Sistema RFID para gerenciamento de cadeia de suprimentos

Artur Pinto Carneiro[1]
Flavius Portella Ribas Martins[2]

Resumo: Sistemas *RFID* (*Radio Frequency Identification*) para identificação e localização de produtos ou equipamentos têm sido cada vez mais utilizados no gerenciamento de cadeias de suprimentos. Como as organizações participantes dessas cadeias possuem, em geral, diversos sistemas de gestão e logística, o desenvolvimento de sistemas *RFID* nesses casos pode ser bastante facilitado, desde que se construa uma camada intermediária de software responsável pela integração e compatibilização dos diversos componentes de software das empresas partícipes. Neste artigo apresenta-se um estudo de caso, baseado em um projeto realizado no Instituto de Pesquisas Tecnológicas do Estado de São Paulo (IPT), que originou um protótipo de um sistema capaz de realizar essa integração numa cadeia de suprimentos de uma concessionária de distribuição de energia elétrica. O sistema desenvolvido aplica-se ao gerenciamento de dispositivos móveis – aparelhos celulares, handhelds, coletores de dados, entre outros –, que, utilizados por diferentes agentes da cadeia, capturam os dados associados a processos realizados durante o ciclo de vida de dado equipamento da rede de distribuição de energia elétrica e os disponibilizam em um banco de dados passível de ser integrado com os demais sistemas de gestão e logística existentes na corporação.

Palavras-chave: Identificação por radiofrequência. gerenciamento de cadeia de suprimentos. RM-ODP.

Abstract: *RFID* (*Radio Frequency Identification*) *systems for identification and tracking of products and equipments have been progressively adopted as an essential tool for supply chain management, a production environment where the members usually share with each other their own logistic and management systems. Therefore, the development of supply chain RFID systems can be strongly simplified through the inclusion of an intermediate software layer responsible for the creation of interfaces to integrate all the heterogeneous software components. In this article we present a case study developed at IPT (Instituto de Pesquisas Tecnológicas do Estado de São Paulo) which gave rise to a prototype system able to implement the required software integration on the supply chain of an electric power distribution company. The developed sistem can be used to manage the interactions with a heterogeneous group of mobile devices – cell phones, handhelds and data colectors, – operated by different supply chain agents that grab data associated to various processes executed by a given electric power distribution equipment during its life cycle and transfer those data to a central database in order to share them with all the logistic and management corporation systems.*

Keywords: *RFID. supply chain management. RM-ODP.*

[1] Curso de Mestrado Tecnológico em Engenharia da Computação, Instituto de Pesquisas Tecnológicas do Estado de São Paulo, Av. Prof. Almeida Prado 532, São Paulo (SP) - Brasil
{ap_carneiro@uol.com.br}
[2] Departamento de Engenharia Mecânica, Escola Politécnica da Universidade de São Paulo, Av. Prof. Mello Moraes, 2231, São Paulo (SP) - Brasil
{flavius.martins@poli.usp.br}

http://dx.doi.org/10.5335/rbca.2011.1337

1 Introdução

Uma cadeia de suprimentos é uma rede de organizações composta por fornecedores, fabricantes, montadores, armazenadores e distribuidores que, mediante a execução de processos colaborativos de aquisição, transformação e distribuição de materiais, fornecem produtos e/ou serviços aos clientes [1]. Conforme enfatizado em [2], uma das principais vantagens inerentes a essa estratégia de produção consiste no aprimoramento do controle e da logística dos produtos durante todo o seu ciclo de vida. Para tanto, é necessário coordenar e sincronizar fluxos de bens e serviços mediante a troca eficiente de informações digitais entre os agentes responsáveis pelos processos logísticos.

Os clássicos sistemas ERP (Enterprise Resource Planning), a despeito de propiciarem a integração dos sistemas de informação distribuídos nas diversas áreas de uma corporação, modelam apenas os aspectos operacionais das transações, mas não possuem ferramentas de auxílio à tomada de decisões estratégicas [3]. Considerando-se que uma típica cadeia de suprimentos abrange diferentes corporações em variados estágios de desenvolvimento tecnológico, a integração das informações geradas pelos seus sistemas computacionais requer a inclusão de novas camadas de informação que permitam que uma dada corporação C_i tenha acesso às informações geradas pelas corporações C_j que compartilham do seu ciclo de produção e transação do produto ou serviço.

Outro fator de grande impacto sobre a qualidade da gestão e da logística dos produtos em uma cadeia de suprimentos é a adoção de um sistema de identificação único que possibilite rastrear cada produto durante seu ciclo de vida e ao longo de seu percurso através das diversas corporações. Com esse propósito, etiquetas de identificação por radiofrequência (RFID) têm sido utilizadas em uma gama cada vez maior de aplicações, como, por exemplo, automação da manufatura [4], manutenção de aeronaves [5] e gerenciamento hospitalar [6], em face das vantajosas condições de comunicação sem contato e em razoável área de abrangência oferecida sobre os tradicionais códigos ópticos e magnéticos, bem como da possibilidade de implantação de um ambiente de computação ubíqua [7], onde a aplicação se adapta de acordo com o contexto em que o usuário está inserido [8], ou seja, de acordo com um conjunto de variáveis que caracterizam o seu ambiente e o seu estado num dado instante.

Embora os sistemas RFID apresentem diversas vantagens comparativamente a outros sistemas de identificação, sua introdução em processos de gestão de uma cadeia de suprimentos requer a construção de várias interfaces com os componentes dos sistemas de informação das empresas membros da cadeia. Tal esforço pode ser minorado mediante a construção de um middleware [9], ou seja, de uma camada de software que conecta o sistema operacional com as aplicações distribuídas no sistema, facilitando o desenvolvimento das mesmas por meio da abstração de componentes comuns de programação, da omissão de detalhes de programação de baixo nível e da geração de interfaces que mascaram a heterogeneidade dos componentes. É importante destacar que as aplicações de RFID possuem um núcleo comum de funcionalidades, abrangendo a filtragem e a agregação dos dados em bruto recebidos pelos dispositivos móveis (ou seja, ligadas ao hardware), bem como a sua transformação em eventos passíveis de serem utilizados pelos consumidores (softwares de alto nível) [10]. Visando dar um tratamento uniforme a esse conjunto comum de funcionalidades, diversos modelos de middleware para sistemas RFID têm sido propostos, como, por exemplo, o WinRFID, o ALE e o Savant. O WinRFID [11], middleware compacto que oferece recursos para a captura, filtragem, agregação e envio de dados, tem sido bastante utilizado em sistemas acadêmicos laboratoriais e demonstrativos [12]. O ALE (Application Level Events) [13] realiza funções similares às do WinRFID (mapeamento entre as etiquetas e os leitores, agregação de dados e geração de comandos); todavia, por ser baseado nos padrões da EPCGlobal, sua difusão no ambiente comercial tem sido mais expressiva. O Savant [14], desenvolvido pela EPCGlobal, é um middleware de maior complexidade que oferece todos os meios para a interconexão de uma rede de etiquetas e leitores RFID baseados no padrão EPC e as aplicações de alto nível das empresas partícipes da cadeia de suprimentos.

Apesar do grande esforço empenhado no desenvolvimento de middlewares projetados para realizar a plena conexão entre os variados dispositivos móveis e sistemas de informação presentes em uma cadeia de suprimentos, os produtos desse esforço têm se mostrado insuficientes para confrontar dois grandes desafios – a evolução da concepção tecnológica dos dispositivos móveis, que, cada vez mais, vêm incorporando recursos que, anteriormente, eram providos pelos middlewares, e a complexidade das interfaces requeridas para atender à heterogeneidade de componentes de software das corporações, tornando-as cada vez mais próximas de sistemas ERP [15]. Tal cenário tem levado muitas empresas a abandonarem a aquisição de middlewares comerciais e a

desenvolverem, em contrapartida, sistemas RFID inteiramente dedicados às suas necessidades. Essa é a linha perseguida pelo middleware RFID apresentado neste artigo – trata-se de uma camada de software dedicado compartilhada por todas as empresas de uma cadeia de suprimentos específica (no caso, constituída pelos fornecedores de produtos e serviços de uma empresa de distribuição de energia elétrica), cujos agentes utilizam coletores de dados baseados em software proprietário para se comunicar com etiquetas RFID compatíveis, afixadas aos variados equipamentos da rede de distribuição, transferindo-as, em seguida, para os seus respectivos sistemas de informação.

