Escola Técnica Professor Everardo Passos

ETEP Faculdades

THIAGO DA SILVA MACHADO

ADAPTAÇÃO PARA LEGISLAÇÃO BRASILEIRA DE UM OPEN SOURCE ERP

Jacareí – SP

2014

Thiago da Silva Machado

ADAPTAÇÃO PARA LEGISLAÇÃO BRASILEIRA DE UM OPEN SOURCE ERP

Trabalho apresentado na ETEP de Jacareí como parte dos requisitos necessários para obtenção do grau de Tecnologia da Informação, sob a orientação do professor Antônio Egydio Graça

Jacareí – SP

2014

Thiago da Silva Machado

ADAPTAÇÃO PARA LEGISLAÇÃO BRASILEIRA DE UM OPEN SOURCE ERP

Trabalho apresentado na ETEP de Jacareí como parte dos requisitos necessários para obtenção do grau de Tecnologia da Informação, sob a orientação do professor Antônio Egydio Graça

_____

Professor (a) Orientador (a): Nome Completo
ETEP Faculdades

_____

Prof.(a) Avaliador(a): Título (Dr, Ms, Esp) Nome Completo
ETEP Faculdades

Jacareí – SP

2014

Dedico este trabalho aos meus pais, as minhas irmãs e a minha namorada que sempre estiveram comigo, me impulsionando à seguir em frente e a enfrentar todos os obstáculos que vieram e virão pela frente, fazendo de mim um vencedor.

RESUMO

MACHADO, Thiago. Adaptação para a Legislação Brasileira de um Open-Source ERP. 2014. 85 f. Trabalho de Conclusão de Curso (Ensino Superior) – Escola Técnica Professor Everardo Passos - ETEP Faculdades, de Jacareí, 2014.

Este trabalho de conclusão de curso visa abordar uma implementação (adaptação) para legislação brasileira de um documento digital conhecido como Manifesto Eletrônico de Documentos Fiscais (MDF-e) em um sistema de gestão empresarial ERP (Enterprise Resource Planning) Open Source denominado ADempiere.

Há necessidade da implantação do módulo do MDF-e no ADempiere veio quando se tornou obrigatório a emissão do manifesto no ambiente de produção da Sefaz. Com isso, as empresas transportadoras de carga (própria ou não) se viram obrigadas a adaptar o ERP a esta necessidade exigida pelo governo.

A não emissão do documento eletrônico proíbe as empresas de transporte de saírem com seus caminhões para entrega das mercadorias, com isso gerando prejuízos e atrasos nas entregas.

Por fim, a adaptação foi realizada de acordo com a necessidade das empresas e obrigatoriedade da emissão eletrônica do manifesto conforme exigência da legislação brasileira.

Palavras-chave: manifesto, ERP, ADempiere, eletrônico, transporte, MDF-e, fiscal.

ABSTRACT

MACHADO , Thiago . Adaptation to Brazilian legislation of an Open - Source ERP . 2014. 85 f . Completion of course work (Higher Education) - Escola Técnica Professor Everardo Passos - ETEP Faculdades, Jacarei , 2014.

This work aims to address ongoing completion of an implementation (adaptation) to Brazilian legislation of a digital document known as Electronic Manifest Tax Documents (MDF-e) on a business management system ERP (Enterprise Resource Planning) called ADempiere Open Source.

There is need for the implementation of the module MDF-e in ADempiere came when it became mandatory to issue the manifesto in the production environment Sefaz. Thus, cargo carriers (own or not) companies have had to adapt to this need ERP required by the government.

The failure to issue the electronic document prohibits the transport companies leave their trucks to deliver the goods , it generates losses and delays in deliveries.

Finally, the adaptation was performed according to the need and requirement of the companies issuing the electronic manifest as required by Brazilian law.

Keywords : clear , ERP , ADempiere , electronic, transportation, MDF-e, tax.

LISTA DE FIGURAS

LISTA DE SIGLAS

API	*Application Programming Interface*
CPU	*Central Processing Unit*
CSV	*Comma-Separated Values*
CT-e	Conhecimento de Transporte Eletrônico
CTRC	Conhecimento de Transporte Rodoviário de Cargas
DAMDFE	Documento Auxiliar do Manifesto Eletrônico de Documentos Fiscais
DHCP	*Dynamic Host Configuration Protocol*
DTD	*Document Type Definition*
ERP	*Enterprise Resource Planning*
FTP	*File Transfer Protocol*
GED	Gerenciamento Eletrônico de Documentos
GUI	*Graphical User Interface*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
HQL	*Hibernate Query Language*
ICMP	*Internet Control Message Protocol*
IDE	*Integrated Development Environment*
IP	*Internet Protocol*
JVM	*Java Virtual Machine*
MDF-e	Manifesto Eletrônico de Documentos Fiscais
NF-e	Nota Fiscal Eletrônica
O.O.	Orientação a Objetos
PDF	*Portable Document Format*
SGBD	Sistema de Gestão de Banco de Dados
SGBDR	Sistema de Gestão de Banco de Dados Relacional
SMTP	*Simple Mail Transfer Protocol*
S.O.	Sistema Operacional
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
TCC	Trabalho de Conclusão de Curso

TCP	*Transmission Control Protocol*
T.I.	Tecnologia da Informação
URL	*Uniform Resource Locator*
W3C	*World Wide Web Consortium*
WS	*Web Service*
WWW	*World Wide Web*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Este trabalho de conclusão de curso (TCC) visa abordar a customização (implementação) de um sistema integrado ERP (Planejamento dos Recursos Empresariais - *Enterprise Resource Planning*) às legislações brasileiras.

Atualmente existem vários softwares no mercado, e um sistema onde se integra vários departamentos de uma empresa (financeiro, fiscal, contas à pagar e etc) muitas vezes não está totalmente pronto para utilização. Normalmente, uma empresa emite notas, conhecimentos e manifestos, tudo dentro da legislação de seu país, e esses sistemas (quase sempre) não possuem os módulos (*models*) necessários para essas funções e operações fiscais, pois grande parte dos programas são estrangeiros, precisando então de uma adaptação ao país onde será executado e colocado em produção.

1.1 OBJETIVO

Com base na necessidade e da falta de adaptação de alguns processos fiscais, foi desenvolvido um módulo para emissão de um documento fiscal brasileiro (atualmente obrigatório) chamado MDF-e (Manifesto Eletrônico de Documentos Fiscais), para as empresas de transporte em um software ERP *Open Source* denominado ADempiere.

Este módulo possibilita a emissão, cancelamento, consulta, encerramento e impressão do documento MDF-e dentro de um sistema que está em execução nas empresas transportadoras de carga (própria ou não).

1.2 METODOLOGIA

A metodologia do TCC é baseada em três fases. A primeira fase consiste em um estudo inicial com referências e revisões bibliográficas específicas para o tema, ou seja, uma abordagem geral sobre os softwares e hardwares necessários para o pleno funcionamento do ERP ADempiere e para o desenvolvimento e implementação do módulo MDF-e.

A segunda parte abordará o ERP ADempiere em si, contando seus principais recursos, vantagens, instalação, ferramentas utilizadas entre outros.

A terceira e última parte visa documentar o módulo desenvolvido, ou seja, o Manifesto Eletrônico de Documentos Fiscais (MDF-e). Nesta será abordada sua finalidade, obrigatoriedade, operacionalidade, vantagens e *models* necessários.

