UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

DANIEL TERZELLA CARDOSO

Coleta e Geração de Dados de Medição de Energia para o Novo Modelo Elétrico Brasileiro

FLORIANÓPOLIS
2004

DANIEL TERZELLA CARDOSO

Coleta e Geração de Dados de Medição de Energia para o Novo Modelo Elétrico Brasileiro

Monografia apresentada à Universidade Federal de Santa Catarina, como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação, sob Orientação do Prof. Frank Augusto Siqueira.

FLORIANÓPOLIS
2004

DANIEL TERZELLA CARDOSO

Coleta e Geração de Dados de Medição de Energia para o Novo Modelo Elétrico Brasileiro

Monografia aprovada em, 21/06/2004, como requisito para obtenção do Grau de Bacharel em Sistemas de Informação.

BANCA EXAMINADORA

_____
Professor Frank Augusto Siqueira
Orientador

_____
Professor João Bosco Mangueira Sobral
Membro

_____
Professor Ronaldo dos Santos Mello
Membro

Resumo

De acordo com o novo modelo do setor elétrico brasileiro as empresas geradoras, distribuidoras e comercializadoras de energia elétrica deverão disponibilizar periodicamente dados de medição de energia que serão registrados no MAE (Mercado Atacadista de Energia Elétrica). Desta forma, será possível determinar as diferenças entre a quantidade de energia produzida e consumida pelos clientes de cada empresa participante do mercado de energia e o que foi contratado no MAE. Todo este processo é necessário para que possa ser feito o acerto de contas entre as empresas geradoras, distribuidoras e comercializadoras de energia elétrica participantes do mercado atacadista de energia. Este processo é totalmente automatizado a partir de um sistema de software disponibilizado pelo MAE, que no entanto precisa ser integrado ao sistema de medição de cada uma das empresas para obter os dados de medição.

O presente trabalho consiste no projeto de implementação de um software que efetuará a interface entre o MAE e os sistemas de bancos de dados das empresas, que contém os dados de medição. Este software deve funcionar como uma ferramenta de geração, coleta e acompanhamento das informações de medição que são disponibilizadas ao MAE na forma de arquivos XML que são coletados em horários previamente agendados.

Sumário

Lista de Tabelas e Figuras

Lista de Abreviaturas e Siglas

ANEEL – Agência Nacional de Energia Elétrica

ASP – Active Server Page

DOM – Document Object Model

DTD – Document Type Definitions

JDBC – Java DataBase Connectivity

MAE – Mercado Atacadista de Energia Elétrica

ME – Medição de Energia

ODBC – Open Database Connectivity

ONS – Operador Nacional do Sistema Elétrico

QEE – Qualidade de Energia Elétrica

SAX – Simple API for XML

SCDE – Sistema de Coleta de Dados de Energia

SGBD – Sistema de Gerenciamento de Banco de Dados

SQL – Structured Query Language

UCM – Unidade Central de Coleta de Dados

W3C – World Wide Web Consortium

XML – Extensible Markup Language

1 Introdução

1.1 Apresentação

O mercado Brasileiro de energia elétrica está passando por uma reestruturação. Dentre as reformulações instituídas, decidiu-se pela implantação de um novo sistema de contabilização da energia elétrica produzida e consumida no Brasil. Esta contabilização leva em consideração toda energia contratada pelos Agentes de Mercado, empresas de distribuição e geração, e toda energia efetivamente consumida ou gerada. Aos Agentes de Mercado coube adquirir e instalar os sistemas de medição e os próprios medidores. É atribuição, ainda, destes Agentes, através de processos automáticos, disponibilizarem os dados de medição.

As empresas geradoras, distribuidoras e comercializadoras de energia elétrica registram no MAE (Mercado Atacadista de Energia Elétrica) os montantes de energia contratada. O projeto proposto nesse documento objetiva implementar um aplicativo que efetuará a coleta dos dados de medição dos Agentes de Mercado e a sua disponibilização para o MAE.

1.2 Justificativa

Com a necessidade de uma evolução no processo de contabilização tornou-se indispensável uma redefinição da sistemática de medição para faturamento. Assim, decidiu-se pela implantação e operação de um novo sistema de medição de faturamento para o Sistema Elétrico Brasileiro.

As concessionárias de energia viram-se obrigadas a adotar um novo modelo de medição. Assim, as empresas geradoras, distribuidoras e comercializadoras de energia elétrica deverão disponibilizar periodicamente dados de medição de energia que serão registrados no MAE, para que desta forma se possa determinar quais as diferenças entre o que foi produzido ou consumido e o que foi contratado.

1.3 Objetivos

1.3.1 Objetivo geral

O projeto proposto objetiva desenvolver um aplicativo que efetuará a interface entre o Client SCDE[1] e os dados de energia. Este aplicativo, ao ser instalado nos servidores de um agente de mercado, se responsabilizará pela geração dos dados de medição referentes ao consumo de energia de seus clientes, conforme solitado pelo MAE. A geração dos dados de medição será efetuada acessando a base de dados do Agente, obtendo as informações de medição e convertendo-as em um arquivo XML padronizado. Após a sua geração, os dados serão coletados de maneira automática pelo MAE, que estabelecerá uma conexão TCP/IP em um horário previamente agendado para obter os dados que permitirão a contabilização do consumo.

1.3.1 Objetivos específicos

Como objetivos específicos, pode-se citar:

- Escolher uma linguagem compatível com banco de dados e arquitetura do sistema utilizada, buscando o melhor desempenho;
- Desenvolver a aplicação de acordo com as especificações do MAE [MAE04];
- Desenvolver uma solução de forma a integrá-la a um sistema de medição de energia;

[1] Client SCDE: software desenvolvido em linguagem JAVA, a ser instalado nos Agentes e que enviará ao MAE, arquivos contendo os dados de medição de energia. Este software é disponibilizado pelo MAE

1.4 Metodologia de Pesquisa

Este trabalho foi desenvolvido da seguinte maneira:

- Primeiramente, foi realizado um estudo do problema a ser resolvido e das tecnologias possíveis de serem utilizadas para a sua solução. Como é no Mercado Atacadista de Energia Elétrica (MAE) [MAE04] que ocorre o processamento da contabilização da Energia Elétrica produzida e consumida no Brasil, grande parte das informações necessárias para o projeto foram obtidas através do MAE [MAE04], ou, alternativamente, nas documentações e publicações disponibilizadas pela ONS [ONS04] e ANEEL [ANE04]. Outras fontes de pesquisa bibliográfica envolvidas neste trabalho tratam das tecnologias empregadas no desenvolvimento do aplicativo.
- Em um segundo momento, foi realizada a análise do problema e desenvolvido o projeto da aplicação, utilizando técnicas de modelagem orientada a objetos.
- Em seguida, foi realizada a implementação da aplicação usando as tecnologias estudadas que se mostraram mais adequadas para serem empregadas neste cenário.
- Na etapa posterior, foram realizados testes para avaliar o funcionamento da aplicação. Durante os testes, foram detectadas falhas que foram solucionadas efetuando-se correções na implementação da aplicação.
- Por fim, foi elaborada a documentação do projeto, da qual faz parte este relatório.

1.5 Estrutura do Trabalho

Este trabalho está organizado em mais cinco capítulos, cujos conteúdos são descritos a seguir.

O capítulo 2 visa apresentar ao leitor as tecnologias utilizadas neste projeto, mostrando uma visão geral de cada tecnologia e justificando a razão da sua adoção para desenvolvimento da aplicação.

O capítulo 3 apresenta uma descrição do Sistema de Coleta de Dados de Energia (SCDE), visando contextualizar o leitor.

O capitulo 4 descreve o funcionamento esperado da Aplicação MAE.

Já o capítulo 5 apresenta ao leitor informações sobre a implementação da Aplicação MAE.

Finalmente, o capítulo 6 mostra as conclusões e os resultados obtidos neste trabalho.

2 Tecnologias

Este capítulo destina-se a apresentar o resultado do estudo sobre as ferramentas e tecnologias utilizadas para o desenvolvimento do projeto, incluindo linguagens de programação e representação de dados, publicação web e SGBD [RAM00].

2.1 Java

Java [DEI01] é uma linguagem de programação orientada a objetos desenvolvida pela SUN Microsystems. Ela foi projetada para ser independente de plataforma, permitindo que o mesmo código possa ser portado entre plataformas sem necessidade de modificação e recompilação do mesmo.

A linguagem Java possui diversas características que a torna uma ótima linguagem para escrever aplicações para servidores. Dentre elas, pode-se listar:

- Simplicidade: Java é mais simples que outras linguagens utilizadas para esse fim por causa de seu direcionamento ao modelo orientado a objetos. O grande conjunto de bibliotecas e classes padrão traz ferramentas poderosas para desenvolvedores Java de todas as plataformas;
- Portabilidade: Java é portável entre plataformas. É possível escrever código dependente de plataforma em Java, mas também é possível escrever programas que podem ser portados transparentemente entre plataformas;
- Gerenciamento Automático de Memória: Em Java, não é necessário alocar e desalocar memória explicitamente. A máquina virtual conta com um sistema

que libera automaticamente objetos em memória que não estejam mais sendo referenciados, conhecido como "*garbage collection*" ("coleta de lixo");

- Tipos Fortes: Uma variável em Java precisa ser declarada antes de ser utilizada, especificando que tipo de objeto será armazenado nela. Isso torna possível fornecer uma solução segura para chamadas inter-linguagens entre Java e PL/SQL [ORA04], por exemplo, e a integrar chamadas para Java e SQL na mesma aplicação;
- Sem Ponteiros: Apesar de herdar muito da sintaxe da linguagem C [KER88], Java não permite manipulação direta de ponteiros, o que elimina "vazamento" e corrupção de memória;
- Tratamento de Exceções: Em Java, não é necessário retornar códigos especiais de erros para tratar situações inesperadas, pois já existem classes e instruções responsáveis pelo tratamento de exceções.
- Segurança: A máquina virtual disponibiliza mecanismos para autenticar e controlar os direitos de acesso dos usuários. A máquina virtual também verifica as classes em tempo de execução para se certificar de que elas não foram corrompidas ou alteradas indevidamente.
- Padrões para Conectividade com Banco de Dados Relacionais: O padrão JDBC (*Java DataBase Connectivity*) [WHI02] permite que se escreva código para acesso a banco de dados de modo independente do fabricante.

A linguagem foi desenvolvida com a preocupação de impedir que sejam inseridos executáveis maliciosos que possam corromper o sistema operacional e executar operações não permitidas. Algumas linguagens, como C [KER88], podem introduzir problemas de

segurança no servidor de banco de dados. Java, graças à sua arquitetura, é uma linguagem segura para ser utilizada neste cenário.

2.2 JDBC

A linguagem Java é servida por diversas APIs (*Application Programming Interface*) que permitem uma constante evolução da linguagem frente as novas tecnologias. Para acesso ao banco de dados, como citado anteriormente, os programas em Java utilizam-se da API JDBC [WHI02].

A API JDBC é uma interface padrão criada para fazer a comunicação com bancos de dados relacionais em Java de forma independente de qual SGBD está sendo utilizado. JDBC consiste de duas partes: a API de alto nível e múltiplos *drivers* de baixo nível para conexão com diferentes bancos de dados. Assim, na teoria, é possível escrever código independente do banco de dados e mudar o SGBD utilizado apenas alterando o *driver* JDBC.

A grande vantagem do JDBC é permitir que uma aplicação acesse virtualmente qualquer fonte de dados e execute em qualquer plataforma com uma máquina virtual Java. Em outras palavras, com a API, não é necessário escrever um programa para acessar um banco de dados SQL Server [SHA01], outro para um banco Oracle [LON00], outro para IBM DB2 [DB204], e assim por diante. Um programa escrito com a API JDBC poderá enviar comandos SQL ou outros comandos para a fonte de dados apropriada, através do *driver* que a implementa.

Além de todas vantagens citadas, a linguagem Java ainda apresenta perfeita sintonia na integração e utilização das tecnologias que serão necessárias neste trabalho. Dentre seus

mecanismos, podemos destacar aqui a possibilidade de invocação de *Stored procedures*, *parsers* para manipulação de arquivos XML e classes para validação de arquivos XML via *Schemas XML*. Estes itens serão descritos com mais detalhes no decorrer do trabalho.

2.3 SGBD

Um banco de dados é uma coleção de dados estruturados que utiliza-se de um sistema de gerenciamento de bancos de dados (SGBD) para efetuar a interface entre os dados e o usuário. O SGBD é responsável pela adição, acesso e processamento dos dados armazenados. Como exemplo de SGBD podemos citar: Oracle [LON00], IBM DB2 [DB204], MySQL [SUE02], Sybase [SYB04] e o Microsoft SQL Server [SHA01].

Os dados podem estar organizados de diferentes formas dependendo do modelo lógico que se queira adotar. Como exemplo de modelos lógicos podemos citar:

- Modelos Orientados a Objetos: são modelos que procuram representar as informações através dos conceitos típicos da Programação Orientada ao Objeto (POO), utilizando o conceito de classes que irão conter os objetos;
- Modelo Relacional: as informações estão organizadas em forma de tabelas (linhas x colunas) inter-relacionadas. É o tipo mais utilizado de banco de dados nos dias atuais.
- Modelo XML Nativo: define um modelo lógico para um documento XML e armazena e recupera documentos de acordo com esse modelo.

Um meio de alcançar um considerável ganho de desempenho no acesso a bancos de dados consiste no uso de S*tored procedures* [DEI01], que são equivalentes às chamadas de funções (ou métodos) nas linguagens de programação tradicionais. Alem disso a utilização

de *Stored procedures* traz diversas vantagens incluindo melhor desempenho, facilidade de uso e manutenção, e maior escalabilidade da solução.

- Desempenho: Stored procedures são compiladas uma vez e armazenadas em formato executável, fazendo com que chamadas a elas sejam rápidas e eficientes. O código executável é compartilhado pelos usuários, reduzindo os requerimentos de memória e o *overhead* de sua invocação. Agrupando os comandos SQL numa *Stored procedure* permite que eles sejam executados em uma única chamada. Isso minimiza a utilização de redes lentas, reduz o tráfego na rede e melhora o tempo de resposta;
- Produtividade e Facilidade de Uso: escrevendo aplicações com um conjunto comum de *Stored procedure*, evita-se repetição de código e aumenta-se a produtividade. Além disso, elas permitem estender a funcionalidade do SGBD. Por exemplo, funções chamadas a partir de comandos PL/SQL [ORA04] aumentam o poder da linguagem.
- Escalabilidade: pelo fato de isolar o processamento realizado pela aplicação no servidor, *Stored procedure* aumentam a escalabilidade.
- Manutenção: uma vez validada, uma *Stored procedure* pode ser chamada a partir de qualquer quantidade de aplicações. Se sua definição mudar, apenas ela é afetada e não as aplicações que a utilizam. Além disso, manter essa funcionalidade no servidor é mais fácil que manter cópias em diferentes clientes.

Buscando aliar às vantagens do uso de *Stored procedure* e o uso da linguagem Java os bancos de dados mais recentes já possuem máquina virtual Java (JVM) embutida no próprio SGBD. Desta forma é possível desenvolver classes Java que são carregadas no

banco e executadas a partir de chamadas de um aplicativo ou no próprio banco na forma de *Stored procedure*.

Esse forma de implementação traz diversas vantagens, como:

- Robustez: ao contrário da linguagem SQL, Java permite a declaração e o lançamento de exceções.
- Escalabilidade: como Java é uma linguagem Orientada a Objetos, pode-se tirar todas as vantagens deste paradigma.
- Portabilidade: como o número de bancos com suporte a JVM vem crescendo, classes Java não só podem ser executas em múltiplas plataformas como permitem uma independência do SGBD.
- Maior complexidade: a linguagem Java permite o desenvolvimento de algoritmos mais complexos que os criados com SQL.

