

**CENTRO UNIVERSITÁRIO DE MARINGÁ  
CURSO DE PÓS-GRADUAÇÃO EM DESENVOLVIMENTO ORIENTADO A  
OBJETOS - JAVA**

**RODRIGO GUEDES DE SOUZA**

**DESENVOLVIMENTO DE UMA APLICAÇÃO FRENTE DE CAIXA  
AUTOMATIZADA PARA O VAREJO UTILIZANDO JAVA E RFID**

MARINGÁ  
2005

RODRIGO GUEDES DE SOUZA

**DESENVOLVIMENTO DE UMA APLICAÇÃO FRETE DE CAIXA  
AUTOMATIZADA PARA O VAREJO UTILIZANDO JAVA E RFID**

Trabalho apresentado ao Centro Universitário de Maringá como requisito parcial para obtenção do título de Especialista em Desenvolvimento Orientado a Objetos em Java, sob orientação do Prof. Msc. Edson Yanaga.

MARINGÁ  
2005

RODRIGO GUEDES DE SOUZA

**DESENVOLVIMENTO DE UMA APLICAÇÃO FRENTE DE CAIXA  
AUTOMATIZADA PARA O VAREJO UTILIZANDO JAVA E RFID**

Trabalho apresentado ao Centro Universitário de Maringá com requisito parcial para obtenção do título de Especialista em Desenvolvimento Orientado a Objetos Java, sob a orientação do Prof. Msc. Edson Yanaga.

Aprovado em: \_\_\_\_\_

**BANCA EXAMINADORA**

---

**Prof. Msc. Edson Yanaga**  
(Centro Universitário de Maringá)

---

---

## **AGRADECIMENTOS**

Primeiramente agradeço a Deus, por me dar coragem, força, vontade e motivação para desenvolver e concluir este trabalho.

A minha família por estarem sempre perto de mim, a minha namorada por ter paciência ao agüentar meu stress e sempre estar ao meu lado quando precisava, me dando forças e esperanças. Ao meu orientador por ter me orientado e acreditado em minhas idéias.

## SUMÁRIO

<b>CENTRO UNIVERSITÁRIO DE MARINGÁ.....</b>	<b>1</b>
<b>CURSO DE PÓS-GRADUAÇÃO EM DESENVOLVIMENTO ORIENTADO A .....1</b>	<b>1</b>
<b>OBJETOS - JAVA.....1</b>	<b>1</b>
<b>RODRIGO GUEDES DE SOUZA.....1</b>	<b>1</b>
<b>DESENVOLVIMENTO DE UMA APLICAÇÃO FRENTE DE CAIXA AUTOMATIZADA PARA O VAREJO UTILIZANDO JAVA E RFID.....1</b>	<b>1</b>
<b>MARINGÁ.....2</b>	<b>2</b>
<b>2005 2</b>	
<b>RODRIGO GUEDES DE SOUZA.....3</b>	<b>3</b>
<b>DESENVOLVIMENTO DE UMA APLICAÇÃO FRENTE DE CAIXA AUTOMATIZADA PARA O VAREJO UTILIZANDO JAVA E RFID.....3</b>	<b>3</b>
<b>TRABALHO APRESENTADO AO CENTRO UNIVERSITÁRIO DE MARINGÁ COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE ESPECIALISTA EM DESENVOLVIMENTO ORIENTADO A OBJETOS EM JAVA, SOB ORIENTAÇÃO DO PROF. MSC. EDSON YANAGA.....3</b>	<b>3</b>
<b>MARINGÁ.....3</b>	<b>3</b>
<b>2005 3</b>	
<b>RODRIGO GUEDES DE SOUZA.....4</b>	<b>4</b>
<b>DESENVOLVIMENTO DE UMA APLICAÇÃO FRENTE DE CAIXA AUTOMATIZADA PARA O VAREJO UTILIZANDO JAVA E RFID.....4</b>	<b>4</b>
<b>TRABALHO APRESENTADO AO CENTRO UNIVERSITÁRIO DE MARINGÁ COM REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE ESPECIALISTA EM DESENVOLVIMENTO ORIENTADO A OBJETOS JAVA, SOB A ORIENTAÇÃO DO PROF. MSC. EDSON YANAGA.....4</b>	<b>4</b>
<b>APROVADO EM:.....4</b>	<b>4</b>
<b>BANCA EXAMINADORA.....4</b>	<b>4</b>
<b>.....4</b>	<b>4</b>
<b>PROF. MSC. EDSON YANAGA.....4</b>	<b>4</b>
<b>(CENTRO UNIVERSITÁRIO DE MARINGÁ).....4</b>	<b>4</b>
<b>.....4</b>	<b>4</b>
<b>.....4</b>	<b>4</b>

<b>AGRADECIMENTOS.....</b>	<b>5</b>
<b>SUMÁRIO.....</b>	<b>6</b>
<b>LISTA DE ILUSTRAÇÕES.....</b>	<b>15</b>
<b>LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS.....</b>	<b>16</b>
<b>RESUMO.....</b>	<b>18</b>
<b>1. INTRODUÇÃO.....</b>	<b>19</b>
<b><u>1.1 JUSTIFICATIVA.....</u></b>	<b><u>19</u></b>
<b><u>1.2 OBJETIVOS.....</u></b>	<b><u>20</u></b>
<b><u>1.3 METODOLOGIA.....</u></b>	<b><u>21</u></b>
<b><u>1.4 CONSIDERAÇÕES FINAIS.....</u></b>	<b><u>21</u></b>
<b>2. IDENTIFICAÇÃO POR RÁDIO FREQUÊNCIA.....</b>	<b>22</b>
<b><u>2.1 HISTÓRICO.....</u></b>	<b><u>22</u></b>
<b><u>2.2 A EVOLUÇÃO E NOVAS APLICAÇÕES.....</u></b>	<b><u>23</u></b>
<b><u>2.3 ARQUITETURA.....</u></b>	<b><u>25</u></b>
2.3.1 ETIQUETAS.....	26
2.3.1.1 Tipos de Etiquetas.....	27
2.3.2 ANTENA.....	28
2.3.3 LEITOR.....	28
2.3.4 RFID MIDDLEWARE.....	30
2.3.5 VISÃO GERAL DE UMA IMPLEMENTAÇÃO RFID.....	30
<b><u>2.4 CONSIDERAÇÕES FINAIS.....</u></b>	<b><u>31</u></b>
<b>3. EPCGLOBAL NETWORK.....</b>	<b>32</b>
<b><u>3.1 DEFINIÇÃO.....</u></b>	<b><u>32</u></b>
<b><u>3.2 EPC.....</u></b>	<b><u>33</u></b>
3.2.1 ESTRUTURA.....	33
3.2.2 TIPO DE EPC.....	34
3.2.3 COMPARATIVO EPC vs. UPC.....	34
<b><u>3.3 PML.....</u></b>	<b><u>35</u></b>
<b><u>3.4 CONSIDERAÇÕES FINAIS.....</u></b>	<b><u>36</u></b>
<b>4. SUN JAVA SYSTEM RFID.....</b>	<b>37</b>
<b><u>4.1 DEFINIÇÃO.....</u></b>	<b><u>37</u></b>
<b><u>4.2 INFRA-ESTRUTURA.....</u></b>	<b><u>37</u></b>
<b><u>4.3 ARQUITETURA.....</u></b>	<b><u>39</u></b>
1.1.1 RFID EVENT MANAGER.....	39
1.1.2 RFID INFORMATION SERVER.....	43
1.1.3 REQUISITOS.....	44
<b>PLATAFORMAS SUPORTADAS.....</b>	<b>44</b>

REQUISITOS DE HARDWARE.....	44
ESPAÇO EM DISCO.....	44
SOLARIS 9 OS (SPARC E X86).....	44
CPU: 500MHZ OU SUPERIOR.....	44
1 OU 2 CPU'S.....	44
RAM: 1GB.....	44
EVENT MANAGER – 150MB.....	44
INFORMATION SERVER – 100MB.....	44
MANAGEMENT CONSOLE – 50 MB.....	44
SOLARIS 10 OS (SPARC E X86).....	44
CPU: 500MHZ OU SUPERIOR.....	44
1 OU 2 CPU'S.....	44
RAM: 1GB.....	44
EVENT MANAGER – 150MB.....	44
INFORMATION SERVER – 100MB.....	44
MANAGEMENT CONSOLE – 50 MB.....	44
RED HAT ENTERPRISE LINUX ES, VERSION 3.....	44
CPU: 500MHZ OU SUPERIOR.....	44
1 OU 2 CPU'S.....	44
RAM: 1GB.....	44
EVENT MANAGER – 150MB.....	44
INFORMATION SERVER – 100MB.....	44
MANAGEMENT CONSOLE – 50 MB.....	44
<b><u>4.4 CONSIDERAÇÕES FINAIS .....</u></b>	<b><u>45</u></b>
<b>5. IMPLEMENTAÇÃO DA APLICAÇÃO.....</b>	<b>46</b>
<b><u>5.1 PROJETO.....</u></b>	<b><u>46</u></b>
<b><u>5.2 SOFTWARE E FERRAMENTAS UTILIZADAS.....</u></b>	<b><u>47</u></b>
<b><u>5.3 REQUISITOS.....</u></b>	<b><u>48</u></b>
<b><u>5.4 ARQUITETURA.....</u></b>	<b><u>48</u></b>
<b><u>5.5 COMUNICAÇÃO ENTRE RFIDPDV E RFID INFORMATION SERVER.....</u></b>	<b><u>50</u></b>
<b>TABELA 5 INFORMAÇÕES DISPONIBILIZADAS DO RFID INFORMATION SERVER.....</b>	<b>51</b>

<b>TABELA:</b> .....	<b>51</b>
<b>SENSOR</b> .....	<b>51</b>
<b>DESCRIÇÃO:</b> .....	<b>51</b>
<b>CONTÉM INFORMAÇÕES DE SENSORES</b> .....	<b>51</b>
<b>CLASSE DE ACESSO DA API CLIENTE:</b> .....	<b>51</b>
<b>COM.SUN.AUTOID.EPCIS.CLIENT.SENSOR</b> .....	<b>51</b>
<b>CONTEÚDO</b> .....	<b>51</b>
<b>DESCRIÇÃO</b> .....	<b>51</b>
<b>EPC 51</b>	
<b>NÚMERO EPC DO SENSOR, DIFERENTE DO EPC DA ETIQUETA</b> .....	<b>51</b>
<b>NAME</b> .....	<b>51</b>
<b>NOME DO SENSOR</b> .....	<b>51</b>
<b>TYPE</b> .....	<b>51</b>
<b>TIPO DE SENSOR (LEITORA, ANTENA, LEITORA VIRTUAL)</b> .....	<b>51</b>
<b>DESCRIPTION</b> .....	<b>51</b>
<b>DESCRIÇÃO DO SENSOR</b> .....	<b>51</b>
<b>GLN 51</b>	
<b>NÚMERO DA LOCALIZAÇÃO GLOBAL</b> .....	<b>51</b>
<b>IP ADDRESS</b> .....	<b>51</b>
<b>NÚMERO IP DO SENSOR NA REDE</b> .....	<b>51</b>
<b>TABELA:</b> .....	<b>51</b>
<b>UNIT51</b>	
<b>DESCRIÇÃO:</b> .....	<b>51</b>
<b>CONTÉM INFORMAÇÕES DE ITENS</b> .....	<b>51</b>
<b>CLASSE DE ACESSO DA API CLIENTE:</b> .....	<b>51</b>
<b>COM.SUN.AUTOID.EPCIS.CLIENT.UNIT</b> .....	<b>51</b>
<b>CONTEÚDO</b> .....	<b>51</b>
<b>DESCRIÇÃO</b> .....	<b>51</b>
<b>EPC 51</b>	
<b>NUMERO EPC DO OBJETO</b> .....	<b>51</b>
<b>SERIAL_NUM</b> .....	<b>51</b>
<b>NUMERO SERIAL DO OBJETO</b> .....	<b>51</b>

<b>SSCC51</b>	
<b>SERIAL SHIPPING CONTAINER CODE.....</b>	<b>51</b>
<b>IS_ACTIVE.....</b>	<b>51</b>
<b>INDICADOR SE O ITEM ESTA ATIVO OU NÃO.....</b>	<b>51</b>
<b>UNIT_TYPE.....</b>	<b>51</b>
<b>TIPO DO ITEM RELACIONADO COM A TABELA CONTAINERTYPE.....</b>	<b>51</b>
<b>OWNER_ID.....</b>	<b>51</b>
<b>NÚMERO PARA O TIPO DE RECIPIENTE DO ITEM RELACIONADO COM A TABELA ORGANIZATION.....</b>	<b>51</b>
<b>PRODUCT_ID.....</b>	<b>51</b>
<b>CÓDIGO DO PRODUTO RELACIONADO COM A TABELA PRODUCT.....</b>	<b>51</b>
<b>MANUFACTURE_DATE.....</b>	<b>51</b>
<b>DATE DE FABRICAÇÃO DO PRODUTO.....</b>	<b>51</b>
<b>EXPIRY_DATE.....</b>	<b>51</b>
<b>DATA DE VENCIMENTO DO PRODUTO.....</b>	<b>51</b>
<b>TABELA:.....</b>	<b>51</b>
<b>PRODUCT.....</b>	<b>51</b>
<b>DESCRIÇÃO:.....</b>	<b>51</b>
<b>CONTÉM INFORMAÇÕES DO PRODUTO.....</b>	<b>51</b>
<b>CLASSE DE ACESSO DA API CLIENTE:.....</b>	<b>51</b>
<b>COM.SUN.AUTOID.EPCIS.CLIENT.PRODUCT.....</b>	<b>51</b>
<b>CONTEÚDO.....</b>	<b>51</b>
<b>DESCRIÇÃO.....</b>	<b>51</b>
<b>PRODUCT_ID.....</b>	<b>51</b>
<b>CÓDIGO IDENTIFICADOR DO PRODUTO.....</b>	<b>51</b>
<b>OBJECT_CLASS.....</b>	<b>51</b>
<b>CÓDIGO DA CLASSE DO PRODUTO.....</b>	<b>51</b>
<b>NAME.....</b>	<b>51</b>
<b>NOME DO PRODUTO.....</b>	<b>51</b>
<b>PART_NUMBER.....</b>	<b>51</b>
<b>OUTRO EXTRA EM CASO DE PRECISAR.....</b>	<b>51</b>
<b>GTIN51</b>	

<b>NÚMERO DE IDENTIFICAÇÃO PARA COMERCIO GLOBAL DO PRODUTO.....</b>	<b>51</b>
<b>DESCRIPTION.....</b>	<b>51</b>
<b>DESCRIÇÃO DO PRODUTO.....</b>	<b>51</b>
<b>IMAGE_URL.....</b>	<b>51</b>
<b>URL ONDE SE CONTEM A IMAGEM DO PRODUTO.....</b>	<b>51</b>
<b>MANUFACTURER_ID.....</b>	<b>51</b>
<b>CÓDIGO DO FABRICANTE RELACIONADA A TABELA ORGANIZATION.....</b>	<b>51</b>
<b>TABELA:.....</b>	<b>51</b>
<b>ORGANIZATION.....</b>	<b>51</b>
<b>DESCRIÇÃO:.....</b>	<b>51</b>
<b>CONTÉM INFORMAÇÕES DO FABRICANTE.....</b>	<b>51</b>
<b>CLASSE DE ACESSO DA API CLIENTE:.....</b>	<b>51</b>
<b>COM.SUN.AUTOID.EPCIS.CLIENT.ORGANIZATION.....</b>	<b>51</b>
<b>CONTEÚDO.....</b>	<b>51</b>
<b>DESCRIÇÃO.....</b>	<b>51</b>
<b>ORGANIZATION_ID.....</b>	<b>51</b>
<b>CÓDIGO DE IDENTIFICAÇÃO DO FABRICANTE.....</b>	<b>51</b>
<b>DOMAIN_MGR.....</b>	<b>51</b>
<b>CÓDIGO GID DO FABRICANTE ASSINADO PELA EPCGLOBAL.....</b>	<b>51</b>
<b>NAME.....</b>	<b>51</b>
<b>NOME DO FABRICANTE.....</b>	<b>51</b>
<b>DESCRIPTION.....</b>	<b>51</b>
<b>DESCRIÇÃO DO FABRICANTE.....</b>	<b>51</b>
<b>IMAGE_URL.....</b>	<b>51</b>
<b>URL ONDE SE ENCONTRA A IMAGEM DO LOGO DO FABRICANTE.....</b>	<b>51</b>
<b>TABELA:.....</b>	<b>52</b>
<b>CURRENT_OBSERVATION OU OBSERVATION_LOG.....</b>	<b>52</b>
<b>DESCRIÇÃO:.....</b>	<b>52</b>
<b>CURRENT_OBSERVATION: CONTÉM OBSERVAÇÕES DE ETIQUETAS QUE ESTÃO NO CAMPO DE SENSOR.....</b>	<b>52</b>
<b>OBSERVATION_LOG: CONTÉM LOGS DE ETIQUETA QUE PASSARAM PELO SENSOR. O OBSERVATION_LOG É PERMITIDO O ACESSO A LEITURA PARA ESTAS INFORMAÇÕES.....</b>	<b>52</b>

<b>CLASSE DE ACESSO DA API CLIENTE:</b> .....	<b>52</b>
<b>COM.SUN.AUTOID.EPCIS.CLIENT.OBSERVATION</b> .....	<b>52</b>
<b>CONTEÚDO</b> .....	<b>52</b>
<b>DESCRIÇÃO</b> .....	<b>52</b>
<b>SENSOR_EPC</b> .....	<b>52</b>
<b>NUMERO DO SENSOR QUE CAPTUROU A ETIQUETA</b> .....	<b>52</b>
<b>OBSERVATION_TYPE</b> .....	<b>52</b>
<b>TIPO DE OBSERVAÇÃO DO SENSOR</b> .....	<b>52</b>
<b>OBSERVATION_VALUE</b> .....	<b>52</b>
<b>VALOR DA OBSERVAÇÃO DO SENSOR NORMALMENTE CONTEM O NÚMERO EPC DO OBJETO</b> .....	<b>52</b>
<b>TIMESTAMP</b> .....	<b>52</b>
<b>DATA E HORA DE CAPTURA</b> .....	<b>52</b>
<b>ACTION</b> .....	<b>52</b>
<b>CONTÉM INFORMAÇÕES DE AÇÃO. ESTE CAMPO DEPENDE O TIPO DE FILTRO QUE ESTA SE UTILIZANDO PELO SENSOR. SE ESTIVER UTILIZANDO O TIPO DELTA O CAMPO PODE CONTER O VALOR TAGIN (O OBJETO ENTROU NO CAMPO DE LEITURA) OU TAGOUT (O OBJETO SAIU DO CAMPO DE LEITURA)</b> .....	<b>52</b>
<b>TABELA:</b> .....	<b>52</b>
<b>CONTAINERTYPE</b> .....	<b>52</b>
<b>DESCRIÇÃO:</b> .....	<b>52</b>
<b>CONTEM INFORMAÇÕES SOBRE OS TIPOS DE RECIPIENTES. PODE SER UMA PALLET, PACOTE OU UM ITEM</b> .....	<b>52</b>
<b>CLASSE DE ACESSO DA API CLIENTE:</b> .....	<b>52</b>
<b>COM.SUN.AUTOID.EPCIS.CLIENT.CONTAINERTYPE</b> .....	<b>52</b>
<b>CONTEÚDO</b> .....	<b>52</b>
<b>DESCRIÇÃO</b> .....	<b>52</b>
<b>NAME</b> .....	<b>52</b>
<b>NOME DO TIPO DE RECIPIENTE</b> .....	<b>52</b>
<b>MAX_CAPACITY</b> .....	<b>52</b>
<b>CAPACIDADE MÁXIMA PARA O RECIPIENTE</b> .....	<b>52</b>
<b>MIN_CAPACTIY</b> .....	<b>52</b>
<b>CAPACIDADE MÍNIMA PARA O RECIPIENTE</b> .....	<b>52</b>