1.3 JUSTIFICATIVA

Este projeto justifica-se pela necessidade e obrigatoriedade das empresas transportadoras e suas filiais na emissão do manifesto.

Diversos documentos fiscais brasileiros são emitidos eletronicamente, substituindo os documentos em papel, com isso, os softwares (integrados ou não) tiveram de ser customizados e adaptados à fim de atender as necessidades.

Muitos dos documentos são obrigatórios, forçando as empresas a procurarem equipes de T.I. (Tecnologia da Informação) especializadas no assunto para realizar tal tarefa. Essa obrigatoriedade de emissão eletrônica acontece pois o governo já percebeu de uns anos para cá que utilizar da tecnologia para controlar os documentos e operações fiscais é muito mais ágil, vantajoso e até mesmo seguro.

2 REVISÃO BIBLIOGRÁFICA

Para o pleno funcionamento de qualquer programa em um computador, é necessário diversas ferramentas (softwares e hardwares) e alguns ajustes na máquina. Isso não é muito diferente quando o programa é um ERP, ainda mais que um sistema integrado necessita de configurações um pouco mais avançadas do que o normal.

Este capítulo começará abordando as ferramentas e as tecnologias necessárias (partindo das mais básicas às mais avançadas) para a perfeita execução do ERP ADempiere e para o desenvolvimento e implementação do módulo do manifesto eletrônico.

2.1 COMPUTADOR

O ato de processar dados (informações) tem um nome: computar. Daí vem o nome computação e o nome do objeto computador.

O computador é uma máquina que computa, ou seja, processa dados. Por definição, um computador é uma máquina que lê dados, efetua cálculos e fornece resultados. Um computador (basicamente) possui três tipos de processos: entrada de dados (ler ou receber os valores iniciais ou constantes), processamento (efetuar o cálculo) e saída de dados (fornecer os resultados obtidos) (CERIBELLI, 2008).

Um computador, para executar tais funções, necessita de dois elementos primordiais: hardware (parte física) e software (parte lógica).

2.1.1 Hardware

Parte física de um computador. É constituído pelos componentes eletrônicos, necessários para fazer um computador funcionar. Hardware é a maneira (termo) usada para designar as peças, circuitos, e peças em geral. Em um computador, há dois tipos de Hardware, os internos (processador, *hard-disk*, memória-ram entre outros) e os externos (monitor, *pendrive*, teclado, mouse, impressora entre outros).

2.1.2 Software

Software é um fluxo de instruções escritas em linguagem de programação. Estes comandos, ou instruções, criam as ações dentro do programa e permitem seu funcionamento. O software é definido como parte lógica, onde a função é fornecer instruções para o hardware (em um computador). Software é tudo que pode ser executado no computador. Os softwares podem ser classificados em três tipos: Software de Sistema, Software de Programação e Software de Aplicação (PACIEVITCH, 2011).

Software de Sistema - É constituído pelos Sistemas Operacionais (S.O.) como Windows e Linux. (PACIEVITCH, 2011).

Software de Programação - São softwares usados para criar outros programas a parir de uma linguagem de programação como: Java, PHP, Pascal, C# entre outras (PACIEVITCH, 2011).

Software de Aplicação - São os programas utilizados para aplicações dentro do sistema operacional (S.O.), que não estejam ligados com o funcionamento do mesmo. Exemplos: Word, Exel, Paint, Bloco de Notas, Calculadora entre outros (PACIEVITCH, 2011).

2.2 SISTEMA OPERACIONAL

Sabe-se que um computador necessita de dois elementos primordiais para seu funcionamento, sendo eles o Software e o Hardware. Todavia um elemento extra (mas não menos importante) “inicia” todo o Hardware de sua máquina. Para que o usuário possa utilizar seu dispositivo, e realizar suas tarefas (criar documentos, enviar arquivos, calcular valores e etc) existe um Software de Operação (Software de Sistema) denominado sistema operacional (S.O.).

Segundo Deitel (2005), no nível mais simples e básico, um S.O. é uma coleção de programas que inicializam o hardware do computador e realiza duas tarefas:

1. Gerenciar os recursos de hardware e software do sistema. Em um computador, esses recursos incluem o processador, a memória, o espaço em disco etc. Em um celular, o S.O. gerencia o teclado, a tela, a agenda, a bateria e etc;
2. Proporcionar uma maneira estável para lidar com o hardware, sem ter de conhecer todos os detalhes do mesmo.

A primeira tarefa (gerenciamento de recursos de software e hardware) é uma tarefa extremamente importante. Diversos programas e métodos de entrada de dados competem pela atenção da CPU (*Central Processing Unit* - Unidade Central de Processamento) e demandam memória, espaço em disco e largura de banda de entrada/saída. O sistema operacional faz o papel do bom pai. Ele cuida para que cada aplicativo tenha os recursos necessários para o funcionamento, e gerencia a capacidade limitada do sistema para atender a todos os usuários e aplicativos (DEITEL, 2005).

Um S.O. assegura que os aplicativos continuem funcionando após as atualizações de hardware. Isso acontece porque é o sistema operacional quem gerencia o hardware e a distribuição dos seus recursos (DEITEL, 2005).

Segundo Deitel (2005), a segunda tarefa é fornecer uma interface consistente para os aplicativos. Um ambiente visual nos permite interagir com o S.O. e com os recursos (dispositivos/hardware) do computador de forma fácil. Existem dois tipos de interfaces:

1. *Interface de Linha de Comando (Command Line Interface)* - Usa comandos alfanuméricos simples para navegar entre os discos, diretórios e pastas, à fim de conseguir outras funções como copiar, formatar deletar, etc., e para executar aplicativos. Exemplos: DOS, Unix and Linex entre outros;
2. *GUI (Graphical User Interface - Interface Gráfica Para Usuários)* - Usam ícones, menus e janelas para acessar programas e etc. Para Windows temos apenas o ambiente gráfico padrão. Nas versões Windows Vista e Windows 7 temos a chamada Windows Aero. Para GNU/Linux temos vários ambientes gráficos, como KDE, Gnome, BlackBox, Xfce, entre outros.

2.3 REDES DE COMPUTADORES E PROTOCOLOS

Podemos pensar em rede de computadores como diversas máquinas interligadas fisicamente entre si (ou não) onde os seus utilizadores promovem a troca de informação (Figura 1). Entretanto, uma rede não pode ser bem estabelecida considerando apenas o hardware como preocupação principal como nas primeiras redes, atualmente o software é considerado uma das partes mais importantes na concepção de novas tecnologias de redes de computadores.