Conforme ficará claro nas seções seguintes, a originalidade deste artigo limita-se, *stricto sensu*, à aplicação focalizada – utilização de sistemas RFID em redes de distribuição de energia elétrica, aplicação da qual poucos artigos tratam na literatura [16], apesar da sua grande importância econômica. Cabe destacar que a ausência de integração dos sistemas de informação dos membros da cadeia de suprimentos das empresas de distribuição de energia elétrica impede a realização de diagnósticos em tempo real e, por conseguinte, dificulta a implantação de uma política de manutenção preditiva dos equipamentos. Tais dificuldades deram ensejo à execução do projeto intitulado "Controle Logístico Inteligente de Equipamentos de Distribuição em Estoque e Instalados no Campo", realizado no IPT (Instituto de Pesquisas Tecnológicas do Estado de São Paulo) e tendo por cliente uma empresa estadual de energia elétrica do estado de São Paulo. Alguns dos resultados alcançados durante a realização desse projeto são relatados neste artigo.

2 Materiais e métodos

O desenvolvimento do sistema RFID referido acima obedeceu ao método clássico de engenharia de software, o qual, conforme descrito em [17], abrange as etapas de comunicação, planejamento, modelagem, construção e implantação, das quais as quatro primeiras serão aqui discutidas.

2.1 A etapa de comunicação

Por meio de reuniões realizadas com os engenheiros e técnicos da empresa de distribuição de energia elétrica envolvidos nos processos de instalação e manutenção de equipamentos, identificaram-se as informações relevantes dos equipamentos, processos e operadores. Desse trabalho resultou um conjunto de requisitos técnicos definidores da arquitetura do sistema bem como de informações relevantes a serem armazenadas no assim denominado "Banco de Dados Logísticos da Web" (BDLW), as quais podem ser atualizadas pelos operadores das corporações que compõem a cadeia de suprimentos à medida que ocorrem eventos que mereçam ser registrados, como, por exemplo, identificação de um defeito em um equipamento ou remoção desse equipamento para o setor de manutenção [18].

Conforme se recomenda em [1], a cadeia de suprimentos foi dividida em dois níveis lógicos – "cooperativo" e "empresa". No primeiro nível concebeu-se uma entidade responsável pela centralização das informações e execução dos processos de tomada de decisão. No nível "empresa" associaram-se as corporações que compõem a cadeia de suprimentos e seus sistemas de gestão próprios. Estabeleceu-se também que os diversos agentes obtêm informações geradas no nível cooperativo e reagem de forma consistente incluindo no sistema novas informações específicas desse nível hierárquico.

Como a identificação dos equipamentos precisava ser feita a uma distância de cerca de 6 m e não havia necessidade de se gravarem dados *in loco*, selecionaram-se etiquetas RFID L-TG800-(IH) (da WaveTrend Co.), com as seguintes características: fonte de potência própria, capacidade exclusiva de leitura (*read only*), alcance de até 30m, frequência de 433,92 MHz, resistência à temperatura de até 60º C e protocolo de comunicação próprio. Considerando-se que o sistema deveria prover, de forma transparente, conexão com um grande número de dispositivos leitores RFID apresentando diferentes configurações de hardware e operando sob variados sistemas operacionais, estabeleceu-se que a construção de um middleware apropriado seria essencial ao projeto. Finalmente, o fato de o banco de dados logísticos (BDLW) fornecer e capturar dados de um sistema computacional distribuído e heterogêneo fez com que se adotasse para descrição da arquitetura do sistema o modelo RM-ODP (Reference Model of Open Distributed Processing), definido na norma ISO/IEC 10746 [19] . Conforme exposto em [20], o principal fundamento desse modelo é o conceito de múltiplas visões (*empresa*, *informação*, *computação*, *engenharia* e *tecnologia*), que possibilita aos agentes que planejam e desenvolvem o sistema observá-lo por uma perspectiva conveniente e segundo um nível de abstração adequado.

2.2 A etapa de planejamento

A partir dos requisitos oriundos da etapa de comunicação, dividiu-se o sistema computacional em dois grandes módulos: o do dispositivo móvel e o do servidor de dados (BDLW). Optou-se pela criação de um ambiente artificial composto por máquinas virtuais que emulassem as do ambiente real (dispositivo leitor RFID de interface serial conectado a um coletor de dados e operado ao nível do solo, comunicando-se com o servidor de dados corporativo e com as etiquetas RFID afixadas a equipamentos elétricos instalados em postes com cerca de 5 m de altura ou distribuídos em pátios de manutenção), pois isso facilitaria a execução das atividades de desenvolvimento de software sem prejuízo à sua posterior conversão para o ambiente físico real. Assim, foram concebidas duas máquinas virtuais – uma para o servidor de dados e outra para o dispositivo móvel –, estabelecendo-se que ambas seriam emuladas na mesma máquina física, de modo a facilitar sua operação simultânea, e que a porta serial do equipamento físico seria mapeada para o dispositivo móvel virtual.

2.3 A etapa de modelagem

Nesta etapa, os pontos de vista do modelo RM-ODP foram desenvolvidos de forma incremental. As informações obtidas durante a fase *comunicação* foram suficientes para suprir os pontos de vista *empresa*, *informação* e *computação*, mas as informações requeridas pelos pontos de vista *engenharia* e *tecnologia* exigiram que se especificassem de forma detalhada as plataformas de desenvolvimento de software para o servidor de dados BDLW e para o dispositivo móvel.

Para o desenvolvimento de software responsável pelo servidor de dados e pela distribuição dos serviços web necessários selecionou-se a plataforma Apache, constituída pelas aplicações Apache Tomcat, responsável pela execução dos Servlets Java, e Apache Axis, responsável pela execução do protocolo de acesso ao servidor HTTP; para o gerenciamento de banco de dados adotou-se MySQL. Tais escolhas se deveram ao fato de que tanto o Apache quanto o MySql atribuem licenciamento GPL às aplicações desenvolvidas. Tomadas essas decisões, construíram-se dois protótipos funcionais de serviços web com busca e inserção no banco de dados para validar o funcionamento da plataforma escolhida.

No que tange ao desenvolvimento de software para os dispositivos móveis, a análise preliminar de alternativas conduziu à escolha da plataforma J2ME (Java 2 Micro Edition), da Sun Microsystems, pois esta atenderia às exigências impostas pelo ambiente – computação distribuída baseada em dispositivos móveis apresentando diversidade de hardware e de sistema operacional, fato esse, aliás, referendado na literatura especializada em sistemas computacionais móveis [21-22]. Utilizando-se essa plataforma, foram criados protótipos para a validação dos requisitos do projeto, abrangendo interface homem-máquina, manipulação de arquivos locais e funcionalidades da conexão serial e da conexão com servidores remotos por protocolo TCP/IP. Contudo, durante a execução desses protótipos em simuladores de equipamentos móveis (telefone celular, SmartPhone e PocketPc), verificou-se não ser possível realizar a comunicação com a interface serial, fato que se deveu a problemas de compatibilidade da Virtual Machine J2ME com a biblioteca nativa para comunicação serial disponível em seu sistema operacional (Windows Mobile 5).

Para fazer frente ao problema de compatibilidade apontado, pesquisaram-se algumas plataformas alternativas de desenvolvimento de sistemas móveis, optando-se ao final pela plataforma SuperWaba (www.superwaba.com.br). Conforme descrito em [23], o SuperWaba, atualmente designado por TotalCross, é uma evolução da plataforma Waba (www.wabasoft.com) em que se acrescentam máquinas virtuais para os mais variados sistemas operacionais de dispositivos portáteis (incluindo-se o Windows Mobile) bem como um novo conjunto de classes que permitem a expansão das funcionalidades da plataforma primitiva. Os testes realizados com os protótipos funcionais desenvolvidos para a validação dos requisitos do sistema revelaram plena compatibilidade do SuperWaba/TotalCross com os requisitos estabelecidos, razão pela qual foi adotada a mesma como plataforma de desenvolvimento de software dos dispositivos móveis. É importante salientar que à época em que essa decisão foi tomada ainda não estavam disponíveis aos usuários alguns sistemas operacionais mais robustos, como, por exemplo, o Android (www.android.com), cuja primeira versão foi lançada no mercado ao final de 2008.