IMAGE_URL.....	52
URL ONDE SE ENCONTRA A IMAGEM PARA O TIPO DE RECIPIENTE.....	52
<b><u>5.6 CONSULTANDO ETIQUETAS NO RFID INFORMATION SERVER.....</u></b>	<b><u>53</u></b>
<b><u>5.7 ALTERANDO INFORMAÇÕES NO RFID INFORMATION SERVER.....</u></b>	<b><u>56</u></b>
<b><u>5.8 FUNCIONAMENTO DA APLICAÇÃO.....</u></b>	<b><u>58</u></b>
<b><u>5.9 CONSIDERAÇÕES FINAIS .....</u></b>	<b><u>63</u></b>
<b>6. CONCLUSÃO.....</b>	<b>64</b>
<b><u>6.1 CONSIDERAÇÕES FINAIS.....</u></b>	<b><u>64</u></b>
<b><u>6.2 TRABALHOS FUTUROS.....</u></b>	<b><u>64</u></b>
<b>REFERÊNCIAS.....</b>	<b>66</b>
<b>BHUPTANI, MANISH; MORADPOUR, SHAHRAM. RFID FIELD GUIDE: DEPLOYING RADIO FREQUENCY IDENTIFICATION SYSTEMS. UPPER SADDLE RIVER, NJ, EDITORA PRENTICE HALL, 2005. 288P. ISBN 0-13-1853554.....</b>	<b>66</b>
<b>BROCK, DAVID L. THE COMPACT ELETRONIC PRODUCT CODE: A 64-BIT REPRESENTATION OF THE ELECTRONIC PRODUCT CODE. 2001, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MA, E.U.A. DISPONÍVEL EM WWW.AUTOIDLABS.ORG/WHITEPAPERS/MIT-AUTOID-WH-008.PDF. ACESSO EM: 27 SET. 2005.....</b>	<b>66</b>
<b>KULPES, SERGIO. AS LOJAS USAM A TECNOLOGIA A SEU FAVOR. E OS CONSUMIDORES TAMBÉM. 2005, DIÁRIO DO COMÉRCIO. DISPONÍVEL EM &lt;HTTP://WWW.DCOMERCIO.COM.BR/ESPECIAIS/AUTOMACAO/ASLOJAS.HTM&gt;. ACESSO EM: 16 OUT. 2005.....</b>	<b>66</b>
<b>MARCHETTI, R. Z. ; PRADO, PAULO. A AUTOMAÇÃO COMERCIAL E A SATISFAÇÃO DO CONSUMIDOR EM SUPERMERCADOS. IN: CLÁUDIO FELISONI DE ANGELO; JOSÉ A. G. DA SILVA. (ORG.). VAREJO COMPETITIVO. 1 ED. SÃO PAULO, 1996, V. 1. ISBN 8522424578.....</b>	<b>66</b>
<b>OLIVEIRA, BÁRBARA, SUPERMERCADO DO FUTURO JÁ FUNCIONA NA ALEMANHA. 2005, DIÁRIO DO COMÉRCIO. DISPONÍVEL EM: &lt;HTTP://WWW.DCOMERCIO.COM.BR/ESPECIAIS/AUTOMACAO/SUPERMERCADO.HTM&gt;. ACESSO EM: 16 OUT. 2005.....</b>	<b>66</b>
<b>SANTA CLARA (CA). SUN MICROSYSTEMS, INC. INSTALLATION GUID: SUN JAVA SYSTEM RFID SOFTWARE 2.0. SANTA CLARA, 2005. 18P.....</b>	<b>66</b>
<b>SOUZA BRUNO. JAVA &amp; JINI. 1999. SENAC, SÃO JOSÉ DO RIO PRETO, SP. DISPONÍVEL EM &lt;HTTP://WWW.JAVAMAN.COM.BR/APRES/JAVA&amp;JINI/INDEX.HTML&gt;. ACESSO EM: 01 OUT. 2005.....</b>	<b>66</b>
<b>SWEENEY, PATRICK J. RFID FOR DUMMIES. HOBOKEN, NJ, EDITORA WILEY, 2005. 388P. ISBN 0-7645-7910-X.....</b>	<b>66</b>

<b>YUAN, MICHAEL. TECNOLOGIA RFID E A INTERNET DE OBJETOS. REVISTA MUNDO JAVA. CURITIBA, N.11, 2005.....</b>	<b>66</b>
<b>66</b>	
<b>7. APÊNDICES.....</b>	<b>67</b>
A. APÊNDICE A - CÓDIGO FONTES DA APLICAÇÃO RFIDPDV.....	68

## LISTA DE ILUSTRAÇÕES

<b>FIGURA 1 HISTÓRIA DA RFID.....</b>	<b>23</b>
<b>FIGURA 2 EVOLUÇÃO DA RFID.....</b>	<b>25</b>
<b>FIGURA 3 ARQUITETURA BASE PARA UM SISTEMA RFID.....</b>	<b>26</b>
<b>FIGURA 4 MODELO DE ETIQUETAS.....</b>	<b>26</b>
<b>FIGURA 5 MODELO DE ANTENAS.....</b>	<b>28</b>
<b>TABELA 1 APLICAÇÕES DA RFID.....</b>	<b>29</b>
<b>FIGURA 6 CAMADA DA RFID.....</b>	<b>31</b>
<b>TABELA 2 TIPO DE EPC.....</b>	<b>34</b>
<b>TABELA 3 COMPARATIVO EPC VS. UPC.....</b>	<b>34</b>
<b>FIGURA 7 EXEMPLO DE INFORMAÇÕES GERADAS PELO SENSOR EM FORMATO PADRÃO PML.....</b>	<b>35</b>
<b>FIGURA 8 ARQUITETURA DO RFID EVENT MANAGER.....</b>	<b>42</b>
<b>FIGURA 9 SUN JAVA SYSTEM RFID FUNCIONANDO EM UMA REDE EPC.....</b>	<b>44</b>
<b>TABELA 4 REQUISITOS PARA MIDDLEWARE SUN JAVA SYSTEM RFID.....</b>	<b>44</b>
<b>FIGURA 10 DIAGRAMA DE ATIVIDADE.....</b>	<b>47</b>
<b>FIGURA 11 DIAGRAMA DE CLASSES DO SISTEMA RFIDPDV.....</b>	<b>49</b>
<b>FIGURA 12 CONEXÃO HTTP COM O RFID INFORMATION SERVER.....</b>	<b>53</b>
<b>FIGURA 13 CONSULTANDO EPC EM UM SENSOR.....</b>	<b>55</b>
<b>FIGURA 14 CONSULTANDO O PRODUTO.....</b>	<b>56</b>
<b>FIGURA 15 EXEMPLO: UPDATEREQUEST &amp; UPDATERESPONSE.....</b>	<b>57</b>
<b>FIGURA 16 EXEMPLO: DELETEREQUEST &amp; DELETERESPONSE.....</b>	<b>58</b>
<b>FIGURA 17 TELA INICIAL.....</b>	<b>59</b>
<b>FIGURA 18 OPÇÃO DE REMOÇÃO DE ITENS.....</b>	<b>59</b>
<b>FIGURA 19 RELA REMOÇÃO DE ITENS DA COMPRA.....</b>	<b>60</b>
<b>FIGURA 20 TELA PROCESSANDO ITENS.....</b>	<b>60</b>
<b>FIGURA 21 TELA DE VENDA.....</b>	<b>62</b>

**LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS**

AIG	E-ZPass Interagency Group
DAO	Data Access Object
EAS	Electronic Article Surveillance
EPC	Electronic Code Product
ERP	Enterprise Resource Planning
GIAI	Global Individual Asset Identifier
GLN	Global Location Number
GRAI	Global Reusable Asset Identifier
GTIN	Global Trade Item Number
HF	Frequency
HTTP	HyperText Transfer Protocol
IC	Circuito Integrado
ICU	Information and Communication University
IFF	Identification Friend-or-Foe
JAXB	Java Architecture for XML Binding
JDMK	Java Dynamic Management Kit
JMS	Java Message Service
JMX	Java Management Extensions
JSTL	Java Server Pages Standard Tag Library
LF	Low Frequency
NDC	National Drug Code
MIT	Massachusetts Institute of Technology
MVC	Model View Control
NRL	Naval Research Laboratory
ONS	Object Name Server
PDA	Personal Digital Assistants
PML	Physical Markup Language

RFID	Radio Frequency Identification
SNMP	Simple Network Management Protocol
TIRIS	Instruments Registration and Identification System
UHF	Ultra High Frequency
UPC	Universal Product Code
XML	Extense Markup Language
WSDP	Web Services Developer Pack
RF	Radio Frequency

## RESUMO

As longas filas nos caixas, do setor varejista, é um dos diversos fatores que causam constrangimentos e insatisfações a consumidores modernos, que esperam por um rápido atendimento na hora de efetuar suas compras. Por este motivo os clientes procuram outros estabelecimentos, na busca de acharem um ambiente ideal, sem transtornos. Deste modo um processo automatizado auxilia efetuar vendas rápidas e eficientes trazendo lucros para o negócio e satisfazendo o desejo do cliente. Neste momento a utilização da tecnologia de identificação por rádio frequência (RFID) é primordial, pois ela elimina a entrada de dados manuais tornando o processo de efetuação de venda mais rápida e precisa. Este trabalho tem, portanto, o objetivo de realizar um estudo sobre a tecnologia de identificação por rádio frequência e o Middleware SUN JAVA SYSTEM RFID para utilizarem como ferramentas. A partir dos resultados obtidos neste estudo, será desenvolvida uma aplicação desktop frente de caixa específica para o varejo, desenvolvido em JAVA para agilizar o processamento dos itens de venda, minimizando o problema das grandes filas de caixas.

Palavras-chave: Identificação por Rádio Frequência, JAVA, Fila de Caixa.

# 1. INTRODUÇÃO

## 1.1 JUSTIFICATIVA

Com a chegada da globalização, a competitividade no mundo dos negócios tornou-se cada vez mais competitiva. Junto a isto, houve uma alteração no hábito dos consumidores, devido às evoluções tecnológicas que se tornou parte das necessidades das pessoas.

Visando as mudanças, o setor varejista buscou alternativas que pudessem fazer seu diferencial dentre a vasta competitividade do mercado, entre estas inovações a automação comercial teve um fator muito importante no aspecto de buscar clientes modernos que buscam um ambiente de compra ideal, onde a alta tecnologia e familiaridade se encontrem.

“O consumidor quer ser tratado como alguém especial, em um ambiente arquitetonicamente interessante, com as de entretenimento e cultura, com funcionários educadíssimos e elegantes e que sabem tratar uma pessoa com a medida exata de intimidade e respeito. Ou então um lugar onde as pessoas desejam estar, para verem e serem vistas e onde o ato de comprar pode ser um detalhe”(Kulpes, 2005).

Num estudo feito por Prado e Marchett (1996), em relação à satisfação do consumidor de supermercado, um ponto que se destaca é que a eficácia nos serviços de caixa eleva o nível de satisfação dos consumidores. Deste modo, podemos observa que o caixa tem um papel importante para atrair cliente.

Atualmente a automação comercial oferecida ao caixa é feita com o auxílio de código de barras que ajudou a melhorar o processo de compra, mas mesmo assim ainda ocorrem algumas deficiências na classificação dos itens, pois o processo é manual,

podendo ocorrer erros na entrada de dados e dependendo do volume de itens, o tempo de processamento aumenta, formando-se assim as indesejadas filas nos caixas. Tais filas que influenciam a insatisfação dos consumidores, levando-os em muitas vezes a busca de outros estabelecimentos.

Em busca de fornecer uma solução para este problema a Identificação por Rádio Freqüência se encaixa perfeitamente, pois com ela é possível processar dezenas e até centenas de itens por segundo, diminuindo bruscamente ou até extinguindo as filas, tornando o processo de compra satisfatório, além do mais, trazendo diversos benefícios ao varejo.

Atualmente a RFID vem sendo utilizada em um baixo grau de implantação por grandes empresas varejistas. Como é o caso da Future Store, supermercado alemão pertencente ao grupo Metro Inc. que utiliza a identificação de etiquetas por Rádio Freqüência e Personal Digital Assistants (PDA) em seus carrinhos e gôndolas, e os clientes fazem o checkout sem a ajuda de funcionários. Tal mudança, em dois anos já identificou um aumento de 30% em sua base de clientes. Em uma prova de que os consumidores se acostumaram rápido com as novidades é que 85% deles já utilizam a tecnologia oferecida, e mesmo os clientes com idade acima de 60 anos não se intimidaram. (Oliveira, 2005).

A RFID fornece para o mercado varejista diversas vantagens: Agilização do processo de leitura "checkout" (leitura simultânea); Redução de contratação de Mão de Obra (processo automatizado); Aumento de precisão de leitura (venda mais eficiente); Verificação de itens com validade vencida (aumenta credibilidade com clientes) entre outros;

## 1.2 OBJETIVOS

O Objetivo da pesquisa é realizar um estudo sobre a tecnologia de RFID que é utilizada para identificação de objetos através de Rádio Freqüência e um Middleware RFID que facilite a implementação de uma aplicação.

Também o objetivo deste, a implementação de uma aplicação desktop frente de caixa multiplataforma desenvolvida em Java em conjunto com o middleware Sun Java System RFID com o intuito de automatizar o processo de classificação de itens, diminuindo as filas de caixas, satisfazendo as necessidades dos clientes.

### 1.3 METODOLOGIA

No capítulo dois deste trabalho, é apresentada uma definição sobre a tecnologia RFID, mostrando um breve histórico, seus conceitos, sua evolução e qual seu estado atual. No capítulo três é apresentado detalhes do padrão internacional para RFID. No capítulo quarto, é apresentado o Middleware Sun Java System RFID, apresentando sua estrutura, tecnologia utilizada, funcionamento para que possa ser desenvolvida uma aplicação. O capítulo quinto, é apresentado com base nos estudos dos capítulos anteriores, uma aplicação implementada para resolver o problema proposto neste trabalho. E para finalizar, no capítulo sexto é feita uma conclusão sobre os resultados obtidos.

### 1.4 CONSIDERAÇÕES FINAIS

Neste capítulo foram mostrados os motivos da realização, bem como os seus objetivos. No capítulo seguinte é feita uma introdução ao tema Identificação por Rádio Freqüência, apresentando assim a base para o entendimento deste estudo proposto.

## 2. IDENTIFICAÇÃO POR RÁDIO FREQUÊNCIA

Este capítulo será apresentado à tecnologia de identificação por rádio frequência, descrevendo como ela surgiu, como funciona e como é utilizada. O presente capítulo foi baseado nas referências (Bhuptani, 2005) do livro **RFID Field Guide: Deploying Radio Frequency Identification Systems** e o livro **RFID For Dummies** (Sweeney, 2005)

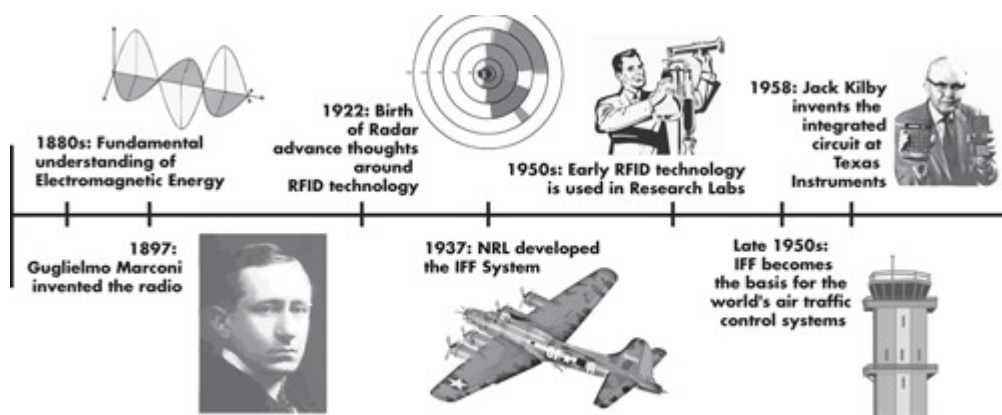
### 2.1 HISTÓRICO

A tecnologia RFID que significa Identificação por Rádio Frequência (Radio Frequency Identification) é utilizada para transmissão de informações de um determinado objeto através de ondas de rádio com o objetivo de identificação.

Apesar de muitas pessoas acharem que seja uma tecnologia recente, ela teve suas raízes em 1897 quando Guglielmo Marconi inventou o rádio, e se difundiu na década de 30, quando o Exército e Marinha Americana tiveram o desafio de projetar um sistema que fosse capaz de identificar objetos no chão, no mar e no ar, para que unidades aliadas pudessem ser distinguidas de inimigos em combates, então em 1937 o Laboratório de Pesquisa Naval Norte Americano (NRL) desenvolveu o Identification Friend-or-Foe (IFF) ao qual se tornou base para o Controle de Tráfego Aéreo e se tornou precursor do que é hoje a RFID.

Este tipo de tecnologia ficou restrito ao exército até os anos 50 devidos ao grande volume de equipamento e o seu alto custo, a partir de então ela foi estudada aprofundada mente em outros laboratórios não militares, e também em paralelo o surgimento de outras tecnologias como de circuitos integrados, chips de memórias, microprocessadores e linguagens de programação forneceram suporte para sua evolução.

Nos meados dos anos 60 e 70 as empresas Sensormatic e Checkpoint Systems desenvolveram uma das primeiras aplicações baseadas em RFID que fazia o controle de segurança de artigos por meio eletrônico chamado de Electronic Article Surveillance (EAS) que utilizava uma etiqueta passiva (sem uso de energia) de 1 bit acopladas aos objetos que ao passarem por um dispositivo de leitura que fica localizada na saída do recinto, acionava um alarme alertando os funcionários de roubos ou em outras situações acionando câmeras de vídeo para capturar o movimento do objeto. Assim percebe-se que até hoje são utilizadas comumente estas tecnologias em diversos locais como supermercados, bibliotecas, livrarias e etc. Na figura abaixo temos uma linha histórica do surgimento da RFID.