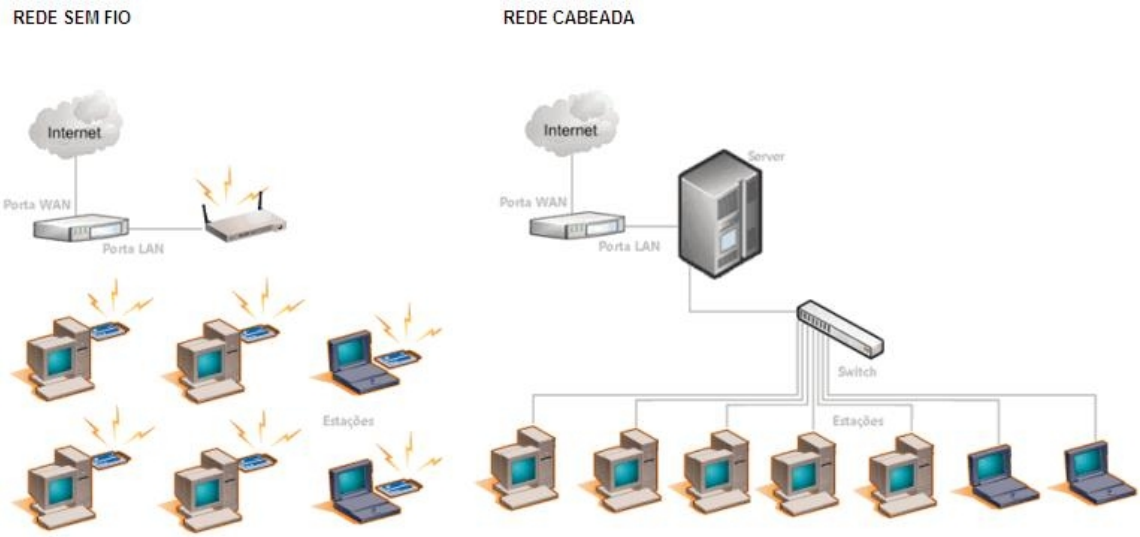

Figura 1: Rede de Computadores
Fonte: www.tecvirtua.com.br

“Protocolo é a 'língua' dos computadores, ou seja, uma espécie de idioma que segue normas e padrões determinados, pois é através dele que é possível a comunicação entre um ou mais computadores” (ALENCAR, 2005).

“O protocolo permite a comunicação entre processos (que se executam eventualmente em diferentes máquinas), isto é, um conjunto de regras e procedimentos a respeitar para emitir e receber dados numa rede” (ALENCAR, 2005).

> “Existem diversos tipos de protocolos, alguns deles, por exemplo, são especializados na troca de ficheiros como o FTP (*File Transfer Protocol* - Protocolo de Transferência de Arquivos), outros poderão servir para gerir simplesmente o estado da transmissão e os erros como o ICMP (*Internet Control Message Protocol* – Protocolo de Internet para Controle de Mensagens) entre outros. Na internet (a maior rede de computadores interligados do mundo), os protocolos utilizados fazem parte de uma sequência/conjunto de protocolos. Esta sequência de protocolos chama-se TCP/IP (*Transmission Control Protocol* - Protocolo de Controle de Transmissão / *Internet Protocol* – Protocolo de Internet)” (ALENCAR, 2005).

A internet, por exemplo, é uma rede de computadores interligadas, que possibilita o acesso as informações em qualquer lugar do mundo, e esta possui diversos protocolos sendo algum deles: IP (*Internet Protocol*), DHCP (*Dynamic Host Configuration*

Protocol), TCP (Transmission Control Protocol), HTTP (*Hypertext Transfer Protocol*), FTP (*File Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*) entre outros" (ALENCAR, 2005).

2.3.1 Hypertext Transfer Protocol

O HTTP (Protocolo de Transferência de Hipertexto - *Hypertext Transfer Protocol*) é um protocolo de comunicação entre sistemas que permite a transferência de dados entre redes de computadores (W3C, 2009, tradução nossa). A troca de informações entre um *Browser* e um servidor Web é toda feita através desse protocolo, que foi criado especificamente para a *World Wide Web* (WWW).

O HTTP é o protocolo utilizado para transferência de páginas HTML (*HyperText Markup Language* - Linguagem de Marcação de Hipertexto) do computador para a Internet, ou seja, o HTTP nada mais é do que uma página na Internet sendo ele o responsável pela forma de conversação no estilo pedido-resposta entre um cliente (*Browser*/Navegador) e um servidor (Sevidor Web). Por isso, as URLs (*Uniform Resource Locator* – Localizador de Recursos) dos sites utilizam no início a expressão "http://", definindo o protocolo usado. Esta informação é necessária para estabelecer a comunicação entre a URL e o servidor Web que armazena os dados, enviando então a página HTML solicitada pelo usuário (W3C, 1999, tradução nossa).

Para que a transferência de dados na Internet seja realizada, o protocolo HTTP necessita estar agregado a outros dois protocolos de rede: TCP e IP Esses dois últimos protocolos formam o modelo TCP/IP, necessário para a conexão entre computadores clientes/servidores (W3C, 1999, tradução nossa).

2.3.2 Cliente/Servidor

A tecnologia cliente/servidor *(client/server)* é uma arquitetura na qual o processamento da informação é dividido em módulos ou processos distintos. Um processo é responsável pela manutenção da informação (servidores) e outros responsáveis (os clientes) pela obtenção dos dados (MENDES, 2002).

Numerosas aplicações funcionam de acordo com um ambiente cliente/servidor, o que significa que máquinas clientes (máquinas que fazem parte da rede) contatam um servidor, uma máquina geralmente bastante potente, em termos de capacidades de entrada/saída, que fornece serviços (Figura 2). Estes serviços são programas que fornecem dados como hora, arquivos, uma conexão, etc (MENDES, 2002).

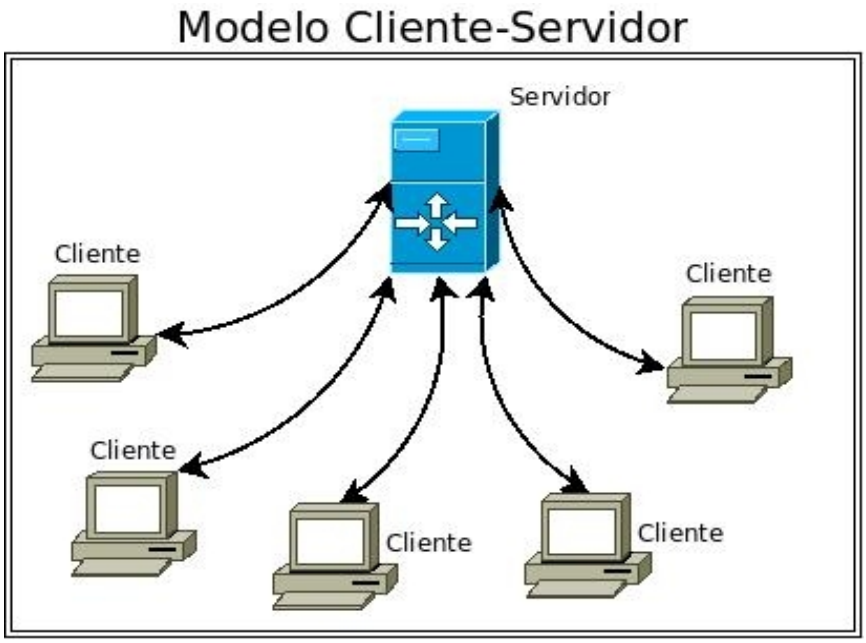

Figura 2: Cliente / Servidor
Fonte: www.pt.wikibooks.org

O cliente possui características como: iniciar pedidos para servidores, esperar por respostas, receber respostas, conecta-se a um pequeno número de servidores de

uma só vez, interagir diretamente com os usuários finais através de qualquer interface e utilizar recursos da rede (MENDES, 2002).

O servidor possui características como: sempre esperar por um pedido de um cliente, atender os pedidos e responder aos clientes com os dados solicitados, se comunicar com outros servidores para atender uma solicitação específica do cliente e fornece recursos de rede (MENDES, 2002).