O SGBD escolhido para este trabalho foi a plataforma Oracle 8i [LON00] utilizando banco de dados relacional. Além de ser uma ótima solução de SGBD para aplicações de grande porte e acopla diversas tecnologias que fazem parte do escopo deste projeto (*Stored procedures*, Java *Stored procedures* e suporte a XML nativo), o ambiente escolhido para fazer a validação do aplicativo desenvolvido neste trabalho utiliza o Oracle 8i.

Apesar de muitas das tecnologias que o Oracle 8i oferece não serem utilizadas neste trabalho, como por exemplo o suporte a XML nativo, elas são consideradas importantes para futuras extensões.

2.4 XML

Extensible Markup Language (XML) [HEI01] é uma linguagem de marcação de dados (*meta-markup language*) que provê um formato para descrever dados estruturados, padronizada pela W3C (*World Wide Web Consortion*). Ela permite declarações mais precisas do conteúdo dos dados, facilita a troca de dados entre aplicativos e também dá suporte a uma nova gama de aplicações de manipulação e visualização de dados via Internet.

Assim como HTML, o XML deriva do SGML (*Standard Generalized Markup Language*). Ambos identificam elementos em uma página e ambos utilizam sintaxes similares. A grande diferença entre HTML e XML é que o HTML descreve a aparência e as ações em uma página na rede, enquanto o XML não descreve nem aparência e ações, mas sim o conteúdo do documento.

O XML permite a definição de um número infinito de *tags*. Assim o usuário pode definir suas próprias *tags*, e dessa forma cada aplicação pode ter *tags* específicas dela, como por exemplo <casa>, <quarto>, <area>, etc. Já no HTML, *tags* predefinidas (<b>, <i>, <table>, etc.) são usadas para definir a formatação de caracteres, parágrafos etc.

XML fornece uma maneira de estruturar e tratar os dados que antes eram tratados como texto puro. No exemplo ilustrado na Figura 1, é mostrada a estrutura de uma mensagem em XML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mensagem>
        <para>Rogerio</para>
```

```
        <de>Fernando</de>
        <assunto>XML</assunto>
        <corpo>Estudar o assunto!</corpo>
    </mensagem>
```

Figura 1 – Exemplo de documento XML [HEI01]

As regras de sintaxe de XML são muito simples de serem entendidas, pois documentos XML usam uma sintaxe auto-descritiva. Na primeira linha estão declaradas a versão de XML e o padrão de codificação de caracteres usado no documento. O restante da mensagem é auto-explicativo. Está bastante claro que este trecho de código significa que Fernando está querendo passar uma mensagem para Rogério informando que ele deve estudar o assunto XML.

2.5.1 XML e XSL

Ao contrário de HTML, quando definimos uma *tag* em XML o browser não sabe como deverá mostrá-la na tela. Isto se explica devido à natureza de XML, que não define uma forma padrão de exibir um documento XML. Para resolver esse problema, é necessário usar um mecanismo para descrever como um documento deve ser exibido. Um dos mecanismos é o CSS (*Cascading Style Sheets*), também usado em HTML. Entretanto, para documentos XML geralmente é usado o XSL (*eXtensible Stylesheet Language*), que é bem mais completo que o CSS.

O XSL é formado por duas partes: um método para transformar documentos e outro para formatar documentos. XSL pode ser usado para definir como um arquivo XML deve ser exibido, transformando o arquivo XML num formato que pode ser reconhecido por um

browser, por exemplo. Além disso, XSL é poderoso o suficiente para rearranjar e ordenar elementos, testar e tomar decisões sobre que elementos exibir, entre outras funcionalidades.

2.5.2 XML e DTD

No XML, as regras que definem um documento são ditadas por DTDs (*Document Type Definitions*), que ajudam a validar os dados quando a aplicação que os recebe não possui internamente uma descrição do dado que está recebendo. Um analisador de documentos pode checar os dados que chegam analisando as regras contidas no DTD para ter certeza de que o dado foi estruturado corretamente.

Pode-se dizer que a finalidade de um arquivo DTD é definir os possíveis elementos para a construção de um documento XML. Basicamente, ele define a estrutura do documento com uma lista de elementos disponíveis para uso. Um DTD pode ser declarado no mesmo arquivo do documento XML, conforme ilustrado na Figura 2, ou fazendo uma referência a um arquivo externo.

```
<?xml version="1.0"?>
<!DOCTYPE mensagem [
  <!ELEMENT mensagem   (para,de,assunto,corpo)>
  <!ELEMENT para       (#PCDATA)>
  <!ELEMENT de         (#PCDATA)>
  <!ELEMENT assunto            (#PCDATA)>
  <!ELEMENT corpo              (#PCDATA)>
]>
<mensagem>
        <para>Rogerio</para>
```

```
        <de>Fernando</de>
        <assunto>XML</assunto>
        <corpo>Estudar o assunto!</corpo>
</mensagem>
```

Figura 2 - Exemplo de documento XML com DTD [HEI01]

2.5.3 XML Schema

A linguagem XML Schema surgiu no intuito de permitir que pessoas definam, através de regras, a estrutura, o conteúdo e a semântica de um documento XML. Um XML Schema possui a mesma função que um DTD, porém possui uma maior poder de validação, como mostrado abaixo:

- Um Schema podem ser estendido, herdando a sintaxe de outros esquemas;
- XML Schema é descrito na própria linguagem XML;
- XML Schema permite não somente especificar a sintaxe de um documento, como um DTD, mas é possível também especificar os tipos de dados reais do conteúdo de cada elemento;
- XML Schema pode restringir os intervalos dos valores que os documentos podem conter;
- XML Schema permite validar um documento XML sem necessitar que este faça referência ao Schema;

A Figura 3 mostra um exemplo de XML Schema que faz a validação do exemplo XML de mensagem que foi apresentado na Figura 1. Pode-se observar que o XML Schema possui uma sintaxe baseada em XML.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
<xs:element name="mensagem">
   <xs:complexType>
    <xs:sequence>
          <xs:element name="para" type="xs:string"/>
          <xs:element name="de" type="xs:string"/>
          <xs:element name="assunto" type="xs:string"/>
          <xs:element name="corpo" type="xs:string"/>
    </xs:sequence>
   </xs:complexType>
</xs:element>
</xs:schema>
```

Figura 3 - Exemplo de XML Schema [HEI01]

Como já citado, o DTD possui uma série de limitações que XML Schema não possui. Uma dessas limitações que torna impossível o desenvolvimento de uma série de aplicações é a falta de mecanismos para especificação de tipos de dados, especialmente para determinar o tipo de conteúdo de um elemento. Através de uma DTD não podemos dizer, que um elemento com o nome VALOR_ENERGIA deva conter um número, muito

menos que ele deva ser inteiro, ou com dois dígitos decimais de precisão. Já o XML Schema permite que sejam definidas estas características, além de poder definir os tipos para o conteúdo de seus elementos e atributos, como *boolean*, *string* e *date,* dentre outros.

A validação dos documentos XML pode ser feita a partir de *parsers* (analisadores) XML. Desta forma, um *parser* pode determinar através de um DTD ou Schema a validade de um documento XML, ou seja, a utilização dessa estrutura poderá determinar se um documento XML foi escrito seguindo as regras de uma determinada gramática. Este processo de determinar se um documento XML está escrito de acordo com as regras de uma DTD ou um XML Schema é denominado de validação. Este processo será melhor descrito no item seguinte.

2.5.4 Parsers XML

Para efetuar a validação de um documento XML é necessário a utilização de *parsers* XML, que permitem efetuar a análise do documento. *Parsers* são disponibilizados na forma de APIs para que possam ser acessados a partir de programas. Estas APIs estão divididas basicamente em duas categorias: as baseadas em eventos e as baseadas em árvores. Nesta sessão serão descritas as principais implementações de cada categoria.

2.5.5 DOM

O DOM (*Document Object Model*) [DOM04], um exemplo de uma API baseado em árvore desenvolvido pela W3C, define uma funcionalidade padrão para navegação e manipulação de conteúdo e de estrutura em documentos XML e HTML. O DOM transforma o arquivo de texto (documento XML) em uma estrutura de árvore onde cada elemento é considerado como um nó da árvore gerada.

Neste modelo, o documento XML inteiro é armazenado na memória num formato de árvore de nodos, todos descendendo de uma raiz. O programador pode então aplicar vários métodos para localizar e manipular os nodos.

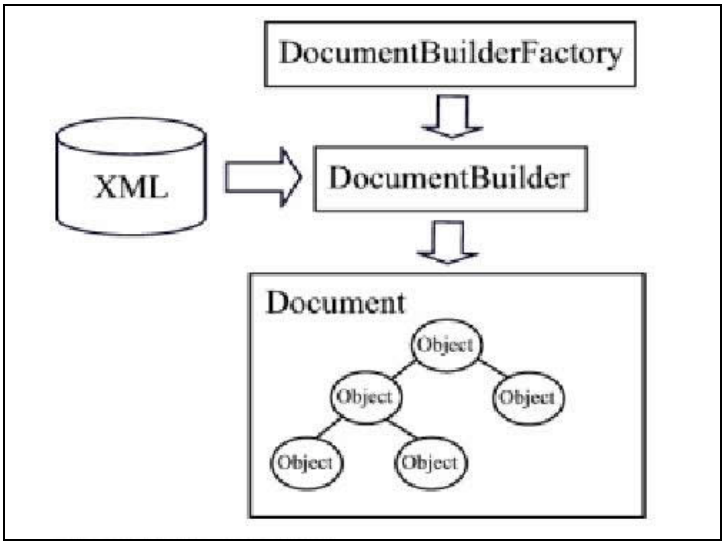


Figura 4 - Análise de um documento XML com parser DOM [JAX04]

O programador configura um *parser* com um XML fonte, e espera terminar. Se existir algum erro, ele recebe um relatório do erro, caso contrário é retornado um objeto DOM que pode ser manipulado. De uma maneira geral, para cada procura ou algum tipo de manipulação, é preciso começar pelo elemento raiz e ir subindo na hierarquia (Figura 4).

Uma vantagem do DOM para manipular documentos XML é que ele tem acesso aleatório, ou seja, o nó pode ser criado ou anexado a qualquer momento e em qualquer lugar da árvore XML.

2.5.6 SAX

O SAX (*Simple API for XML*) [SAX02], uma API baseada em eventos. É uma interface que permite escrever aplicações para ler documentos XML. Ela traz a vantagem

de se poder analisar um arquivo XML de qualquer tamanho, já que, contrária ao DOM [DOM04], ela não carrega o documento inteiro na memória.

Ela é uma interface baseada em eventos que são disparados sempre que é encontrado algum elemento no documento XML. Esses eventos são disparados durante a leitura do documento em diversos momentos: na abertura e no fechamento do documento, na abertura e no fechamento de cada elemento (*tag*), e quando cada caractere é lido pelo *parser*. O *parser* que implementa a SAX acessa de forma seqüencial o documento XML efetuando apenas a leitura do mesmo. Desta forma, o *parser* lê o documento XML sem a necessidade da criação de um modelo de objeto na forma de árvore, tornando-se, assim, mais rápido. A Figura 5 apresenta o funcionamento do *parser* que implementa o SAX.

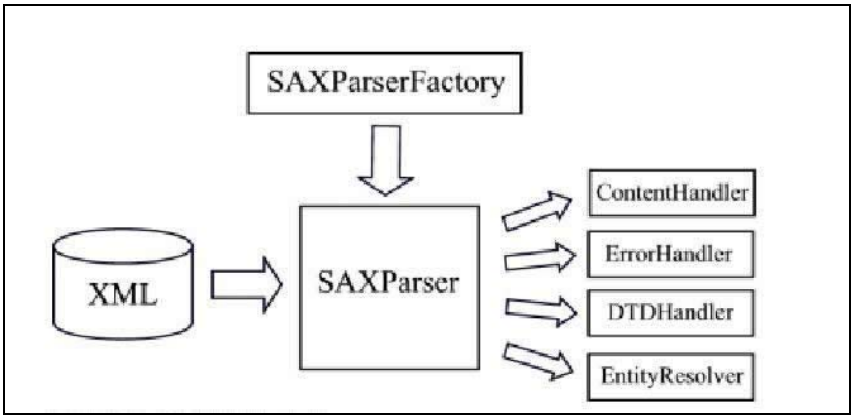


Figura 5 - Análise de um documento XML com parser SAX [JAX04]

Resumidamente, o programador configura um *parser* com uma fonte de entrada e o conecta a um conjunto de métodos de tratamento de eventos (*handler methods*). Quando o *parser* é executado, ele dispara eventos que são capturados pelos *handler methods*. Cada vez que o *parser* detecta uma parte importante do documento XML, ele dispara o *handler* correspondente. Um erro pode acontecer em qualquer momento durante a leitura. Neste

caso, o programador receberá informações sobre os elementos do documento até a detecção do erro.

Ao contrário das APIs baseadas em árvores, as baseadas em eventos geralmente não constroem representações dos documentos XML em memória. A SAX se mostra muito útil quando se quer criar sua própria estrutura de dados e também quando se quer apenas um pequeno subconjunto de informações. A SAX não tem métodos que permitem a alteração do documento gerado. E ela o acessa apenas em modo de leitura.

2.5 ASP

A tecnologia ASP (*Active Server Pages*) [MAC00] é um recurso que permite o processamento de comandos no servidor, com a conseqüente geração dinâmica de páginas HTML para o cliente. Dentre as linguagens que utilizam esta tecnologia estão PHP [NIE01], JSP [NIE01] e ASP (linguagem da Microsoft que herdou o mesmo nome da tecnologia). A partir de agora quando for citado ASP neste trabalho estará se referindo à linguagem.

As páginas em ASP consistem em arquivos de extensão .asp que contêm combinações de scripts mantidos no servidor (*server-side scripts*) e *tags* HTML. Todo o código de programação existente em páginas ASP é executado no servidor, e este retorna ao cliente somente respostas em HTML padrão - o que faz com que aplicações ASP possam ser acessadas por qualquer *browser* existente no mercado. Essas páginas devem estar hospedadas num servidor com suporte à linguagem ASP, como por exemplo o *Microsoft Internet Information Services* (IIS).

O *Internet Information Services* é um servidor de páginas de Internet padrão do Microsoft Windows. Ele possui grande performance quando utilizado junto ao ASP, além de fácil administração e segurança. O IIS já vem com suporte à ASP integrado, mas pode facilmente integrar outras tecnologias de processamento no servidor.

Neste trabalho optou-se pela utilização de ASP ao invés de outras linguagens já que a funcionalidade desenvolvida neste trabalho integrará um módulo Web previamente desenvolvido em ASP.

3 Sistema de Coleta Dados de Energia (SCDE)

O objetivo deste capítulo é contextualizar o leitor em relação ao novo modelo do setor elétrico, apresentando de forma genérica o funcionamento dos sistema de medição de dados e o novo processo de contabilização de energia controlado pelo MAE.

3.1 O Setor Elétrico Brasileiro

Para facilitar o entendimento do projeto segue abaixo uma breve descrição do atual sistema de energia elétrica segundo MAE [MAE04].

Depois de uma série de acontecimentos: o esgotamento da capacidade de geração de energia elétrica das hidrelétricas existentes, o aquecimento da economia provocado pelo Plano Real, a necessidade de novos investimentos e a escassez de recursos do Governo para atender a esta necessidade, o Governo viu-se obrigado a encontrar alternativas que viabilizassem uma reforma e expansão do setor de energia.