**Figura 1 História da RFID.**

## 2.2 A EVOLUÇÃO E NOVAS APLICAÇÕES

A difusão da RFID em outros setores se deu a partir dos anos 70 devido à integração de circuitos integrados aos seus componentes, ao qual se aplicou a possibilidade de leitura de objetos em um campo mais distante do dispositivo de leitura de sinal, também houve uma evolução no tempo de leitura, isso possibilitou o surgimento de novas aplicações para RFID como sistema de monitoramento de rebanho de gado e

monitoração de caminhões de transporte. Apesar de surgirem novas aplicações muitas delas eram patenteadas.

No início dos anos 90 houve um marco importante da tecnologia, onde empresas de pedágios dos Estados Unidos se juntaram e criaram o E-Zpass Interagency Group (AIG) onde foram definidos padrões de interoperabilidade entre diversas concessionárias de pedágio onde definiram padrões de frequências, protocolos de operação e hardware de comunicação que até então não existiam tais padrões. Tal sistema permitia que carros pudessem ser identificados por diversas praças de pedágios de concessionárias diferentes, utilizando o mesmo identificador, então substituindo mecanismos de identificação manual por códigos ao qual era utilizada.

A evolução se concretizou quando a empresa Texas Instruments deu uma nova visão à Tecnologia RFID criando a Texas Instruments Registration and Identification System (TIRIS) onde ao qual implementava novas funcionalidades. Nesta época ocorreu também à evolução das etiquetas passivas que trabalhavam em Baixa frequência (LF) e Alta Frequência (HF) ao qual era muito restrito devido a sua pouca distância de leitura, criaram então, uma etiqueta de Frequência Extremamente Alta (UHF) que permitiu o estouro na utilização comercial, pois com este tipo de etiqueta era possível ter centenas e até milhas de etiquetas ao mesmo tempo passando pelo leitor e com alta velocidade de leitura devido a alta frequência alcançadas pelas etiquetas.

E em 1999 foi fundado pelo MIT o Centro Auto-ID que tinha o objetivo de tornar a tecnologia RFID viável para cadeia de suprimentos, devido a necessidade que empresas varejistas já tinham naquele momento em ter visibilidade em seus produtos, pois as quais já haviam analisado os benefícios na utilização no departamento de defesa dos Estado Unidos desde o início dos anos 90. E em 2000 o Centro Auto-ID desenvolveu sob coordenação Dr. David Brock o EPC que significa Código Eletrônico de Produto (Electronic Product Code) que tinha como objetivo principal a substituição do Código de Barra (UPC).

E em 2003 empresas dos Estados Unidos e Europa junto com o Centro de Auto-ID, sentiram a necessidade de se criar um órgão com os mesmos princípios do W3C (World Wide Web Consortium) responsável pela padronização da tecnologia Web e criaram a EPCGlobal que tem o objetivo de ser o principal órgão responsável pela padronização da tecnologia RFID, deste modo o já criado Centro de Auto-ID passou toda sua base de pesquisa ao EPCGlobal e passou a se chamar Auto-ID Labs com o intuito de aproximar as instituições universitárias de pesquisas ao EPCGlobal criando novas tecnologias. Atualmente o Auto-ID Labs é formada por sete universidades mundiais, MIT (Estados Unidos), Universidade de Cambridge (Inglaterra), Universidade de Adelaide (Australia), Universidade de Keio (Japão), Universidade de Fudan (China), Universidade de St. Gallen (Alemanha) e ICU (Korea) com propostas diferentes mas tendo a mesma visão sobre a tecnologia.

A Figura abaixo mostra um histórico a evolução e novas aplicações da RFID.

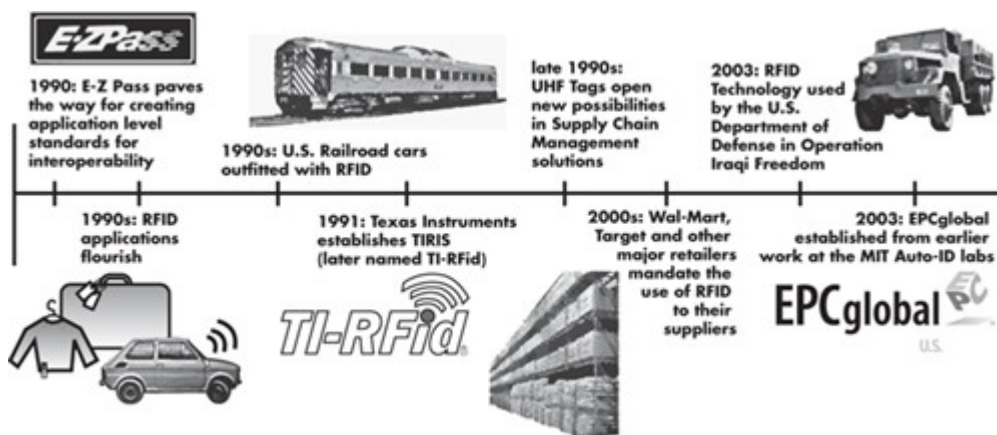
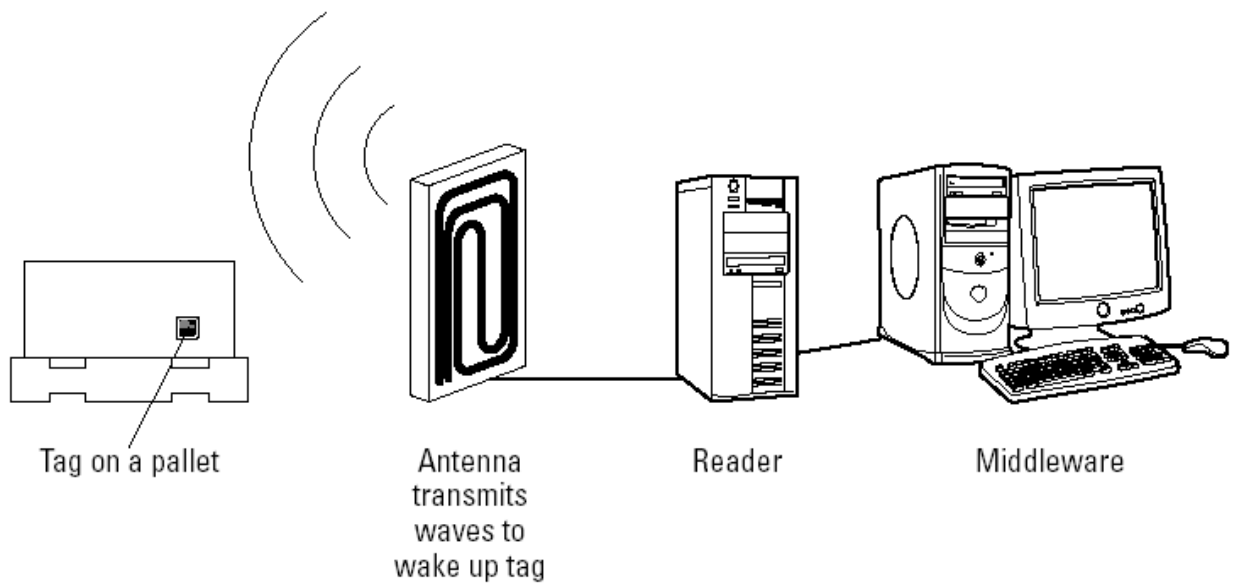


Figura 2 Evolução da RFID

## 2.3 ARQUITETURA

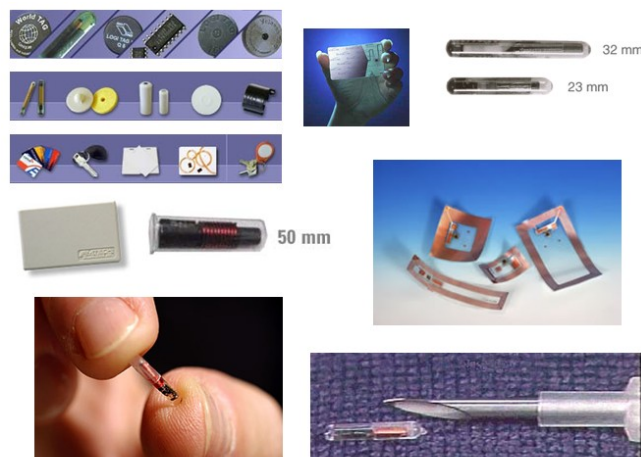
Basicamente a arquitetura da tecnologia RFID se baseia nos seguintes componentes: Etiqueta, Antena, Leitora e Middleware.



**Figura 3 Arquitetura Base para um Sistema RFID**

### 2.3.1 Etiquetas

A Etiqueta, RF Tag ou simplesmente Tag é um componente que tem a função de armazenar informações do objeto e retransmitir através de ondas de rádio quando interrogada pelo leitor. Atualmente existem no mercado diversos modelos de etiqueta que podem ser acoplada ou injetada ao objeto, algumas do tamanho de uma ponta de lápis. Na figura 4 temos exemplos de modelos de etiquetas.



**Figura 4 Modelo de Etiquetas**

Existem dois tipos de etiquetas:

**Baseada por IC** (Circuito Integrado) que contém memória e microprocessador acoplados as etiquetas. Este tipo de etiqueta tem a capacidade de processar informação quando interrogada pelo leitor. Sua capacidade de armazenamento varia de acordo com as especificações.

**Chipless** diferente da etiqueta baseada por IC, contém somente memória para armazenar 1 bit, este tipo de memória é comumente utilizado para controle de segurança de objetos.

### 2.3.1.1 Tipos de Etiquetas

Existem três tipos de etiquetas baseada por IC:

**Ativa:** é um tipo de etiqueta que possui uma bateria integrada que permite o envio de sinal, deste modo ela propaga seu sinal a uma área maior, normalmente este tipo de etiqueta permite regravação dos dados. Ela é comumente usada para rastreamento de cargas. Sua utilização foi muito limitada, pois não houve uma padronização para este tipo de etiqueta, sendo assim ela não é muito utilizada em aplicações que envolvem a rede EPC.

**Passiva:** é um tipo de etiqueta que não se necessita de bateria, ela utiliza o próprio sinal enviado pela leitora para re-enviar o sinal de retorno contendo os dados do objeto, e por este motivo seu campo de leitura é muito pequeno variando da leitora. A maioria das aplicações atuais utiliza esta etiqueta como padrão na rede EPC.

**Semi-Passiva:** é um tipo de etiqueta que se utiliza da bateria para operação do chip da etiqueta e também se utiliza do sinal da leitora para re-enviar o sinal. Esta etiqueta

por necessitar da onda de rádio do leitor, seu campo também é restrito a área da leitura.

### 2.3.2 Antena

A Antena é encontrada tanto nas etiquetas quanto na leitora e tem a função de servir como canal de comunicação entre os dispositivos através de sinais de rádio. Na etiqueta a antena é conectada diretamente ao seu microprocessador, já no dispositivo de leitura a antena pode estar conectada diretamente ao próprio dispositivo ou pode estar conectada a grandes distâncias da leitora. Na figura 7 temos modelos de antenas.



Figura 5 Modelo de Antenas

### 2.3.3 Leitor

O Leitor tem a função de criar um campo eletromagnético (rádio frequência) e processar todos os dados de etiquetas capturados pela antena ao passar pelo seu campo de leitura e retransmitir a um computador através de rede ethernet ou wireless

dependendo do dispositivo de leitura. Além de poder ler etiquetas, existem alguns modelos de leitores que tem função de alterar o conteúdo da etiqueta.

### Frequência de Transmissão e Recepção

A frequência tem um papel muito importante para a comunicação entre etiquetas e leitora. Quando vai ser implantado um sistema utilizando RFID, existem diversos fatores que devem ser observados quanto ao tipo de aplicação, normas nacionais e especificações. Na Tabela abaixo é demonstrado um panorama sobre as operações de frequência para cada tipo de aplicação.

**Tabela 1 Aplicações da RFID**

<b>Frequência</b>	<b>Características</b>	<b>Tipo de Aplicação</b>
Baixa Frequência (Low Frequency LF) Menos de 135Khz	Em uso desde 1980 Trabalha melhor sobre objetos que contenha metal e líquidos Baixa taxa de transferência Área de leitura medida em polegadas	Identificação Animal Automatização Industrial Controle de Acesso
Alta Frequência (High Frequency HF) 13.56Mhz	Em uso em meados de 1990 Padrão Mundial Maior área de leitura do que a de baixa frequência Menor custo sobre a de baixa frequência Melhor desempenho sobre metal e líquido	Smart Label Controle de Acesso Anti-Pirataria Localização de diversos itens com livros, bagagem, vestuário, etc. Prateleira Inteligente Identificação e Monitoramento de Pessoas
Ultra Frequência (Ultra High Frequency UHF) 433Mhz e 860 a 930Mhz	Em uso logo após a década de 90 Maior área de leitura do que a de alta frequência Longa área de transmissão para 433Mhz Grande Aceitação do mercado mundial devido à utilização da cadeia de suprimentos e varejo Incompatibilidade entre padrões nacionais Sujeito à interferência de metais e líquidos	Fornece a Cadeia de Suprimentos e Logística: - Controle de Inventário - Administração de Estoque - Localização de Itens
Microonda 2.45GHz e 5.8GHz	Alta taxa de transferência Comumente utilizado para etiquetas ativas e semi-passivas	Controle de Acesso Coleta eletrônica de Pedágios Automação Industrial

	Área de leitura similar a de UHF Perda de desempenho sobre metal e líquido	
--	---	--

#### 2.3.4 RFID Middleware

O Middleware é um software que tem a função de gerenciar os dados capturados pelas leitoras e também tem a função de integrar outros sistemas externos ao sistema de RFID como, por exemplo, um sistema ERP. Veremos em detalhe mais aprofundado no próximo capítulo.

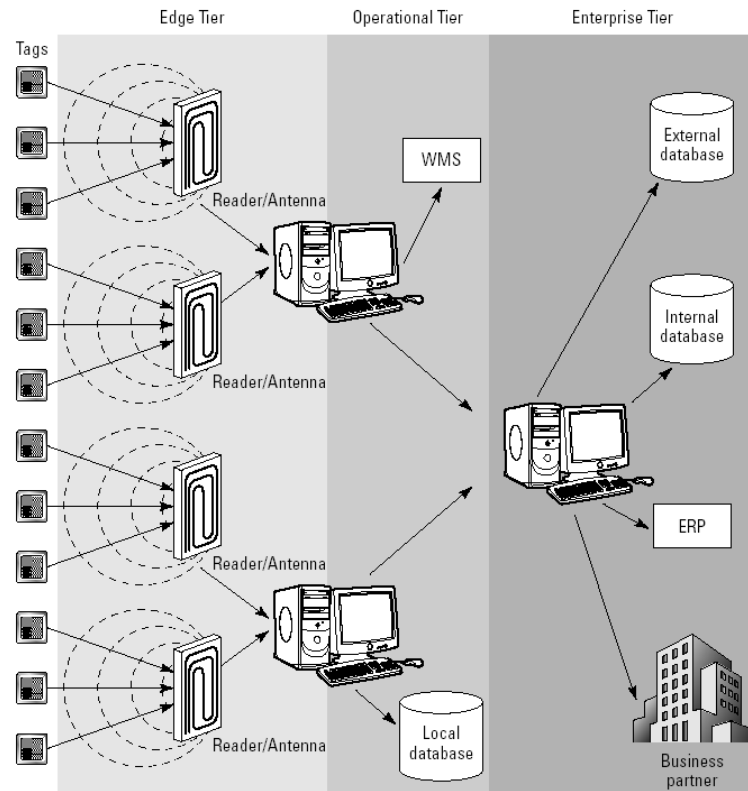
#### 2.3.5 Visão Geral de uma Implementação RFID

Após ter visto todos os componentes da RFID podemos classificar em 3 camadas uma aplicação utilizando a arquitetura:

**Camada de Transferência de Dados:** Nesta camada estão contidos os diversos modelos de etiquetas e leitoras.

**Camada Operacional:** Nesta camada intermediária se encontra o middleware que faz a integração entre as diversas leitoras com os diversos sistemas existentes.

**Camada de Negócio:** Nesta camada encontra-se toda a infra-estrutura de uma empresa que se utiliza da RFID. Esta camada se caracteriza por sua heteriogêndade devido a diversos tipos de plataformas e sistemas muitas vezes legados. A figura 6 representa as camadas de uma aplicação.



**Figura 6 Camada da RFID**

## 2.4 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os principais conceitos da arquitetura RFID, mostrando seus componentes e a tecnologia em sua arte atual, percebendo-se assim que a identificação por rádio frequência não é uma tecnologia da moda, pois ela sofreu por um grande período, uma intensa pesquisa evolutiva por diversos setores ao qual possibilitou que hoje ela esteja em um estado padronizado e confiável. Já no próximo capítulo será apresentado o EPCGlobal Network responsável pela atual padronização da arquitetura de Identificação por Rádio Frequência para a cadeia de suprimentos.

### 3. EPCGLOBAL NETWORK

#### 3.1 DEFINIÇÃO

A EPCGlobal Network ou chamada por alguns de Rede EPC (EPC Network), teve seu lançamento em setembro de 2003, ela é um conjunto de padrões abertos criada pela organização EPCGlobal Inc. com o intuito de se tornar uma especificação padrão para implementações RFID no mercado mundial. A rede EPC utiliza-se da RFID como base para seu funcionamento, com ela é possível ter visibilidade e informações sobre os itens em toda cadeia de suprimentos, com isto, criando uma rede de informações em tempo real entre empresas, fornecedores, parceiros e clientes.

Apesar da proposta da Rede EPC ter sido focalizada para empresas que necessitasse de visibilidade e rastreabilidade em tempo real para seus ativos na sua cadeia de suprimentos ela pode ser aplicado a outras áreas.

A Rede EPC se caracteriza pelos seus cinco elementos básicos:

**Número EPC:** Identificador global e único, que serve com um ponteiro para realizar consultas sobre um objeto que ele identifica.

**Etiqueta EPC (tag):** Portador de dados do EPC que se comunica com as leitoras por RF. É constituído por memória, microprocessador e por uma antena.

**Reader/Leitora:** Dispositivo de captura de dados; portátil ou fixo (instalado), que se conecta-se à rede EPC.

**Savant (EPC Middleware):** Software que controla as leitoras. Pode funcionar com um repositório local de números EPCs e informações associadas.

**ONS (Object Name Service):** recurso distribuído que “conhece” onde as informações associadas ao número EPC podem ser encontradas (assim como o DNS para internet).

**EPC-IS (EPC Information Service):** Serviço de informações de EPCs que mantém todos os dados relativos a um EPC . (Utiliza o PML que é o vocabulário definido em XML, para permitir consultas e obter dados relacionados aos números EPCs).

### 3.2 EPC

O principal propósito do EPC é servir como um indicador de referência na rede EPC usando em conjunto, o ONS, para vincular o objeto físico as suas informações escrito em uma linguagem de marcação física chamada de PML. (Brock, 2001)

O EPC não se limita somente a produtos, ele pode ser empregado a qualquer objeto individualmente ao qual se queira ter visibilidade sobre o mesmo. Ele também foi projetado para integrar e facilitar a migração entre os padrões já existentes como os GTIN (Global Trade Item Number), GIAI (Global Individual Asset Identifier), SSCC (Serialized Shipping Container Code), GRAI (Global Reusable Asset Identifier), GLN (Global Location Number) e NDC (National Drug Code).