2.4 A etapa de construção

O sistema foi desenvolvido utilizando-se uma grande gama de ferramentas computacionais.

Como ambiente de programação, tanto do aplicativo do dispositivo móvel como dos serviços do servidor de aplicação, adotou-se o Eclipse versão 3.5.2, ao qual se adicionaram o plug-in EclipseME (para desenvolvimento em J2ME) e o plug-in EclipseUML (para elaboração do diagrama de classes). Para elaborar os diagramas UML requeridos à construção da arquitetura do middleware utilizou-se o JUDE Community versão 5.2, ferramenta freeware distribuída pela empresa ChangeVision; a elaboração e análise dos diagramas do banco de dados, por sua vez, foi auxiliada pelo software MySQL Workbench, da Sun Microsystems.

Com o intuito de facilitar a emulação do ambiente computacional do projeto utilizaram-se máquinas virtuais capazes de reproduzir os comportamentos do dispositivo móvel e do servidor de dados em uma única máquina real. Para tanto, adotou-se a ferramenta Microsoft Virtual PC, executada sobre o sistema operacional Microsoft Windows Vista. Dessa forma, o ambiente de emulação ficou constituído por duas máquinas virtuais – uma emulando a plataforma PocketPC sob o sistema operacional Windows Mobile 5.0, outra emulando uma plataforma PC sob o sistema operacional Ubuntu Linux. No ambiente Windows Mobile 5.0 foram instalados o KVM (Kilo Virtual Machine) para J2ME J9 da IBM e o VM (de virtual machine) SuperWaba/TotalCross para Windows Mobile, ambos capazes de executar os MIDLets J2ME e a aplicação SuperWaba/TotalCross. No ambiente Linux Ubuntu incluíram-se, além do sistema gerenciador de banco de dados MySQL, os pacotes responsáveis pelo servidor de Apache Tomcat, bem como o Apache Axis. Na máquina virtual PocketPC configurou-se o leitor RFID, conectado através de um cabo conversor serial/USB à máquina real. Esta porta, por sua vez, foi mapeada para a máquina virtual do PocketPC.

É importante ressaltar que, pelo fato de possuir a mesma sintaxe da linguagem Java, os programas SuperWaba/TotalCross podem ser desenvolvidos com qualquer ferramenta utilizada pela plataforma Java. Para tanto, compila-se o código Java via compilador JDK, geram-se os arquivos do tipo '.class', os quais são transformados em pacotes '.pdb' interpretados pela máquina virtual SuperWaba. O aplicativo gerado é independente de plataforma; logo, é suficiente que haja no ambiente uma máquina virtual desenvolvida para a plataforma fim.

Seguindo as recomendações da W3C [24], as ferramentas e recursos utilizados para desenvolver os serviços web do sistema foram o XML (Extensible Markup Language), o SOAP (Simple Object Access Protocol) e o WSDL (Web Services Description Language). O XML é um formato de dados normalizado, flexível e extensível que potencializa a interoperabilidade entre diversas plataformas; o SOAP é um protocolo de acesso a objetos que utiliza XML como formato das mensagens trocadas entre dois pontos; o WSDL, por sua vez, é uma linguagem baseada em XML para descrições de modelos de serviços web, começando pela descrição das mensagens que serão trocadas entre o requerente e o provedor.

3 Construção da arquitetura do sistema

Considerando-se que o sistema de gerenciamento da cadeia de suprimentos da empresa de distribuição de energia elétrica compõe-se de aplicações executadas em dispositivos móveis distribuídos, seu respectivo middleware deve ser responsável pela execução das seguintes tarefas: 1) identificação da etiqueta RFID de um equipamento; 2) disponibilização das informações armazenadas no BDLW sobre o referido equipamento; 3) fornecimento de uma interface ao usuário para que essas informações sejam coletadas e/ou modificadas de acordo com as tarefas realizadas pelo operador. Além disso, é necessário que o middleware funcione independentemente de existir ou não conectividade com o servidor de banco de dados.

Os requisitos definidos acima caracterizam o sistema em foco como um sistema de processamento aberto e distribuído, razão pela qual a modelagem de sua arquitetura foi realizada segundo o modelo de referência RM-ODP, conforme já mencionado

Nos itens que seguem, essa arquitetura será descrita segundo os cinco pontos de vista do modelo RM-ODP. Cabe salientar que, muito embora o protótipo desenvolvido aborde apenas o acompanhamento do ciclo de vida de transformadores, sua extensão aos demais equipamentos de interesse poderá ser realizada sem grandes dificuldades.

3.1 Ponto de vista 'empresa'

Este ponto de vista permite estabelecer as regras de negócio às quais o sistema estará submetido.

De acordo com os requisitos estabelecidos para o sistema, identificam-se os seguintes objetos *empresa*: *Usuário*, *Identificador de Equipamentos*, *Gerenciador de Tarefas* e *Gerenciador de Configuração*. Dentre os objetos citados, o *Usuário* é o ator, executando as funcionalidades disponíveis nos demais objetos. Na Figura 1 apresenta-se o diagrama de caso de uso para o sistema proposto.

O usuário interage com o objeto *Gerenciador de Configuração,* configurando *login* de acesso, senha, endereço do servidor de banco de dados e configurações do leitor de RFID. Por meio do objeto *Identificador de Equipamentos,* o usuário pode mostrar a lista dos equipamentos visíveis e consultar seus detalhes. Já o objeto *Gerenciador de Tarefas* permite que o usuário consulte as tarefas que lhe são destinadas e as execute, adicionando ou alterando dados relativos aos equipamentos em formulários. As informações disponibilizadas ao usuário são filtradas pelo sistema, dependendo do contexto onde está inserido.

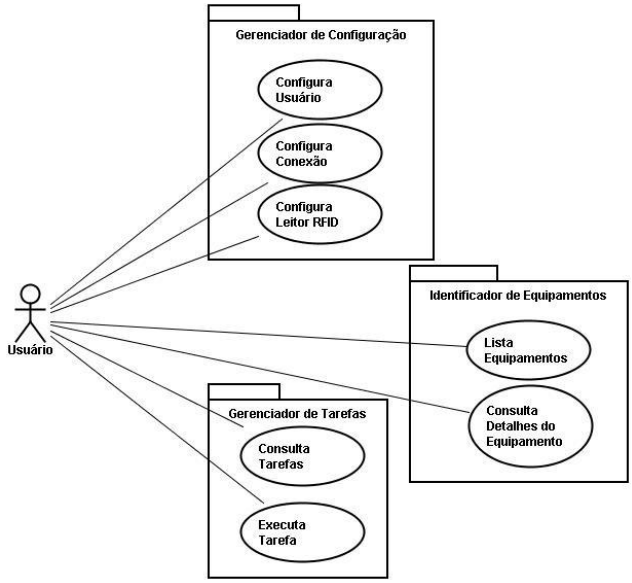


Figura 1. Diagrama de caso de uso – visão *Empresa***.**

Conforme ilustrado na Figura 2, os contextos existentes na cadeia de suprimentos dos equipamentos de redes de distribuição de energia elétrica são: *Fornecedor*, *Reformador*, *Central de Logística*, *Base Operacional* e *Local de Instalação*. O usuário de cada um desses contextos possui direitos distintos às informações dos equipamentos, os quais se referem às tarefas que podem ser executadas em cada contexto.

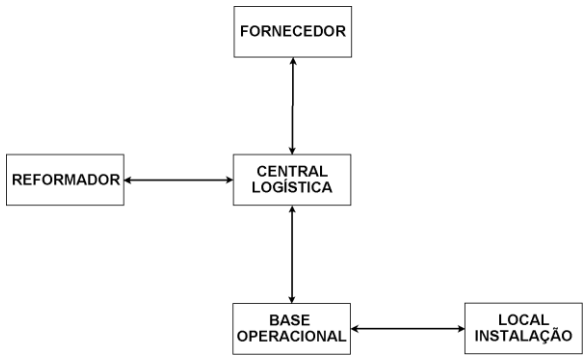


Figura 2. Contextos possíveis.

