicid"
        preservelastmodified="true">
        <path>
          <fileset dir="dist" includes="*.jar" />
        </path>
        <flattenmapper />
    </signjar>
  </target>
</project>
```

Caminho dos Arquivos de Configuração no JBoss

Como o JBoss AS 7 tem uma nova estrutura de arquivos, foi necessário criar uma classe para ajudar as aplicações a encontrar o caminho correto do diretório de configuração do servidor do JBoss:

TccFiapCommon

com.robsonmartins.fiap.tcc.util.JBossUtil

```
package com.robsonmartins.fiap.tcc.util;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
/** Encapsula funcionalidades uteis de acesso ao servidor de aplicacao JBoss.
 *  @author Robson Martins (robson@robsonmartins.com)  */
public class JBossUtil {
  /** Retorna caminho absoluto de um arquivo de configuracao no JBoss.
   *  @param filename Nome do arquivo, relativo ou absoluto.
   *  @return Nome de arquivo com caminho absoluto.
   *  @throws Exception */
  public static String getJBossAbsFilePath(String filename) throws Exception {
    /* substitui system properties no nome de arquivo */
    int sprop = -1;
    int eprop = -1;
    StringBuilder fNameBuilder = new StringBuilder(filename);
    StringBuilder newFileName = null;
```

```
    do {
      sprop = fNameBuilder.toString().indexOf("${");
      eprop = fNameBuilder.toString().indexOf("}");
      if (sprop >= 0 && eprop >= 0) {
        newFileName = new StringBuilder();
        newFileName.append(fNameBuilder.toString().substring(0, sprop));
        newFileName.append(
          System.getProperty(
            fNameBuilder.toString().substring(sprop+2,eprop)));
        newFileName.append(fNameBuilder.toString().substring(eprop+1));
        fNameBuilder = newFileName;
      }
    } while (sprop >= 0 && eprop >= 0);
    filename = fNameBuilder.toString();
    /* tenta obter path "conf" do JBoss */
    String confDir = getJBossConfigPath();
    InputStream inputStream = null;
    if (confDir != null) {
      /* tenta obter arquivo no path especificado via classloader */
      try {
        inputStream =
            JBossUtil.class.getClassLoader().getResourceAsStream(
              String.format("%s%s%s", confDir, File.separator, filename));
        filename = String.format("%s%s%s", confDir, File.separator, filename);
      } catch (Exception e) { }
    }
    if (inputStream == null) {
      /* se nao achou no dir JBoss, tenta obter arquivo via classloader */
      try {
        inputStream =
            JBossUtil.class.getClassLoader().getResourceAsStream(filename);
      } catch (Exception e) {  }
    }
    if (inputStream == null) {
      /* se nao achou, tenta obter arquivo diretamente no path */
      try {
        inputStream = new FileInputStream(
            String.format("%s%s%s", confDir, File.separator, filename));
        filename = String.format("%s%s%s", confDir, File.separator, filename);
      } catch (Exception e) {  }
    }
```

```
    if (inputStream == null) {
      /* se nao achou, tenta obter arquivo diretamente */
      try {
        inputStream = new FileInputStream(filename);
      } catch (Exception e) {  }
    }
    if (inputStream == null) {
      throw new FileNotFoundException(
          String.format("File %s not found.", filename));
    }
    try { inputStream.close(); } catch (Exception e) { }
    return filename.replace('/', File.separatorChar);
  }
  /**  Retorna o diretorio de configuracao do JBoss AS.
   *   @return Caminho do diretorio de configuracao do JBoss,
   *   ou null se nao encontrado.  */
  public static String getJBossConfigPath() {
    String confDir = null;
    /* tenta obter path "conf" do JBoss */
    confDir = System.getProperty("jboss.server.config.url");
    if (confDir == null) {
      confDir = System.getProperty("jboss.server.config.dir");
    }
    if (confDir == null) {
      /* tenta obter path "serverhome"/conf do JBoss */
      confDir = System.getProperty("jboss.server.home.url");
      if (confDir == null) {
        confDir = System.getProperty("jboss.server.base.dir");
      }
      if (confDir != null) {
        File f = new File(confDir + File.separatorChar + "conf");
        if (f.exists()) {
          confDir += File.separatorChar + "conf";
        } else {
          f = new File(confDir + File.separatorChar + "configuration");
          if (f.exists()) {
            confDir += File.separatorChar + "configuration";
          }
        }
      }
    }
    return confDir;
  }
}
```

E nos arquivos **`icp.model.IcpAdmin`** (linha 608), **`banco.ws.BancoSeguroClient`** (linhas 265 e 301), **`sicid.ws.SICidClient`** (linhas 265 e 301), foi adicionada uma chamada semelhante a esta:

```
/* tenta obter path absoluto do arquivo no JBoss */
filename = JBossUtil.getJBossAbsFilePath(file);
```

Configuração do Java Server Faces (JSF)

No JBoss AS 7.x, é necessário especificar que uma aplicação fará uso do *framework Java Server Faces* (JSF), através de uma entrada no arquivo MANIFEST.MF do EAR:

TccFiapEApp

EarContent/META-INF/MANIFEST.MF

```
Manifest-Version: 1.0
Dependencies: javax.faces.api export
```

Configuração dos Security Domains nos EJB's e Web Services

Enquanto no JBoss 6.x a configuração de segurança do JAAS para um EJB é realizada pelo arquivo **`META-INF/jboss.xml`**, no JBoss AS 7.x, essa configuração passou a ser realizada pelo arquivo **`META-INF/jboss-ejb3.xml`**, com a seguinte estrutura:

SICidService

ejbModule/META-INF/jboss-ejb3.xml

```
<?xml version="1.0"?>
<jboss:ejb-jar xmlns:jboss="http://www.jboss.com/xml/ns/javaee"
                  xmlns="http://java.sun.com/xml/ns/javaee"
                  xmlns:s="urn:security"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.jboss.com/xml/ns/javaee
                     http://www.jboss.org/j2ee/schema/jboss-ejb3-2_0.xsd
                     http://java.sun.com/xml/ns/javaee
                     http://java.sun.com/xml/ns/javaee/ejb-jar_3_1.xsd"
                  version="3.1"
                  impl-version="2.0">
  <assembly-descriptor>
    <s:security>
      <ejb-name>SICidService</ejb-name>
      <s:security-domain>SICidService</s:security-domain>
    </s:security>
  </assembly-descriptor>
</jboss:ejb-jar>
```

BancoSeguroService

ejbModule/META-INF/jboss-ejb3.xml