#### 3.2.1 Estrutura

O EPC é constituído por quatro partes fundamentais independente de qual versão seja: Header, EPCManager, Object Class e o Serial Number.

**Header:** Este campo contém o número da versão EPC utilizada pela etiqueta que serve para o leitor identificar o tamanho de cada campo podendo assim decodificar.

**EPC Manager:** Este campo contém o código do fabricante ou a entidade por manter os códigos subsequentes. Ele equivale ao prefixo da companhia no sistema EAN.UCC.

**Object Class:** Este campo contém o identificador que classifica o tipo do objeto por classes. Ele equivale ao código de referência do artigo no sistema EAN.UCC.

**Serial Number:** Este campo contém o identificador único do objeto que pode ser definido de acordo a cada classe do Objeto.

### 3.2.2 Tipo de EPC

Atualmente existem sete tipos de EPC com os tamanhos de 64, 96 e 256 bits como descritos na tabela abaixo.

**Tabela 2 Tipo de EPC**

Descrição	Tipo	Header (bit)	EPC Manager (bit)	Object Class (bit)	Serial Number (bit)
EPC-64	Tipo I	2	21	17	24
	Tipo II	2	15	13	34
	Tipo III	8	26	13	23
EPC-96	Tipo I	8	26	24	36
EPC-256	Tipo I	8	32	56	192
	Tipo II	8	64	56	128
	Tipo III	8	128	56	64

### 3.2.3 Comparativo EPC vs. UPC

Fazendo um comparativo entre o EPC e o UPC pode encontrar tais diferenças descritas da tabela 2.

**Tabela 3 Comparativo EPC vs. UPC**

Características	UPC	EPC
Baseado em Padrões	Sim	Sim
Identificação única por Item	Não	Sim
Operação automática	Não	Sim
Modo de Leitura	Óptica – Com área de visão da leitura	Rádio Frequência sem área de visão do leitor
Tipo de Leitura	Leitura Sequencial	Leitura Simultânea

Possibilidade de Escrita	Não	Sim (dependendo do tipo de etiqueta)
Funções Adicionais	Sim	Não
Segurança	Baixa	Alta
Custo Inicial	Baixo	Alto
Custo de Manutenção	Baixo	Alto
Vida Útil	Alta	Baixa
Dados Armazenados	Baixa	Alta

### 3.3 PML

A PML (Physical Markup Language) é uma linguagem de marcação baseada em xml que faz parte das especificações criada pela EPCGlobal Inc. que tem o objetivo de representar e distribuir informações de objetos na Rede EPC. Atualmente a especificação que defini os elementos da linguagem se encontra na versão 1.0

Os dados formatados nesta especificação podem ser encontrados tanto na comunicação, entre sensores e middleware, quanto no compartilhamento de informações fora da empresa utilizando o EPC Information Service um exemplo desses dados podemos visualizar na Figura 7 abaixo.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <!--Elemento Raiz PML Core -->
<Sensor xmlns="urn:autoid:specification:interchange:PMLCore:xml:schema:1">
  <!--O EPC do leitor especificado no arquivo de configuração de Leitores do RFID Event Manager -->
  <ns1:ID xmlns:ns1="urn:autoid:specification:universal:Identifier:xml:schema:1">urn:epc:id:gid:1.1.3</ns1:ID>
  <!--Elemento Raiz Observation que são leitura efetuado pelo leitor -->
  <Observation>
    <ns2:ID xmlns:ns2="urn:autoid:specification:universal:Identifier:xml:schema:1">4</ns2:ID>
    <!--Data Hora em que o RFID Event Manager capturou o evento EPC gerado pelo leitor -->
    <DateTime>2004-05-27T15:36:25.758-07:00</DateTime>
    <!--Campo de comando do Filtro do leitor recebido pelo Execution Agent-->
    <!--Neste caso o PML foi gerado filtro do tipo Delta, Poderia ter dois tipos de comando TagsIn our TagsOut -->
    <Command>TagsIn</Command>
    <!--Elemento Tag identificado pelo leitor -->
    <Tag>
      <!--O EPC da etiqueta identificada -->
      <ns3:ID xmlns:ns3="urn:autoid:specification:universal:Identifier:xml:schema:1">
        urn:epc:id:gid:1.1.209
      </ns3:ID>
    </Tag>
  </Observation>
</Sensor>
```

Figura 7 Exemplo de informações geradas pelo sensor em formato padrão PML

A PML consiste em dois tipos:

**PML Core** é utilizada para descrever os dados gerados pela infra-estrutura local da Rede EPC, ela contém informações únicas a cada objeto como, por exemplo, data de leitura efetuada pelos sensores.

**PML Extension ou Savant Extension** é utilizada para integrar informações que não é gerada pela infra-estrutura da Rede EPC como informações relacionadas ao objeto, por exemplo, data de validade. Ela é utilizada na integração com outros sistemas da empresa que são geradas pelo EPC Information Service.

### 3.4 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado o padrão EPCGlobal Network, mostrando suas especificações criadas pela EPCGlobal Inc. para implantação de sistemas baseados em RFID. No próximo capítulo será apresentado o middleware Sun Java System RFID escolhido para a implementação do sistema frente de caixa.

## **4. SUN JAVA SYSTEM RFID**

### **4.1 DEFINIÇÃO**

O Sun Java System RFID é um software criado pela empresa SUN Microsystems sobre especificações definidas pela EPCGlobal Inc. para facilitar a implementação de um infra-estrutura baseada na Rede EPC. Este software representa de forma unificada os componentes EPC Middleware e EPC Information Service na arquitetura da Rede EPC. Ele foi desenvolvido sobre a linguagem Java.

A função do Sun Java System RFID é processar e filtrar fluxo de EPCs capturados pelos leitores e fornecer estas informações de forma fácil e padronizada a outras aplicações corporativas que fazem parte da cadeia de suprimentos como por exemplo sistemas ERP's, Sistemas de Gerenciamento de Armazéns entre outros.

### **4.2 INFRA-ESTRUTURA**

Atualmente a última versão do Sun Java System RFID é a 2.0. Ela é composta por diversas tecnologias que fazem com que ela seja um sistema altamente robusto devido à credibilidade que as tecnologias que a compõem têm sobre o mundo. Elas são definidas abaixo:

#### **Jini 2.0**

Jini é uma tecnologia aberta desenvolvida pela Sun que trabalha com sistemas distribuídos altamente plugavel. Os componentes do Jini são os serviços aos quais podem ser tanto software quando hardware em uma rede.

Jini é um sistema distribuído, baseado na idéia de uma federação de grupos de usuários e recursos. O objetivo final é tornar a rede em uma ferramenta flexível e facilmente administrável, na qual clientes humanos e computacionais possam facilmente encontrar e utilizar recursos (Software ou Hardware). (Bruno Souza 1999)

Na arquitetura Sun Java System RFID esta tecnologia pode ser encontrada no módulo Sun Event Manger, para facilitar a instalação e administração dos leitores de etiquetas.

### **Rio 3.1**

Rio é um subprojeto do JINI que tem a função de prover uma arquitetura de rede adaptável e dinâmica. Ele permite o gerenciamento de serviços na rede automaticamente, identificando alterações e tomando devidas decisões autônomas, para deixar a rede estável e funcional. Através dele o JINI se torna plugavel, pois ele permite descobrir dinamicamente novos Jini Lookup Service que estão disponíveis, como também permite descobrir serviços que estão fora do ar ou que houve algum problema e não forneça mais serviço.

O Rio junto com o JINI trabalha verificando o estado do Sun Java System RFID, ajudando a descobrir novos componentes disponíveis na rede “Leitores” e descobrir componentes não operantes devido a alguma falha.

### **Java Web Services Developer Pack 1.5**

O Java Web Services Developer Pack (Java WSDP) é um conjunto de ferramentas integradas e gratuitas que permite desenvolvedores Java implementarem e testarem aplicações web com XML e web services. O Java WSDP engloba as APIs de Java para XML, Java Architecture for XML Binding (JAXB), JavaServer Faces, Web Services Interoperability Sample Application, Web Services Security, JavaServer Pages Standard Tag Library (JSTL) e Java WSDP Registry Server. As ferramentas Ant

Build Tool e Apache Tomcat container também fazem parte deste pacote distribuído pela Sun.

### **Java Dynamic Management Kit 5.1**

O Java Dynamic Management Kit (JDMK) é um produto da Sun Microsystems orientado para o desenvolvimento de soluções de gestão de sistemas de Rede. Sendo mais orientado para a construção de agentes de gestão nos próprios elementos de rede, baseia-se numa infra-estrutura modular guiada por serviços de gestão distribuídos por toda a rede. Este pacote oferece integração com vários protocolos de gestão (destacando-se o SNMP) e permite, por exemplo, comunicar com aplicações de gestão baseadas neste protocolo. Deve no entanto mencionar-se que os agentes do JDMK não são agentes móveis, ainda que possam ser dinamicamente instalados nos elementos de rede por mecanismos push/pull.

## **4.3 ARQUITETURA**

O Sun Java System RFID é composto pelos módulos RFID Event Manager e RFID Information Service.

### **1.1.1 RFID Event Manager**

O RFID Event Manager é um modulo que tem a função de capturar, filtrar informações extras a todos os dados vindos de um ou de diversos dispositivos de leitura em uma rede EPC e disponibilizar estas informações para o modulo RFID Information Server, visto na Figura 8. Ele pode ser montado em uma estrutura distribuída, sendo que seus componentes “Leitores” podem estar em localizações geograficamente distintas. Devido a esta compressibilidade de ser distribuído, por trabalhar ao extremo devido a

grande leitura de EPC's ao mesmo tempo e em alguns casos, conter centenas de leitores que por algum motivo pode sofrer falhas, sua estrutura foi desenvolvida em cima da tecnologia Jini e Rio provendo uma alta escalabilidade, confiança e disponibilidade em suas operações.

O RFID Event Manager consiste em dois componentes como visto na figura 8, o Control Station e o Execution Agent.

O **Execution Agent** (Agente de Execução) é responsável por filtrar as informações, ele é o elemento que fica ouvindo os eventos EPC gerados por um leitor e repassa estas informações ao Control Station. Ele trabalha de acordo com as configurações do tipo de adaptador, tipo de filtro e tipo de logger.

As opções de algoritmo de filtragem que os Execution Agent são:

- BandPass: Filtra somente as etiquetas que passarem por leitores específicos predefinidos.
- Delta: É um tipo de filtro que informa ao Control Station quando etiqueta entra e sai do campo de leitura.
- EPCFilter: Filtra somente as etiquetas específicas predefinidas.
- Smoothing: É um tipo de filtro que identifica a etiqueta somente quando receber "n" eventos do leitor, este filtro é usado devido os leitores não terem 100% de precisão sobre a leitura das etiquetas.

As opções de Logger usada pelo Execution Agent são:

- FileLogger: Transfere as informações PML Core em um arquivo.
- HttpPmlLogger: Transfere as informações PML Core via conexão http.
- JMSLogger: Transfere as informações PML core via JMS usando fila ou tópico.
- SocketLogger: Cria uma conexão socket e transfere as informações PML Core via Socket.
- SsocketLogger: Cria uma conexão socket do tipo servidor e transfere as informações PML core via socket cliente permitidas.

O **Control Station** (Estação de Controle) é responsável por gerenciar a inicialização de diversos Execution Agent, distribuir e monitorar a carga de trabalho entre os agentes.

O control Station consiste em cinco componentes:

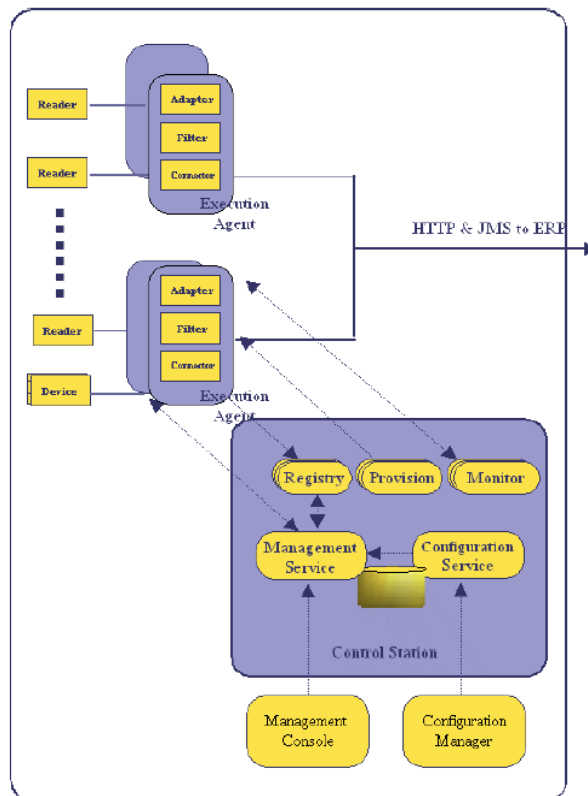
- Control Station Registry: é um JIni Lookup Server que registra os Execution Agent.

- Control Station Code Provisioner: Utiliza o Rio Webster para prover arquivos JAR necessários para executar a carga de trabalho.

- Control Station Workload Monitor: Utiliza o Rio Provisioner/Monitor que é responsável por monitorar os Execution Agent e distribuir a carga de trabalho.

- Configuration Service: é um repositório onde se encontra informações de configuração dos componentes. Suas informações são persistidas em banco de dados.

- Management Service: é um ponto de acesso para toda a administração do Event Manager provendo acesso via SNMP e JMX. O Management Service se utiliza da base de dados do configuration service.



**Figura 8** Arquitetura do RFID Event Manager.

O Sun Java System RFID possui duas ferramentas que ajudam o administrador da Rede EPC administrar o RFID Event Manager:

**Management Console** é uma aplicação de interface visual com acesso via web, que tem a finalidade de administrar os leitores e componentes do RFID Event Manager. O Management Console acessa as informações via Management Service para poder estar monitorando todos os componentes. Veja Figura 8.

**Configuration Manager** é uma aplicação de interface visual que tem o objetivo de montar a Semântica de Processos Empresariais (Business Processing Semantics - BPS) que é responsável pela infra-estrutura de funcionamento dos componentes “Serviços JIni”.

### 1.1.2 RFID Information Server

O RFID IS (Information Server) é um módulo desenvolvido sobre a arquitetura J2EE que tem a função de servir como uma camada de integração que agrega valor aos eventos EPCs gerado pelo RFID Event Manager e fornecer informações de alto nível a aplicações externas.

O IS (Information Server) permite somente rodar em cima do servidor de aplicação da Sun (Sun Java System Application Server 7 e 8) e o servidor da BEA (WebLogic). E seus dados são persistidos em RDBMS (Oracle ou PostgreSQL).

O IS possibilita a integração dos eventos de negócio EPC a outros sistemas através de dois tipos de comunicação: HTTP ou via JMS utilizando comunicação síncrona sem estado. Para facilitar o acesso as informações o RFID IS fornece um conjunto de bibliotecas que permite desenvolvedores utilizá-las para efetuarem manipulações de dados.

Segundo Michael Yuan (2005) “O RFID Information Server é tipicamente usado para traduzir um conjunto de observações de baixo nível em funções de negócio de alto nível”.

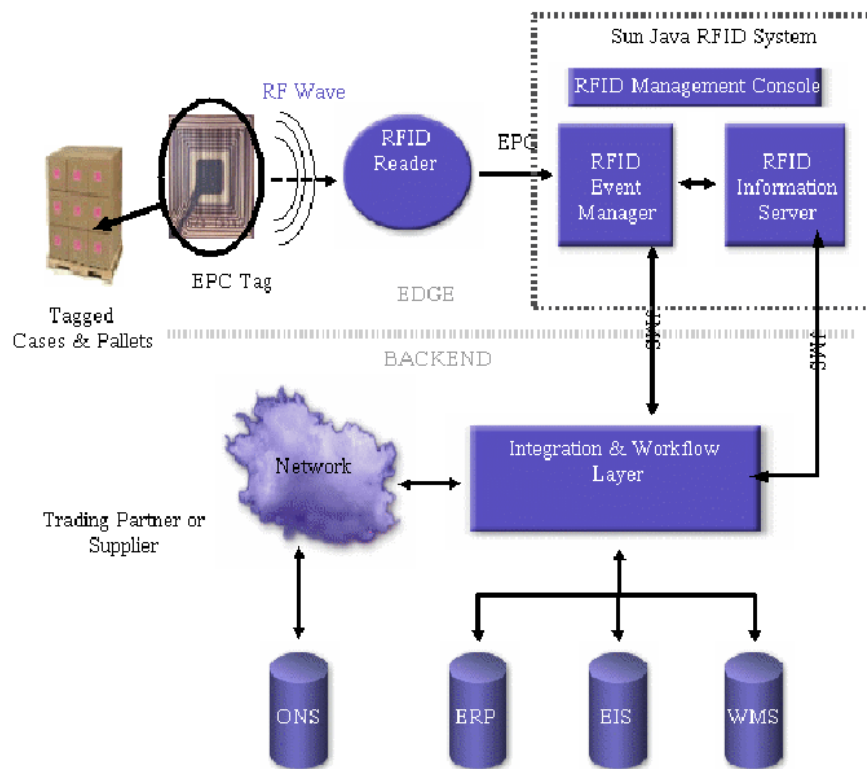


Figura 9 Sun Java System RFID funcionando em uma Rede EPC

### 1.1.3 Requisitos

Os requisitos necessários para instalação do middleware de acordo com o Guia de Instalação fornecido pela Sun Microsystems, Inc, 2005 são:

Tabela 4 Requisitos para Middleware Sun Java System RFID

Plataformas Suportadas	Requisitos de Hardware	Espaço em disco
Solaris 9 OS (Sparc e x86)	CPU: 500Mhz ou Superior 1 ou 2 CPU's RAM: 1GB	Event Manager – 150MB Information Server – 100MB Management Console – 50 MB
Solaris 10 OS (Sparc e x86)	CPU: 500Mhz ou Superior 1 ou 2 CPU's RAM: 1GB	Event Manager – 150MB Information Server – 100MB Management Console – 50 MB
Red Hat Enterprise Linux ES, Version 3	CPU: 500Mhz ou Superior 1 ou 2 CPU's RAM: 1GB	Event Manager – 150MB Information Server – 100MB Management Console – 50 MB

#### 4.4 CONSIDERAÇÕES FINAIS

Neste capítulo foi mostrado a infra-estrutura do middleware escolhido para a implementação da frente de caixa automatizado proposto pelo estudo. Observou-se que o middleware Sun Java System RFID proporciona uma infra-estrutura altamente integrada as especificações da EPCGlobal Network, formada sobre diversas tecnologias de renome, deste modo proporciona aos profissionais a confiança de escolher este middleware na hora de montar uma estrutura RFID baseada em padrões. No próximo capítulo será apresentado o protótipo frente de caixa proposto pelo estudo.

## 5. IMPLEMENTAÇÃO DA APLICAÇÃO

Neste capítulo será apresentado um aplicativo desenvolvido com base no problema proposto no presente trabalho.

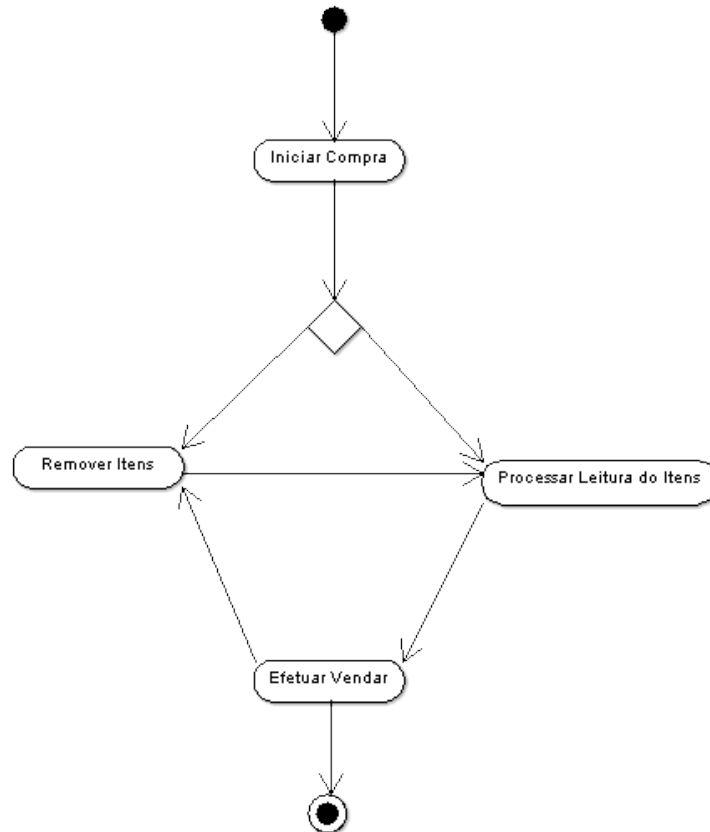
### 5.1 PROJETO

#### **Descrição e Proposta**

O aplicativo desenvolvido se chama RFIDPDV que tem o objetivo de agilizar e automatizar o processo de classificação de itens em uma loja do setor varejista.

O RFIDPDV é uma aplicação cliente com interface desktop multiplataforma, que trabalha sobre a borda da rede EPC e que utiliza informações do módulo Information Server do Middleware Sun Java System RFID para processamento das informações.

Para se ter uma visão sobre a proposta do sistema, pode-se observar na Figura 10 que representa um diagrama de atividades utilizando a UML, que descrevem as ações em que ocorrem em uma compra, utilizando-se de uma arquitetura RFID. Num primeiro momento o usuário do sistema “consumidor” da loja varejista inicia o processo de compra, no segundo momento o sistema permitirá que o usuário possa tomar uma decisão de remover algum item da compra ou processar os itens que estão no campo de leitura. Caso o usuário decida pela remoção de alguns itens, o sistema parará no estágio de remoção e em um próximo momento prosseguirá com o processamento dos itens. E do terceiro e último momento o sistema permitirá a efetuação da compra dos itens.



**Figura 10 Diagrama de Atividade**

O aplicativo desenvolvido tem como objetivo, somente agilizar o processo de classificação dos produtos em uma compra de supermercado e também possibilitará avisar o cliente quando existir um item em que seu prazo de validade esteja vencido, confirmado-se assim os benefícios que a identificação por rádio frequência pode trazer a este ambiente problemático, deste modo ele não terá outros recursos como pagamento da compra e relatórios dos itens da compra.

## 5.2 SOFTWARE E FERRAMENTAS UTILIZADAS

Para o desenvolvimento e testes da aplicação RFIDPDV, foram utilizadas as seguintes ferramentas e software:

Java 2 SDK Standard Edition versão 1.4.2

NetBeans 5.

API Client de conexão com RFID Information Server fornecido pelo Sun Java System RFID.

Servidor de Aplicação Sun Application Server 8.1

Banco de Dados PostgreSQL 7.2.8

Sistema Operacional Suse Linux 3.1 Professional

### 5.3 REQUISITOS

#### **Requisitos da Aplicação**

Devido à aplicação ser desenvolvida em Java e ser uma linguagem multiplataforma, não existe uma restrição para o sistema operacional, desde que o qual tenha suporte a máquina virtual Java e esteja instalado. Como descrito do capítulo 2, a aplicação RFIDPDV não tem acesso direto as leitoras RFID, sendo assim não é necessário a instalação de drivers para o dispositivo de leitura.

#### **Requisitos da Arquitetura**

Para que a aplicação RFIDPDV funcione deve ser montada uma arquitetura RFID de acordo com a figura 9, deste modo deve ser instalado o Middleware Sun Java System RFID em uma máquina servidora. Os requisitos para instalação do Middleware podem ser observados na tabela 4.

### 5.4 ARQUITETURA

No desenvolvimento do projeto foram abordados dois Designer Patterns DAO (Data Access Objeto) e MVC (Model View Controller) que ajudaram no desenvolvimento.

A arquitetura das classes do sistema é dividida em cinco pacotes como mostra a figura 11.

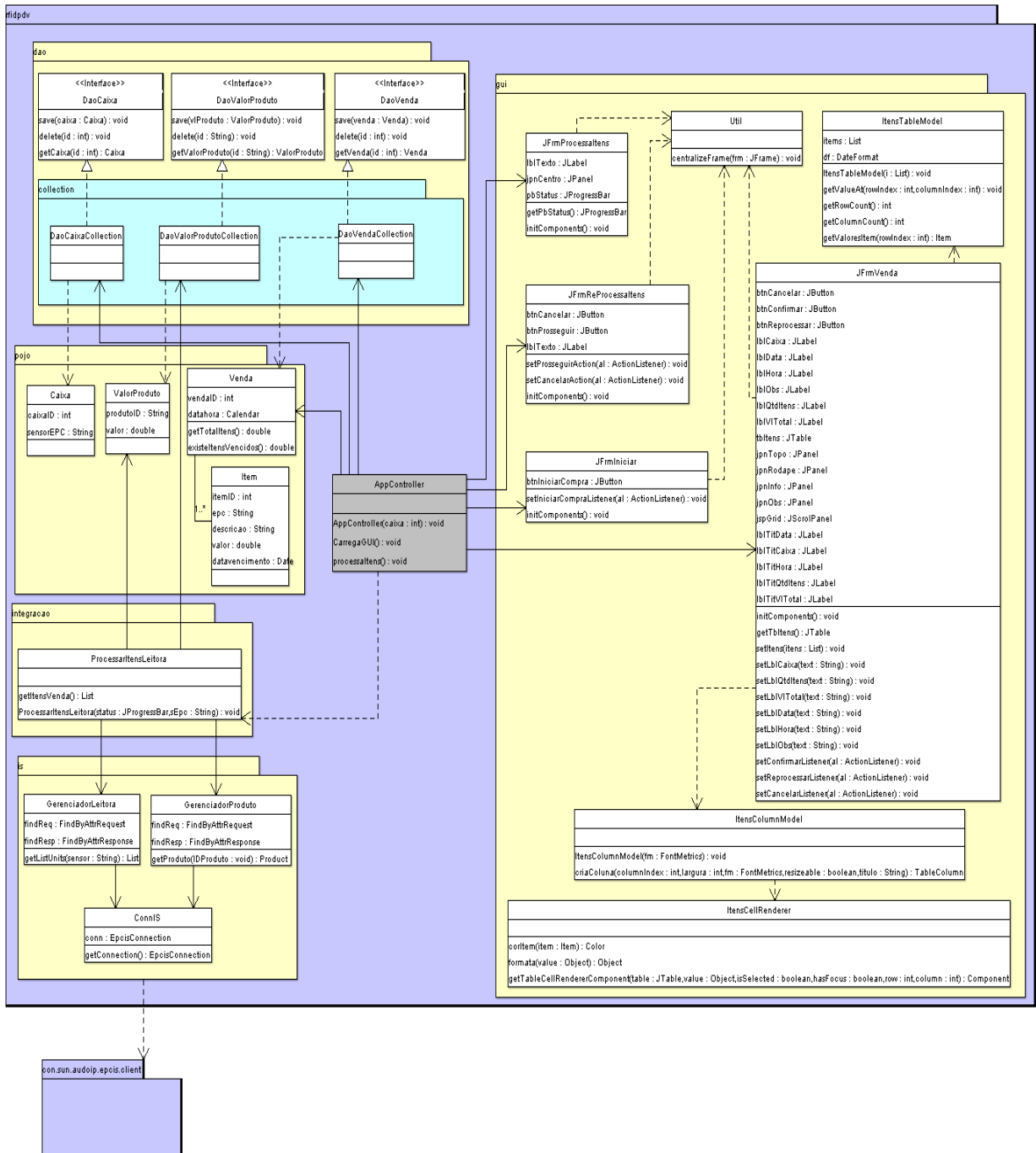


Figura 11 Diagrama de Classes do Sistema RFIDPDV

No pacote **dao**, contém as interfaces para implementação dos objetos que gerenciam a persistência dos objetos de negócio. Dentro do pacote dao, existe um sub-pacote chamado **collection** que implementa as interfaces Dao.

No pacote **pojo**, contém as classes de negócios.

No pacote **is**, contém as classes que são utilizadas para acessarem o RFID Information Server, estas classes utilizam-se da API cliente fornecida pelo Middleware.

No pacote **gui**, contém as classes relacionadas à interface gráfica da aplicação.

No pacote **integração**, contém as classes que unifica as informações obtidas pela leitora com as informações das regras de negócios da aplicação.

No pacote raiz **rfidpdv** contém a classe ApplicationController que tem o objetivo de controlar as ações entre a interface gráfica com as regras de negócios, aplicando-se assim o pattern MVC.

## 5.5 COMUNICAÇÃO ENTRE RFIDPDV E RFID INFORMATION SERVER

### **Informações fornecidas pelo RFID Information Server**

A aplicação RFIDPDV é um cliente consumidor dos serviços fornecidos pelo RFID Information Server, que podem ser acessado via serviço JMS ou HTTP através de XML. O conteúdo de informações disponibilizado do Banco de Dados pelo RFID Information Server pode ser visto na Tabela 4, estas informações são acessadas via classes da API Cliente.

Tabela 5 Informações Disponibilizadas do RFID Information Server

<b>Tabela:</b>		Sensor
<b>Descrição:</b>		Contém informações de Sensores
<b>Classe de Acesso da API Cliente:</b>		Com.sun.autoid.epcis.client.Sensor
<b>Conteúdo</b>		<b>Descrição</b>
EPC	Número EPC do Sensor, diferente do EPC da Etiqueta	
Name	Nome do Sensor	
Type	Tipo de Sensor (Leitora, Antena, Leitora Virtual)	
Description	Descrição do Sensor	
GLN	Número da Localização Global	
IP ADDRES	Número ip do sensor na rede	
<b>Tabela:</b>		Unit
<b>Descrição:</b>		Contém informações de itens
<b>Classe de Acesso da API Cliente:</b>		Com.sun.autoid.epcis.client.Unit
<b>Conteúdo</b>		<b>Descrição</b>
EPC	Numero EPC do Objeto	
Serial_Num	Numero Serial do Objeto	
SSCC	Serial Shipping Container Code	
Is_Active	Indicador se o item esta ativo ou não	
Unit_Type	Tipo do Item relacionado com a tabela ContainerType	
Owner_Id	Número para o tipo de recipiente do item relacionado com a tabela Organization	
Product_id	Código do Produto relacionado com a tabela Product	
Manufacture_date	Date de fabricação do Produto	
Expiry_Date	Data de Vencimento do Produto	
<b>Tabela:</b>		Product
<b>Descrição:</b>		Contém informações do Produto
<b>Classe de Acesso da API Cliente:</b>		Com.sun.autoid.epcis.client.Product
<b>Conteúdo</b>		<b>Descrição</b>
Product_Id	Código Identificador do Produto	
Object_Class	Código da Classe do Produto	
Name	Nome do Produto	
Part_number	Outro extra em caso de precisar	
GTIN	Número de itendificação para comercio global do produto	
Description	Descrição do produto	
Image_Url	Url onde se contem a imagem do produto	
Manufacturer_Id	Código do Fabricante relacionada a tabela Organization	
<b>Tabela:</b>		Organization
<b>Descrição:</b>		Contém informações do fabricante
<b>Classe de Acesso da API Cliente:</b>		Com.sun.autoid.epcis.client.Organization
<b>Conteúdo</b>		<b>Descrição</b>
Organization_Id	Código de Identificação do Fabricante	
Domain_Mgr	Código GID do fabricante assinado pela EPCGlobal	
Name	Nome do Fabricante	
Description	Descrição do Fabricante	
Image_Url	Url onde se encontra a Imagem do Logo do Fabricante	

<b>Tabela:</b>		Current_Observation ou Observation_Log
<b>Descrição:</b>		Current_Observation: Contém observações de etiquetas que estão no campo de Sensor. Observation_Log: Contém logs de etiqueta que passaram pelo Sensor. O Observation_Log é permitido o acesso a leitura para estas informações
<b>Classe de Acesso da API Cliente:</b>		Com.sun.autoid.epcis.client.Observation
<b>Conteúdo</b>		<b>Descrição</b>
Sensor_EPC	Numero do Sensor que capturou a etiqueta	
Observation_Type	Tipo de Observação do Sensor	
Observation_Value	Valor da Observação do Sensor normalmente contem o número EPC do objeto	
TimeStamp	Data e Hora de Captura	
Action	Contém informações de Ação. Este campo depende o tipo de filtro que esta se utilizando pelo sensor. Se estiver utilizando o tipo Delta o campo pode conter o valor TagIn (o objeto entrou no campo de leitura) ou TagOut (o objeto saiu do campo de leitura)	
<b>Tabela:</b>		ContainerType
<b>Descrição:</b>		Contem informações sobre os tipos de Recipientes. Pode ser uma Pallet, pacote ou um item.
<b>Classe de Acesso da API Cliente:</b>		Com.sun.autoid.epcis.client.ContainerType
<b>Conteúdo</b>		<b>Descrição</b>
Name	Nome do Tipo de Recipiente	
Max_Capacity	Capacidade Máxima para o recipiente	
Min_Capactiy	Capacidade Mínima para o Recipiente	
Image_Url	Url onde se encontra a imagem para o tipo de recipiente	

## Conexões

Como já dito, existem dois métodos de acessos ao RFID Information Server, via JMS ou HTTP+XML. A API Cliente traz uma classe chamada de **com.sun.autoid.epcis.client.EpcisConnection** que é responsável por gerenciar uma conexão com o servidor. Na figura 12 é mostrado um exemplo de conexão através de HTTP, este tipo de conexão foi utilizado para o desenvolvimento do RFIDPDV.