3.2 Pontos de vista 'informação'

Este ponto de vista define os objetos *Informação* que descrevem as informações manipuladas ou geradas pelos objetos *Empresa*.

Os objetos *Informação* identificados no sistema são: *Configuração*, *Equipamento*, *Tarefa*, *Usuário* e *Contexto*. Na Figura 3 ilustram-se esses objetos por meio do diagrama de classe UML, que permite explorar sua estrutura interna bem como suas relações

O objeto *Contexto*, apesar de não estar representado no ponto de vista *Empresa*, está caracterizado nos requisitos do sistema. Tal objeto descreve os tipos de contexto possíveis, visualizados pelas seguintes subclasses do diagrama de classes UML: *Fornecedor*, *Reformador*, *Central de Logística*, *Base Operacional* e *Local de Instalação*.

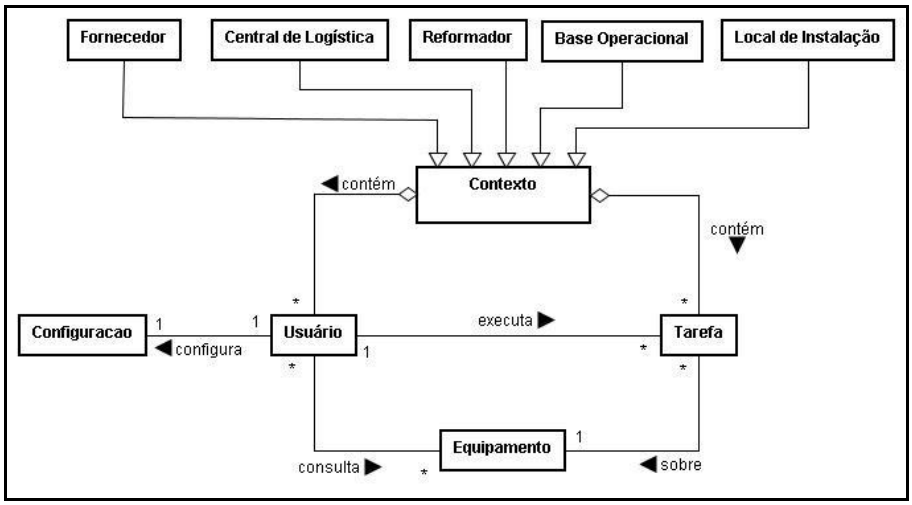


Figura 3. Diagrama de Classes – Visão 'Informação'.

Existem tarefas específicas para cada contexto e, da mesma forma, os usuários também estão subordinados a determinados contextos, sendo compelidos a executar as tarefas desses contextos. Cada tarefa, por sua vez, possui um conjunto de informações específicas a serem armazenadas sobre um dado equipamento *X*. Assim, quando o usuário solicitar detalhes sobre *X,* as informações disponibilizadas devem estar coerentes com o contexto onde este usuário está inserido. Na Tabela 1 são relacionadas as tarefas disponíveis para cada um dos contextos em que o sistema deverá atuar.

Tabela 1. Tarefas executadas em cada contexto.

Contexto	Tarefa
Fornecedor	Atualização de cadastro do equipamento
	1ª Inspeção elétrica
	1ª Inspeção do óleo
	Expedição para a Central de Logística (Fornecedor)
Reformadora	Recepção do transformador (Reformadora)
	Expedição para Central de Logística (Reformadora)
Base Operacional	Recepção do transformador (Base Operacional)
	Expedição para local de instalação
Local de Instalação	Recepção do transformador (Local de Instalação)
	Inspeção
Central de Logística	Recepção de transformador novo
	Recepção de transformador avariado
	Recepção de transformador reformado
	Expedição para Base Operacional
	Expedição para Reformadora
	Triagem

3.3 Ponto de vista ‘computação’

Este ponto de vista descreve as transformações das informações durante a execução das funcionalidades previstas, permitindo, com isso, caracterizar uma decomposição funcional do sistema. Para representar a dinâmica dessas transformações durante as interações entre os objetos utiliza-se o diagrama de sequência UML.

Os objetos *Computação* definidos neste ponto de vista, são: *Gerenciador de Configuração*, *Identificador de Equipamentos*, *Gerenciador de Tarefas*, *Núcleo* e *Servidor de Dados*. Os itens configuráveis são: login (usuário e senha), caminho do servidor de dados e comunicação com o leitor RFID (porta serial e velocidade). Essas configurações ficam armazenadas no sistema de arquivos do dispositivo móvel em um arquivo formatado chamado ‘config.ini’ e são utilizadas pelos demais objetos do sistema.

O objeto *Identificador de Equipamentos*, já abordado no ponto de vista *empresa*, é responsável por disponibilizar as funcionalidades de identificação dos equipamentos por meio dos identificadores RFID. Esse objeto gera uma lista de equipamentos visíveis (Figura 4a) e permite que o usuário selecione um equipamento para obter detalhes. Quando o usuário realiza tal ação, o objeto *Identificador de Equipamentos* (Figura 4b) solicita os detalhes do equipamento ao objeto *Servidor de Dados*, armazenando o resultado no sistema de arquivos do dispositivo móvel em um arquivo formatado, cujas informações são exibidas em um formulário. Caso não haja conexão com o objeto *Servidor de Dados*, o *Identificador de Equipamentos* armazena esta requisição para que seja executada tão logo a conexão esteja disponível.

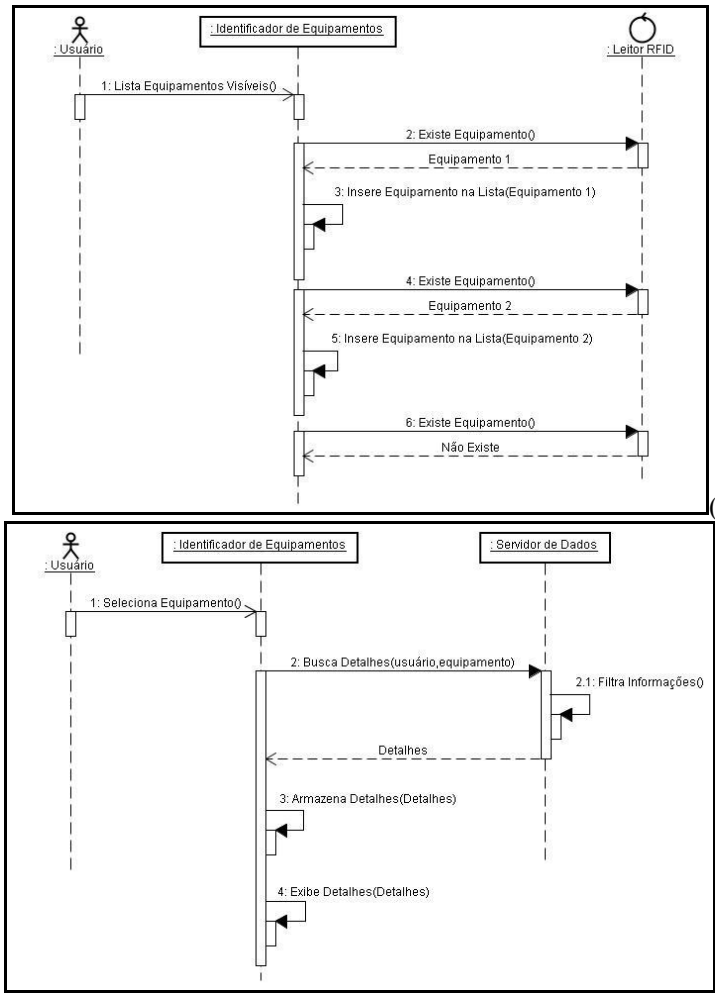


Figura 4. Diagramas de sequência: (a) lista de equipamentos visíveis; (b) detalhes de um equipamento.