Através do Projeto RE-SEB (Projeto de Reestruturação do Setor Elétrico Brasileiro), iniciou-se a fase de concepção do novo modelo, sob a coordenação da Secretaria Nacional de Energia do Ministério de Minas e Energia, chegando-se à conclusão de que era preciso criar uma Agência Reguladora (ANEEL - Agência Nacional de Energia Elétrica), um operador para o sistema (ONS – Operador Nacional do Sistema Elétrico) e um ambiente (MAE – Mercado Atacadista de Energia Elétrica), através de uma operadora (ASMAE - Administradora de Serviços do Mercado Atacadista de Energia Elétrica), onde fossem transacionadas as compras e vendas de energia elétrica.

Dentre outras reformulações, coube ao MAE a implantação de um Sistema de Coleta de Dados de Medição de Energia, o SCDE, ficando os Agentes de Mercado com a

responsabilidade de adquirir e instalar os sistemas de medição, incluindo os transformadores de instrumentos e os próprios medidores. É atribuição, ainda, destes Agentes, através de processos automáticos, disponibilizarem para o MAE os dados de medição.

Apesar deste modelo ser recente, o Ministério das Minas e Energia já possui novas reformulações que deverão entrar em vigor ainda este ano. Novas instituições como a Empresa de Pesquisa Energética (EPE), o Comitê de Monitoramento do Setor Elétrico (CMSE) e a Câmara de Comercialização de Energia Elétrica (CCSE), que exercerá as atuais funções do MAE, serão criadas. No entanto, ainda existe pouca documentação sobre este Novo Modelo e preferiu-se neste trabalho adotar as definições do modelo atual.

É bom ressaltar que as especificações do modelo do setor elétrico não se restringem apenas às citadas neste trabalho. Procurou-se apresentar apenas os pontos que se enquadram no escopo deste projeto.

3.2 Sistema de Medição de Energia Elétrica

Para atender a contabilização do Mercado Atacadista de Energia Elétrica, os Agentes de Mercados registram os montantes de energia consumido, gerado ou transmitido. Estes utilizam-se de equipamentos que possibilitam a medição de diferentes variáveis elétricas. Os medidores de energia são instalados em diferentes pontos da rede elétrica, dependendo da classificação na qual o Agente se enquadra (usina, consumidores livres, distribuidor...) e da estrutura física que possui.

Além de atender a requisitos de certificação, características elétricas e classes de exatidão, o MAE especifica que os medidores de energia devem permitir a medição e o registro de pelo menos três grandezas elétricas (Energia Ativa[2] , Energia Reativa[3] e adicionalmente uma saída para as medições instantâneas (Potência Ativa e Reativa[4], Corrente e Tensão). Outra exigência é que os medidores devem possuir memória de massa, ou seja, capacidade de armazenar os dados de energia em intervalos de 5 minutos durante o período mínimo de 32 dias. O medidor também deverá coletar informações para análise de qualidade de energia (QEE).

Normalmente os Agentes utilizam-se de uma central de aquisição de dados que efetua a comunicação remota direta com os medidores, possibilitando o processamento da leitura dos valores registrados e da memória de massa. Isto permite que tanto o MAE como o Agente tenham acesso aos dados em tempo real. Os dados são armazenados em um banco de dados de grande porte (Figura 6).

---

[2] Energia Ativa: energia elétrica que pode ser convertida em outra forma de energia, expressa em quilowatts-hora (kWh).
[3] Energia Reativa: energia elétrica que circula continuamente entre os diversos campos elétricos e magnéticos de um sistema de corrente alternada, sem produzir trabalho, expressa em quilovolt-ampère-reativo-hora (kVArh).
[4] Potência ativa e reativa: quantidade de energia elétrica solicitada na unidade de tempo, expressa em quilowatts (kW).

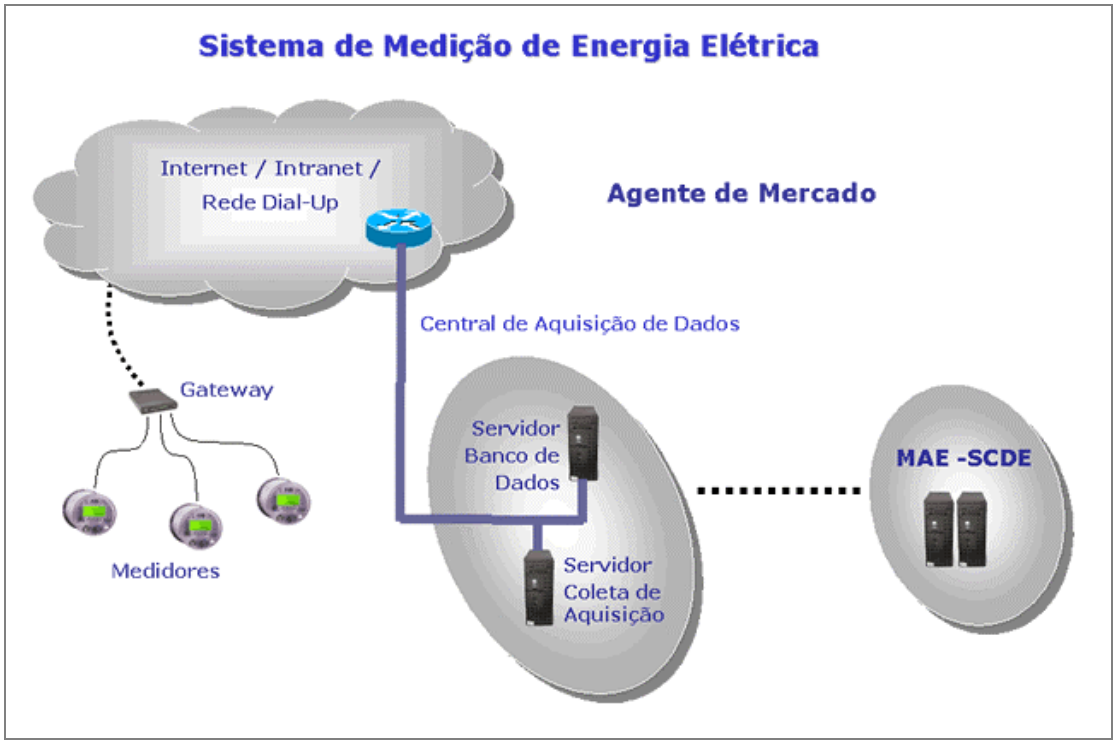


Figura 6 Arquitetura do Sistema de Medição de Energia.

3.3 Agendamento de Aquisição de Leituras

Depois dos dados de medição estarem devidamente armazenados na base de dados, estes já estão disponíveis para serem transformados em arquivos XML e transferidos ao MAE.

Para tanto, cada Agente de Mercado possui um arquivo de configuração fornecido pelo MAE, através do Client SCDE. A aplicação responsável pela geração dos arquivos de coleta[5] deverá consultar o arquivo de configuração para verificar a data, hora e qual tipo de arquivo de coleta que deverá ser disponibilizado.