```

conn = new EpcisConnection(
    "http://localhost:8080/epcis/service" //URL do service RFID IS
    ,"http://localhost:" // Endereço do Proxy se Existir
    ,"8080" //Porta de Conexão
    ,"usuario" // login do usuário configurado no servidor de aplicação
    ,"123456"); // senha do usuário configurado no servidor de aplicação

```

**Figura 12 Conexão HTTP com o RFID Information Server**

A classe que é responsável por gerar esta conexão no RFIDPDV é a **rfidpdf.is.ConnIS** (ConnIS.java), ela possui um método chamado getConnection que retorna um objeto EpcisConnection já configurado para acessar via HTTP.

## 5.6 CONSULTANDO ETIQUETAS NO RFID INFORMATION SERVER

Na API Cliente fornecida pelo Middleware, existem duas classe para tratamento das informações no RFID Information Server, a FindByAttrRequest que é responsável por efetuar a consulta e a classe FindByAttrResponse que é responsável por processar o resultado obtido pelo objeto da classe FindByAttrRequest.

O RFIDPDV necessita somente de duas consulta ao RFID Information server para seu funcionamento, ao qual está descrito na figura 13, que mostra o código utilizado pela classe **rfidpdv.is.GerenciadorLeitora** (GerenciadorLeitora.java) que tem o objetivo de capturar todos as etiquetas que estão no campo de leitura de um determinado sensor e a outra classe **rfidpdv.is.GerenciadorProduto** que tem o objetivo de capturar um produto específico listada na figura 14.

Descrevendo o processo de captura de objetos que estão no campo de leitura do sensor mostrado da figura 13, primeiramente é criado um objeto da Classe FindByAttrRequest passando como parâmetro do construtor um Objeto EpcisConnection e o nome da Tabela que se deseja obter informações (A tabela 4 fornece uma descrição detalhada das opções disponíveis) neste caso "CURRENT\_OBSERVATION" que contém as etiquetas que estão no campo de leitura

do sensor. Depois é adicionada uma condição a consulta através do método `addCondition`, passando por parâmetro o nome do campo relacionado a tabela desejada no construtor, caso "SENSOR\_EPC", também o número EPC do sensor e um operador. Existem três tipos de operado: (eq) que representa "igual a"; (LT) que representa "maior que" e (GT) que representa "menor que". Em uma consulta podem ser adicionadas diversas condições. Depois de adiciona as condições e chamado método `process()` que envia informações a servidor e retorna um objeto do tipo `FyndByAttrResponse` ao qual é passado por referencia ao objeto `findResp`. E para capturar as informações utiliza-se do método `get+NomeDaTabela` que retorna uma lista de objetos. Após capturar todas as etiquetas que estão no campo de leitura do sensor selecionado é feito uma estrutura de repetição "For", onde para cada EPC localizada é pesquisado informações da tabela Unit para o código EPC e assim criando uma lista de Objetos de alto nível do tipo Unit.

```

package rfidpdv.is;

import com.sun.autoid.epcis.EPCISException;
import com.sun.autoid.epcis.client.FindByAttrRequest;
import com.sun.autoid.epcis.client.FindByAttrResponse;
import com.sun.autoid.epcis.client.Observation;
import com.sun.autoid.epcis.client.Unit;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;

public class GerenciadorLeitora {
    ConnIS conn = new ConnIS();
    FindByAttrRequest findReq = null;
    FindByAttrResponse findResp = null;

    public GerenciadorLeitora() { }

    //Retorna uma lista da Units que estao no campo da leitura (sensor)
    public List getListUnits(String sensor){
        List arrayUnits = new ArrayList();
        try{
            findReq = new FindByAttrRequest(conn.getConnection(),
"CURRENT_OBSERVATION");
            findReq.addCondition("SENSOR_EPC",sensor,"eq");
            findResp = findReq.process();
            FindByAttrRequest findReqUnit = null;
            FindByAttrResponse findRespUnit = null;
            for (int i=0;i<findResp.getObservations().size();i++){
                Observation obs = (Observation)
findResp.getObservations().get(i);
                findReqUnit = new FindByAttrRequest(conn.getConnection(),
"UNIT");
                findReqUnit.addCondition("EPC",obs.getObservationValue(),"eq");
                findRespUnit = findReqUnit.process();
                Unit unit = (Unit) findRespUnit.getUnits().get(0);
                arrayUnits.add(unit);
            }
        }catch(Exception e){
            JOptionPane.showMessageDialog(null,e.getMessage());
        }
        return arrayUnits;
    }
}

```

**Figura 13 Consultando EPC em um Sensor**

Na figura 14 é mostrado como o RFIDPDV captura um determinado produto. A consulta aos produtos é necessária, pois todas as informações referentes aos produtos são tratadas da base de dados do módulo RFID Information Server não pela aplicação RFIDPDV.