O objeto *Gerenciador de Tarefas* mostra a lista de tarefas ao usuário e gera um formulário para que o mesmo execute uma tarefa. O formulário é gerado por meio da leitura do arquivo formatado da tarefa, previamente armazenado pelo objeto *Núcleo*, mediante um "*parser*". Esse arquivo possui os seguintes dados: *Título do formulário*, *Número de identificação* e *Itens do formulário*. Estes últimos são formados pelos seguintes campos: *Tipo do item*, *Descrição do Item*, *Tamanho do Item*, *Tipo de caractere permitido* e *Conteúdo do item*. A lista de tarefas exibida para o usuário, gerada a partir dos arquivos de tarefas existentes no dispositivo móvel, pode estar em dois estados: *pendente* e *finalizada*.

O objeto *Núcleo* (Figura 5) é responsável por realizar as funções periódicas de '*background*' no sistema, executadas sem que o usuário as perceba, de modo a não demandar qualquer interação de sua parte. A primeira dessas funções é a consulta ao objeto *Servidor de Dados* sobre a existência ou não de tarefas alocadas para o usuário configurado; caso existam, são armazenadas no sistema de arquivos do dispositivo móvel. Outra função desempenhada pelo objeto *Núcleo* é a de enviar ao objeto *Servidor de Dados* as tarefas já executadas (finalizadas). A última tarefa do objeto *Núcleo* consiste em consultar o objeto *Servidor de Dados* sobre detalhes de equipamentos das tentativas pendentes.

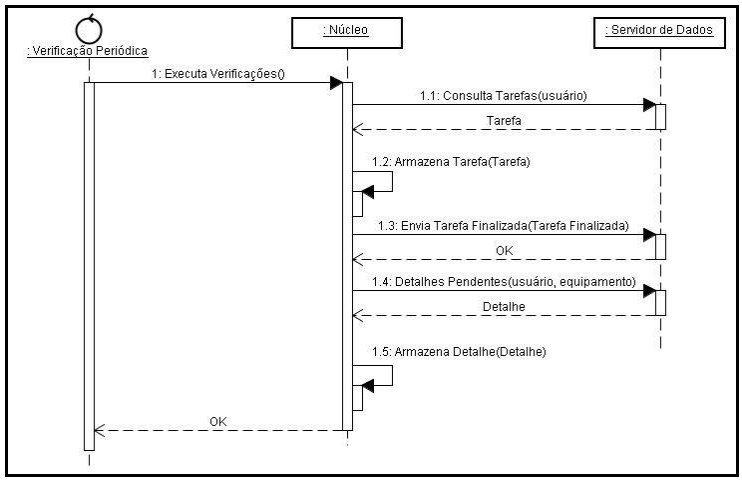


Figura 5. Diagrama de sequência da execução de funções em *background.*

O objeto *Servidor de Dados* é responsável por executar as consultas de tarefas pendentes, obter detalhes de equipamentos e de alterações geradas pelas tarefas. Este objeto realiza a filtragem das informações, dependendo do contexto do usuário que as requisitou.

3.4 Ponto de vista 'engenharia'

Considerando-se que o sistema pode ser dividido em três componentes distintos – o *Leitor RFID*, o *Dispositivo Móvel* e o *Servidor de Dados* –, o ponto de vista 'engenharia' representa-os por meio de objetos *Engenharia*, que dão consistência ao sistema modelado segundo os pontos de vista anteriores (Figura 6).

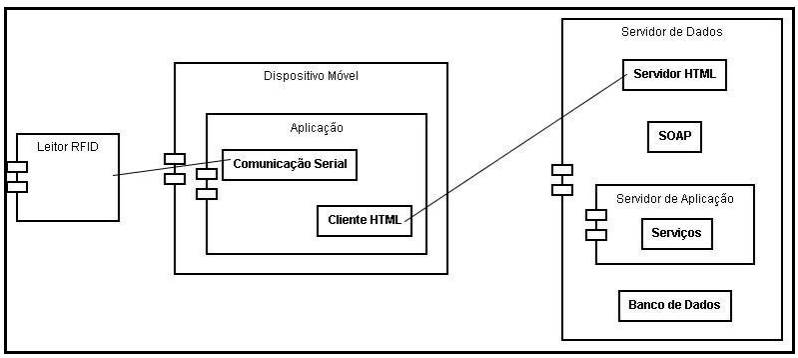


Figura 6. Diagrama de componentes do ponto de vista *engenharia*.

O *Dispositivo Móvel* contém o objeto *Aplicação,* responsável pela interação do usuário com o leitor RFID e com o *Servidor de Dados*. A comunicação com o leitor RFID ocorre por meio do padrão RS232; já a comunicação com o *Servidor de Dados* é estabelecida através do protocolo TCP/IP utilizando HTML para realizar as requisições de serviços. O submódulo *Comunicação Serial* do objeto *Aplicação* realiza a comunicação com o leitor RFID por meio do protocolo especificado pelo fabricante, o qual é baseado em dois tipos de pacotes – os de comando e os de resposta. Conforme descrito na Tabela 2, ambos possuem até 72 bytes de tamanho total e são distinguidos por um *header* (primeiro byte do pacote); para pacotes de comando esse *header* tem valor 0xAA e, para pacotes de resposta, 0x55.

Existem 21 comandos possíveis para o leitor RFID selecionado, dos quais apenas dois serão utilizados: o comando de "Reset Network" (vide Tabela 3), para capturar os campos *Identificação da rede* e *Identificação do receptor* do Leitor conectado, e o comando "Get Tag Packet" (vide Tabela 4), para capturar os dados das etiquetas eletrônicas visíveis pelo dispositivo *Leitor*.

Tabela 2. Estrutura dos pacotes de comando ou resposta.

Pacote	
Campos do Pacote	**Tamanho**
1. Header	1 byte (0xAA – Comando / 0x55 - Resposta)
2. Tamanho	1 byte (número de bytes da parte Dados)
3. Identificação da rede	1 byte
4. Identificação do receptor	1 byte
5. Identificação do Nó	1 byte
6. Código do Comando	1 byte
7. Dados	Até 64 bytes de dados
8. Checksum (campos 2-7)	1 byte, *CRC*

Tabela 3. Estrutura do comando "*Reset Network*".

Comando "*Reset Network*":						
0xAA	0x00	0x00	0x00	0xFF	0x00	*Checksum*
Resposta:						
0x55	0x00	ID da rede	ID do receptor	0x01	0x00	*Checksum*

Tabela 4. Estrutura do comando "*Get Tag Packet*".

Comando "*Get Tag Packet*":							
0xAA	0x00	ID da rede	ID do receptor	0x01	0x06		*Checksum*
Resposta (nenhuma etiqueta visível):							
0x55	0x00	ID da rede	ID do receptor	0x01	0x06		*Checksum*
Resposta (etiqueta visível):							
0x55	Tamanho dos dados	ID da rede	ID do receptor	0x01	0x06	Dados	*Checksum*

O campo *Dados* da resposta ao comando "Get Tag Packet" contém as informações disponíveis na etiqueta eletrônica. A informação utilizada por este projeto é o código dessa etiqueta, de 4 bytes, encontrado nos bytes 17 a 20 do campo *Dados*.

O submódulo *Cliente HTML* do objeto *Aplicação* é o responsável pela comunicação com o *Servidor de Dados* mediante requisições de serviços por *HTML*. O Cliente *HTML* captura a resposta à requisição e a armazena em um arquivo formatado que será tratado pelo objeto *Aplicação*.

O componente *Servidor de Dados* contém os objetos *Servidor HTML*, *SOAP*, *Servidor de Aplicação* e *Banco de Dados*, dos quais os três primeiros provem os serviços web mediante recepções de requisições do objeto *Aplicativo* e transmissões dos dados referentes obtidos por meio de consultas ao objeto *Banco de Dados*.

No objeto *Servidor de Aplicação* são disponibilizados os serviços que serão utilizados pelo objeto *Aplicação* do componente *Dispositivo Móvel*. Esses serviços abrangem a extração de detalhes do equipamento, retomada de tarefa pendente e obtenção de informações de tarefa concluída.