A figura 7 apresenta um exemplo de arquivo de configuração. Neste arquivo o MAE está solicitando que sejam gerados diariamente arquivos de coleta do tipo diário, para dois medidores (SC-SETOR-A--01 e SC-SETOR-B--1C) com as informações de energia (ME), e que o Cliente SCDE estará apto à transmitir estes arquivos no intervalo de 08:00 às 18:00.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <XML>
  - <Medidores>
      <Medidor nome="SC-SETOR-A--01" />
      <Medidor nome="SC-SETOR-B--1C" />
    </Medidores>
  - <Agendamento tipo="ME">
    - <Periodicidade tipo="diaria" id="111" intervalo="1">
        <Hora inicio="08:00:00" fim="18:00:00" />
      </Periodicidade>
    </Agendamento>
    <Agendamento tipo="QEE" />
    <Agendamento tipo="ME_QEE" />
  </XML>
```

Figura 7 - Exemplo arquivo de configuração

[5] Arquivo de coleta: arquivo no formato XML que armazena os dados de medição de energia conforme padrão MAE.

Na Figura 7, o elemento Medidor possui o número de identificação do medidor. Este número, conhecido como número MAE, permite que o Agente identifique a qual medidor o MAE está se referindo. O elemento Agendamento informa quais tipos de dados que o arquivo de coleta deverá conter. O tipo ME refere-se a informações de energia (energia ativa, energia reativa, tensão e corrente). O tipo QEE solicita dados de qualidade de energia. Já para o tipo ME_QEE deverão ser gerado dois arquivos: um de energia, outro de qualidade.

O elemento Periodicidade informa qual a periodicidade que deverá ser efetuada coleta. Para Periodicidade do tipo diária deverá ser gerado um arquivo de coleta que possua informações de medição de um dia. Este elemento pode ainda possuir os tipos: horário, mensal, semanal e único. Já o elemento Hora especifica o período no qual o Client SCDE fará a transmissão dos arquivos de coleta.

3.4 Geração de Arquivos de Coleta

Os arquivos de coleta são gerados por uma aplicação de responsabilidade do Agente. Esta aplicação, que será desenvolvida neste trabalho, efetua a leitura do arquivo de configuração e gera arquivos do tipo XML, com formato definido pelo MAE.

O aplicativo deve consultar periodicamente o arquivo de configuração para efetuar o agendamento da geração dos arquivos de coleta. O arquivo de configuração, através do elemento Agendamento, informa qual o tipo de coleta que deverá conter o arquivo. Como apresentado nas Figuras 8 e 9, abaixo, os arquivos de coleta do tipo ME e QEE devem possuir determinadas *tags* XML.

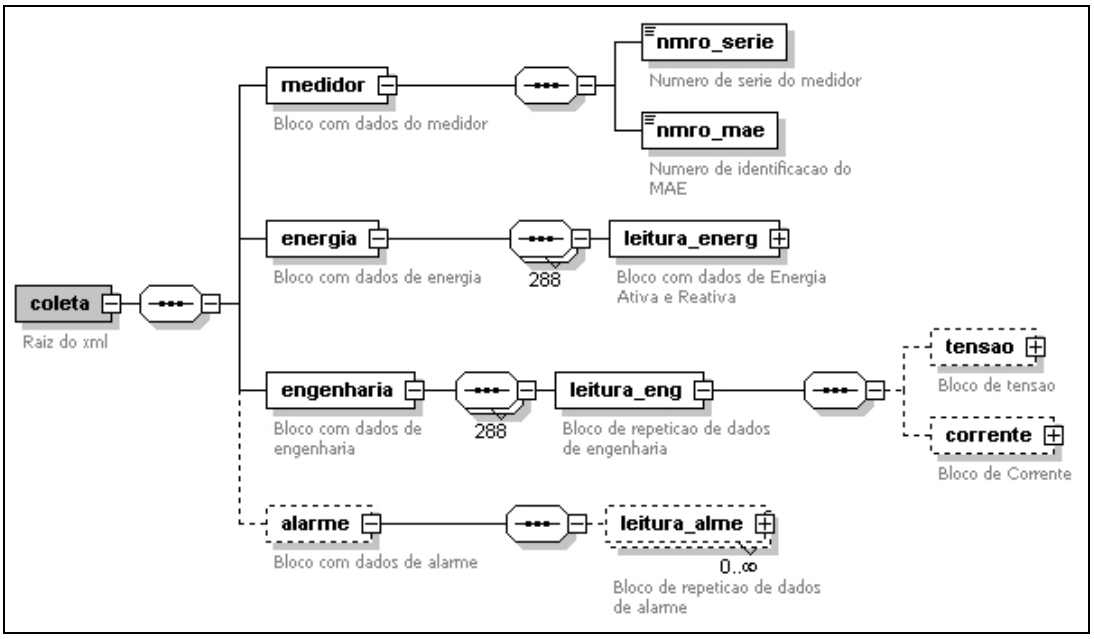


Figura 8 - Descrição dos *tags* XML para arquivos do tipo ME (energia) [RGM04]

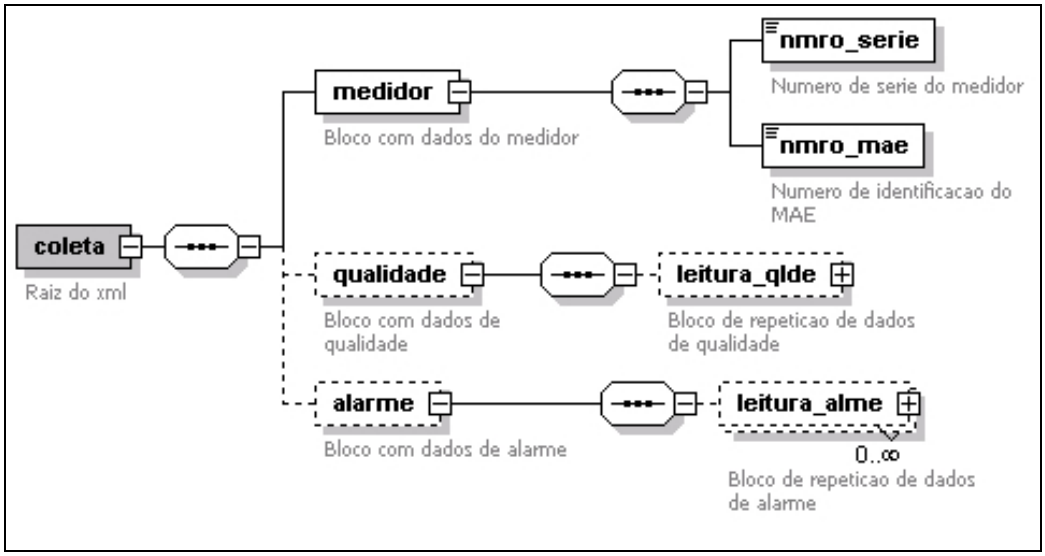


Figura 9 - Descrição dos *tags* XML parar arquivos do tipo QEE (qualidade) [RGM04]

Cada *tag*, com exceção de *coleta*, corresponde a um grupo de informações de medição com sua respectiva data e hora de coleta. A *tag* medidor possui dois elementos que indicam o número de série e o número MAE do medidor. Na *tag* energia, por exemplo,

encontram-se as medições de energia ativa e energia reativa com a data e hora da medição. Veja o exemplo de arquivo de coleta (tipo ME) do medidor SC-SETOR-B--1C na Figura 10.

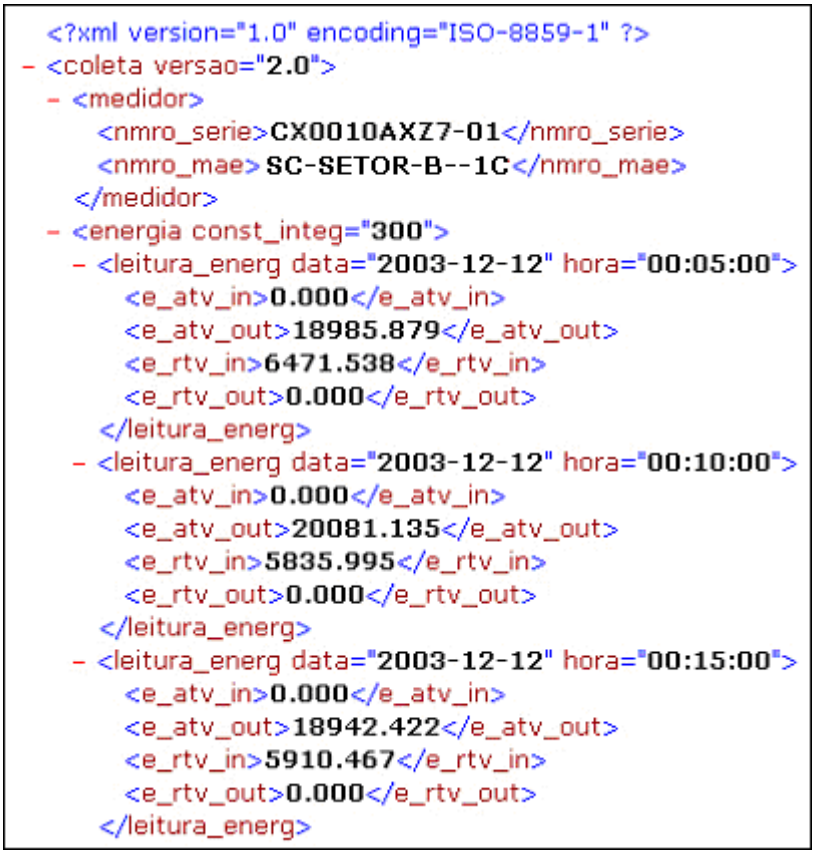


Figura 10 - Exemplo de arquivo de coleta XML do tipo ME.

Depois de definida a estrutura do arquivo, este será nomeado seguindo a uma nomenclatura: "SCDE_IdentificadorMedidor_TipoColeta_DataHoraColeta.xml", onde *IdentificadorMedidor* corresponde ao número MAE de identificação do medidor, o *TipoColeta* define qual o tipo de coleta (ME ou QEE) e *DataHoraColeta* informa a data e hora que o arquivo de coleta foi gerado.

3.5 Funcionamento do Client SCDE

O Client SCDE realizará a leitura das datas e horários de transmissão agendados no arquivo de configuração, comparando-os com o horário atual do sistema. No horário estabelecido para transmissão, será iniciada a conexão ao SCDE. A partir desse ponto, o Client SCDE estará pronto para transmitir (o)s arquivo(s) de coleta XML gerado pelo Agente. Cada arquivo enviado do Cliente para o SCDE só será considerado transmitido com sucesso após a validação do formato XML.

4 Aplicação MAE – Análise e Projeto

Este capítulo objetiva mostrar o resultado da etapa de análise e projeto [LAR00].

Para facilitar o entendimento, a Aplicação MAE foi dividida em duas: Aplicação MAE Desktop (APD) e Aplicação MAE Web (APW). Estes constituirão um módulo do AES (*Airgate Energy Suíte*) (figura 12),um sistema de medição responsável pela aquisição e disponibilizarão de dados de energia para os Agentes do Mercado.

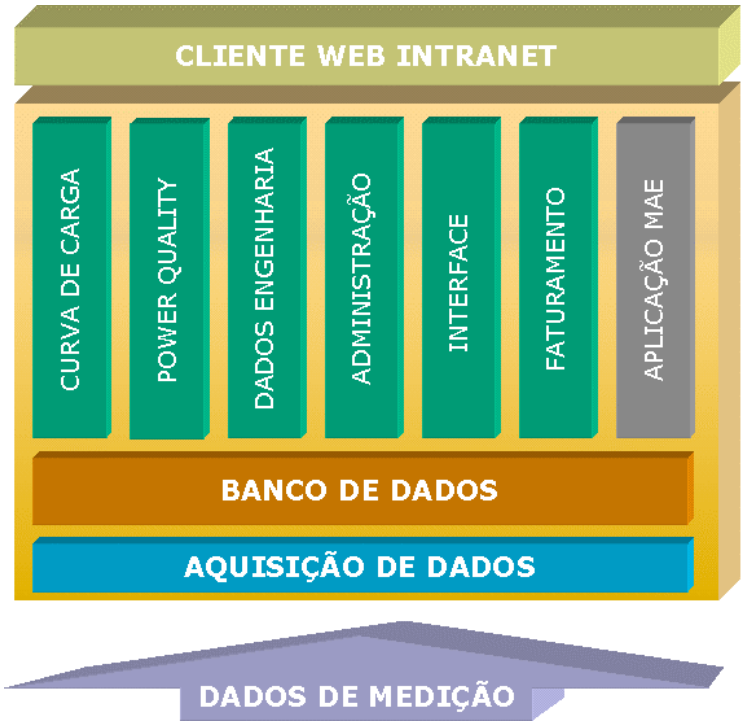


Figura 11 - Módulos do AES

4.1 Aplicação MAE Desktop

A Aplicação MAE Desktop é representada por um programa desenvolvido em JAVA que efetua a geração de arquivos de coleta no formato XML. Os casos de uso abordados serão: agendamento de coleta e geração de arquivos de coleta.

4.2.1 Agendamento de Coletas

O agendamento de coletas consiste na leitura do arquivo de configuração fornecido pelo MAE de modo a agendar a geração dos arquivos que serão coletados posteriormente. A tabela 1 descreve este caso de uso da aplicação MAE.

Caso de Uso

Caso de uso:	**Agendamento de Coletas**
Atores:	Gerente
Tipo:	Primário
Finalidade:	Fazer a leitura do arquivo de configuração e efetuar o agendamento para geração dos arquivos de coleta.
Visão Geral:	Depois de iniciado a APD é feita a leitura do arquivo de configuração, definindo a agenda para geração dos arquivos de coleta.

Tabela 1 - Caso de uso Agendamento de Coletas

Diagrama de Classes

O diagrama de classes da Figura 12 mostra as classes utilizadas para agendamento de coletas. A classe *Gerente* é responsável por dar inicio ao agendamento de coletas e às principais funcionalidades do APD. Já o *SaxHandler* representa o *parser* que fará a leitura

do arquivo de configuração. A classe *Agenda* armazena os dados obtidos pelo *parser*. A classe abstrata *Evento* possui as subclasses *EventoDiario*, *EventoHorário*, *EventoUnico*, *EventoSemana* e *EventoMensal*, que são usadas para modelar as coletas agendadas na agenda de coleta (classe Agenda). A classe *Medidor* corresponde a representação das *tags* que identificam o medidor no arquivo de configuração.

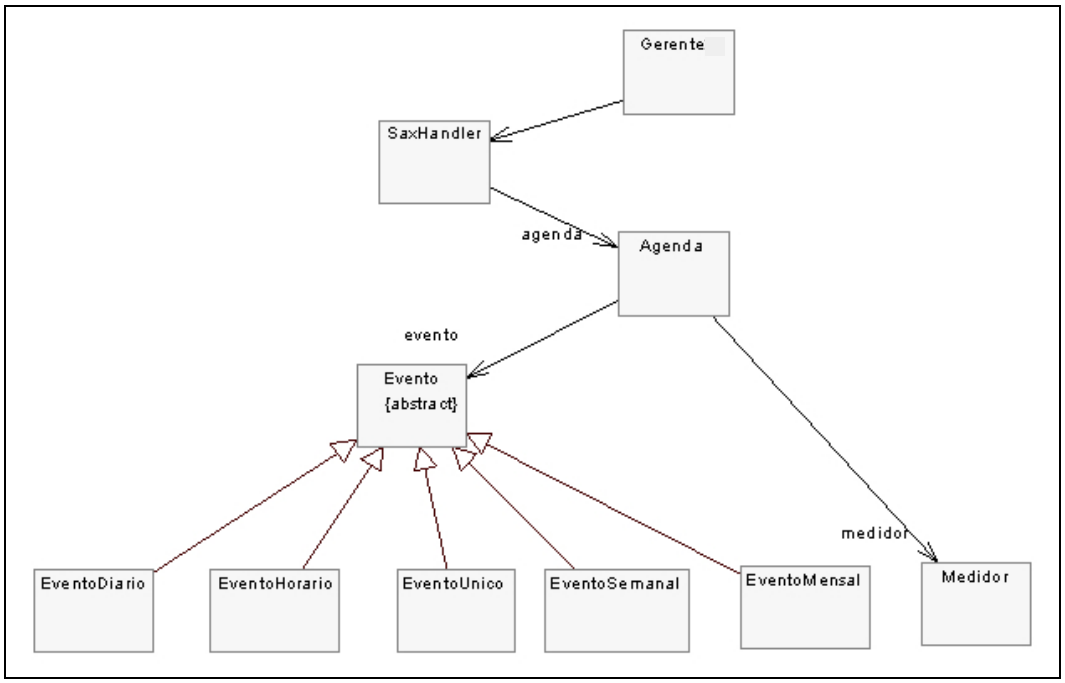


Figura 12 - Diagrama de classes

Diagrama de Seqüência

O diagrama de seqüência apresentado na figura 13 possui objetos representados pelas classes: *Gerente*, *SaxHandler*, *Agenda*, *Medidor*, *EventoTP*.

O agendamento de coleta ocorre sempre quando a aplicação é iniciada ou quando se verifica alguma modificação no arquivo de configuração. A operação começa com o *Gerente* criando uma nova instância da classe *SaxHandler*. E esta fará a leitura sobre o arquivo de configuração e retornará um objeto da classe *Agenda*. Este objeto criará

instâncias das classes *Medidor* e *Evento* armazenando-as em coleções de objetos (*cl_e* – coleção de eventos e *cl_m* – coleção de medidores), que mais tarde serão retornadas ao *Gerente,* finalizando o processo.

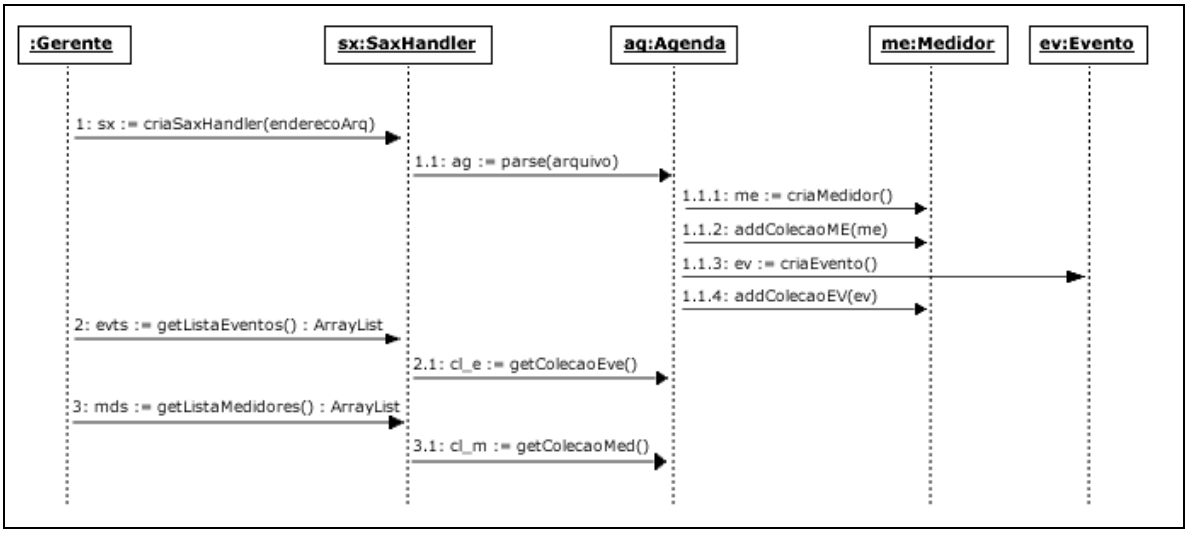


Figura 13 - Diagrama de seqüência

4.2.2 Geração de Arquivos de Coleta

A geração dos arquivos de coleta, descrita no caso de uso especificado na tabela 2, ocorre no momento agendado previamente, disponibilizando para coleta os arquivos solicitados pelo MAE.

Caso de Uso

Caso de uso:	**Geração de Arquivos de Coleta**
Atores:	Gerente
Tipo:	Primário e Real
Finalidade:	Gerar os arquivos de coleta no formato XML conforme especificações do MAE.