```
<?xml version="1.0"?>
<jboss:ejb-jar xmlns:jboss="http://www.jboss.com/xml/ns/javaee"
                  xmlns="http://java.sun.com/xml/ns/javaee"
                  xmlns:s="urn:security"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.jboss.com/xml/ns/javaee
                    http://www.jboss.org/j2ee/schema/jboss-ejb3-2_0.xsd
                    http://java.sun.com/xml/ns/javaee
                    http://java.sun.com/xml/ns/javaee/ejb-jar_3_1.xsd"
                  version="3.1"
                  impl-version="2.0">
  <assembly-descriptor>
    <s:security>
      <ejb-name>BancoSeguroService</ejb-name>
      <s:security-domain>BancoSeguroService</s:security-domain>
    </s:security>
  </assembly-descriptor>
</jboss:ejb-jar>
```

OpenShift e Aplicações em Idle

O serviço *OpenShift* apresenta uma característica de economia de recursos, que é a de desligar automaticamente o servidor de aplicação quando não há requisições num determinado período de tempo (48 horas), conforme explicado no tópico <https://www.openshift.com/faq/what-happens-if-my-application-is-not-used-for-a-long-time> (acesso em 28 dez. 2013). Esse comportamento faz com que o JBoss AS seja desligado (*shutdown*) automaticamente, sendo necessário esperar pelo tempo de *boot-up* quando a primeira requisição for realizada após esse estado *idle*.

O problema é que o tempo de inicialização (*boot-up*) do servidor JBoss se torna elevado, pois é realizado o *re-deploy* de todas aplicações que compõem a prova de conceito. Isso provoca *timeout* no navegador do usuário durante o primeiro acesso após o estado *idle*.

Para contornar esse problema, é necessário realizar requisições à aplicação periodicamente, mas fora do domínio da aplicação (ou seja, a partir de outro servidor).

Desta forma, optou-se por utilizar uma conta gratuita do serviço *Site 24x7*, localizado em <http://site24x7.com> (acesso em 28 dez. 2013). Esse serviço oferece o monitoramento de até 5 sites, enviando requisições a cada 10 minutos e verificando a resposta do servidor (para monitorar se está ativo ou não). Com isso, as aplicações rodando no *OpenShift* podem funcionar ininterruptamente, mesmo que não haja requisições num intervalo maior que 48 horas.

Funções desabilitadas na demonstração

Para evitar alterações indevidas realizadas por visitantes da Prova de Conceito, e que venham a comprometer a integridade da demonstração, algumas funcionalidades foram desabilitadas nas aplicações hospedadas no serviço *OpenShift*:

- **ICPAdmin**:
    - Adição e remoção de usuários com perfil administrador;
    - Remoção da AC Raiz "*icprmartins*" e da AC Intermediária "*acrmartins*";
    - Download de cerificados PKCS#12 e de chaves privadas da AC Raiz "*icprmartins*" e da AC Intermediária "*acrmartins*", e dos certificados emitidos por elas;
    - Emissão, revogação e renovação de certificados da AC Raiz "*icprmartins*" e da AC Intermediária "*acrmartins*", e dos certificados emitidos por elas;
    - Download do Backup da ICP.
- **SICid**:
    - Adição e remoção de usuários com perfil administrador;
    - Remoção de certificados confiáveis;
    - Remoção de consumidores confiáveis;
    - Remoção de cidadãos do cadastro.
- **BancoSeguro**:
    - Adição e remoção de usuários com perfil administrador;
    - Remoção de consumidores confiáveis.
- **ReceitaNacional**:
    - Adição e remoção de usuários com perfil administrador.

Conclusões

Após a realização das configurações propostas, foi possível, apesar de algumas restrições, a migração satisfatória das aplicações e serviços da prova de conceito para o serviço em nuvem do *OpenShift*.

Esta migração proporcionou o aprimoramento do conhecimento sobre a plataforma JBoss AS 7 e sobre o serviço *RedHat OpenShift*.

REFERÊNCIAS

AMAZON. **Amazon Elastic Compute Cloud (Amazon EC2)**. 2012. Disponível em: <http://aws.amazon.com/pt/ec2/>. Acesso em: 28 dez. 2013.

H2. **H2 Database Engine**. 2005. Disponível em: <http://www.h2database.com>. Acesso em: 28 dez. 2013.

HSQL, The hsql Development Group. **HyperSQL Database**. 2001. Disponível em: <http://hsqldb.org>. Acesso em: 28 dez. 2013.

MARTINS, Robson de Sousa. **Integração de Sistemas: Simplificando a Vida do Cidadão**. 2012. 348p. Trabalho de Conclusão de Curso (Pós-Graduação em Desenvolvimento de Aplicações Corporativas em Java/SOA) – Faculdade de Informática e Administração Paulista, São Paulo, 2012.

ORACLE. **Java 7u45 Frequently Asked Questions (FAQ)**. 2013. Disponível em: <http://www.java.com/pt_BR/download/faq/release_changes.xml>. Acesso em: 28 dez. 2013.

REDHAT; JBOSS Community. **JBoss Application Server 7**. 2011. Disponível em: <http://www.jboss.org/jbossas/>. Acesso em: 28 dez. 2013.

REDHAT. **OpenShift**: The Open Hybrid Cloud Application Platform by Red Hat. 2013. Disponível em: <https://www.openshift.com>. Acesso em: 28 dez. 2013.

ZOHO Corporation Pvt. Ltd. **Site 24x7**: Website Monitoring Service. 2013. Disponível em: <http://site24x7.com>. Acesso em: 28 dez. 2013.