2.3.2.1 Tipos de Servidores

Existem vários tipos de servidores. Os mais conhecidos são: servidor de fax, de arquivos, web, e-mails, imagens e FTP. Cada um destes servidores executa uma função, por exemplo, para você visualizar a página do Google, você está utilizando o servidor web, o qual é responsável pelo armazenamento das páginas de um site (MENDES, 2002).

Para que você consiga visualizar as páginas web, você utiliza um navegador, portanto o navegador é o cliente e o site é o servidor, pois o primeiro acessa informações disponibilizadas pelo segundo. Desta forma, as redes que utilizam servidores são chamadas do tipo cliente/servidor (MENDES, 2002).

2.3.2.2 Vantagens e Desvantagens da Tecnologia Cliente/Servidor

Segundo Mendes (2002), a tecnologia cliente/servidor possui seus pontos positivos e negativos. Abaixo é listado as vantagens e desvantagens dessa arquitetura:

- *Recursos centralizados (Vantagem)* - já que o servidor está no centro da rede, pode gerenciar recursos comuns a todos os usuários, como por exemplo uma base de dados centralizada, a fim de evitar os problemas de redundância e de contradição de informações;

- *Maior segurança (Vantagem)* - porque o número de pontos de entrada que permitem o acesso aos dados é menos importante;
- *Administração a nível do servidor (Vantagem)* - como os clientes têm pouca importância neste modelo, têm menos necessidade de ser administrados;
- *Rede evolutiva (Vantagem)* - graças a esta arquitetura, é possível suprimir ou acrescentar clientes sem perturbar o funcionamento da rede e sem modificação essencial;
- *Clientes (Desvantagem)* - Podem solicitar serviços, mas não podem oferecê-los para outros, com isso, sobrecarregando o servidor;
- *Arquitetura cliente/servidor (Desvantagem)* - Se um servidor crítico falha, os pedidos dos clientes não poderão ser cumpridos.

2.3.3 Web Services

Web Service (WS) é um sistema de software projetado para suportar a interoperabilidade entre máquinas sobre rede, sendo este muito utilizado em aplicações B2B[1] (*Business - to - Business*). Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis (W3C, 2004, tradução nossa).

> "Os Web Services são componentes que permitem às aplicações enviar e receber dados em formato XML (*eXtensible Markup Language* - Linguagem Extensível de Marcação). Cada aplicação pode ter a sua própria "linguagem", que é traduzida para uma linguagem universal, o formato XML" (W3C, 2004, tradução nossa).

Resumindo, o WS é a solução utilizada na integração de sistemas e na comunicação entre aplicações (diferentes ou não), permitindo às aplicações enviar e receber dados em formato XML (Figura 3).

1 Comércio eletrônico associado a operações de compra e venda, de informações, de produtos e de serviços através da Internet ou através da utilização de redes privadas partilhadas entre parceiros de negócios, substituindo assim os processos físicos que envolvem as transações comerciais.

> “Os Web Services possuem como base os padrões XML e SOAP (*Simple Object Access Protocol* - Protocolo Simples de Acesso a Objetos). O transporte dos dados é realizado normalmente via protocolo HTTP. Os dados são transferidos no formato XML e encapsulados pelo protocolo SOAP” (W3C, 2004, tradução nossa).

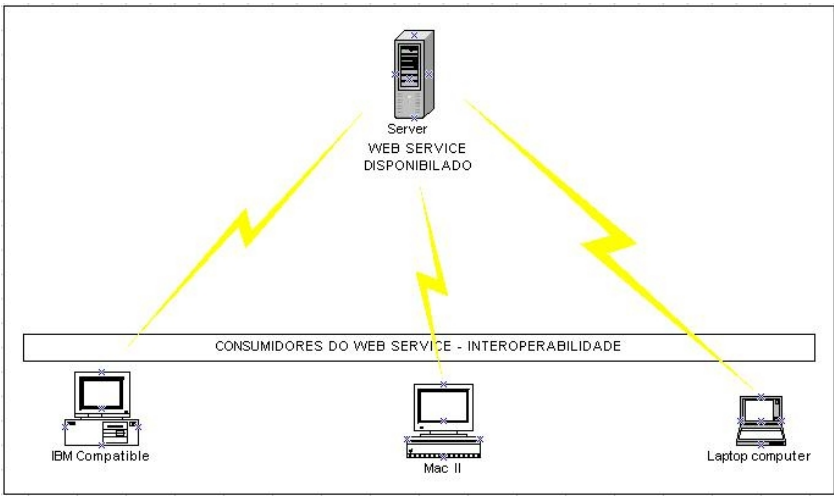

Figura 3: Web Service
Fonte: www.devmedia.com.br

O WS traz agilidade para os processos e eficiência na comunicação entre cadeias de produção ou de logística. Toda e qualquer comunicação entre sistemas passa a ser dinâmica e principalmente segura, pois não há intervenção humana. (W3C, 2004, tradução nossa)

Os Web Services permitem que a integração de sistemas seja realizada de maneira compreensível, reutilizável e padronizada (W3C, 2004, tradução nossa).

2.4 LINGUAGEM DE MARCAÇÃO

As linguagens de Marcação são utilizadas para definir formatos, maneiras de exibição e padrões dentro de um documento qualquer. Normalmente, elas não possuem qualquer estrutura de controle como as linguagens de programação tradicionais (por exemplo, comandos condicionais ou de repetição). Dessa forma, elas servem

basicamente para definir como um determinado conteúdo será exibido na tela ou como os dados estarão estruturados ao trafegar entre os diferentes módulos de um sistema ou na rede (W3C, 2013, tradução nossa).

As linguagens de marcação se utilizam do conceito de marcador ou *tag*, que já trazem algum significado e que quando forem visualizados por algum sistema que as reconheça, irão saber como o conteúdo deve ser exibido (W3C, 2013, tradução nossa).

Exemplo: a linguagem HTML é uma linguagem de marcação, pois com ela você "marca" os seus elementos e os define (títulos, textos, imagens, parágrafos e etc). Exemplo: para colocar a palavra "Java" em *itálico*, basta colocar a palavra entre as Tags .

Existem diversos tipos e padrões para linguagens de marcação e a linguagem HTML é mais uma delas, mas que se popularizou por causa do advento da internet, já que a maioria dos documentos que trafegam na rede a utiliza para exibir suas informações .

Outro exemplo de linguagem de marcação é o XML, que diferente do HTML, seu foco está na estruturação dos dados que serão enviados entre dois pontos de um mesmo sistema e uma maior rigidez na forma de escrita da sua estrutura (W3Schools, 1999, tradução nossa).

2.4.1 *eXtensible Markup Language*

O XML (Linguagem de Marcação Extensível - *eXtensible Markup Language*) é uma linguagem de marcação e uma recomendação da W3C (*World Wide Web Consortium*) para a criação de documentos com dados organizados hierarquicamente.

A linguagem XML é classificada como extensível porque permite definir os elementos de marcação, ou seja, ela padroniza uma sequência de dados com o objetivo

de organizar, separar o conteúdo de forma hierárquica e integrá-lo com outras linguagens à fim de manipular as informações. O propósito principal da linguagem é a facilidade de compartilhamento de informações através da internet (W3Schools, 1999, tradução nossa).