Visão Geral:	Depois de definido a agenda de coletas, periodicamente é feito uma verificação da lista de eventos buscando eventos de coleta. Caso exista algum evento, é gerado o arquivo de coleta correspondente.

Tabela 2 - Caso de uso Agendamento de Coletas

Diagrama de Classes

O diagrama de classes da figura 14 mostra os objetos envolvidos na geração dos arquivos de coleta e os relacionamentos entre estes.

O Gerenciador de Eventos (classe *GerenciadorEventos*) é responsável por manter o registro dos eventos de coleta. A classe *TimerHandler*, que consiste em uma extensão da classe *Timer* [SUN03] que permite executar métodos de forma temporal, é responsável pela busca por coletas programadas para ocorrer.

O Gerenciador de Coleta (classe GerenciadorColeta) é responsável por efetuar as operações de coleta, utilizando um coletor para o banco de dados utilizado. No nosso caso, foi utilizado o banco de dados Oracle e foi criado um coletor específico para acesso a este SGBD, chamado *ColetorOracle*.

O gerenciador de coleta gera os dados de coleta em XML usando a classe *GeradorXML*. Estes dados são validados usando a classe *SAXValidacao* e depois salvos em um arquivo, representado pela classe *Arquivo*. Este arquivo é armazenado no disco para que seja coletado posteriormente.

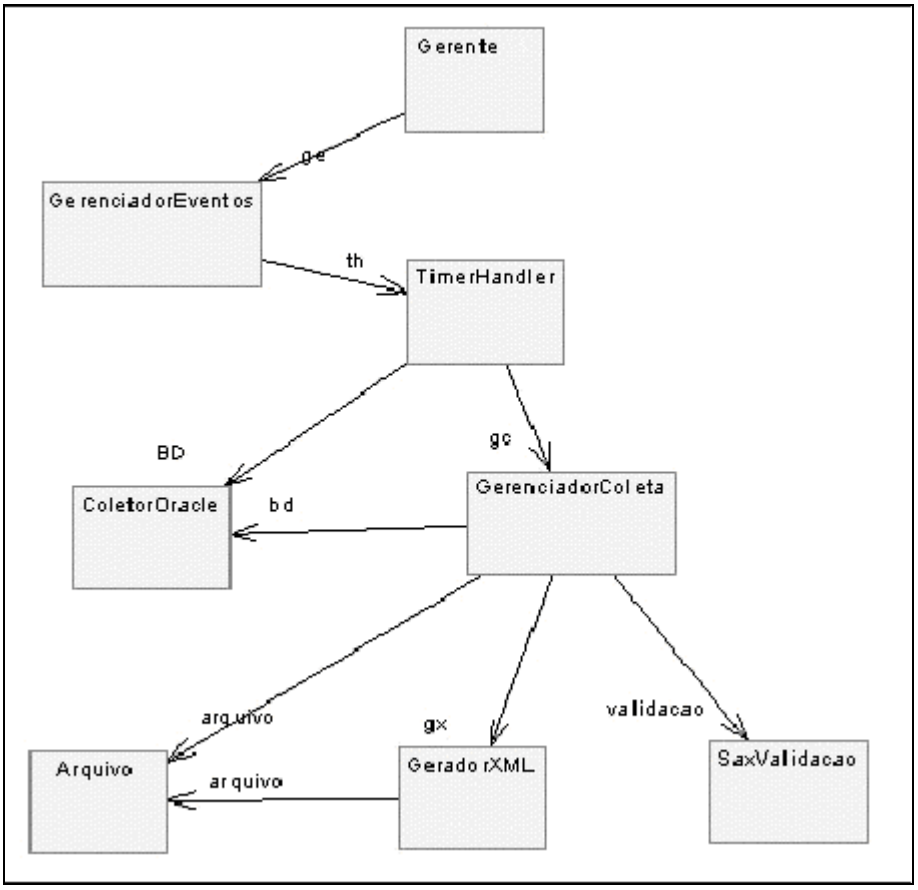


Figura 14 - Diagrama de classes

Diagrama de Seqüência

Os diagramas de seqüência mostrados nas figuras 15, 16 e 17 exibem as trocas de mensagens entre os objetos para efetuar a geração dos arquivos de coleta. Tendo em vista a complexidade deste processo, optou-se por apresentá-la dividida em três etapas.

A primeira etapa (figura 15) consiste em preparar a função *timer* para permitir à APD efetuar a verificação periódica de eventos. As seguintes classes estão envolvidas: *Gerente, GerenciadorEventos, TimerHandler* e *DescricaoColeta*. Este etapa é iniciada assim que a Aplicação MAE Desktop é executada. O *Gerente* cria uma instância da classe *GerenciadorEventos* passando como parâmetro as coleções de *Evento* e *Medidor*

(*listaEventos* e *listaMedidores*) criado no processo de Agendamento de Coletas (descrito no item anterior). Depois disso é criado uma instância da classe *TimerHandler,* que verifica se existem eventos programados para ocorrer. Em caso positivo, é criado um objeto da classe *DescricaoColeta* que será armazenado numa lista de coletas a serem feitas (*listaColetas*).

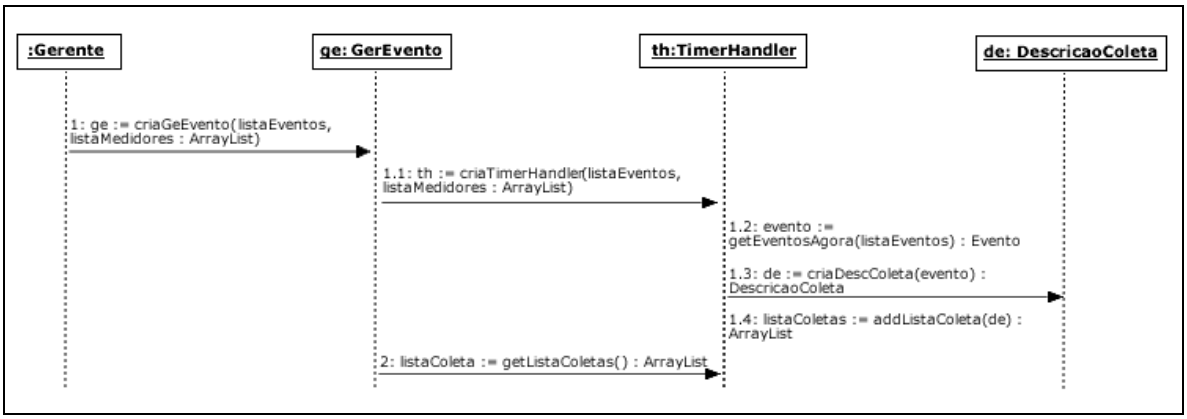


Figura 15 - Diagrama de seqüência – Gerenciador de Eventos (1/3)

A segunda etapa (figura 16) compreende a coleta dos dados e a criação do arquivo de coleta. Se existir alguma coleta programada na lista de coletas (*listaColetas)* o *GerenciadorEvento* cria uma instância da classe *GerenciadorColeta*. Depois de iniciado, este objeto efetua coleta dos dados de medição presentes na base de dados. Para isso ele faz chamada à instância *bd* da classe *ColetorOracle,* que faz a interface com o Banco de dados.

Com os dados de medição em mão o próximo passo é montar a *string* que formará o arquivo de coleta. Isso é feito criando uma instância da classe *GeradorXML* e enviando como parâmetro os dados coletados. O objeto montará a *string* do arquivo de coleta, conforme estrutura definida pelo MAE, retornando-a em seguida para o *GerenciadorColeta.*

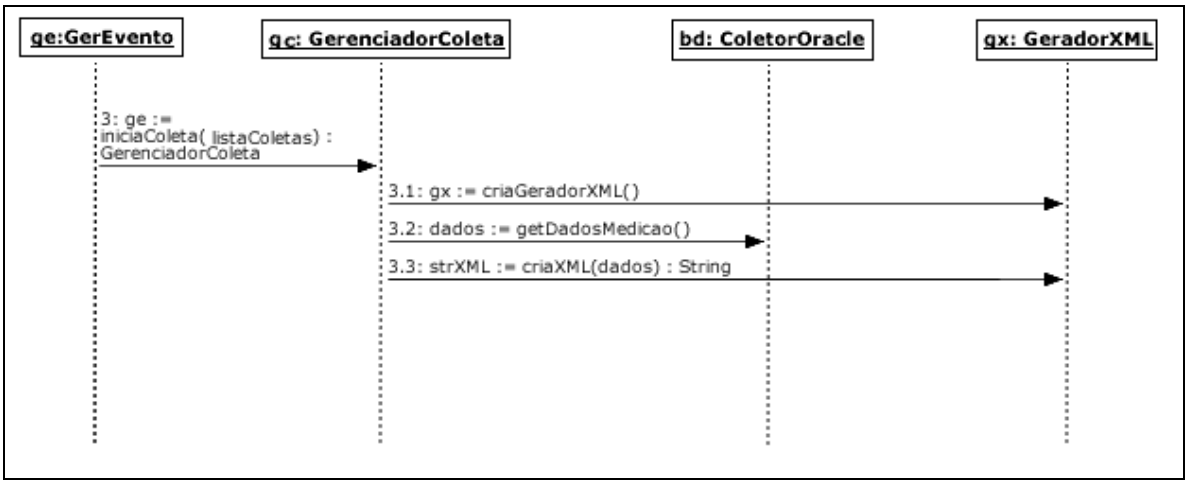


Figura 16 – Diagrama de seqüência - Geração de Arquivos de Coleta (2/3)

Por fim, a última etapa (figura 17) consiste na validação do *string* do arquivo de coleta gerado na etapa anterior. A validação é feita a partir de um arquivo XML Schema fornecido pelo MAE. A classe responsável pela validação é a *SaxValidacao*. Desta forma o *GerenciadorColeta* inicia uma instância desta classe e verifica a integridade da *string* do arquivo de coleta. Caso a *string* seja válida, o próximo passo é a geração do arquivo de coleta. Este é gerado por uma instância (*arq*) da classe *Arquivo.* Ela recebe como parâmetro a *string* que contem o conteúdo do arquivo de coleta, e cria o arquivo de coleta em disco.

Por último, através da instância da classe *ColetorOracle*, é guardado um *log* do processo de geração, armazenando as informações do arquivo de coleta.

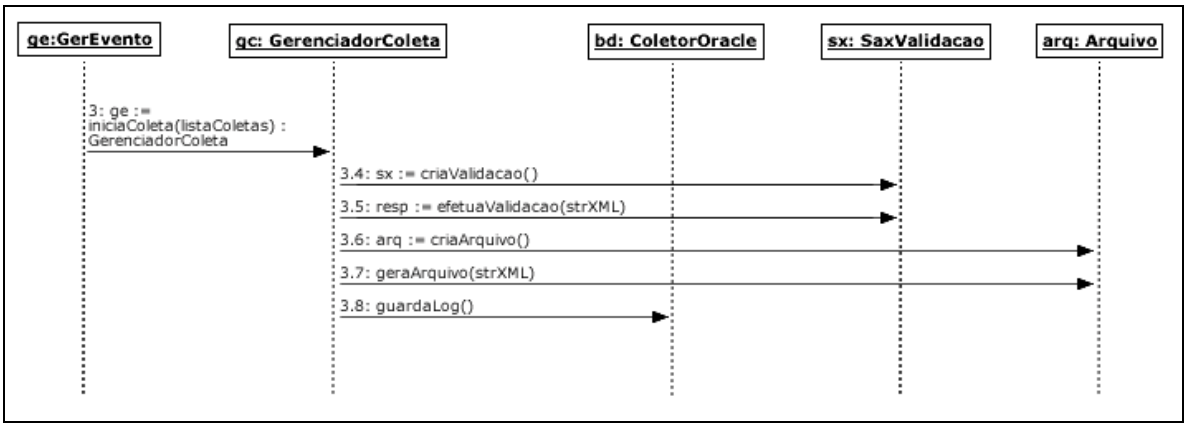


Figura 17 – Diagrama de seqüência – Validação dos Arquivos de Coleta (3/3)

É importante observar que alguns relacionamentos mostrados no diagrama de classes (Figura 14) foram ocultados nos diagramas de seqüência para simplificá-los, como o caso da classe *TimerHandler* que faz acesso ao banco de dados (através da classe *ColetorOracle*), durante a criação dos objetos da classe *DescricaoColeta,* verificando se estas coletas já foram efetuadas. Outro exemplo é a classe *GeradorXML*, que apesar de não utilizar a classe *Arquivo* durante a geração do arquivo de coleta, possui um relacionamento com a classe *Arquivo* utilizado para execução de outra tarefa.

4.2 Aplicação MAE Web

A Aplicação MAE Web constitui de uma interface desenvolvida em ASP, publicada em um servidor rodando IIS (*Internet Information Service*), através do qual o usuário do sistema fará o monitoramento da aplicação. O caso de uso abordado neste item será a geração de arquivos de coleta no módulo web.

Caso de Uso

Caso de uso:	**Geração de Arquivos de Coleta**
Atores:	Usuário, Aplicação MAE Web
Tipo:	Primário e Real
Finalidade:	Gerar os arquivos de coleta no formato XML conforme especificações do MAE.
Visão Geral:	Depois de acessar a função Gera Retroativo o usuário seleciona o ponto ou pasta o qual deseja gerar o(s) arquivo(s) de coleta, preenche o campo tipo de coleta (ME ou QEE) e define o período que deseja gerar os arquivos. Por fim basta clicar em "gerar" que os arquivos serão salvos na pasta especificada.

Tabela 3 - Caso de uso Geração de Arquivos de Coleta via APW

Os diagrama de seqüência e classes deste caso de uso são semelhantes àqueles mostrados no item 4.2.1. A diferença se encontra apenas no fato das classes estarem armazenadas no banco de dados, na forma de *Java Stored Proecedures*.

4.3 Banco de Dados

O banco de dados contém as informações de todo sistema de medição, incluindo registros de medição de energia, configurações, *Stored procedures,* usuários do sistema, etc.

4.4.1 Dados de Medição de Energia

Neste item serão apresentadas as tabelas que armazenam os registros utilizados para geração dos arquivos de coleta. Estes estão organizados nas seguintes tabelas: **tb_coleta_ene** (Figura 18), **tb_coleta_eng** (Figura 19), **tb_coleta_qua** (Figura 20) e **tb_coleta_eve** (Figura 21):

- **tb_coleta_ene**: armazena os dados de energia (*timestamp*, Energia Ativa Distribuída, Energia Ativa Recebida, Energia Reativa Distribuída, Energia Reativa Recebida);

Columns of TB_COLETA_ENE

Name	Type	Nullable	Default	Comments
CD_PONTO	NUMBER(10)			
DT_HR_INSTANTE	DATE			
VL_EN_ATIVA_DEL	FLOAT	Y		
VL_EN_ATIVA_REC	FLOAT	Y		
VL_EN_REATIVA_DEL	FLOAT	Y		
VL_EN_REATIVA_REC	FLOAT	Y		
VL_DEMANDA	FLOAT	Y		

Figura 18 - Tabela com os dados de energia

- **tb_coleta_eng**: armazena os dados de engenharia (*timestamp*, Tensão fase A, Tensão fase B, Tensão fase C, Corrente na fase A, Corrente na fase B, Corrente na fase C);

Columns of TB_COLETA_ENG

Name	Type	Nullable	Default	Comments
CD_PONTO	NUMBER(10)			
DT_HR_INSTANTE	DATE			
VL_TENSAO_AB	FLOAT	Y		
VL_TENSAO_BC	FLOAT	Y		
VL_TENSAO_CA	FLOAT	Y		
VL_TENSAO_AVG	FLOAT	Y		
VL_TENSAO_DES	FLOAT	Y		
VL_CORRENTE_A	FLOAT	Y		
VL_CORRENTE_B	FLOAT	Y		