Os parâmetros de entrada do serviço *Detalhes de Equipamento* são as identificações do usuário e do equipamento, as quais são utilizadas na consulta ao objeto *Banco de Dados*, de onde se extraem as informações do equipamento pertinentes ao contexto do usuário que as solicita, emitidas por meio de um arquivo formatado. O serviço *Tarefa Pendente* retorna uma tarefa pendente para o usuário que a solicita, caso exista tal tarefa. O parâmetro de entrada é a identificação do usuário e a resposta é um arquivo formatado que detalha os campos a serem preenchidos para a tarefa pendente. O serviço *Tarefa Completa* recebe as informações coletadas em uma tarefa executada e as insere no *Banco de Dados*; a resposta é uma mensagem indicando sucesso ou fracasso da atualização do banco de dados.

3.5 Ponto de vista 'tecnologia'

Este ponto de vista estabelece as tecnologias utilizadas para desenvolver os objetos *engenharia*.

A aplicação do *Dispositivo Móvel* utiliza a tecnologia *SuperWaba/TotalCross*, selecionada em virtude de suas características de portabilidade, conforme apresentado anteriormente. O *Servidor de Aplicação* é executado com o *Apache Tomcat versão 6.0.24*, que é responsável por executar os *Servlets Java*, tecnologia utilizada para desenvolver os serviços a serem disponibilizados, com versões disponíveis para diversas plataformas sob licenciamento *GPL*. Como SOAP utiliza-se o *Apache Axis versão 1.4*, tecnologia que executa o SOAP para o *Apache Tomcat*. O *Banco de Dados*, por sua vez, é executado em *MySQL Server versão 5.0.75*.

4 Desenvolvimento do sistema protótipo

Com o propósito de facilitar o desenvolvimento do sistema de gerenciamento da cadeia de suprimentos de empresa de distribuição de energia elétrica, utilizando-se os elementos da arquitetura apresentados nas seções anteriores, elaborou-se um sistema protótipo constituído por três grandes módulos – banco de dados, servidor de aplicação e aplicativo móvel. Os casos de uso associados às funcionalidades previstas para o protótipo – *configuração da aplicação*, *localização de etiquetas*, *identificação de detalhes das etiquetas*, *identificação de tarefas pendentes* e *execução* – serviram de guia para a concepção e validação do protótipo. Na Tabela 5 descreve-se de forma detalhada, e a título de ilustração, o caso de uso *Consulta Etiqueta*.

Tabela 5. **Caso de uso *Consulta Etiqueta***

Descrição	Consulta etiquetas "visíveis" ao leitor RFID.
Atores	Usuário.
Pré-condições	Configurações da aplicação realizadas. Existência de um leitor RFID conectado ao dispositivo móvel.
Pós-condições	O usuário visualiza uma lista de etiquetas "visíveis".
Caminho primário	Usuário seleciona no menu o item "Consulta Tags". A aplicação interroga o leitor RFID sobre a existência de etiquetas detectáveis. A aplicação recebe a lista de etiquetas detectáveis do leitor *RFID* e exibe uma lista com estas etiquetas para o usuário.
Caminho de exceção	Usuário seleciona no menu o item "Consulta Tags". A aplicação interroga o leitor RFID sobre a existência de etiquetas detectáveis. A aplicação detecta erro na comunicação com o leitor RFID. A aplicação exibe um erro de comunicação para o usuário.

4.1 Modelagem dos dados

O banco de dados foi criado de maneira a atender ao objetivo central do projeto, ou seja, captura de dados dos equipamentos durante seu ciclo de vida em uma cadeia de suprimentos. Na Figura 7 apresenta-se o diagrama desse banco de dados, gerado com auxílio da ferramenta MySQL WorkBench. Conforme se pode observar nesse diagrama, o banco de dados é constituído por onze tabelas, responsáveis pelo armazenamento e manutenção da coerência dos dados. Sua estrutura permite que um equipamento esteja associado a dados coletados por usuários em tarefas executadas em diversos contextos. O modelo concebido possibilita ainda que se restrinja o acesso às informações de um dado equipamento, visto que o usuário de um contexto somente terá acesso aos dados de informações coletadas por esse contexto.

Nas tabelas *Usuário*, *Contexto* e *Tarefa* são armazenados, respectivamente, os dados dos usuários, os contextos possíveis e as tarefas passíveis de serem realizadas na cadeia de suprimentos. Os dados dos equipamentos são armazenados na tabela *Equipamento*; as informações capturadas, na tabela *Informacao*; os dados associados a um determinado equipamento, na tabela *Dado*. Na tabela *TipoEquipamento* definem-se os tipos esperados de equipamentos e, ao mesmo tempo, realiza-se a abstração do relacionamento equipamento-tarefa, pois as tarefas são relacionadas não com um equipamento em particular, mas com um dado *tipo* de equipamento. Na tabela *Tarefa_exe* armazenam-se os dados das tarefas executadas nos equipamentos pelos usuários da cadeia de suprimentos.

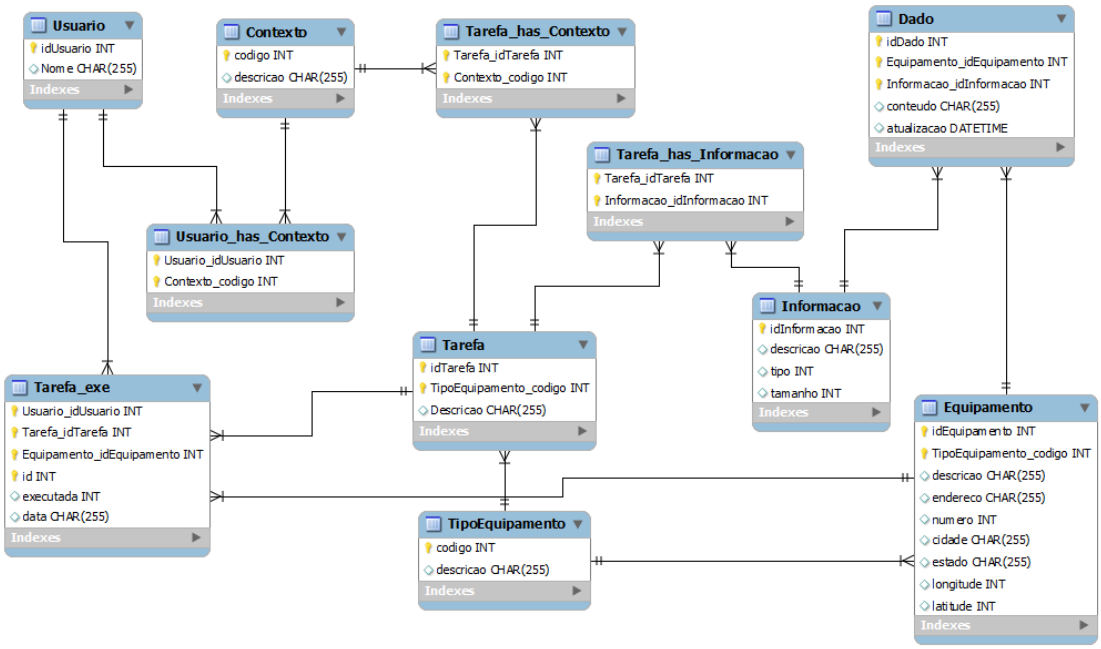


Figura 7. Modelo do banco de dados.

Para garantir a coerência dos dados criaram-se três tabelas: a tabela *Usuário_has_Contexto*, que estabelece a qual contexto um usuário pertence; a tabela *Tarefa_has_Contexto*, que define as tarefas possíveis em cada contexto; a tabela *Tarefa_has_Informacao*, que define as informações pertinentes a cada tarefa. Além dessas tabelas, elaboraram-se três *procedures* – *carrega_tarefa, detalhe_tag_pross* e *tarefas_pendentes* –, responsáveis pela coleta de dados dessas tabelas e disponibilização dos mesmos às aplicações que se conectam ao banco de dados; assim, evita-se a necessidade de se montarem comandos SQL específicos.

4.2 Servidor de aplicação

No protótipo desenvolvido, o servidor de aplicação é executado sobre um servidor Apache Tomcat e a publicação de serviços de web é realizada por meio do Apache Axis. Por se tratar de um protótipo, optou-se pela utilização do método elementar de publicação de serviços no Axis, conhecido pela sigla JWS (Java Web Services), o qual gera automaticamente o WSDL referente a uma classe Java, publicando os métodos desta classe como serviços. A publicação desses *serviços* requer apenas que se copie o código fonte da classe Java para

a pasta do *Axis* no *Tomcat* e que se atribua ao nome da mesma a extensão *.jws*. Quatro serviços são consumidos pelo aplicativo móvel: *detalhes*, *tarefa*, *tarefas_pend* e *insereDados*.