Figura 4: XML
Fonte: www.blog.segr.com.br

Utilizar do XML pode trazer vantagens e desvantagens dependendo do que se quer fazer com ele. Abaixo é listado as principais vantagens e desvantagens de se usar essa tecnologia:

- Vantagem - ser padrão aberto (você não precisa pagar nada para utilizar);
- Vantagem - existem várias ferramentas e editores grátis bons no mercado;
- Vantagem - simplicidade e legibilidade tanto para humanos quanto para computadores;
- Vantagem - concentração na estrutura da informação (não na sua aparência);
- Vantagem - possibilidade de criar sua própria sintaxe de dados (estruturar os dados da forma que achar melhor, através da criação ilimitada de tags);
- Vantagem - permitir validação (torna os teste unitários mais efetivos, e a construção de aplicações bem mais fácil);
- Desvantagem – a sintaxe do XML é redundante ou torna-se grande em relação a representações de dados semelhantes;

- Desvantagem - a redundância pode afetar a eficiência quando utiliza-se o XML para armazenamento afetando também transmissão e processamento (os custos ficam muito mais elevados);
- Desvantagem - o XML pode ser substituído por documentos com formatos mais simples, como os arquivos properties ou texto;
- Desvantagem - a grande quantidade de informação repetida pode prejudicar a velocidade de transferência real da informação (se estiver em formato XML);

2.4.2 Simple Object Access Protocol

O SOAP (Procolo Simples de Acesso a Objetos - *Simple Object Access Protocol*) é um protocolo de comunicação baseado em XML que permite a comunicação de mensagens entre aplicações via HTTP, normalmente utilizado em Web Services. Uma das grandes qualidades desse protocolo é sua independência de plataforma e linguagem, além de ser simples e extensível por utilizar XML.

Geralmente servidores SOAP são implementados utilizando-se servidores HTTP, embora isto não seja uma restrição para o funcionamento do protocolo. As mensagens SOAP são documentos XML que aderem a uma especificação fornecida pelo órgão W3C (W3Schools, 1999, tradução nossa).

2.4.2.1 Mensagem SOAP

Uma mensagem SOAP é um documento XML comum contendo um elemento chamado *Envelope* que identifica o documento XML como uma mensagem SOAP, um elemento *Header* que contém informações sobre o cabeçalho do documento, e um elemento *Body* que é o corpo do documento contendo informações de chamada e

resposta. Dentro do corpo (*Body*) possui um elemento *Fault* que contém erros e informações de status (W3Schools, 1999, tradução nossa).

Resumindo, a mensagem SOAP possui: mecanismo para definir a unidade de comunicação, mecanismo para lidar com erros, mecanismo de extensão que permite evolução e mecanismo entre as mensagens SOAP e HTTP (permitindo representar tipos de dados em XML).

2.4.2.2 Elementos SOAP

Segundo o site da W3Schools (1999, tradução nossa), a sintaxe de uma mensagem SOAP é bem simples e contem algumas regras assim como o XML, ou seja, o uso de *namespaces*[2] específicos. Seu *namespace* deve utilizar a codificação definida pela W3C, não deve conter referencia DTD[3] (*Document Type Definition* – Tipo de Definição do Documento) e nem ter instruções de processamento XML. Os elementos da mensagem SOAP são:

- *Envelope* - elemento raiz da mensagem SOAP. Este elemento define o documento XML como uma mensagem SOAP;
- *Header* - contem informações especificas do aplicativo da mensagem SOAP. É um elemento opcional;
- *Body* - contem a mensagem SOAP pretendida que o usuário espera;
- *Fault* - elemento de falha aonde possui erros e informações de status de uma mensagem SOAP. É um elemento opcional.

2 Atributos colocados nas *tags* de abertura identificando o documento como um arquivo XML, XHTML e etc. Sua principal funcionalidade é evitar conflito de *tags* quando uma concatenação de arquivos distintos acontecer.

3 Conjunto de regras que define quais tipos de dados e entidades farão parte de um documento XML, ou seja, funciona como uma espécie de validador de arquivo.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

Figura 5: Pedido SOAP
Fonte: www.helpdev.com.br

Podemos interpretar na Figura 5 como uma chamada a um método. A *tag* <m:GetPrice> seria o nome do método, e está passando um parâmetro <m:Item> (valor “Apples”). Por fim, o cliente recebe a resposta com o preço da maça como na Figura 6.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

Figura 6: Resposta SOAP
Fonte: www.helpdev.com.br

2.5 LINGUAGEM DE PROGRAMAÇÃO

Uma linguagem de programação é um método padronizado para comunicar instruções para um computador, ou seja, um conjunto de regras sintáticas e semânticas usadas para definir um software. Esta permite que um programador especifique precisamente sobre quais dados um computador vai atuar, como estes dados serão

armazenados ou transmitidos e quais ações devem ser tomadas sob várias circunstâncias.

As linguagens de programação podem ser classificadas em níveis de linguagens, sendo que os níveis mais baixos são mais próximas da linguagem interpretada pelo processador e outras mais distante das linguagens naturais. Hoje, existem diversos tipos de linguagens de programação, as quais são escritas pelos programadores, algumas dessas linguagens são compreendidas pelo computador e outras ajudam na forma de tradutores. As linguagens podem ser classificadas como: Linguagem de Máquina (baixo nível), Linguagens Assembly (baixo nível) e Linguagens de Alto Nível.

2.5.1 Linguagem de Máquina (Baixo Nível)

É uma linguagem “crua”, ou seja não muda seu estado natural. Esta linguagem é composta somente por números representados de forma binária, que sob o ponto de vista do computador, representam as operações e os operandos que serão usados no processamento do programa. Para um ser humano, a linguagem de máquina é dificílima de se compreender (MANZANO, 2012).

Características da linguagem de máquina: consistem geralmente de números, qualquer computador entende diretamente a sua própria linguagem de máquina, são dependentes de máquina e são complicadas para a leitura do código (MANZANO, 2012).

2.5.2 Linguagem Assembly (Baixo Nível)

Linguagem Assembly ou Linguagem de Montagem consiste de abreviações de expressões em inglês que são operações elementares, onde se originou a base da

linguagem Assembly. Embora o código seja mais claro para seres humanos, ele é incompreensível para computadores até ser traduzido em linguagem de máquina (MANZANO, 2012).

A conversão da Linguagem Assembly para o código de máquina é feita pelo montador ou Assembler (Figura 7), que é basicamente um tradutor de comandos, sendo mais simples que um compilador. O Assembly é a linguagem de mais baixo nível depois da linguagem de máquina. O montador ou Assembler (não confundir com Assembly) é um programa que cria o código objeto traduzindo as instruções da Linguagem Assembly para código de máquina (MANZANO, 2012).

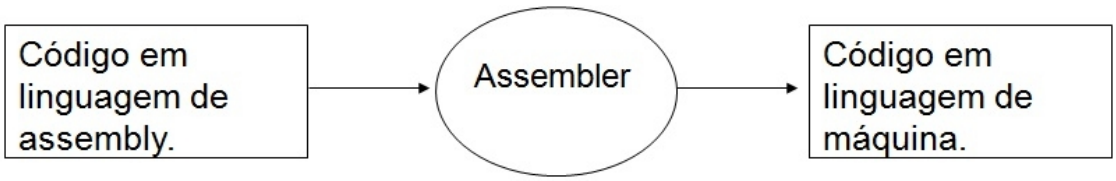

Figura 7: Assembler
Fonte: www.raulparadeda.wordpress.com

2.5.3 Linguagem de Alto Nível

É uma linguagem com um nível de abstração relativamente elevado, longe do código de máquina e mais próximo à linguagem humana. A Linguagem de Alto Nível permite aos programas escrever instruções que se pareçam com o inglês e contêm notações matemáticas comumente utilizadas. As linguagem em C, C++, .NET e o JAVA são exemplos deste tipo linguagem.