VL_CORRENTE_C	FLOAT	Y		
VL_CORRENTE_AVG	FLOAT	Y		
VL_FREQUENCIA	FLOAT	Y		

Figura 19 - Tabela com os dados de engenharia

- **tb_coleta_qua**: armazena os dados de qualidade de energia (*timestamp*, Duração do Distúrbio, Distúrbio Nominal, Valor máximo 1, Valor mínimo 1, Valor máximo 2, Valor mínimo 2, Valor máximo 3, Valor mínimo 3, Limite de *Sag*, Limite *Swell*);

Columns of TB_COLETA_QUA

Name	Type	Nullable	Default	Comments
CD_PONTO	NUMBER(10)			
DT_HR_INSTANTE	DATE			
VL_DIST_DURACAO	FLOAT	Y		
VL_DIST_NOMINAL	FLOAT	Y		
VL_MAX1	FLOAT	Y		
VL_MIN1	FLOAT	Y		
VL_AVG1	FLOAT	Y		
VL_MAX2	FLOAT	Y		
VL_MIN2	FLOAT	Y		
VL_AVG2	FLOAT	Y		
VL_MAX3	FLOAT	Y		
VL_MIN3	FLOAT	Y		
VL_AVG3	FLOAT	Y		
VL_SAG	FLOAT	Y		
VL_SWELL	FLOAT	Y		
NUM_MILISEGUNDO	FLOAT	Y		

Figura 20 - Tabela com dados de qualidade de energia

- **tb_coleta_eve**: armazena os dados de alarme (Prioridade do evento, Causa do evento, Valor da causa do Evento, Efeito do Evento, Valor do efeito do evento);

Columns of TB_COLETA_EVE

Name	Type	Nullable	Default	Comments
CD_PONTO	NUMBER(10)			
DT_HR_INSTANTE	DATE			
STR_CAUSA	VARCHAR2(100)	Y		
NUM_CAUSA	NUMBER(10)			
STR_EFEITO	VARCHAR2(100)			
NUM_EFEITO	NUMBER(10)			
NUM_PRIORIDADE	NUMBER(5)			
NUM_REPLACE	NUMBER(5)			
NUM_MILISEGUNDO	FLOAT	Y		

Figura 21 - Tabela com dados de alarme

4.4.2 Histórico de Geração

A Aplicação MAE armazena o histórico de geração de arquivos de coleta. A tb_mae, ilustrada na figura 22, guarda estes registros. Dentre os dados armazenadas estão as informações sobre o arquivo de coleta, como: nome do arquivo, endereço o qual foi salvo, período de coleta, a situação (gerado com falha ou não), etc.

Columns of TB_MAE

Name	Type	Nullable	Default	Comments
CD_MAE	NUMBER(10)			
STR_NOME_ARQ	VARCHAR2(100)			
STR_ENDERECO_ARQ	VARCHAR2(100)	Y		
DT_INICIO_COLETA	DATE			
DT_FIM_COLETA	DATE			
FL_SITUACAO	VARCHAR2(30)			
STR_DESC_ERRO	VARCHAR2(400)	Y		
STR_TIPO_COLETA	VARCHAR2(20)			
DT_HR_INSTANTE	DATE			
STR_ID_MAE	VARCHAR2(14)	Y		
STR_PERIODICIDADE	VARCHAR2(20)	Y		
CD_APLICACAO	NUMBER(10)	Y		
STR_ID_PERIODICIDADE	VARCHAR2(10)	Y		

Figura 22 - Tabela com histórico de geração de arquivos de coleta

4.4.3 Pacotes e *Stored procedures*

Grande parte do acesso a base de dados se da através de *Stored procedures*. Estas estão organizadas em pacotes, de acordo com sua função. Na figura 23, abaixo, são mostrados os pacotes utilizados no AES.

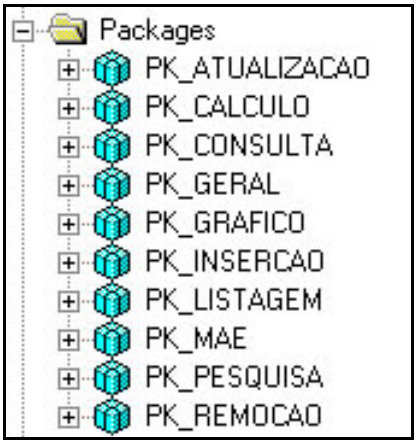


Figura 23 - Pacotes do banco de dados

O pacote MAE (PK_MAE) armazena um conjunto de *Stored Procedures* utilizado pela Aplicação MAE. A figura 24 apresenta o conteúdo deste pacote.

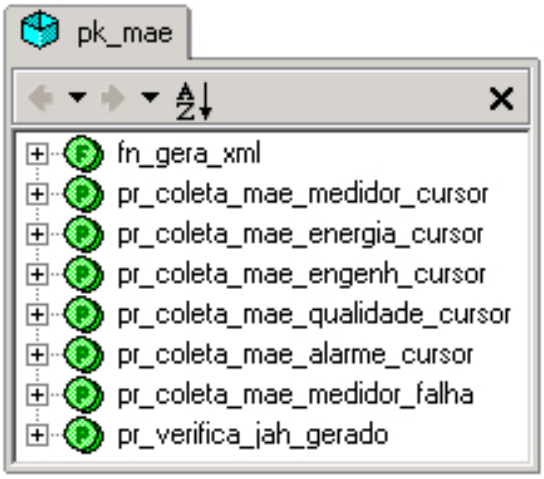


Figura 24 - Pacote com *Stored Procedures*

A primeira função (fn_gera_xml) é uma *Java Stored Procedure*, que faz chamada a uma classe Java carregada no banco de dados. Ela é responsável pela geração dos arquivos de coleta via Aplicação MAE Web. As demais *Stored Procedures* efetuam consultas sobre tabelas do banco de dados, buscando dados de medição para compor o arquivo de coleta.

5 Aplicação MAE – Implementação e Funcionamento

Para facilitar o seu entendimento, a Aplicação MAE pode ser dividida em três módulos: Aplicação MAE Desktop, Aplicação MAE Web e Banco de Dados. No módulo Banco de Dados, além de possuir o histórico da geração de arquivos de coleta, são armazenadas as *Stored procedures* que permitem a manipulação de informações tanto para Aplicação MAE Desktop como para Aplicação MAE Web (Figura 25).

Figura 25 Estrutura Aplicação MAE

5.1 Ferramentas Utilizadas

Na etapa de desenvolvimento foram utilizadas as seguintes ferramentas:

- **Java 2 Platform Standard Edition** (J2SE – versão 1.4.2): compilador, máquina virtual e APIs Java;
- **Java 2 Platform Enterprise Edition** (J2EE – versão 1.4): serviços e APIs Java adicionais;
- **JCreator Pro 2.0**: ambiente de desenvolvimento para programação em Java;
- **EditPlus Text Editor v2.10c**: ambiente para edição de texto, utilizado para programação em ASP e HTML.
- **XMLSPY 2004 Professional Edition**: ambiente para criação e edição de arquivos XML;
- **PL/SQL Developer** (Versão 4.0): ambiente para manipulação do banco de dados;
- **PowerDesigner** (versão 9.5.1): ferramenta utilizada para modelagem do sistema, criação de diagramas e casos de uso.

5.2 Aplicação MAE Desktop

A Aplicação MAE Desktop (Figura 26) é representada por um software localizado na barra de ferramentas do Windows (*system tray*), que é responsável pela geração dos arquivos de coleta.

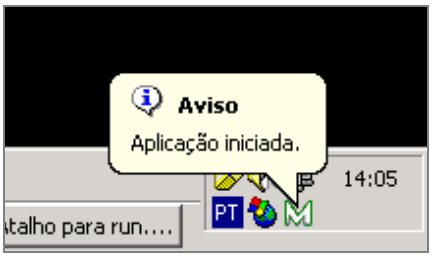


Figura 26 Iniciando Aplicação

Para acessar as funcionalidades da Aplicação basta clicar com o botão direito do mouse sobre o ícone do aplicativo e selecionar a opção desejada no menu, como mostrado na figura 27.

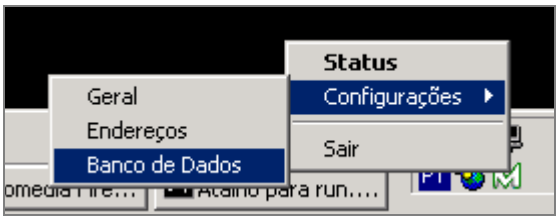


Figura 27 Configuração Banco de Dados

5.2.1 Geração de Arquivos de Coleta

Ao ser iniciada, a Aplicação MAE Desktop efetua a leitura do arquivo de configuração fornecido pelo Cliente SCDE e a partir deste define a agenda de coletas para o dia. Quando chega a hora programada, automaticamente, o programa gera o(s) arquivo(s) de coleta.

Caso ocorra algum problema na coleta é exibida uma mensagem de erro (figura 28). Novas tentativas serão feitas até o arquivo ser gerado com sucesso ou até expirar o prazo de coleta definido no arquivo de configuração.

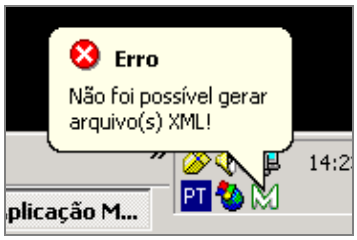


Figura 28 - Mensagem de erro

Durante o processo de geração do arquivo de coleta deve-se levar em conta algumas operações:

Configuração do Arquivo de Coleta

O arquivo de coleta é gerado conforme configurado na Aplicação MAE Web onde é possível definir a estrutura do arquivo.

Validação

Durante o processo de geração do arquivo é feita uma validação sobre seu conteúdo, verificando se está em conformidade com as especificações do MAE. Caso o processo de validação não encontre erro, o arquivo de coleta é gerado e armazenado na pasta configurada pelo usuário. Caso contrário, como explicado acima, novas tentativas serão feitas até o arquivo ser gerado com sucesso ou até expirar o prazo de coleta definido no arquivo de configuração.

Com o processo de validação evita-se que o Client SCDE rejeite o arquivo, garantido a transferência apenas dos arquivos de coleta completos. A tabela abaixo apresenta as mensagens de erro mais comuns com um breve comentário:

Mensagem de Erro	Comentário
cvc-complex-type.2.4.b: The content of element 'engenharia' is not complete. It must match '((""):leitura_eng)){288}'	Indica que as medições de engenharia no dia especificado não estão completas.
cvc-complex-type.2.4.b: The content of element 'energia' is not complete. It must match '((""):leitura_energ)){288}'	Indica que as medições de energia no dia especificado não estão completas.

5.2.2 Log da Aplicação

O Log da Aplicação constitui a principal interface de interação com o usuário na Aplicação MAE Desktop. Nela são exibidas as mensagens de erro bem como a situação dos arquivos de coleta gerados no dia. Para acessar a sessão Log da Aplicação basta clicar com o botão direito sobre o ícone da Aplicação MAE Desktop e selecionar a opção Log da Aplicação.

Ao iniciar a Aplicação MAE Desktop serão exibidas informações sobre a inicialização da aplicação (Figura 29). Em seguida é apresentado um breve resumo do arquivo de configuração (Figura 30). O exemplo informa que o MAE espera receber diariamente, no período de 08:00 a 12:00, o(s) arquivo(s) de coleta do(s) ponto(s) de medição cadastrado.