```

package rfidpdv.is;

import com.sun.autoid.epcis.client.FindByAttrRequest;
import com.sun.autoid.epcis.client.FindByAttrResponse;
import com.sun.autoid.epcis.client.Product;

public class GerenciadorProduto {
    ConnIS conn = new ConnIS();
    FindByAttrRequest findReq = null;
    FindByAttrResponse findResp = null;

    public GerenciadorProduto() { }

    public Product getProduto(String IDProduto){
        try{
            findReq = new FindByAttrRequest(conn.getConnection(), "PRODUCT");
            findReq.addCondition("PRODUCT_ID",IDProduto,"eq");
            findResp = findReq.process();
        }catch(Exception e){
        }
        return (Product) findResp.getProducts().get(0);
    }
}

```

**Figura 14 Consultando o Produto**

## 5.7 ALTERANDO INFORMAÇÕES NO RFID INFORMATION SERVER

A API Cliente possui também um conjunto de classe que possibilita que a aplicação possa alterar, excluir ou incluir alguma informação ao RFID Information Server. Apesar do sistema RFIDPDV não utilizar estes recursos, será descrito as principais classes.

### **Alteração e Inclusão**

As classes utilizadas para alteração e inclusão de uma informação são: `com.sun.autoid.epcis.client.UpdateRequest` e a classe `com.sun.autoid.epcis.client.UpdateRespose`.

A classe `UpdateRequest` é responsável por fazer a conexão com o RFID Information Server, passada como parâmetro em seu construtor um objeto do tipo `EpcisConnection`. O método `modify()` fornecida pela classe, permite fazer a alteração da tabela no RFID IS, passando como parâmetro um objeto de negócio listada na

tabela 4. O método add() é responsável com adicionar uma nova informação ao RFID IS, passando como parâmetro um objeto de negócio listada na tabela4.

A classe UpdateResponse é responsável por informar o status da alteração efetuada por um objeto do tipo UpdateRequest, ela recebe este objeto no seu construtor. O succes() fornecida pela classe, retorna um valor booleano que representa se a alteração ou inclusão da informação foi efetuada com sucesso, caso o método retorne false, dizendo que a operação não foi efetuada, é possível saber o motivo através do método getStatus() que retorna um texto dizendo o motivo da omissão do comando.

```
EpcisConnection conn = new EpcisConnection(...)

UpdateRequest updateReq = new UpdateRequest(conn);

Unit unit = new Unit();
unit.setEpc("urn:epc:id:gid:2.1.1");
unit.setExpiryDate(Calendar.getInstance());
unit.setProductId("1");
unit.setUnitType("ITEM");
dataUpdateResponse updateResp = updateReq.add(unit);
System.out.println("ModifyUnit. Success -> " +
updateResp.success() + ". Should be true");
```

**Figura 15 Exemplo: UpdateRequest & UpdateResponse**

## Exclusão

As classes utilizadas para exclusão de informações são: com.sun.autoid.epcis.client.DeleteRequest e com.sun.autoid.epcis.client.DeleteResponse.

A classe DeleteRequest é responsável por fazer a conexão com o RFID Information Server, passada como parâmetro em seu construtor um objeto do tipo EpcisConnection. Esta classe possui um método chamado deleteByPk() que permite a exclusão de uma informação ou um conjunto de informações, passando como parâmetro um ArrayList com objetos do tipo PrimaryKey que contém a chave primária

da informação e o tipo de informação fornecida na listagem 4. Veja exemplo na figura 16

A classe DeleteResponse têm as mesmas funcionalidades da classe UpdateResponse o qual é responsável por retornar informações sobre a operação.

```

DeleteRequest deleteReq = new DeleteRequest(conn);
PrimaryKey pk = new PrimaryKey("urn:epc:id:gid:1.402.1", "CONTAINMENT");
ArrayList pkList =new ArrayList();
pkList.add(pk);
DeleteResponse deleteResp = deleteReq.deleteByPk(pkList);

DeleteResponse deleteResp = deleteReq.deleteByPk(pkList);
System.out.println("Success -> " + deleteResp.success());

```

**Figura 16 Exemplo: DeleteRequest & DeleteResponse**

## 5.8 FUNCIONAMENTO DA APLICAÇÃO

Esta seção tem o objetivo de demonstrar o funcionamento da aplicação desenvolvida.

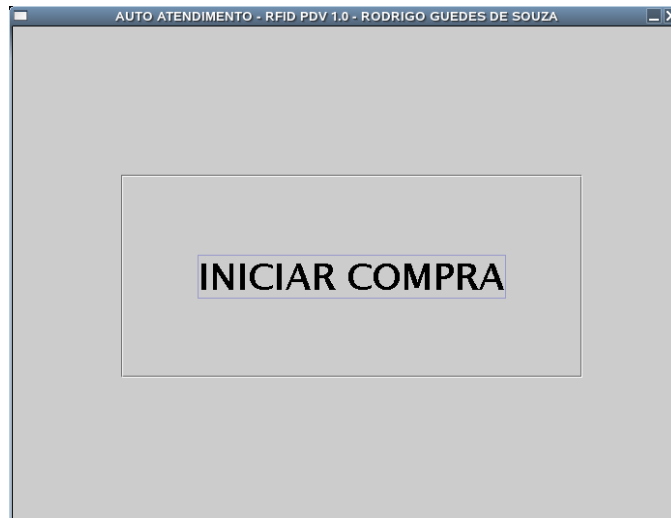
### Iniciar a Aplicação

Para iniciar o sistema deve ser informado o número do caixa a ser aberto.

Comando: java -jar RFIDPDV.jar 1

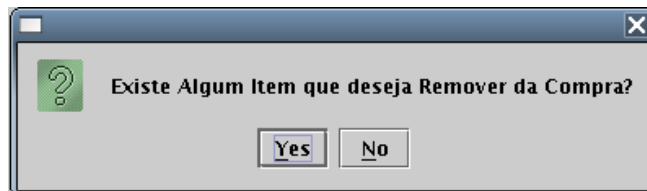
### Tela Inicial

Ao carregar a aplicação será mostrada a tela inicial do sistema como visto da figura 17, esta tela é o ponto inicial da aplicação. Neste momento a aplicação já configurou qual o sensor que o caixa deve buscar as informações, quando solicitar ao RFID Information Server. Para o usuário iniciar o processo de compra deve-se apertar o botão iniciar compra.



**Figura 17 Tela Inicial**

Logo após o usuário apertar o botão iniciar, será apresentada uma caixa de diálogo, visto na figura abaixo, onde o sistema pergunta para o usuário se existe algum item que ele deseja remover do carrinho de compra.



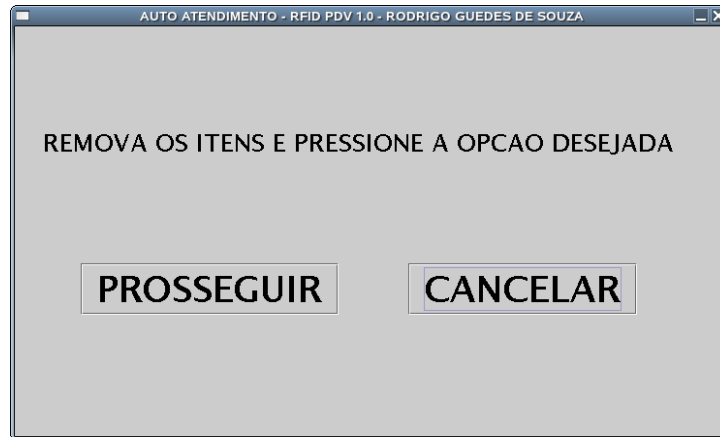
**Figura 18 Opção de Remoção de Itens**

Se o usuário apertar o botão Yes, será apresentado a Tela de Remoção de itens, Caso contrário será iniciado o processamento de itens.

### **Tela Remoção de Itens da Compra**

A tela de remoção de itens da compra permite que o usuário possa remover os itens do carrinho antes de efetuar o processamento de itens. A aplicação permite que o usuário prossiga após ter removido o item do carrinho ou cancele a operação de

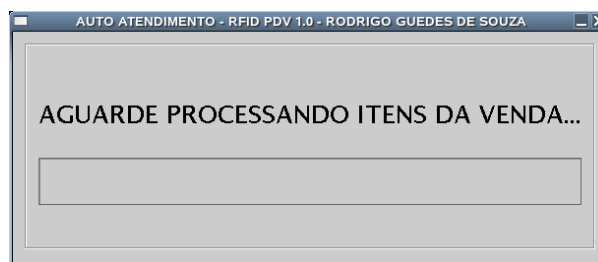
compra neste momento, voltando assim para a tela inicial. Se o usuário optar por prosseguir, neste momento serão processados os itens.



**Figura 19** Tela Remoção de Itens da Compra

### **Processando Itens**

Após o usuário ter adequado seu carrinho com os produtos, a aplicação começa a processar as etiquetas RF dos objetos que estão no campo da leitura (este processo pode ser visto na figura 13) apresentando assim a tela de status de leitura visto da figura 18.



**Figura 20** Tela Processando Itens

### **Tela Efetuação da Compra**

Após o processamento dos itens, será apresentada a tela de efetuação de compra onde conterão em uma grade, todos os itens que estão no carrinho, mostrando o

código EPC do objeto, descrição de produto relacionado ao objeto, valor e validade do produto.

Os itens contidos nesta grade, é gerada pelo método `getItensVenda()` da classe **rfidpdv.integracao.ProcessarItensLeitora** vista na figura 22, ao qual é responsável por integrar os dados de baixo nível (captura de etiquetas pelos sensores) com os dados de alto nível (valor do produto).

A aplicação consegue identificar os produtos vencidos automaticamente. Quando encontrado será apresentado o item na grade com a cor vermelha e será adicionada uma mensagem no campo observações, dizendo que existem itens vencidos na compra.

Nesta tela também é apresentado alguns dados informativos sobre a compra, como o número do caixa, quantidade total de itens processados, valor total da compra, data e hora da leitura.

Aplicação fornece três tipos de operações ao usuário, que são acionados pelos botões: Confirmar, Reprocessar e Cancelar. O botão cancelar, permite que seja desfeita a compra, voltando à tela inicial. O botão Reprocessar, permite voltar à tela de reprocessamento de itens, para que o usuário possa remover algum item que não gostaria de comprar. O botão confirmar, finaliza a compra e volta a tela inicial para uma nova operação de compra.

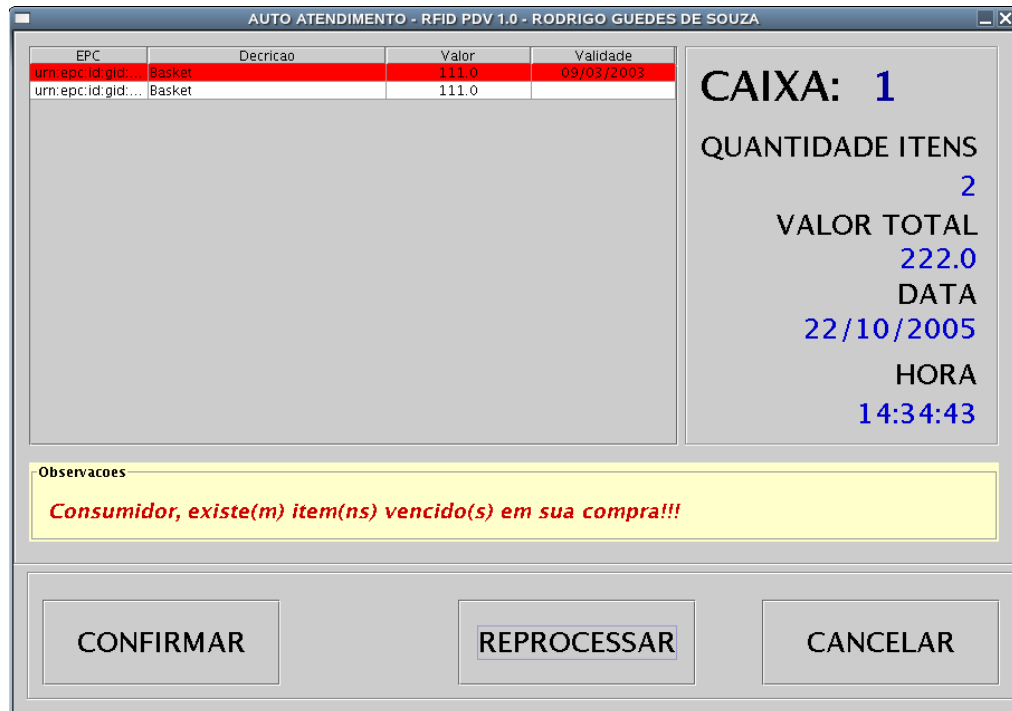


Figura 21 Tela de Venda

```
//FAZ A INTEGRACAO ENTRE O SUN JAVA SYSTEM RFID COM OS DADOS LOCAIS
public List getItensVenda(){
    List arrayTable = new ArrayList();
    //Localiza Todos os Itens que esta no campo da leitura
    List itens = genLeitora.getListUnits(sensorEPC);
    status.setMaximum(itens.size());
    status.setValue(0);
    for (int i=0;i<itens.size();i++){
        Unit unit = (Unit)itens.get(i);
        Product product = genProduto.getProduto(unit.getProductId());

        ValorProduto vlProd = daoVlProd.getValorProduto(
            product.getProductId());

        Item item = new Item();
        item.setItemID(i+1);
        item.setEpc(unit.getEpc());
        item.setDescricao(product.getName());
        item.setValor(vlProd.getValor());
        if (unit.getExpiryDate() != null){
            //Data de Vencimento
            item.setDatavencimento(unit.getExpiryDate().getTime());
        }
        arrayTable.add(item);
        status.setValue(i+1);
    }
}
```

Figura 22 Método de Integração da Classe ProcessaltensLeitora

## 5.9 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado o protótipo proposto pelo trabalho com o objetivo de automatizar o processo de classificação de itens em caixas do varejo, propondo-se assim a solução para o problema nas filas dos caixas. Tal capítulo mostra de forma clara como implementar um sistema utilizando a identificação por rádio frequência através do middleware Sun Java System RFID.

## **6. CONCLUSÃO**

### **6.1 CONSIDERAÇÕES FINAIS**

Com esta pesquisa, percebeu-se que a utilização da identificação por rádio frequência pode aumentar o processo de classificação devida o alto grau de automação que ela proporciona, diminuindo as filas de caixas.

A implementação da aplicação mostrou que, com a utilização de um middleware facilita muito a administração das informações que percorrem a arquitetura EPC e facilita o desenvolvimento das aplicações. O Desenvolvedor fica concentrado em solucionar problemas em um nível mais alto, como a integração entre os eventos EPC com regras de negócios de outros sistemas como Enterprise Resource Planning (ERP), Warehouse Management System (WMS) e Executive Information Systems (EIS) entre outros como desenvolvidos no trabalho, sem se preocupar com a captura e controle de dados e etiquetas em um nível mais baixo da arquitetura.

Este trabalho também serve como um guia de estudos para outros desenvolvedores ou profissionais afins que queiram implementar um sistema utilizando identificação por rádio frequência.

### **6.2 TRABALHOS FUTUROS**

Como trabalhos futuros podemos citar a complementação do protótipo desenvolvido neste trabalho, acrescentando as funções de pagamento, relatório e a integração a outros sistemas comumente utilizada pelas empresas do setor varejista.

Outra referência a trabalhos futuros é um estudo comparativo entre o grau de automação nos caixas do varejo, utilizando o UPC e o EPC, analisando o tempo de classificação de produtos, o nível de satisfação do cliente, o custo de instalação e manutenção.

Finalmente, os trabalhos futuros não se limitam somente a aplicação da tecnologia sobre o setor varejista, pode-se fazer um estudo sobre a utilização da RFID para automatização do controle de produção em tempo real.

## REFERÊNCIAS

BHUPTANI, Manish; MORADPOUR, Shahram. **RFID Field Guide: Deploying Radio Frequency Identification Systems**. Upper Saddle River, NJ, Editora Prentice Hall, 2005. 288p. ISBN 0-13-1853554.

BROCK, David L. **The Compact Eletronic Product Code: A 64-bit Representation of the Electronic Product Code**. 2001, Massachusetts Institute of Technology, Cambridge, MA, E.U.A. Disponível em [www.autoidlabs.org/whitepapers/mit-autoid-wh-008.pdf](http://www.autoidlabs.org/whitepapers/mit-autoid-wh-008.pdf). Acesso em: 27 Set. 2005.

KULPES, Sergio. **As lojas usam a tecnologia a seu favor. E os consumidores também**. 2005, Diário do Comércio. Disponível em <http://www.dcomercio.com.br/especiais/automacao/aslojas.htm>. Acesso em: 16 Out. 2005.

MARCHETTI, R. Z. ; PRADO, Paulo. **A Automação comercial e a satisfação do consumidor em supermercados**. In: Cláudio Felisoni de Angelo; José A. G. da Silva. (Org.). Varejo Competitivo. 1 Ed. São Paulo, 1996, v. 1. ISBN 8522424578.

OLIVEIRA, Bárbara, **Supermercado do futuro já funciona na Alemanha. 2005, Diário do Comércio**. Disponível em: <http://www.dcomercio.com.br/especiais/automacao/supermercado.htm>. Acesso em: 16 Out. 2005.

SANTA CLARA (CA). Sun Microsystems, Inc. **Installation Guid: Sun Java System RFID Software 2.0**. Santa Clara, 2005. 18p.

SOUZA Bruno. **JAVA & JINI**. 1999. SENAC, São José do Rio Preto, SP. Disponível em <http://www.javaman.com.br/apres/java&jini/index.html>. Acesso em: 01 Out. 2005.

SWEENEY, Patrick J. **RFID For Dummies**. Hoboken, NJ, Editora Wiley, 2005. 388p. ISBN 0-7645-7910-X.

YUAN, Michael. Tecnologia RFID e a Internet de Objetos. **Revista Mundo Java**. Curitiba, n.11, 2005

## 7. APÊNDICES

## a. APÊNDICE A - Código Fontes da Aplicação RFIDPDV

### AppController.Java

```

import com.sun.autoid.epcis.client.Product;
import com.sun.autoid.epcis.client.Unit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import javax.swing.JOptionPane;
import rfidpdv.dao.DaoCaixa;
import rfidpdv.dao.DaoValorProduto;
import rfidpdv.dao.collection.DaoCaixaCollection;
import rfidpdv.dao.collection.DaoValorProdutoCollection;
import rfidpdv.dao.collection.DaoVendaCollection;
import rfidpdv.gui.JFrmIniciar;
import rfidpdv.gui.JFrmProcessaItens;
import rfidpdv.gui.JFrmReprocessaItens;
import rfidpdv.gui.JFrmVenda;
import rfidpdv.integracao.ProcessarItensLeitora;
import rfidpdv.is.GerenciadorLeitora;
import rfidpdv.is.GerenciadorProduto;
import rfidpdv.pojo.Item;
import rfidpdv.pojo.ValorProduto;
import rfidpdv.pojo.Venda;

public class AppController {
    private int caixaID;