Os programas tradutores são conhecidos também pelo nome de compiladores. Os compiladores convertem os programas de linguagem em alto nível em linguagem de máquina. Exemplo de um comando em uma Linguagem de Alto Nível (Java) onde uma variável “nome” recebe o nome de uma pessoa: String nome = “Elaine”;

2.6 ORIENTAÇÃO A OBJETOS

As Linguagens de Programação de Alto Nível utilizam paradigmas[4] de desenvolvimento. Uma delas é a O.O. (Orientação a Objetos – *Object Oriented*).

A O.O. é um paradigma para desenvolvimento de software que baseia-se na utilização de componentes (objetos) que colaboram para construir sistemas mais complexos. A colaboração entre os objetos é feita através de mensagens. Foi uma das tentativas de trazer a programação para um nível de linguagem mais semelhante ao cotidiano (MCLAUGHLIN; POLLICE; WEST, 2008).

A O.O. é um conceito que esta relacionado com a ideia de classificar, organizar e abstrair coisas. O termo O.O. significa organizar o mundo real como uma coleção de objetos que incorporam estrutura de dados e um conjunto de operações que manipulam estes dados (MCLAUGHLIN; POLLICE; WEST, 2008). Os principais conceitos da O.O. são:

- *Classe* - Representa um conjunto de objetos com características similares (*atributos* e *métodos*). Uma classe define o comportamento dos objetos através de seus *métodos*, e quais estados ele é capaz de manter através de seus *atributos*. Uma classe define a estrutura e o comportamento de qualquer objeto da classe, atuando como um padrão para a construção de objetos. Exemplo de classes: Animal, Pessoa, Veiculo...;
- *Objeto / Instância de uma Classe* - Todo objeto é instância de uma classe. Uma classe é um molde para os objetos que serão criados dela. Exemplo: Temos a classe Animal, através dessa classe podemos ter um objeto chamado cachorro, um outro chamado galinha e etc. A mesma coisa com a classe Pessoa, podemos ter o objeto vanessa, camila e etc.

4 Paradigma é um conjunto de regras que estabelecem fronteiras e descrevem como resolver problemas dentro desta fronteira. Um paradigma ajuda-nos a organizar e a coordenar a maneira como olhamos o mundo.

- *Atributo* - São as características de um objeto. Pegando o exemplo do objeto cachorro, podemos ter os seguintes atributos: peso, tamanho, raça entre outros. Já com o objeto vanessa, podemos ter os atributos nome, idade, sexo, data de nascimento entre outros.
- *Método* - Os métodos são as ações que os objetos conseguem realizar. O objeto cachorro pode ter ações como: latir, correr, comer, pegar bolinha e etc. O objeto vanessa pode possuir os métodos: andar, falar, olhar e etc.
- *Mensagem* - É uma chamada a um objeto para invocar um de seus métodos ou até atributos, ativando um comportamento descrito por sua classe. Exemplo: do que adianta o cachorro saber “pegar bolinha” se ninguém chamá-lo para tal coisa? Ou do que adianta o objeto vanessa ter uma data de nascimento mas ninguém informá-la qual é?
- *Herança ou Generalização* - É o mecanismo pelo qual uma classe (sub-classe) pode estender outra classe (super-classe), aproveitando seus comportamentos (métodos) e variáveis possíveis (atributos). Um exemplo de herança: com a super-classe Animal podemos criar sub-classes como Anfíbios, Répteis e Mamíferos. Essas sub-classes herdariam os métodos e atributos de Animal, ou seja, reaproveitaria o que já existe sem criar novamente as mesmas coisas.
- *Abstração* - Habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. Em modelagem O.O., uma classe é uma abstração de entidades existentes no domínio do sistema de software . Por exemplo, imaginamos a abstração referente a classe Animal. Há várias entidades na classe Animal como Anfíbios, Répteis e Mamíferos que são também sub-classes da classe Animal, onde há objetos que contêm cada sub-classe como Ser-humano, Jacaré e outros.

Além destes conceitos citados acima, há outros como Polimorfismo, Encapsulamento e etc. Diversas linguagens de programação de alto nível utilizam o paradigma de O.O. como: C++, C♯, VB.NET, Java, Object Pascal, Objective-C, Python, SuperCollider, Ruby entre outros.

2.7 JAVA

Segundo o site oficial do Java (2012), “Java é uma linguagem de programação orientada a objetos e plataforma computacional lançada pela primeira vez pela Sun Microsystems em 1995”. Em 2009 a Oracle comprou a Sun, e o Java vem sendo mantido pela mesma desde então.

2.7.1 Programação Java

> “Por volta de 1990, existiam alguns problemas quando se programava, sendo algum deles: ponteiros, gerenciamento de memória, organização, falta de bibliotecas, reescrever parte do código ao mudar de sistema operacional, custo financeiro de usar a tecnologia e portabilidade. A linguagem Java resolve bem esses problemas, que até então apareciam com frequência nas outras linguagens. Alguns desses problemas foram particularmente atacados porque uma das grandes motivações para a criação da plataforma Java era de que essa linguagem fosse usada em pequenos dispositivos, como tvs, videocassetes, aspiradores, liquidificadores e outros” (CAELUM, 2004).

2.7.2 Vantagens da Plataforma Java

A tecnologia Java suporta processamento paralelo múltiplo e é grátis, ou seja, custo quase totalmente zero e seus editores e ambientes de produção também são gratuitos (NetBeans, Eclipse, Jcreator e afins). Ainda tem a gratuidade dos Servidores de Aplicação (TomCat, Jboss, Jetty e outros) (CAELUM, 2004).

O mesmo código Java roda em diversas plataformas sem a necessidade de alteração de código e as aplicações podem ser facilmente migradas entre servidores, tornado desnecessário ficar preso a somente um fabricante, além de existirem diversas bibliotecas disponíveis na internet que auxiliam e ajudam os desenvolvedores. Não há a necessidade de se reinventar a roda. Se algo já existe, está disponível o acesso e resolve o seu problema, por que martelar a cabeça para criar outro? (CAELUM, 2004)

Por fim, mas não menos importante, há dezenas de fóruns e blogs onde é possível discutir tudo sobre Java. Exemplos de fóruns: GUJ e Javafree.

2.7.3 Compilação

> "Em uma linguagem de programação como C e Pascal por exemplo, temos a seguinte situação quando vamos compilar um programa (Figura 8). O código fonte é compilado para o código de máquina específico de uma plataforma e sistema operacional. Esse código executável (binário) resultante da compilação será executado pelo sistema operacional e por esse motivo ele deve saber conversar com o S.O. em questão" (CAELUM, 2004).