```
[Aplicação Iniciada]
>> Arquivo com configurações do sistema carregado.
>> Arquivo de configuração (MAE) carregado.
>> Gerenciador de eventos iniciado.
```

Figura 29 - Iniciando APD.

Figura 30 - Descrição do arquivo de configuração.

Depois de apresentada a descrição do arquivo de configuração, a aplicação buscará as coletas selecionadas para serem realizadas, conforme mostrado na figura 31. Em "Coletas Selecionadas" o número apresentado entre parêntese indica o número de arquivos de coleta que a Aplicação tentará gerar. Para cada tentativa serão apresentadas as informações referente à coleta.

De acordo com a figura, no primeiro bloco de informações, o arquivo de coleta do tipo "ME" (energia) com periodicidade "diária" e identificador MAE "SC-SETOR-A--01" foi gerado com sucesso. As datas de inicio e término representam o intervalo relativo aos dados de energia que foram coletados.

Já para o segundo ponto de medição, com número MAE "SC-SETOR-B--1C", houve falha durante a geração do arquivo de coleta. A última linha apresenta a mensagem "Nova(s) tentativa(s) serão feitas" significando que de acordo com o arquivo de configuração, a Aplicação MAE Desktop poderá fazer novas tentativas até expirar o período de transferência dos arquivos. Se esta mensagem não aparecer, a Aplicação MAE Desktop não fará novas tentativas, já que o prazo para o Client SCDE buscar o arquivo de coleta excedeu. Desta forma o usuário terá que gerar o arquivo de coleta manualmente, via Aplicação MAE Web.

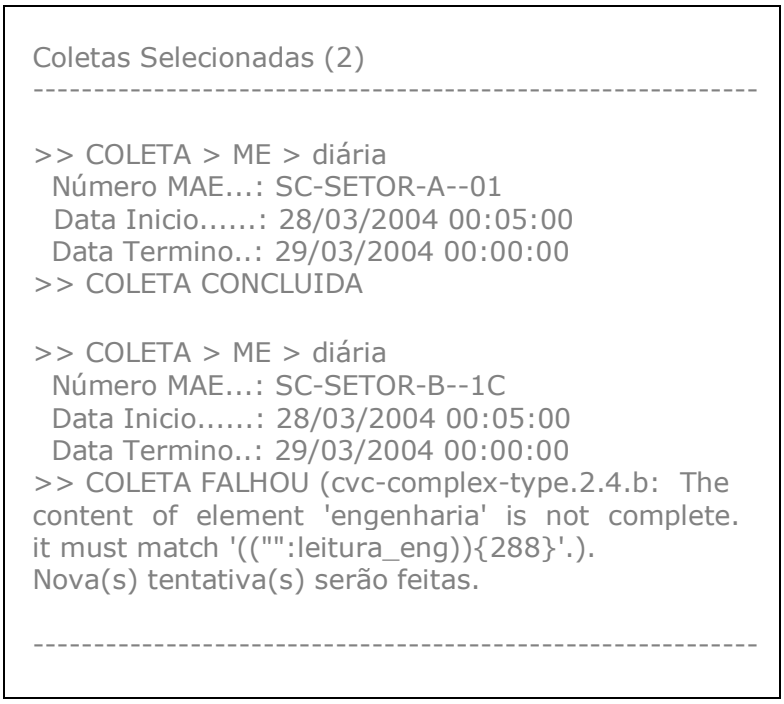


Figura 31 - Coletas selecionadas.

5.2.3 Configurações de Endereços

Para que a Aplicação MAE Desktop efetue suas funções é obrigatório que os endereços estejam corretamente configurados. Para acessar esta sessão basta clicar com o botão direito do mouse sobre o ícone da Aplicação MAE, passar o mouse sobre o item *Configurações* e depois selecionar *Endereços*.

Abaixo é descrito cada campo da janela de configuração (Figura 32):

Arquivo de Configuração

O arquivo de configuração compreende o arquivo "config.xml" fornecido pelo MAE, que normalmente fica localizado na pasta onde foi instalado o Cliente SCDE. As novas configurações só terão efeito da próxima vez que a Aplicação MAE Desktop for iniciada.

Pasta MAE Web

Este endereço aponta para a localização da pasta MAE na estrutura de diretórios da Aplicação MAE Web. Caso o endereço esteja incorreto, a sessão de visualização do arquivo de configuração na APW não funcionará.

Repositório XML

O repositório XML compreende a pasta onde serão armazenado os arquivos de coleta gerados pela Aplicação MAE Desktop. Caso este endereço esteja incorreto os arquivos de coleta não serão gerados. Normalmente este campo aponta para a pasta *Upload*, onde o Client SCDE transfere os arquivos.

Cópia do Repositório XML

Este campo especifica o endereço o qual serão armazenados as cópias dos arquivos de coleta gerados. Desta forma é possível fazer um backup dos arquivos XML.

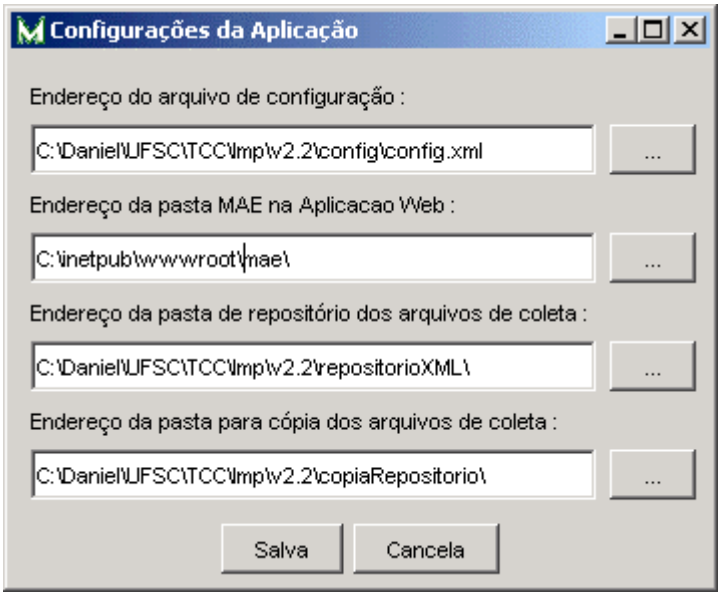


Figura 32 - Configurando endereços da aplicação.

5.2.4 Configurações do Banco de Dados

Para acessar esta sessão basta clicar com o botão direito do mouse sobre o ícone da Aplicação MAE, passar o mouse sobre o item Configurações e depois selecionar Banco de Dados. É essencial que as configurações do Banco de Dados estejam corretas. Os dados requeridos são Host, Port, Sid, Usuário e Senha (figura 33).

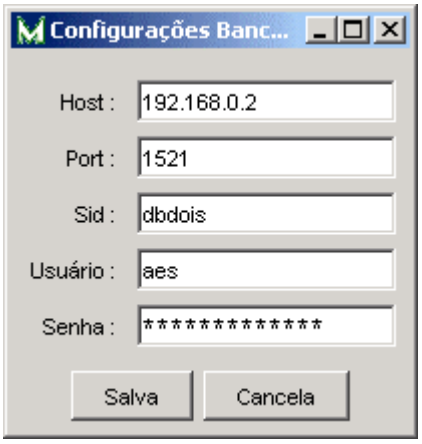


Figura 33 - Configurações do Banco de Dados

5.3 Aplicação MAE Web

A Aplicação MAE Web compreende o módulo MAE da página Web do Airgate Energy Suíte[6] - AES (Figura 34).

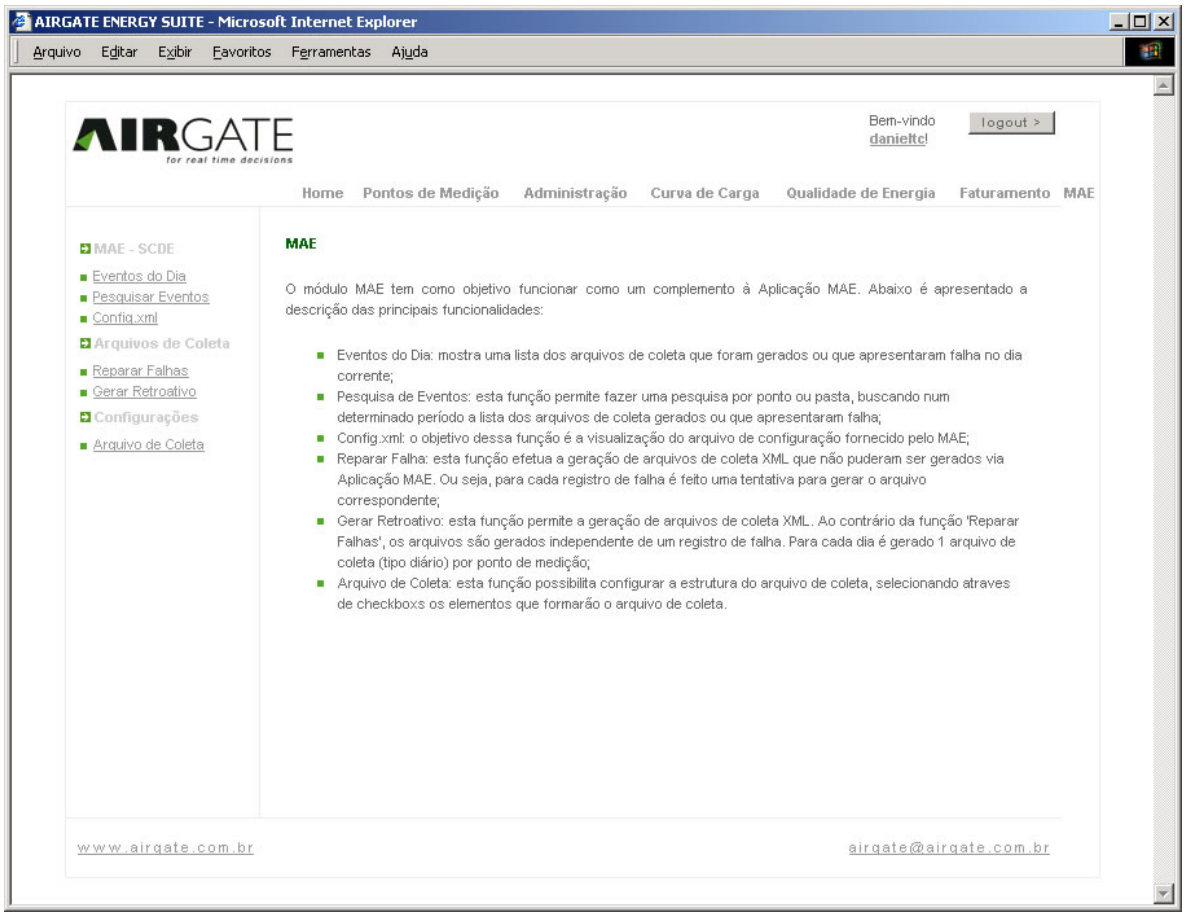


Figura 34 - Módulo MAE do AES

A Aplicação MAE Web permite o controle e a geração automática de arquivos de coleta que não foram validados ou que apresentaram falha durante a transmissão. A seguir serão apresentadas as principais funcionalidades deste módulo.

[6] Airgate Energy Suite : sistema de medição de energia, presente no mercado, responsável pela aquisição, gerenciamento de dados de faturamento e qualidade de energia.

5.4.1 Pesquisa de Eventos & Eventos do Dia

A pesquisa de eventos pode ser entendida como uma consulta sobre o histórico do log de geração dos arquivos de coleta. Para acessar esta função basta clicar sobre o item Pesquisar Eventos no menu esquerdo da sessão MAE. Será apresentada na tela principal, conforme figura 35, um formulário de pesquisa.

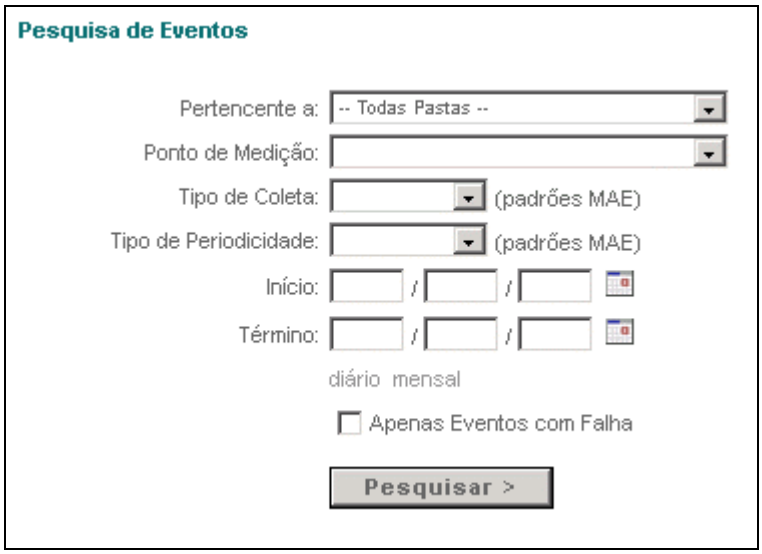
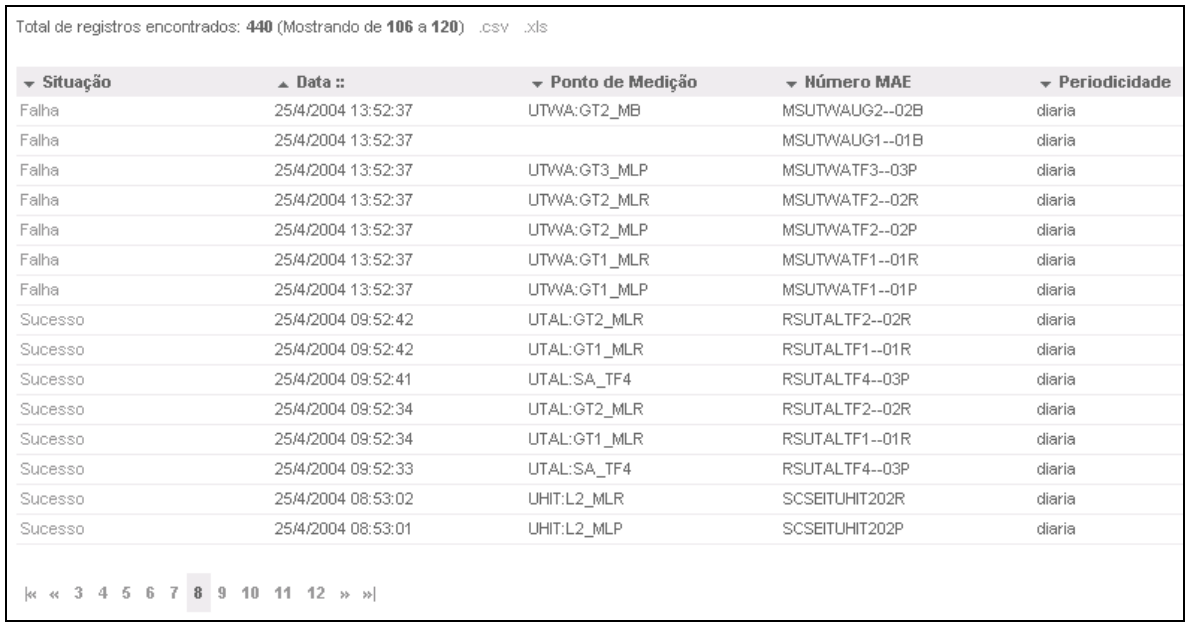


Figura 35 - Pesquisa de Eventos

O usuário efetua a pesquisa selecionando um ponto ou um conjunto de pontos pertencentes a uma pasta, escolhe o tipo de coleta (Energia, Qualidade ou Energia/Qualidade) e define um período.

A função Eventos do Dia é semelhante, no entanto ela busca apenas os eventos gerados no dia corrente. Para acessá-la basta clicar sobre o apontador Eventos do Dia no menu esquerdo. Um exemplo do resultado da pesquisa é apresentado nas Figuras 36 e 37 abaixo.

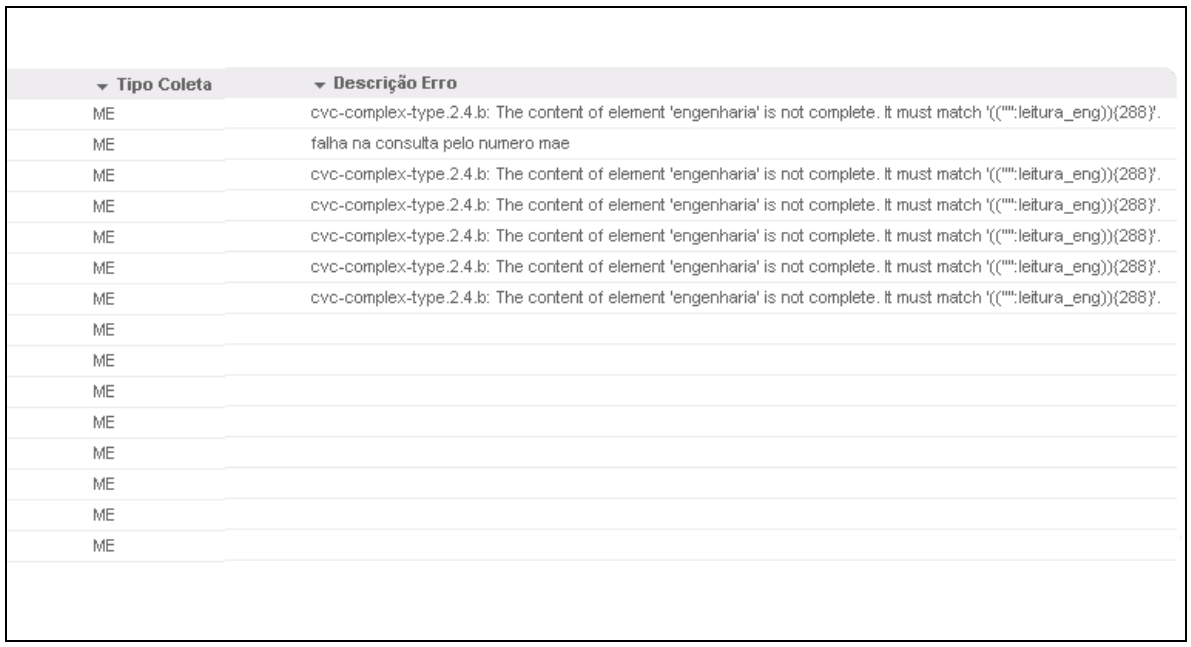


Total de registros encontrados: **440** (Mostrando de **106** a **120**) .csv .xls

Situação	Data ::	Ponto de Medição	Número MAE	Periodicidade
Falha	25/4/2004 13:52:37	UTWA:GT2_MB	MSUTWAUG2--02B	diaria
Falha	25/4/2004 13:52:37		MSUTWAUG1--01B	diaria
Falha	25/4/2004 13:52:37	UTWA:GT3_MLP	MSUTWATF3--03P	diaria
Falha	25/4/2004 13:52:37	UTWA:GT2_MLR	MSUTWATF2--02R	diaria
Falha	25/4/2004 13:52:37	UTWA:GT2_MLP	MSUTWATF2--02P	diaria
Falha	25/4/2004 13:52:37	UTWA:GT1_MLR	MSUTWATF1--01R	diaria
Falha	25/4/2004 13:52:37	UTWA:GT1_MLP	MSUTWATF1--01P	diaria
Sucesso	25/4/2004 09:52:42	UTAL:GT2_MLR	RSUTALTF2--02R	diaria
Sucesso	25/4/2004 09:52:42	UTAL:GT1_MLR	RSUTALTF1--01R	diaria
Sucesso	25/4/2004 09:52:41	UTAL:SA_TF4	RSUTALTF4--03P	diaria
Sucesso	25/4/2004 09:52:34	UTAL:GT2_MLR	RSUTALTF2--02R	diaria
Sucesso	25/4/2004 09:52:34	UTAL:GT1_MLR	RSUTALTF1--01R	diaria
Sucesso	25/4/2004 09:52:33	UTAL:SA_TF4	RSUTALTF4--03P	diaria
Sucesso	25/4/2004 08:53:02	UHIT:L2_MLR	SCSEITUHIT202R	diaria
Sucesso	25/4/2004 08:53:01	UHIT:L2_MLP	SCSEITUHIT202P	diaria

|« « 3 4 5 6 7 8 9 10 11 12 » »|

Figura 36 - Resultado da pesquisa de eventos (1/2)



Tipo Coleta	Descrição Erro
ME	cvc-complex-type.2.4.b: The content of element 'engenharia' is not complete. It must match '((":leitura_eng)){288}'.
ME	falha na consulta pelo numero mae
ME	cvc-complex-type.2.4.b: The content of element 'engenharia' is not complete. It must match '((":leitura_eng)){288}'.
ME	cvc-complex-type.2.4.b: The content of element 'engenharia' is not complete. It must match '((":leitura_eng)){288}'.
ME	cvc-complex-type.2.4.b: The content of element 'engenharia' is not complete. It must match '((":leitura_eng)){288}'.
ME	cvc-complex-type.2.4.b: The content of element 'engenharia' is not complete. It must match '((":leitura_eng)){288}'.
ME	cvc-complex-type.2.4.b: The content of element 'engenharia' is not complete. It must match '((":leitura_eng)){288}'.
ME	
ME	
ME	
ME	
ME	
ME	
ME	
ME	

Figura 37 - Resultado da pesquisa de eventos (2/2)

O resultado da pesquisa apresenta as seguintes informações sobre o(s) ponto(s) de medição: **Situação**, **Data**, **Ponto de Medição**, **Número MAE**, **Periodicidade**, **Tipo Coleta**, **Descrição Erro**:

- **Situação** (Sucesso/Falha): indica se o arquivo de coleta foi gerado. Para registros com a indicação de Sucesso basta clicar sobre a linha desejada para exibir o arquivo de coleta. Já registros com falha apresentam uma mensagem de erro no campo **Descrição Erro**;
- **Data**: para registros com situação igual a Sucesso apresenta a data na qual o arquivo de coleta foi gerado; no caso de Falha, indica a data da última tentativa de geração;
- **Ponto de Medição**: exibe o nome do ponto de medição;
- **Número MAE**: exibe o número MAE do ponto de medição;
- **Periodicidade**: mostra o tipo de periodicidade (diária, mensal, única, semanal, horária) do arquivo de coleta gerado;
- **Tipo Coleta**: exibe o tipo de coleta (ME - Energia ou QEE - Qualidade) do arquivo de coleta gerado;
- **Descrição Erro**: para registros que apresentaram falha durante a geração do arquivo de coleta, este campo é preenchido com a causa do erro.

5.4.2 Visualização do Arquivo de Configuração

A função Config.xml permite visualizar o arquivo de configuração fornecido pelo Cliente SCDE. Para acessar esta função basta clicar sobre o item Config.xml no menu esquerdo. Veja exemplo na Figura 38.