    //Views
    JFrmIniciar telaInicial;
    JFrmReprocessaItens telaReprocessa;
    JFrmProcessaItens telaProcessando;
    JFrmVenda telaVenda;

    //Model
    DaoCaixa daoCaixa = new DaoCaixaCollection();
    DaoVendaCollection daoVenda = new DaoVendaCollection();

    //Temp
    Venda venda;

    public AppController(int caixa) {
        this.caixaID = caixa;
        CarregaGUI();
    }

    public void CarregaGUI(){
        telaInicial = new JFrmIniciar();
        telaReprocessa = new JFrmReprocessaItens();
        telaProcessando = new JFrmProcessaItens();
        telaVenda = new JFrmVenda();

        telaInicial.setVisible(true);

        // Acoes da Tela Inicial-----

```

```

telaInicial.setIniciarCompraListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            int result;
            result = JOptionPane.showConfirmDialog(telaInicial,"Existe
Algum Item que deseja Remover da Compra?", "",JOptionPane.ERROR_MESSAGE);
            if (result == 0){
                telaReprocessa.setVisible(true);
            }else if (result == 1){
                telaInicial.setVisible(false);
                processaItens();
            }
            telaInicial.setVisible(false);
        }
    }
);

// Acoes da Tela ReProcessar Itens-----
telaReprocessa.setProsseguirAction( //BOTAO PROSSEGUIR
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            telaReprocessa.setVisible(false);
            processaItens();
        }
    }
);
telaReprocessa.setCancelarAction( //BOTAO CANCELAR
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            //Volta a Tela Inicial
            telaReprocessa.setVisible(false);
            telaInicial.setVisible(true);
        }
    }
);

//Acoes da Tela de Venda-----
telaVenda.setConfirmarListener( //BOTAO CONFIRMAR
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (venda.existeItensVencidos()){
                JOptionPane.showMessageDialog(null,"Nao e possivel
confirmar, Existe itens vencidos!");
            }else{
                daoVenda.save(venda);
                JOptionPane.showMessageDialog(null,"Operacao de Compra
efetuada com sucesso!");
                telaVenda.setVisible(false);
                telaInicial.setVisible(true);
            }
        }
    }
);
telaVenda.setReprocessarListener( //BOTAO REPROCESSAR
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            venda = null;
            telaVenda.setVisible(false);
            telaReprocessa.setVisible(true);
        }
    }
);
telaVenda.setCancelarListener( //BOTAO CANCELAR

```

```

        new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                venda = null;
                telaVenda.setVisible(false);
                telaInicial.setVisible(true);
            }
        }
    );
}

public void processaItens(){
    telaProcessando.setVisible(true);

    venda = new Venda();
    venda.setDatahora(Calendar.getInstance());

    ProcessarItensLeitora pItensLeitura = new
    ProcessarItensLeitora(telaProcessando.getPbStatus(),daoCaixa.getCaixa(caixaID
    ).getSensorEPC());

    venda.setItens(pItensLeitura.getItensVenda());

    telaVenda.setItens(new ArrayList(venda.getItens()));

    //Display Tela de Venda
    telaVenda.setLblCaixa(Integer.toString(caixaID));
    telaVenda.setLblQtDItens(Integer.toString(venda.getItens().size()));
    telaVenda.setLblVlTotal(Double.toString(venda.getTotalItens()));

    String formatDia = "dd/MM/yyyy";
    String formatHora = "kk:mm:ss";

    SimpleDateFormat sdfDia = new SimpleDateFormat(formatDia);
    telaVenda.setLblData(sdfDia.format(venda.getDatahora().getTime()));
    SimpleDateFormat sdfHora = new SimpleDateFormat(formatHora);
    telaVenda.setLblHora(sdfHora.format(venda.getDatahora().getTime()));

    if (venda.existeItensVencidos()){
        telaVenda.setLblObs("Consumidor, existe(m) item(ns) vencido(s) em
sua compra!!!");
    }

    telaProcessando.setVisible(false);

    telaVenda.setVisible(true);
}

public static void main(String args[]){
    int cx = 0;
    if (args.length == 0){
        System.out.println("RFIPDV INFO:Informe o numero do caixa!");
        System.exit(0);
    }else{
        cx = Integer.parseInt(args[0]);
    }
    AppController app = new AppController(cx);
}
}

```

### DaoCaixa.java

```
package rfidpdv.dao;
```

```
import rfidpdv.pojo.Caixa;
public interface DaoCaixa {
    public void save(Caixa caixa);
    public void delete(int id);
    public Caixa getCaixa(int id);
}
```

### **DaoValorProduto.java**

```
package rfidpdv.dao;
import rfidpdv.pojo.ValorProduto;
public interface DaoValorProduto {
    public void save(ValorProduto vlProduto);
    public void delete(String id);
    public ValorProduto getValorProduto(String id);
}
```

### **DaoVenda.java**

```
package rfidpdv.dao;
import rfidpdv.pojo.Venda;
public interface DaoVenda {

    public void save(Venda venda);
    public void delete(int id);
    public Venda getVenda(int id);

}
```

### **DaoCaixaCollection.java**

```
package rfidpdv.dao.collection;

import java.util.TreeMap;
import rfidpdv.dao.DaoCaixa;
import rfidpdv.pojo.Caixa;

public class DaoCaixaCollection implements DaoCaixa{
    public static TreeMap caixas = new TreeMap();

    public DaoCaixaCollection() {
        Caixa cx = new Caixa();
        cx.setCaixaID(1);
        cx.setSensorEPC("urn:epc:id:gid:1.700.2");

        this.save(cx);
    }

    public void delete(int id) {
        caixas.remove(new Integer(id));
    }

    public rfidpdv.pojo.Caixa getCaixa(int id) {
        return (Caixa)caixas.get(new Integer(id));
    }

    public void save(rfidpdv.pojo.Caixa caixa) {
```

```

        caixas.put(new Integer(caixa.getCaixaID()), caixa);
    }
}

```

### **DaoValorProdutoCollection.java**

```

//Classe que representa Dados dos Valores de Cada Produtos do IS
//A Classe representa a regra de negocios em um nivel mais alto da EPC

package rfidpdv.dao.collection;

import java.util.TreeMap;
import rfidpdv.dao.DaoValorProduto;
import rfidpdv.pojo.ValorProduto;

public class DaoValorProdutoCollection implements DaoValorProduto{
    public static TreeMap valoresProdutos = new TreeMap();

    public DaoValorProdutoCollection(){
        ValorProduto vlProd = new ValorProduto();
        vlProd.setProdutoID("1");//Valor do produto no Sun Java System RFID
        (Information Service)
        vlProd.setValor(111.00);

        valoresProdutos.put(vlProd.getProdutoID(), vlProd);
    }

    public void save(ValorProduto vlProduto) {
        valoresProdutos.put(vlProduto.getProdutoID(), vlProduto);
    }

    public void delete(String id) {
        valoresProdutos.remove(id);
    }

    public ValorProduto getValorProduto(String id) {
        return (ValorProduto) valoresProdutos.get(id);
    }
}

```

### **DaoVendaCollection.java**

```

package rfidpdv.dao.collection;

import java.util.TreeMap;
import rfidpdv.dao.DaoVenda;
import rfidpdv.pojo.Venda;

public class DaoVendaCollection implements DaoVenda{
    public static TreeMap vendas = new TreeMap();

    public DaoVendaCollection() {
    }

    public void save(Venda venda) {
        vendas.put(new Integer(venda.getCompraID()), venda);
    }

    public void delete(int id) {

```

```

        vendas.remove(new Integer(id));
    }

    public Venda getVenda(int id) {
        return (Venda) vendas.get(new Integer(id));
    }
}

```

## Caixa.java

```

package rfidpdv.pojo;

public class Caixa {
    private int caixaID;
    private String sensorEPC;

    public int getCaixaID() {
        return caixaID;
    }

    public void setCaixaID(int caixaID) {
        this.caixaID = caixaID;
    }

    public String getSensorEPC() {
        return sensorEPC;
    }

    public void setSensorEPC(String sensorEPC) {
        this.sensorEPC = sensorEPC;
    }
}

```

## Item.java

```

package rfidpdv.pojo;

import java.util.Calendar;
import java.util.Date;

public class Item {
    private int itemID;
    private String epc;
    private String descricao;
    private double valor;
    private Date datavencimento;

    public int getItemID() {
        return itemID;
    }

    public void setItemID(int itemID) {
        this.itemID = itemID;
    }

    public String getEpc() {
        return epc;
    }

    public void setEpc(String epc) {

```

```

        this.epc = epc;
    }

    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }

    public double getValor() {
        return valor;
    }

    public void setValor(double valor) {
        this.valor = valor;
    }

    public Date getDatavencimento() {
        return datavencimento;
    }

    public void setDatavencimento(Date datavencimento) {
        this.datavencimento = datavencimento;
    }
}

```

### **ValorProduto.java**

```

package rfidpdv.pojo;

public class ValorProduto {
    private String produtoID;
    private double valor;

    public String getProdutoID() {
        return produtoID;
    }

    public void setProdutoID(String produtoID) {
        this.produtoID = produtoID;
    }

    public double getValor() {
        return valor;
    }

    public void setValor(double valor) {
        this.valor = valor;
    }
}

```

### **Venda.java**

```

package rfidpdv.pojo;

```

```

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Collection;
import java.util.Date;

public class Venda {
    private int compraID;
    private Calendar datahora;
    private Collection itens;

    public int getCompraID() {
        return compraID;
    }

    public void setCompraID(int compraID) {
        this.compraID = compraID;
    }

    public Calendar getDatahora() {
        return datahora;
    }

    public void setDatahora(Calendar datahora) {
        this.datahora = datahora;
    }

    public Collection getItens() {
        return itens;
    }

    public void setItens(Collection itens) {
        this.itens = itens;
    }

    public double getTotalItens(){
        double tot = 0;
        ArrayList array = (ArrayList) itens;
        for (int i=0;i<array.size();i++){
            Item item = (Item)array.get(i);
            tot += item.getValor();
        }
        return tot;
    }

    public boolean existeItensVencidos(){
        boolean exist = false;
        ArrayList array = (ArrayList) itens;
        for (int i=0;i<array.size();i++){
            Item item = (Item)array.get(i);
            if (item.getDatavencimento() != null){
                if (item.getDatavencimento().before(new Date())){
                    exist = true;
                }
            }
        }
        return exist;
    }
}

```

**ConnIS.java**

```

package rfidpdv.is;

import com.sun.autoid.epcis.client.EpcisConnection;

public class ConnIS {

    EpcisConnection conn = null;

    public ConnIS(){
        try{
            conn = new
EpcisConnection("http://localhost:8080/epcis/service","http://localhost:", "80
80", "epcis", "marcia74");
        }catch(Exception e){

        }

    }

    public EpcisConnection getConnection(){
        return conn;
    }

    protected void finalize() throws Throwable {
        conn.close();
    }
}

```

### **GerenciadorLeitora.java**

```

package rfidpdv.is;

import com.sun.autoid.epcis.EPCISException;
import com.sun.autoid.epcis.client.FindByAttrRequest;
import com.sun.autoid.epcis.client.FindByAttrResponse;
import com.sun.autoid.epcis.client.Observation;
import com.sun.autoid.epcis.client.Unit;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;

public class GerenciadorLeitora {
    ConnIS conn = new ConnIS();
    FindByAttrRequest findReq = null;
    FindByAttrResponse findResp = null;

    public GerenciadorLeitora() { }

    //Retorna uma lista da Units que estao no campo da leitura (sensor)
    public List getListUnits(String sensor){
        List arrayUnits = new ArrayList();
        try{
            findReq = new FindByAttrRequest(conn.getConnection(),
"CURRENT_OBSERVATION");
            findReq.addCondition("SENSOR_EPC", sensor, "eq");

            findResp = findReq.process();

            FindByAttrRequest findReqUnit = null;
            FindByAttrResponse findRespUnit = null;

            for (int i=0;i<findResp.getObservations().size();i++){

```

```

        Observation obs = (Observation)
findResp.getObservations().get(i);

        findReqUnit = new FindByAttrRequest(conn.getConnection(),
"UNIT");
        findReqUnit.addCondition("EPC", obs.getObservationValue(), "eq"
);
        findRespUnit = findReqUnit.process();

        Unit unit = (Unit) findRespUnit.getUnits().get(0);

        arrayUnits.add(unit);
    }
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, e.getMessage());
}
return arrayUnits;
}
}

```

### GerenciadorProduto.java

```

package rfidpdv.is;

import com.sun.autoid.epcis.client.FindByAttrRequest;
import com.sun.autoid.epcis.client.FindByAttrResponse;
import com.sun.autoid.epcis.client.Product;

public class GerenciadorProduto {
    ConnIS conn = new ConnIS();
    FindByAttrRequest findReq = null;
    FindByAttrResponse findResp = null;

    public GerenciadorProduto() { }

    public Product getProduto(String IDProduto){
        try{
            findReq = new FindByAttrRequest(conn.getConnection(), "PRODUCT");
            findReq.addCondition("PRODUCT_ID", IDProduto, "eq");
            findResp = findReq.process();

        } catch (Exception e) {
        }
        return (Product) findResp.getProducts().get(0);
    }
}

```

### ProcessarItensLeitora.java

```

package rfidpdv.integracao;

import com.sun.autoid.epcis.client.Product;
import com.sun.autoid.epcis.client.Unit;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JProgressBar;
import rfidpdv.dao.DaoValorProduto;
import rfidpdv.dao.collection.DaoValorProdutoCollection;
import rfidpdv.is.GerenciadorLeitora;

```

```

import rfidpdv.is.GerenciadorProduto;
import rfidpdv.pojo.Item;
import rfidpdv.pojo.ValorProduto;

public class ProcessarItensLeitora {
    private String sensorEPC;
    private JProgressBar status;

    GerenciadorLeitora genLeitora = new GerenciadorLeitora();
    GerenciadorProduto genProduto = new GerenciadorProduto();

    DaoValorProduto daoVlProd = new DaoValorProdutoCollection();

    public ProcessarItensLeitora(javax.swing.JProgressBar status, String
sEpc) {
        this.sensorEPC = sEpc;
        this.status = status;
    }

    //FAZ A INTEGRACAO ENTRE O SUN JAVA SYSTEM RFID COM OS DADOS LOCAIS
    public List getItensVenda(){
        List arrayTable = new ArrayList();

        //Localiza Todos os Itens que esta no campo da leitura
        List itens = genLeitora.getListUnits(sensorEPC);

        status.setMaximum(itens.size());
        status.setValue(0);

        for (int i=0;i<itens.size();i++){
            Unit unit = (Unit)itens.get(i);
            Product product = genProduto.getProduto(unit.getProductId());

            ValorProduto vlProd =
daoVlProd.getValorProduto(product.getProductId());

            Item item = new Item();
            item.setItemID(i+1);
            item.setEpc(unit.getEpc());
            item.setDescricao(product.getName());
            item.setValor(vlProd.getValor());
            if (unit.getExpiryDate() != null){
                item.setDataVencimento(unit.getExpiryDate().getTime()); //Data
de Vencimento
            }

            arrayTable.add(item);
            status.setValue(i+1);
        }
        return arrayTable;
    }
}

```

### ItensCellRenderrer.java

```

package rfidpdv.gui;

import java.awt.Color;
import java.awt.Component;
import java.text.DateFormat;
import java.text.SimpleDateFormat;

```

```

import java.util.Date;
import javax.swing.JLabel;
import javax.swing.table.DefaultTableCellRenderer;
import rfidpdv.pojo.Item;

public class ItensCellRenderere extends DefaultTableCellRenderer{

    public ItensCellRenderere() {super(); }

    public Color corItem(Item item){
        if (item.getDataVencimento() != null ){
            if (item.getDataVencimento().before(new Date())){
                return Color.RED;
            } else{
                return Color.WHITE;
            }
        }else{
            return Color.WHITE;
        }
    }

    private Object formata(Object value){
        if (value instanceof Date){
            Date data = (Date)value;
            String formato = "dd/MM/yyyy";
            SimpleDateFormat formatter = new SimpleDateFormat(formato);
            return formatter.format(data);
        }else{
            return value;
        }
    }

    public Component getTableCellRendererComponent(
        javax.swing.JTable table,
        Object value, boolean isSelected, boolean hasFocus,
        int row, int column){
        value = formata(value);
        JLabel label =
(JLabel)super.getTableCellRendererComponent(table,value,isSelected,hasFocus,r
ow,column);
        if (column != 1)
            label.setHorizontalAlignment(JLabel.CENTER);
        Item item = ((ItensTableModel)table.getModel()).getValoresItem(row);

        if (isSelected){
            label.setForeground(corItem(item));
            label.setBackground(Color.GRAY);
        } else{
            label.setForeground(Color.BLACK);
            label.setBackground(corItem(item));
        }
        return label;
    }
}

```

### **ItensColumnModel.java**

```

package rfidpdv.gui;

import java.awt.FontMetrics;
import javax.swing.table.DefaultTableColumnModel;

```

```

import javax.swing.table.TableColumn;

public class ItensColumnModel extends DefaultTableColumnModel{

    private TableColumn criaColuna(int columnIndex, int largura,
        FontMetrics fm, boolean resizable, String titulo){
        int larguraTitulo = fm.stringWidth(titulo+" ");
        if (largura< larguraTitulo)
            largura = larguraTitulo;
        TableColumn col = new TableColumn(columnIndex);
        col.setCellRenderer(new ItensCellRenderer());
        col.setHeaderRenderer(null);
        col.setHeaderValue(titulo);
        col.setPreferredWidth(largura);
        if (!resizable){
            col.setMaxWidth(largura);
            col.setMinWidth(largura);
        }
        col.setResizable(resizable);
        return col;
    }

    public ItensColumnModel(FontMetrics fm) {
        int digito = fm.stringWidth("0");
        int letra = fm.stringWidth("M");
        addColumn(criaColuna(0,3*digito, fm,true,"EPC"));
        addColumn(criaColuna(1,20*letra, fm,true,"Descricao"));
        addColumn(criaColuna(2,3*letra, fm,true,"Valor"));
        addColumn(criaColuna(3,10*digito, fm,true,"Validade"));
    }
}

```

### **ItensTableModel.java**

```

package rfidpdv.gui;

import java.text.DateFormat;
import java.util.List;
import javax.swing.table.AbstractTableModel;
import rfidpdv.pojo.Item;

public class ItensTableModel extends AbstractTableModel {
    private List items;
    private DateFormat df = DateFormat.getDateInstance(DateFormat.SHORT);

    public ItensTableModel(List i) {
        this.items = i;
    }

    public Object getValueAt(int rowIndex, int columnIndex){
        Item umItem = (Item)items.get(rowIndex);
        switch(columnIndex){
            case 0: return umItem.getEpc();
            case 1: return umItem.getDescricao();
            case 2: return new Double(umItem.getValor());
            case 3: return umItem.getDatavencimento();
        }
        return null;
    }

    public int getRowCount(){

```

```

        return items.size();
    }

    public int getColumnCount(){
        return 4;
    }

    public Item getValoresItem(int rowIndex){
        return (Item) items.get(rowIndex);
    }
}

```

## **JFrmIniciar.java**