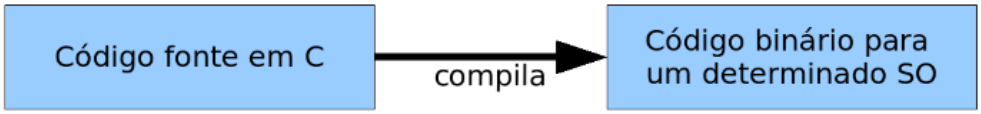

Figura 8: Compilador Código C
Fonte: www.caelum.com.br

> "Com isso temos um código executável, mas para cada S.O., pois será preciso compilar uma vez para Windows, outra para o Linux, e assim por diante para que o software possa ser utilizado em várias plataformas (Figura 9). Esse é o caso de aplicativos como o OpenOffice, Firefox e outros" (CAELUM, 2004).

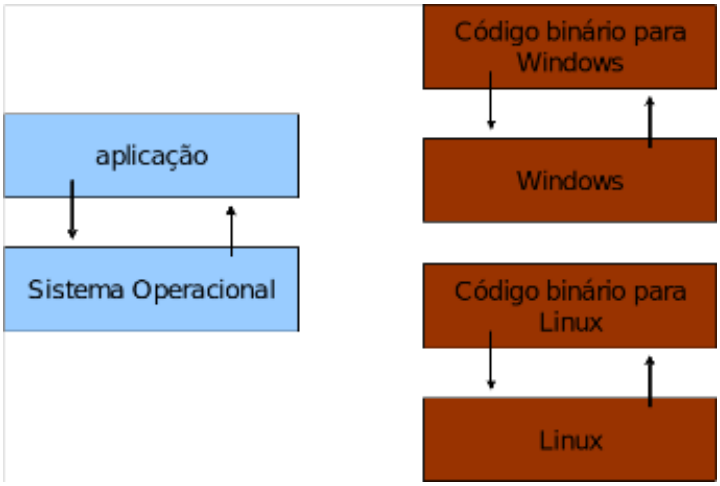

Figura 9: Código Binário Para Plataformas
Fonte: www.caelum.com.br

A aplicação utiliza das bibliotecas do sistema operacional., como a interface gráfica para desenhar as "telas" por exemplo. A biblioteca de interface gráfica do Windows são diferente das do Linux, ou seja, é preciso reescrever uma parte da aplicação para diferentes sistemas. Contudo, a JVM (*Java Virtual Machine* – Máquina Virtual Java) cuida de tudo isso, “traduzindo” a aplicação para o sistema operacional. Então a maneira com a qual você abre uma janela no Linux ou no Windows é a mesma, com isso você ganha independência de plataforma em geral (CAELUM, 2004).

2.7.4 Java Virtual Machine

> Segundo a Caelum (2004), “o Java utiliza do conceito de máquina virtual (JVM), onde existe, entre o sistema operacional e a aplicação, uma camada extra responsável por 'traduzir' o que sua aplicação deseja fazer para as respectivas chamadas do sistema operacional onde ela está rodando no momento”. A Figura 10 demonstra de modo simples como a JVM funciona.

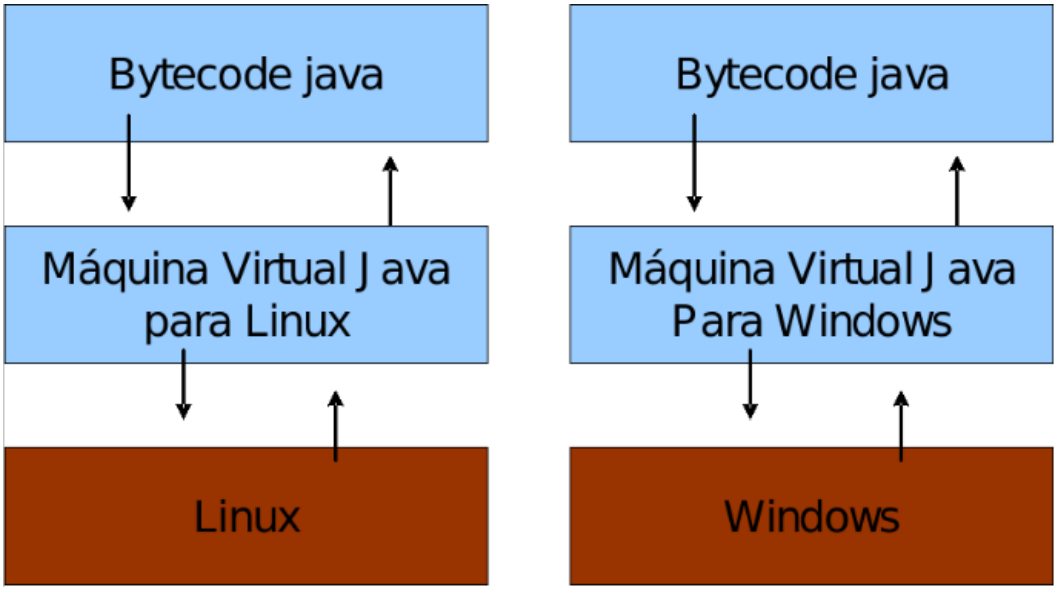

Figura 10: Bytecode e Máquina Virtual Java
Fonte: http://www.caelum.com.br

"Uma máquina virtual é como um "computador de mentira": tem tudo que um computador tem. Em outras palavras, ela é responsável por: gerenciar memória, *Threads*, e etc" (CAELUM, 2004).

> "Aplicação Java roda sem nenhum envolvimento com o sistema operacional. Sempre conversando apenas com a JVM. Essa característica é interessante: como tudo passa pela JVM, ela pode tirar métricas, decidir onde é melhor alocar a memória, entre outros. Uma JVM isola totalmente a aplicação do sistema operacional. Se uma JVM termina abruptamente, só as aplicações que estavam rodando nela irão terminar: isso não afetará outras JVMs que estejam rodando no mesmo computador, nem afetará o sistema operacional. Essa camada de isolamento é interessante quando pensamos em um servidor que não pode se sujeitar a rodar código que possa interferir na boa execução de outras aplicações" (CAELUM, 2004).

2.7.5 Bytecode

Como a JVM não entende código Java, é necessário que o código escrito em Java seja convertido para linguagem de máquina. Esse código de máquina (conhecido por *"bytecode"*) é gerado por um compilador Java (*javac*). "O compilador Java gera esse

bytecode que, diferente das linguagens sem máquina virtual, vai servir para diferentes sistemas operacionais, já que ele vai ser 'traduzido' pela JVM” (CAELUM, 2004).

2.7.6 XStream

XStream é uma biblioteca Java *open source* para serializar objetos para o formato XML, facilitando desta forma a persistência dos dados no próprio arquivo.

Além disso, o XStream facilita o processamento do XML, pois este não precisa necessariamente ser manipulado por outras bibliotecas Java como Sax ou DOM. Essa tarefa poderá ser executada manipulando o objeto carregado a partir do próprio arquivo XML.

Isso acontece porque o próprio XStream se encarrega de desserializar o XML e carregar o objeto, retirando do programador a parte chata do processamento do arquivo, dando mais produtividade, diminuindo substancialmente a geração de erros de programação e ainda por cima contém um ótimo desempenho.

A utilização do XStream no projeto foi na criação dos arquivos XMLs enviados para o Web Service da receita federal. Através da programação Java, faz-se a serialização do objeto carregado com os seus devidos valores. Com o objeto serilializado, basta convertê-lo para uma cadeia de caracteres (tipo String) e enviá-lo para Sefaz através do protocolo SOAP.