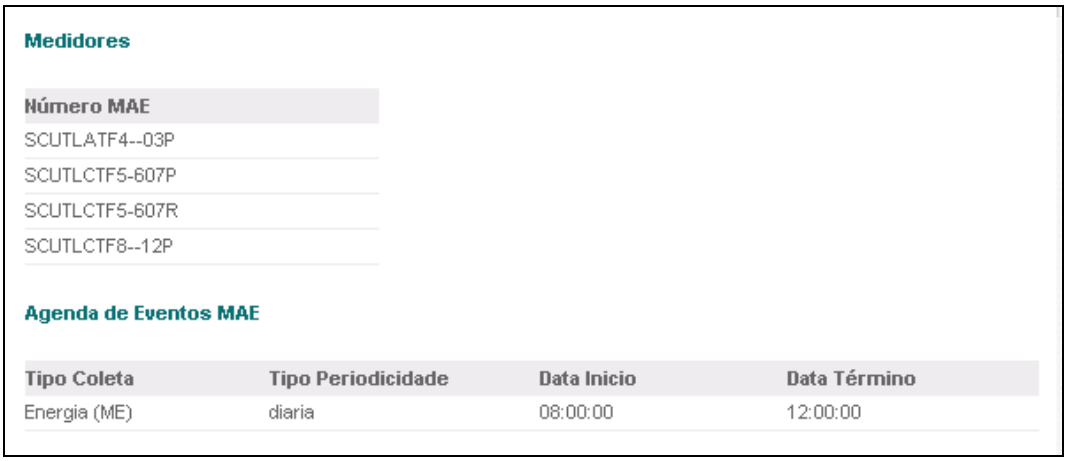


Figura 38 - Visualizando arquivo de configuração (config.xml)

5.4.3 Reparar Falhas durante Geração

Esta função efetua a geração de arquivos de coleta que não puderam ser gerados via Aplicação MAE Desktop. Ou seja, para cada registro de falha é feito uma tentativa para gerar o arquivo correspondente.

Por exemplo, ao executar a função com os parâmetros da Figura 39, a Aplicação MAE Web fará uma pesquisa sobre os pontos da pasta UTLC buscando registros de falha no período especificado. Caso encontre, é feita uma nova tentativa para gerar o arquivo correspondente.

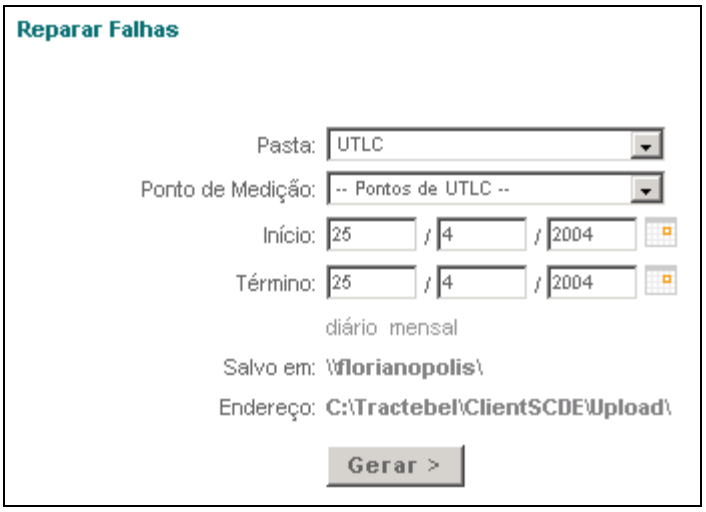


Figura 39 - Configurando a função Reparar Falhas

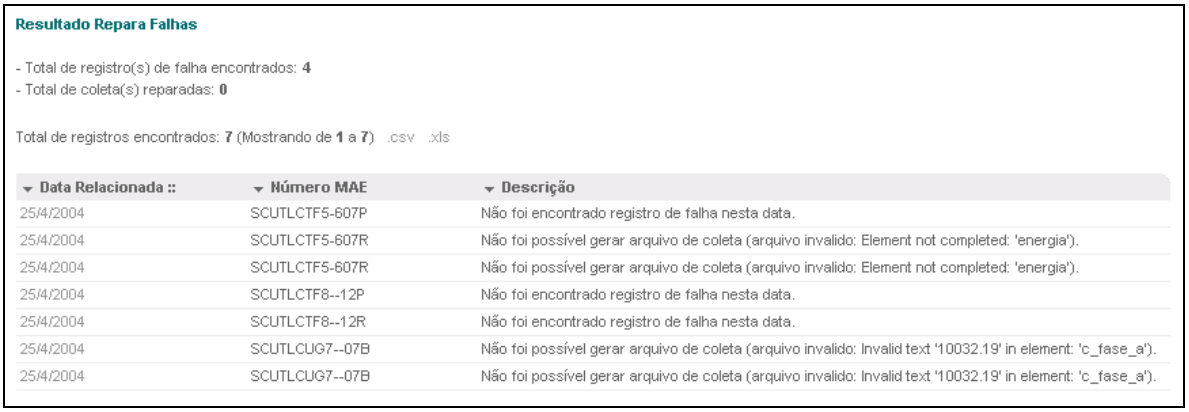


Figura 40 - Resultado após execução da função

Como mostrado na Figura 40, no dia 25/04/2004 foram encontrados 4 registros de falha. Os demais pontos pertencentes a pasta UTLC não apresentaram problemas na geração dos arquivos de coleta neste dia. Para cada registro de falha encontrado foram feitas novas tentativas de geração, no entanto, no exemplo nenhum arquivo de coleta pode ser criado, a descrição dos erros aparece no campo Descrição.

5.4.4 Gerar Arquivos de Coleta Retroativo

Esta função permite a geração de arquivos de coleta XML. Ao contrário da função Reparar Falhas, os arquivos são gerados independente de um registro de falha. Para cada dia é gerado 1 arquivo de coleta (tipo diário) por ponto de medição.

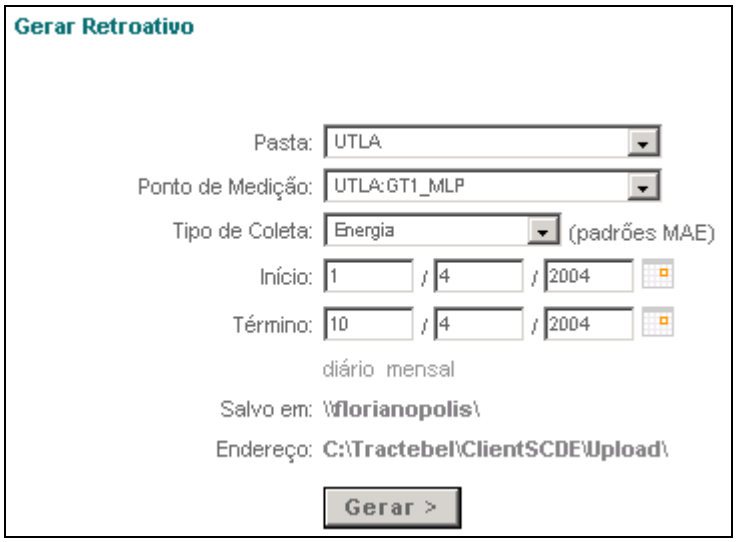


Figura 41 - Configurando a função Gerar Retroativo

No exemplo da figura 41, acima, serão gerados 10 arquivos de coleta, um para cada dia (iniciando dia 01/04/2004 até 10/04/2004), do ponto de medição UTLA:GT1_MLP. Estes serão salvos na máquina \\florianópolis\ no endereço C:\Tractebel\ClientSCDE\Upload. Para visualizar o arquivo basta clicar sobre a linha do registro correspondente, mostrado na figura 42 abaixo.

Figura 42 - Resultado da geração de arquivos retroativos

5.4.5 Configuração do Arquivo de Coleta

Esta função possibilita configurar a estrutura do arquivo de coleta, selecionando através de *checkboxs* os elementos que formarão o arquivo de coleta (figura 43).

De acordo com a especificação do MAE os arquivos de coleta podem ser de dois tipos: Energia (ME) e Qualidade (QEE). Conforme a necessidade, pode-se definir os elementos que constituirão o tipo de coleta selecionado. Após selecionado, a configuração desejada terá efeito tanto nos arquivos de coleta gerado pela Aplicação MAE Desktop como os gerado pela Aplicação MAE Web.

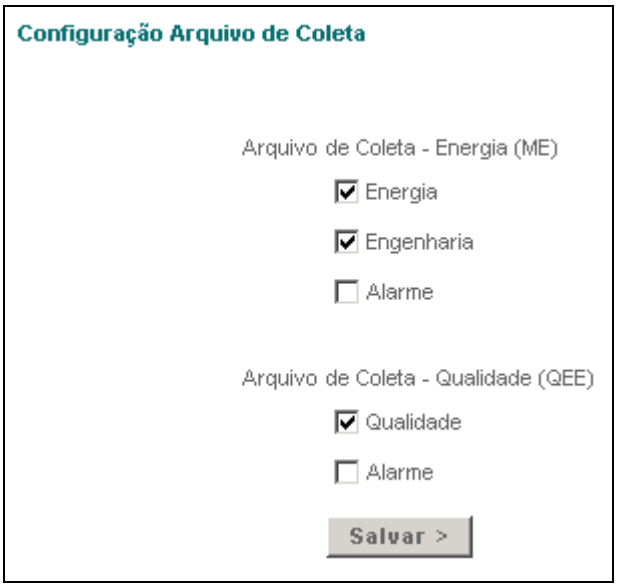


Figura 43 - Configuração do Arquivo de Coleta

6 Considerações Finais

Para adaptar-se às exigências do novo modelo do Setor Elétrico Brasileiro as empresas geradoras, distribuidoras e comercializadoras de energia elétrica deverão disponibilizar periodicamente dados de medição de energia para serem registrados no MAE. O aplicativo desenvolvido neste Trabalho de Conclusão de Curso está atualmente cumprindo esta tarefa.

As tecnologias e ferramentas escolhidas para a implementação do aplicativo apresentaram resultado adequado, com destaque para o uso de Java, Oracle e XML, que apresentam várias possibilidades para trabalharem juntas, com grande flexibilidade e interoperabilidade.

Outro ponto a se destacar é a utilização da linguagem de marcação XML na aplicação MAE, que permite fácil integração do sistema desenvolvido com outros aplicativos que necessitem de dados referentes ao consumo de energia.

6.1 Resultados

Depois de passar por um período de testes em bancada, a Aplicação MAE está funcionando de forma operacional na central de aquisição de dados da Tractebel Energia. A Tractebel é a maior empresa privada geradora de energia elétrica do país. Possui 13 usinas hidrelétricas e termoelétricas nas regiões sul e centro-oeste, com 110 medidores de energia cadastrados no MAE.

Já em operação desde 1º de Abril, a Aplicação MAE, integrada ao sistema de medição de energia AES – Airgate Energy Suíte, está se mostrando eficaz no cumprimento de seus objetivos.

Atualmente o arquivo de configuração está definindo duas coletas de periodicidade diária, desta forma, a Aplicação MAE está gerando diariamente 220 arquivos de coleta.

A Aplicação MAE provou corresponder às expectativas. As tecnologias escolhidas mostraram perfeita sintonia e os arquivos de coleta gerados estão sendo validados pelo MAE.

6.2 Trabalhos Futuros

Como a maioria dos *softwares* de mercado, novas funcionalidades são sempre bem vindas. Com a Aplicação MAE não é diferente, ainda mais porque o SCDE é um sistema recente e alterações estão ocorrendo constantemente.

Recentemente foi definida uma nova versão (2.0) para a estrutura dos arquivos de coleta, especificando regras novas na gramática de validação (XML Schema e DTD) e alterando alguns *tags*. Estas alterações nos obrigam a efetuar uma manutenção no aplicativo, atualizando o método gerador e recompilando as classes.

Pretende-se futuramente verificar a viabilidade de efetuar a geração dos arquivos por meio de XML Schema e DOM. Sendo assim, a estrutura do arquivo de coleta não ficaria especificada no aplicativo e sim no arquivo XML Schema definido pelo MAE. Portando, caso ocorra alguma alteração na estrutura dos arquivos de coleta, basta atualizar o arquivo XML Schema, não necessitando da intervenção do desenvolvedor.

Também pretende-se deixar a Aplicação MAE o mínimo dependente do banco de dados utilizado. A principio a idéia é transcrever todas as *Stored Procedures* para classes Java para depois utilizá-las como *Java Stored Procedures.*

Outra característica que pode ser incluída numa nova versão é um módulo de tolerância a falhas. Neste caso, uma ou mais réplicas do serviço funcionariam em *standby*, tomando o lugar da Aplicação MAE sempre que esta parasse de operar. Para que isto seja possível, torna-se necessário replicar também o banco de dados.

Referências Bibliográficas

[RAM00] RAMEZ, Elmasri; SHAMKANT, Navathe: *Fundamentals of DataBase Systems*. 3ed, Addison Wesley, 2000

[SUN03] SUN, M. *Java™ 2 Platform, Standard Edition, v 1.4.2 API Specification.* Disponível em: http://java.sun.com/j2se/1.4.2/docs/api/ . Acesso em 17 mai. 2004.

[DEI01] H. M. Deitel e P. J. Deitel. Java, como programar – 3.ed – Porto Alegre : Bookman, 2001.

[WHI02] WHITE Seth, et al. *JDBC™ API Tutorial and Reference: Universal Data Access for the Java™ 2 Platform* 2ed, 2002.

[KER88] KERNIGHAN Brian W. *The C Programming Language*, 2ed. Prentice Hall, Inc., 1988.

[LON00] LONEY, K. *Oracle 8i: O Manual do DBA*; TRAD de Kátia Roque. Rio de Janeiro: Campus, 2000.

[SHA01] SHAPIRO, Jeffrey R. *SQL Server 2000: Completo e Total*. 1ed Makron Books, 2001

[HEI01] HEITLINGER, Paulo. *O Guia Prático da XML*. Lisboa: Centro Atlântico, 2001.

[SAX02] SAX, *Simple API for XML – Java API* Disponível em: http://www.saxproject.org/. Acesso em 10 abr 2004.

[LAR00] LARMAN, Craig *Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos* – Porto Alegre : Bookman, 2000

[NIE01] NIEDERAUER, Juliano - *Desenvolvendo Websites com PHP 4.* 1.ed. Novatec, 2001

[DOM04] WORLD WIDE WEB CONSORTIUM, W3C. *Document Object Model (DOM)* Disponível em: http://www.w3c.org/DOM/. Acesso em 29 mai 2004.

[DB204] DB2, Product Family Disponível em: http://www-306.ibm.com/software/data/db2/. Acesso em 29 mai 2004

[SUE02] SUEHRING, Steve. *MySQL: a Bíblia.* 1 ed. Campus, 2002

[SYB04] SYBASE, *Unwired Enterprise solutions via integration of databases*. Disponível em http://www.sybase.com/home. Acesso em 29 mai 2004.

[ORA04] ORACLE TECHNOLOGY NETWORK– PL/SQL. Disponível em http://otn.oracle.com/tech/pl_sql/index.html. Acesso em 20 mai 2004.

[MAE04] MAE - Mercado Atacadista de Energia Elétrica. *Implantação da Medição.* Disponível em http://www.mae.org.br/index.jsp. Acesso 10 abr 2004.

[ONS04] ONS - Operador Nacional do Sistema Elétrico. Disponível em http://www.ons.org.br. Acesso em 10 abr 2004.

[ANE04] ANEEL - Agência Nacional de Energia Elétrica. Disponível em http://www.aneel.gov.br/. Acesso em 10 abr 2004.

[MAC00] MACORATTI, Jose Carlos. *Aprenda Rápido: ASP*. 2 ed. Visual Books, 2000

[RGM04] MAE, Requisitos dos Gateways de Medição, Disponíbel em:

http://www.mae.org.br/implantacao_medicao/ucm/docs/MAE1_IT3_10204_v01.pdf. Acesso em 20 mai 2004 .

[JAX04] JAXP – TheServerSide.com. Disponível em: http://www.theserverside.com/articles/article.tss?l=JAXP. Acessado em 10 abr 2004