```

package rfidpdv.gui;

import java.awt.event.ActionListener;

public class JFrmIniciar extends javax.swing.JFrame {

    public JFrmIniciar() {
        initComponents();
        Util.centralizeFrame(this);
    }

    private void initComponents() {
        btnIniciarCompra = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("AUTO ATENDIMENTO - RFID PDV 1.0 - RODRIGO GUEDES DE
SOUZA");
        setResizable(false);
        btnIniciarCompra.setFont(new java.awt.Font("Dialog", 1, 36));
        btnIniciarCompra.setText("INICIAR COMPRA");

        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
G)
                .add(org.jdesktop.layout.GroupLayout.TRAILING,
layout.createSequentialGroup()
                    .addContainerGap(112, Short.MAX_VALUE)
                    .add(btnIniciarCompra,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 477,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(100, 100, 100)
                    );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
G)
                .add(org.jdesktop.layout.GroupLayout.TRAILING,
layout.createSequentialGroup()
                    .addContainerGap(152, Short.MAX_VALUE)
                    .add(btnIniciarCompra,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 208,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(145, 145, 145)
                    );
        pack();
    }
}

```

```

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new JFrmIniciar().setVisible(true);
        }
    });
}

public void setIniciarCompraListener(ActionListener al){
    btnIniciarCompra.addActionListener(al);
}

private javax.swing.JButton btnIniciarCompra;
}

```

### **JFrmProcessaItens.java**

```

package rfidpdv.gui;

import java.awt.event.ActionListener;

public class JFrmProcessaItens extends javax.swing.JFrame {

    public JFrmProcessaItens() {
        initComponents();
        Util.centralizeFrame(this);
    }

    private void initComponents() {
        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        pbStatus = new javax.swing.JProgressBar();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("AUTO ATENDIMENTO - RFID PDV 1.0 - RODRIGO GUEDES DE
SOUZA");
        setResizable(false);
        jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder());
        jLabel1.setFont(new java.awt.Font("Dialog", 1, 24));
        jLabel1.setText("AGUARDE PROCESSANDO ITENS DA VENDA...");

        org.jdesktop.layout.GroupLayout jPanel1Layout = new
org.jdesktop.layout.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout
.LEADING)
                .add(org.jdesktop.layout.GroupLayout.LEADING,
jPanel1Layout.createSequentialGroup()
                    .addContainerGap()
                    .add(jPanel1Layout.createParallelGroup(org.jdesktop.layout.Gr
oupLayout.LEADING)
                        .add(jLabel1)
                        .add(pbStatus,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 552, Short.MAX_VALUE))
                    .addContainerGap()
                );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout
.LEADING)
                .add(org.jdesktop.layout.GroupLayout.LEADING,
jPanel1Layout.createSequentialGroup()

```

```

        .add(56, 56, 56)
        .add(jLabel1)
        .add(29, 29, 29)
        .add(pbStatus,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 48,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(43, Short.MAX_VALUE)
    );

    org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
G)
        .add(org.jdesktop.layout.GroupLayout.TRAILING,
layout.createSequentialGroup()
        .addContainerGap(org.jdesktop.layout.GroupLayout.DEFAULT_SIZE
, Short.MAX_VALUE)
        .add(jPanel1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
        .addContainerGap())
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
G)
        .add(org.jdesktop.layout.GroupLayout.LEADING,
layout.createSequentialGroup()
        .addContainerGap()
        .add(jPanel1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(org.jdesktop.layout.GroupLayout.DEFAULT_SIZE
, Short.MAX_VALUE))
    );
    pack();
}

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new JFrmProcessaItens().setVisible(true);
        }
    });
}

public javax.swing.JProgressBar getPbStatus(){
    return this.pbStatus;
}
}

```

### **JFrmReprocessaItens.java**

```

package rfidpdv.gui;

import java.awt.event.ActionListener;

public class JFrmReprocessaItens extends javax.swing.JFrame {

```

```

public JFrmReprocessaItens() {
    initComponents();
    Util.centralizeFrame(this);
}

private void initComponents() {
    lblTexto = new javax.swing.JLabel();
    btnProsseguir = new javax.swing.JButton();
    btnCancelar = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("AUTO ATENDIMENTO - RFID PDV 1.0 - RODRIGO GUEDES DE
SOUZA");
    setResizable(false);
    lblTexto.setFont(new java.awt.Font("Dialog", 1, 24));
    lblTexto.setText("REMOVA OS ITENS E PRESSIONE A OPCAO DESEJADA");

    btnProsseguir.setFont(new java.awt.Font("Dialog", 1, 36));
    btnProsseguir.setText("PROSSEGUIR");

    btnCancelar.setFont(new java.awt.Font("Dialog", 1, 36));
    btnCancelar.setText("CANCELAR");

    org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING
G)
        .add(org.jdesktop.layout.GroupLayout.TRAILING,
layout.createSequentialGroup())
        .add(68, 68, 68)
        .add(btnProsseguir)
        .add(72, 72, 72)
        .add(btnCancelar)
        .add(92, 92, 92)
        .add(org.jdesktop.layout.GroupLayout.LEADING,
layout.createSequentialGroup())
        .add(29, 29, 29)
        .add(lblTexto)
        .add(54, 54, 54)
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING
G)
        .add(org.jdesktop.layout.GroupLayout.LEADING,
layout.createSequentialGroup())
        .add(106, 106, 106)
        .add(lblTexto)
        .add(109, 109, 109)
        .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayo
ut.BASELINE)
        .add(btnCancelar)
        .add(btnProsseguir)
        .addContainerGap(125, Short.MAX_VALUE))
    );
    pack();
}

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new JFrmReprocessaItens().setVisible(true);
        }
    });
}

```

```

    });
}

public void setProsseguirAction(ActionListener al){
    btnProsseguir.addActionListener(al);
}

public void setCancelarAction(ActionListener al){
    btnCancelar.addActionListener(al);
}

private javax.swing.JButton btnCancelar;
private javax.swing.JButton btnProsseguir;
private javax.swing.JLabel lblTexto;
}

```

## **JFrmVenda.java**

```

package rfidpdv.gui;

import java.awt.FontMetrics;
import java.awt.event.ActionListener;
import java.util.List;
import javax.swing.JTable;

public class JFrmVenda extends javax.swing.JFrame {

    public JFrmVenda() {
        initComponents();
        tbItens.setAutoCreateColumnsFromModel(false);
        FontMetrics fm = tbItens.getFontMetrics(tbItens.getFont());
        tbItens.setColumnModel(new ItensColumnModel(fm));
        Util.centralizeFrame(this);
    }

    private void initComponents() {
        jpnTopo = new javax.swing.JPanel();
        jpnInfo = new javax.swing.JPanel();
        lblTitCaixa = new javax.swing.JLabel();
        lblCaixa = new javax.swing.JLabel();
        lblTitQtd = new javax.swing.JLabel();
        lblTitVlTotal = new javax.swing.JLabel();
        lblTitData = new javax.swing.JLabel();
        lblTitHora = new javax.swing.JLabel();
        lblQtdItens = new javax.swing.JLabel();
        lblVlTotal = new javax.swing.JLabel();
        lblData = new javax.swing.JLabel();
        lblHora = new javax.swing.JLabel();
        jspGrid = new javax.swing.JScrollPane();
        tbItens = new javax.swing.JTable();
        jpnObs = new javax.swing.JPanel();
        lblObs = new javax.swing.JLabel();
        jpnRodape = new javax.swing.JPanel();
        btnConfirmar = new javax.swing.JButton();
        btnReprocessar = new javax.swing.JButton();
        btnCancelar = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("AUTO ATENDIMENTO - RFID PDV 1.0 - RODRIGO GUEDES DE
SOUZA");
        setResizable(false);
    }
}

```

```

jpnTopo.setBorder(javax.swing.BorderFactory.createEtchedBorder());
jpnInfo.setBorder(javax.swing.BorderFactory.createEtchedBorder());
lblTitCaixa.setFont(new java.awt.Font("Dialog", 1, 36));
lblTitCaixa.setText("CAIXA:");

lblCaixa.setFont(new java.awt.Font("Dialog", 1, 36));
lblCaixa.setForeground(new java.awt.Color(0, 0, 153));
lblCaixa.setText("01");

lblTitQtd.setFont(new java.awt.Font("Dialog", 1, 24));
lblTitQtd.setText("QUANTIDADE ITENS");

lblTitVlTotal.setFont(new java.awt.Font("Dialog", 1, 24));
lblTitVlTotal.setText("VALOR TOTAL");

lblTitData.setFont(new java.awt.Font("Dialog", 1, 24));
lblTitData.setText("DATA");

lblTitHora.setFont(new java.awt.Font("Dialog", 1, 24));
lblTitHora.setText("HORA");

lblQtdItens.setFont(new java.awt.Font("Dialog", 1, 24));
lblQtdItens.setForeground(new java.awt.Color(0, 0, 204));
lblQtdItens.setText("0.000,00");

lblVlTotal.setFont(new java.awt.Font("Dialog", 1, 24));
lblVlTotal.setForeground(new java.awt.Color(0, 0, 204));
lblVlTotal.setText("0,00");

lblData.setFont(new java.awt.Font("Dialog", 1, 24));
lblData.setForeground(new java.awt.Color(0, 0, 204));
lblData.setText("19/10/2005");

lblHora.setFont(new java.awt.Font("Dialog", 1, 24));
lblHora.setForeground(new java.awt.Color(0, 0, 204));
lblHora.setText("02:10:00 AM");

org.jdesktop.layout.GroupLayout jpnInfoLayout = new
org.jdesktop.layout.GroupLayout(jpnInfo);
jpnInfo.setLayout(jpnInfoLayout);
jpnInfoLayout.setHorizontalGroup(
    jpnInfoLayout.createParallelGroup(org.jdesktop.layout.GroupLayout
.LEADING)
        .add(org.jdesktop.layout.GroupLayout.LEADING,
jpnInfoLayout.createSequentialGroup()
            .addContainerGap()
            .add(jpnInfoLayout.createParallelGroup(org.jdesktop.layout.Gr
oupLayout.LEADING)
                .add(org.jdesktop.layout.GroupLayout.LEADING,
jpnInfoLayout.createSequentialGroup()
                    .add(lblTitCaixa)
                    .add(28, 28, 28)
                    .add(lblCaixa)
                    .add(org.jdesktop.layout.GroupLayout.LEADING,
jpnInfoLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
                        .add(lblQtdItens)
                        .add(lblTitQtd)
                        .add(lblVlTotal)
                        .add(lblData)
                        .add(lblHora)
                        .add(lblTitVlTotal)
                        .add(lblTitData)
                        .add(lblTitHora)))
            )
    )

```

```

        .addContainerGap(17, Short.MAX_VALUE)
    );
    jpnInfoLayout.setVerticalGroup(
        jpnInfoLayout.createParallelGroup(org.jdesktop.layout.GroupLayout
.LEADING)
        .add(org.jdesktop.layout.GroupLayout.LEADING,
jpnInfoLayout.createSequentialGroup()
        .addContainerGap()
        .add(jpnInfoLayout.createParallelGroup(org.jdesktop.layout.Gr
oupLayout.BASELINE)
            .add(lblTitCaixa)
            .add(lblCaixa)
            .add(19, 19, 19)
            .add(lblTitQtd)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(lblQtdItens)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(lblTitVlTotal)
            .add(1, 1, 1)
            .add(lblVlTotal)
            .add(7, 7, 7)
            .add(lblTitData,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 21,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(lblData)
            .add(12, 12, 12)
            .add(lblTitHora)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(lblHora)
            .addContainerGap())
        );

    tbItens.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null}
        },
        new String [] {
            "Title 1", "Title 2", "Title 3", "Title 4"
        }
    ));
    jspGrid.setViewportView(tbItens);

    jpnObs.setBackground(new java.awt.Color(255, 255, 204));
    jpnObs.setBorder(javax.swing.BorderFactory.createTitledBorder("Observ
acoes"));
    lblObs.setFont(new java.awt.Font("Dialog", 3, 18));
    lblObs.setForeground(new java.awt.Color(204, 0, 0));

    org.jdesktop.layout.GroupLayout jpnObsLayout = new
org.jdesktop.layout.GroupLayout(jpnObs);
    jpnObs.setLayout(jpnObsLayout);
    jpnObsLayout.setHorizontalGroup(
        jpnObsLayout.createParallelGroup(org.jdesktop.layout.GroupLayout
.LEADING)
        .add(org.jdesktop.layout.GroupLayout.LEADING,
jpnObsLayout.createSequentialGroup()
        .addContainerGap()
        .add(lblObs, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
834, Short.MAX_VALUE)

```

```

        .addContainerGap()
    );
    jpnObsLayout.setVerticalGroup(
        jpnObsLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.
LEADING)
        .add(org.jdesktop.layout.GroupLayout.LEADING,
jpnObsLayout.createSequentialGroup()
        .addContainerGap()
        .add(lblObs, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
16, Short.MAX_VALUE)
        .addContainerGap()
    );

    org.jdesktop.layout.GroupLayout jpnTopoLayout = new
org.jdesktop.layout.GroupLayout(jpnTopo);
    jpnTopo.setLayout(jpnTopoLayout);
    jpnTopoLayout.setHorizontalGroup(
        jpnTopoLayout.createParallelGroup(org.jdesktop.layout.GroupLayout
.LEADING)
        .add(org.jdesktop.layout.GroupLayout.LEADING,
jpnTopoLayout.createSequentialGroup()
        .addContainerGap()
        .add(jpnTopoLayout.createParallelGroup(org.jdesktop.layout.Gr
oupLayout.LEADING)
        .add(org.jdesktop.layout.GroupLayout.LEADING,
jpnTopoLayout.createSequentialGroup()
        .add(jspGrid,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 597, Short.MAX_VALUE)
        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELA
TED)
        .add(jpnInfo,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
        .add(jpnObs)
        .addContainerGap()
    );
    jpnTopoLayout.setVerticalGroup(
        jpnTopoLayout.createParallelGroup(org.jdesktop.layout.GroupLayout
.LEADING)
        .add(org.jdesktop.layout.GroupLayout.LEADING,
jpnTopoLayout.createSequentialGroup()
        .addContainerGap()
        .add(jpnTopoLayout.createParallelGroup(org.jdesktop.layout.Gr
oupLayout.LEADING, false)
        .add(jspGrid, 0, 0, Short.MAX_VALUE)
        .add(org.jdesktop.layout.GroupLayout.TRAILING, jpnInfo))
        .add(15, 15, 15)
        .add(jpnObs, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(25, Short.MAX_VALUE)
    );

    jpnRodape.setBorder(javax.swing.BorderFactory.createEtchedBorder());
    btnConfirmar.setFont(new java.awt.Font("Dialog", 1, 24));
    btnConfirmar.setText("CONFIRMAR");

    btnReprocessar.setFont(new java.awt.Font("Dialog", 1, 24));
    btnReprocessar.setText("REPROCESSAR");

    btnCancelar.setFont(new java.awt.Font("Dialog", 1, 24));
    btnCancelar.setText("CANCELAR");

```

```

        org.jdesktop.layout.GroupLayout jpnRodapeLayout = new
org.jdesktop.layout.GroupLayout(jpnRodape);
        jpnRodape.setLayout(jpnRodapeLayout);
        jpnRodapeLayout.setHorizontalGroup(
jpnRodapeLayout.createParallelGroup(org.jdesktop.layout.GroupLayo
ut.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING,
jpnRodapeLayout.createSequentialGroup()
                .addContainerGap()
                .add(btnConfirmar,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 213,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED,
162, Short.MAX_VALUE)
                    .add(btnReprocessar,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 213,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(59, 59, 59)
                        .add(btnCancelar,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 213,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                            .addContainerGap()
                );
        jpnRodapeLayout.setVerticalGroup(
jpnRodapeLayout.createParallelGroup(org.jdesktop.layout.GroupLayo
ut.LEADING)
            .add(org.jdesktop.layout.GroupLayout.LEADING,
jpnRodapeLayout.createSequentialGroup()
                .add(23, 23, 23)
                .add(jpnRodapeLayout.createParallelGroup(org.jdesktop.layout.
 GroupLayout.LEADING)
                    .add(org.jdesktop.layout.GroupLayout.TRAILING,
btnConfirmar, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 77,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .add(org.jdesktop.layout.GroupLayout.TRAILING,
btnCancelar, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 77,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                            .add(org.jdesktop.layout.GroupLayout.TRAILING,
btnReprocessar, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 77,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                                .addContainerGap()
                    );
            );

        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADIN
G)
            .add(org.jdesktop.layout.GroupLayout.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .add(jpnRodape)
                .add(jpnTopo)
            );
        layout.setVerticalGroup(
layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADIN
G)
            .add(org.jdesktop.layout.GroupLayout.LEADING,
layout.createSequentialGroup()
                .add(jpnTopo)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

```

```

        .add(jpnRodape,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
        .addContainerGap()
    );
    pack();
}
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new JFrmVenda().setVisible(true);
        }
    });
}

public JTable getTbItens(){
    return tbItens;
}

public void setItens(List itens){
    this.tbItens.setModel(new ItensTableModel(itens));
}

public void setLblCaixa(String text){
    lblCaixa.setText(text);
}

public void setLblQtdItens(String text){
    lblQtdItens.setText(text);
}

public void setLblVlTotal(String text){
    lblVlTotal.setText(text);
}

public void setLblData(String text){
    lblData.setText(text);
}

public void setLblHora(String text){
    lblHora.setText(text);
}

public void setLblObs(String text){
    lblObs.setText(text);
}

public void setConfirmarListener(ActionListener al){
    btnConfirmar.addActionListener(al);
}

public void setReprocessarListener(ActionListener al){
    btnReprocessar.addActionListener(al);
}

public void setCancelarListener(ActionListener al){
    btnCancelar.addActionListener(al);
}

private javax.swing.JButton btnCancelar;
private javax.swing.JButton btnConfirmar;
private javax.swing.JButton btnReprocessar;

```

```

private javax.swing.JPanel jpnInfo;
private javax.swing.JPanel jpnObs;
private javax.swing.JPanel jpnRodape;
private javax.swing.JPanel jpnTopo;
private javax.swing.JScrollPane jspGrid;
private javax.swing.JLabel lblCaixa;
private javax.swing.JLabel lblData;
private javax.swing.JLabel lblHora;
private javax.swing.JLabel lblObs;
private javax.swing.JLabel lblQtdItens;
private javax.swing.JLabel lblTitCaixa;
private javax.swing.JLabel lblTitData;
private javax.swing.JLabel lblTitHora;
private javax.swing.JLabel lblTitQtd;
private javax.swing.JLabel lblTitVlTotal;
private javax.swing.JLabel lblVlTotal;
private javax.swing.JTable tbItens;
}

```

## Util.java

```

package rfidpdv.gui;

import java.awt.Rectangle;
import javax.swing.JFrame;

public class Util {

    public static void centralizeFrame(JFrame frm) {
        int x,y;
        Rectangle scr = frm.getGraphicsConfiguration().getBounds();
        Rectangle form = frm.getBounds();
        x = (int) ( scr.getWidth() - form.getWidth() ) / 2;
        y = (int) ( scr.getHeight()- form.getHeight() ) / 2;
        frm.setLocation( x , y );
    }
}

```