O serviço *detalhes* conecta-se ao banco de dados utilizando a classe *jdbc* para *mysql* e executando a *procedure detalhe_tag_pross*. Uma vez terminada a execução desta *procedure*, o serviço gera uma resposta que inclui a descrição da informação, o caractere ':' e o dado correspondente a esta informação; ao fim de cada item ele insere o caractere '\n'. Com isso, gera-se a resposta com os detalhes solicitados pelo usuário.

O serviço *tarefas_pend* executa a *procedure tarefas_pendentes* sobre o banco de dados, processa os dados de retorno e devolve, como resposta, uma lista de tarefas pendentes de execução por parte do usuário, na forma de uma cadeia de caracteres em XML. O serviço *tarefa* recebe o identificador da tarefa pendente, utilizando-o para aplicar a *procedure carrega_tarefa* ao banco de dados, cujos dados de retorno relativos à tarefa pendente de execução são processados de modo a gerar uma cadeia apropriada de caracteres XML. O serviço *insereDados* recebe uma cadeia de caracteres no formato XML com os itens a serem inseridos ou atualizados no banco de dados, e, de posse dessa cadeia de caracteres, monta um arquivo temporário. Finalmente, o serviço executa um *parser* XML sobre este arquivo, de modo tal que onde o campo *idDado* está preenchido com um dado diferente de 0 ocorre atualização do dado; em caso contrário, realiza-se a sua inclusão.

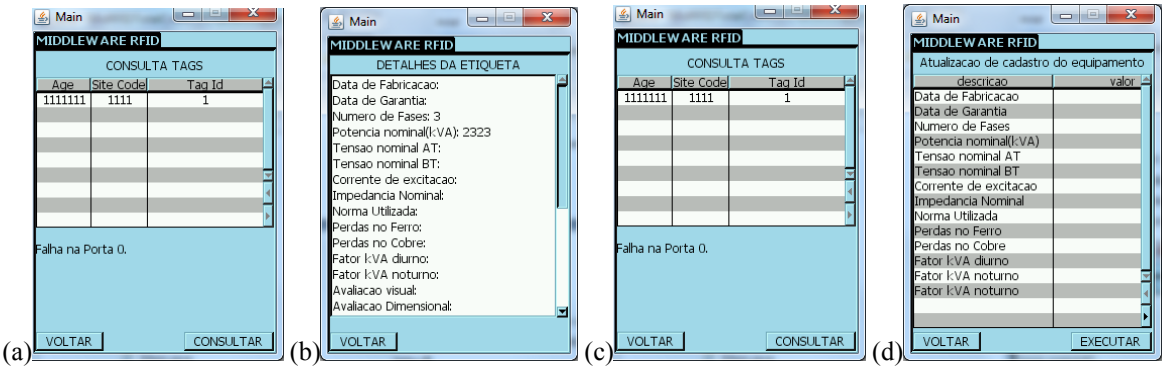
4.3 Aplicação do dispositivo móvel

A classe principal desta aplicação é a classe *Main*, extensão da classe *MainWindow* que é chamada pela *virtual machine* quando a aplicação é executada. É responsável pela criação das instâncias de todas as demais classes e responsável por eliminá-las quando não estiverem mais ativas. Nesta classe monta-se um menu de acesso às funcionalidades da aplicação e geram-se instâncias da classe *Núcleo* a cada 10 segundos. Além de armazenar até dez tarefas do usuário pendentes de execução, esta classe mantém o controle do *status* de conexão com o servidor de aplicação, o qual é utilizado pelas demais classes para selecionar seu modo de operação.

A classe *Configurações* cria uma interface para o preenchimento das configurações do aplicativo do dispositivo móvel. Os campos deste formulário abrangem o nome do usuário, sua senha, a porta serial utilizada para o leitor de etiquetas e o endereço do servidor de aplicação. Esta classe é responsável pelo preenchimento da instância da classe *Config*, criada pela classe *Main*, que armazena as configurações do dispositivo móvel.

A classe *Núcleo* é responsável por realizar as tarefas de *background* da aplicação, consumindo o serviço de tarefas do servidor de aplicação pendentes de execução e montando, na pasta *tarefas,* arquivos em formato XML com os dados a serem coletados. A mesma classe ainda consome o serviço *insereDados*, enviando os dados das tarefas já executadas pelo usuário, armazenadas na pasta *tarefas_exe*. Havendo conexão com o servidor de aplicação, a classe N*úcleo* envia os arquivos, removendo-os em seguida do dispositivo móvel; em caso contrário, os arquivos permanecem armazenados no dispositivo móvel.

A classe *LeitorEtiquetas* executa o protocolo de comunicação com o leitor de etiquetas eletrônicas, exibindo uma lista de etiquetas visíveis (vide Figura 8a). A comunicação com o leitor de etiquetas é realizada pela porta serial do dispositivo móvel. A classe *LeitorEtiquetas* utiliza instâncias da classe *Checksum* para executar o cálculo do byte de verificação do conteúdo enviado ou recebido pelo leitor, enquanto a classe *LeitorComandos* especifica os tipos de mensagens utilizadas na comunicação Leitor-Etiqueta. É importante salientar que a classe *LeitorEtiquetas* contém um temporizador responsável por ciclos de verificação da existência de etiquetas nas vizinhanças do leitor. Cabe destacar também que, ao selecionar uma etiqueta na lista exibida, o usuário pode requisitar detalhes da mesma. Nessas circunstâncias, a classe *LeitorEtiquetas* cria uma instância da classe *Detalhes*, caso exista conexão com o servidor de aplicação. A classe *Detalhes*, por sua vez, é responsável por consumir o serviço *detalhes* do servidor de aplicação, exibindo todos os dados do equipamento selecionado na lista da classe *LeitorEtiquetas* permitidos ao usuário. Os dados são exibidos em um campo de texto, não sendo possível a alteração dos mesmos (vide Figura 8b).



(a) (b) (c) (d)

Figura 8. Interface exibida pela classe: (a)*LeitorEtiquetas*; (b)*Detalhes*; (c) *ListaTarefas*; (d) *ExecutaTarefa*.

A classe *ListaTarefas* é responsável por exibir uma lista das tarefas solicitadas pelo usuário e pendentes de execução (vide Figura 8c), que se encontram armazenadas no dispositivo móvel. Em tais circunstâncias, essa classe oferece os meios para que o usuário selecione e execute uma dessas tarefas, mediante a ativação da classe *ExecutaTarefa*, a qual recebe a tarefa selecionada na classe *ListaTarefa* e abre o arquivo XML referente à mesma, armazenado na pasta *tarefas* do dispositivo móvel. Os dados deste arquivo são submetidos a um *parser* XML, que preenche uma planilha contendo duas colunas: a primeira, com a informação a ser coletada; a segunda com o dado coletado para ser posteriormente alterado, ou um campo em branco para a coleta de um novo dado (vide Figura 8d). Ao ser ativado um botão para finalizar a execução da tarefa, gera-se um arquivo na pasta *tarefas_exe* contendo os dados a serem enviados ao servidor de aplicação.

4.4 Testes conceituais preliminares

Após o desenvolvimento do sistema protótipo, executaram-se testes conceituais baseados nos casos de uso citados, ou seja, verificou-se se todas as funcionalidades previstas nesses diagramas de casos de uso estavam efetivamente disponíveis aos usuários.

Finda a etapa de identificação e correção de erros de implementação, atingiu-se a plena concordância entre as funcionalidades previstas nos diagramas de caso de uso e o efetivo comportamento do protótipo, que se mostrou capaz, portanto, de ler os códigos RFID de equipamentos da cadeia de suprimentos por meio de dispositivos leitores móveis e, utilizando esses códigos, extrair e/ou modificar informações desses equipamentos mediante consulta a uma base de dados central.

Embora esses testes conceituais tenham auxiliado a ajustar o protótipo às especificações de projeto, é importante destacar que, em virtude de o ambiente de execução ser totalmente virtual, seria necessário submeter o protótipo a variados testes adequados de concorrência e carga antes de se poder considerá-lo como uma aplicação finalizada.

5 Conclusões

Tendo em vista os objetivos traçados e os resultados dos testes baseados nos casos de uso definidos na seção anterior, pôde-se concluir que, conceitualmente, o sistema protótipo é capaz de coletar os dados sobre os processos que ocorrem durante o ciclo de vida dos equipamentos utilizados em uma rede de distribuição de energia elétrica, identificados por meio de etiquetas RFID, e de disponibilizá-los em um banco de dados único. Quanto ao requisito de multiplataforma e as limitações de desempenho dos dispositivos móveis, a escolha do *Superwaba /Totalcross* mostrou-se adequada, já que disponibiliza *virtual machines* para uma gama de sistemas operacionais (*Apple iPhone, Blackberry, Palm OS, Android, Windows CE/Pocket PC* e *Opticon*) que abrange a maioria dos dispositivos móveis disponíveis na atualidade, possibilitando a execução do sistema nos mesmos. Durante os testes baseados nos casos de uso não se identificou nenhuma restrição de desempenho operacional; contudo, pôde-se evidenciar que, como arquivos temporários são gravados no dispositivo móvel, a disponibilidade de espaço na área do usuário é um fator determinante para sua operacionalidade.

O uso do RM-ODP como modelo de referência para o projeto da arquitetura do que viria a ser o sistema protótipo mostrou-se importante na obtenção de um sistema que atendesse aos requisitos estabelecidos, uma vez

que se observa grande similaridade entre os documentos do sistema protótipo desenvolvido e a arquitetura proposta.

A utilização do servidor de aplicação como forma de abstrair o acesso ao banco de dados e descentralizar o processamento no tratamento desses dados torna possível o desenvolvimento de outras aplicações que se utilizem do mesmo servidor de aplicação, bem como dos serviços publicados para outros propósitos. A própria integração de dados entre o banco de dados desenvolvido e os ERPs dos participantes da cadeia de suprimentos pode ocorrer mediante o uso desse servidor de aplicação.

Uma expansão natural do sistema projetado poderia vir a ocorrer mediante a inclusão nas etiquetas eletrônicas de informações provenientes de uma rede de sensores capazes de realizar leituras de variáveis físicas, como temperatura, pressão, tensão e corrente. Com tais medições, realizadas em tempo real, seria possível implantar na corporação programas de manutenção preditiva de equipamentos da rede de distribuição elétrica.

Referências

[1] FURTADO, P.C.; CARVALHO, M.F.H. Compartilhamento da informação como elemento de coordenação da produção em cadeia de suprimento. **Gestão & Produção**. v. 12, no. 1, 2005, p.39-53.

[2] FAWCETT, S.E.; STANLEY, L.L.; SMITH, S.R. Developing a logistics capability to improve the performance of international operations. **Journal of Business Logistics**, v.18, no. 2, 1997, p. 101-107.

[3] CHOPRA,S.; MEINDL, P. **Gerenciamento da Cadeia de Suprimentos – Estratégia, Planejamento e Operação.** Prentice Hall, 2006, 477p.

[4] LU, B.H., BATEMAN, R.J., CHENG, K., 2006, RFID enabled manufacturing: fundamentals, methodology and applications. **International Journal of Agile Systems and Management**, v.1, no. 1, p.73-92.

[5] NICOLAI, T., SINDT, T., KENN, H., WITT, H., Case study of wearable computing for aircraft maintenance. In: 2nd INTERNATIONAL FORUM ON APPLIED WEARABLE COMPUTING (IFAWC), Zürich, Switzerland, March 17-18, 2005. **Proceedings of the 2nd International Forum on Applied Computing**. Zürich: 2005. 14p.

[6] WANG, S.W., CHEN, W.H., ONG, C.S., LIU, L., CHUANG, Y.W., RFID applications in hospitals: a case study on a demonstration RFID project. In: 39th HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, Kauai, Hawaii, 2006. **Proceedings of the 39th International Conference on System Sciences**. Kauai, Hawaii: 2006. 10p.

[7] WEISER, M., Some computer science issues on ubiquitous computing. **Communications of the ACM**, v.36, no. 7, 1993, p. 75-84.

[8] SCHILIT, B.N., ADAMS, N., WANT, R., Context-Aware Computing Applications. IN: 1st WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS, Santa Cruz, California, Dec 8-9, 1994. **Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications**. Santa Cruz, California: 1994. p. 85-90.

[9] KRAKOWIAK, S., **Middleware Architecture with Patterns and Frameworks**, 2009. Disponível em: http://sardes.inrialpes.fr/~krakowia/MW-Book/.

[10] FLOERKEMEIER, C., LAMPE, M., RFID middleware design – addressing both application needs and RFID constraints. In: CONFERENCE ON SMART OBJECTS & AMBIENT INTELLIGENCE, Oct. 12-14, 2005, Grenoble, France. **ACM International Conference Proceeding Series**, vol. 121, 2005, p. 219-224.

[11] PRABHU, B.S., SU, X., RAMAMURTH, H., CHU, C.C., Rajit GADH, R., WinRFID – a middleware for the enablement of Radio Frequency Identification (RFID) based Applications. In: **Mobile, Wireless and**

Sensor Networks: Technology, Applications and Future. Eds. Shorey, R., Choon, C.M., Tsang, O.W., Ananda, A., 2005.

[12] LIM, F., CHEN, B., CHAN, C.Y., WU, C.H., IP, W.H., MAI, A., WANG, H., LIU, W., The design of a lightweith RFID middleware. **International Journal of Engineering Business Management**, Vol. 1, no. 2, 2009, p. 73-78.

[13] EPC GLOBAL, Application Level Events (ALE) Standard, version 1.1.1. In: http://www.gs1.org/gsmp/kc/epcglobal/ale. Data do Último acesso: 12/03/2011.

[14] MIT AUTO-ID CENTER, Auto-ID Savant Specification 1.0, version of 1 September 2003. In: http://www.amece.org.mx/amece/Documentos/estandares/epc/WD-savant-1_0-20030911.pdf Center, 2002.

[15] NURMINEM, T., The End of RFID Middleware? **RFID Journal**, p.2. In: http://www.rfidjournal.com/article/view/2035/2. Data do Último Acesso: 12/03/2011.

[16] KIM, Y., YI, B., SONG, J., SHIN, J., LEE, J., Implementing prototype system for power facility management using RFID/WSN. **International Journal of Mathematical and Computer**, Vol. 2, no. 2, 2006., P.70-75.

[17] PRESSMAN, R., **Engenharia de Software**. McGraw-Hill, 6ª edição. 2006. 720p.

[18] IPT, **Controle logístico inteligente de equipamentos de distribuição em estoque e instalados em campo: especificações iniciais do software e do middleware**. Instituto de Pesquisas Tecnológicas do Estado de São Paulo: Relatório técnico no. 94981-205. 2007.

[19] ISO, **International Standard ISO/IEC 10746-1: Information technology – Open Distributed Processing – Reference Model: Overview**. 1998.

[20] FERREIRA, C.L., **Maestro: um middleware para suporte a aplicações distribuídas baseadas em componentes de software.** Dissertação apresentada à Escola Politécnica da Universidade de São Paulo. 2001.

[21] PALUDO, L., **Um Estudo sobre as Tecnologias Java de Desenvolvimento de Aplicações Móveis**. Monografia apresentada ao Departamento de Informática e Estatística da Universidade Federal de Santa Catarina (UFSC). 2003.

[22] MORAES, D., LOPES, P.R.L., PISA, I.T., Protótipo para coleta de informações em saúde utilizando dispositivos móveis. In: IX CONGRESSO BRASILEIRO DE INFORMÁTICA EM SAÚDE, Ribeirão Preto, São Paulo, 7-10 nov. 2004. **Anais do IX Congresso Brasileiro de Informática em Saúde.** 2004. 4p.

[23] LOUTFI, M.S., **Desempenho da Linguagem Java em Dispositivos Embutidos.** Dissertação apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia. 2003.

[24] W3C. **Web Services Architecture.** Disponível em http://www.w3.org/TR/ws-arch. 2004.