2.8 BANCO DE DADOS

> Segundo Susviela (2000, p. 7), “entende-se por banco de dados (ou base de dados) qualquer sistema que reúna e mantenha organizada uma série de informações relacionadas a um determinado assunto em uma determinada ordem. Por exemplo uma lista telefônica. Nela percebemos que todos os dados referentes a uma pessoa estão na mesma linha, a isso chamamos 'registros'**.** O tipo ou

> categoria da informação (nome, telefone e etc) sobre uma pessoa está separada em colunas, as quais chamamos 'campos'".

2.8.1 Utilidade de um Banco de Dados

Um banco de dados permite colocar dados à disposição de usuários para uma consulta, uma inserção, atualização e exclusão. Isso é ainda mais útil quando as informações (dados) são cada vez mais numerosos.

Um banco de dados pode ser local, quer dizer utilizável em uma máquina por um usuário, ou repartida, quer dizer que as informações são armazenadas em máquinas distantes (Figura 11). A vantagem essencial da utilização dos bancos de dados é a possibilidade de poder ser acessada por vários usuários simultaneamente e quantas vezes for necessário (SUSVIELA, 2000).

2.8.2 Gestão de um Banco de Dados

Para que se possa controlar os dados (e também os usuários) é necessário um software chamado SGBD (Sistema de Gestão de Banco de Dados - *Database Managment System*).

O SGBD é um conjunto de serviços (aplicações software) que permitem gerenciar os bancos de dados, quer dizer: permitir o acesso aos dados de maneira simples, autorizar um acesso às informações a múltiplos usuários e manipular os dados presentes no banco de dados (inserção, supressão, modificação e etc.). O SGBD é genérico e pode se referir a qualquer tipo de banco de dados como: orientado a objetos, hierárquico, relacional e etc (SUSVIELA, 2000).

Um SGBDR (Sistema de Gestão de Banco de Dados Relacional - *Database Managment System Relational*) é apenas relativo ao modelo relacional, baseado em tabelas, campos, índices e relacionamento entre as tabelas. Como exemplos de SGBDR temos o Access, ORACLE, MySQL, DB2 da IBM, PostgreSQL, Firebird, SQLServer da Microsoft entre outros (SUSVIELA, 2000).

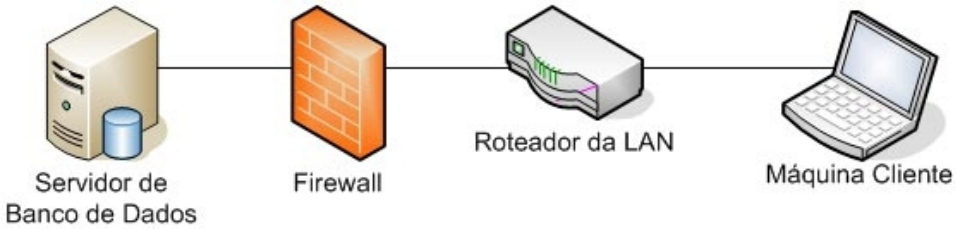

Figura 11: Conexão ao Banco de Dados
Fonte: www.linhadecodigo.com.br

2.8.3 - PostgreSQL

O PostgreSQL é um sistema gerenciador de banco de dados relacional (SGBDR), desenvolvido como projeto software livre. Em termos de software livre, tem como concorrente o MySQL, sendo este bem mais novo que o PostgreSQL e que tem como desvantagem pertencer a uma companhia só (Oracle). O PostgreSQL se comporta melhor quando precisa escalar em hardware multiprocessado em comparação ao MySQL. Exemplo: Às vezes no MySQL, o programador precisa “ensinar” ao banco de dados como se faz uma consulta, o que não ocorre no PostgreSQL (POSTGRESQL, 1996, tradução nossa).

Hoje, o PostgreSQL é um dos SGBDR de código aberto mais avançados, contando com recursos como: consultas complexas, chaves estrangeiras, integridade transacional, controle de concorrência multi-versão, suporte ao modelo híbrido objeto relacional, *trigger*, *views*, *stored procedures* em várias linguagens entre várias outras funcionalidades (POSTGRESQL, 1996, tradução nossa).

2.8.4 – PL/Java

O PL/Java fornece informações que descrevem tanto características da linguagem Java, bem como sua aplicação na linguagem procedural[5] PL/Java, transformando-a em um poderoso artifício no aumento das funcionalidades do SGBD (ORACLE, 2013, tradução nossa).

A linguagem PL/Java é um módulo complementar do *back-end*[6] do PostgreSQL, ou seja, todo o código relacionado é executado no servidor de banco de dados, da mesma forma que as outras linguagens procedurais, como PL/SQL, PL/TCL, PL/Perl, PL/Python e etc. Quando o PL/Java é instalado, funções e *triggers* podem utilizar classes Java, desenvolvidas em qualquer ambiente de desenvolvimento Java, como Eclipse ou NetBeans. Estas classes, por sua vez, são instaladas dentro da base de dados e utilizadas como em qualquer outra linguagem procedural (ORACLE, 2013, tradução nossa).

Resumindo, o PL/Java é uma forma de se programar dentro do banco de dados, com isso obtendo resultados de forma mais rápida e eficiente e pode-se destacar como vantagens de se utilizar essa forma de programação: rodar dentro do banco (acesso mais rápido aos dados) e portabilidade para outros SGBDs como Oracle e DB2.

2.9 - IREPORT

“Há várias maneiras de se criar relatórios para um aplicativo. Para muitos desenvolvedores web, um relatório significa apenas criar uma página web, que produz

5 As instruções dos programas são seguidas sequencialmente e eventualmente desviadas pelos comandos condicionais como “*if”* e de *loop* como “*while”*, “*for”* e etc. Resumindo, com a linguagem/programação procedural trabalha-se com a resolução de problemas através da solução de tarefas básicas as quais se centra o problema.

6 É o Banco e o Armazenamento de dados. MySQL, Oracle, SAP e outros sistemas fazem parte do *Back-end,* que é responsável por todo processamento final dos dados recebidos do *Front-end* (inputs, formulários, requisições, etc).

bons resultados em tela, mas pobres em papel" (JASPERSOFT COMMUNITY, 2000, tradução nossa).

> "Para criar um relatório em PDF (*Portable Document Format* - Formato Portátil de Documento) significa escrever muitos códigos, na verdade, toneladas de códigos, tornando os relatórios difíceis de se manter e até mesmo de escrever. Ao trabalhar com outras tecnologias o problema não muda muito. Por exemplo, o Java fornece uma extensa API (*Application Programming Interface* - Interface de Programação de Aplicativos) para impressão, mas ainda há muito trabalho para se escrever código específico para cada formato de documento. Uma solução para este problema é a utilização de uma biblioteca de relatórios. O JasperReports Library é uma biblioteca Java de código aberto para geração de relatórios (JASPERSOFT COMMUNITY, 2000, tradução nossa)".

O iReport Designer (Figura 12) é um relatório de design visual para o JasperReports . A biblioteca é um mecanismo de relatório que pode ser integrado em seu aplicativo aberto ou comercial para gerar os relatórios projetados com iReport Designer, exibi-las na tela ou exportá-las em um formato final como PDF, OpenOffice, DOCX e muitos outros. Alternativamente, você pode transmitir o resultado através de uma aplicação web ou enviar o documento final diretamente a uma impressora. O JasperReports é o núcleo do iReport Designer (JASPERSOFT COMMUNITY, 2000, tradução nossa).

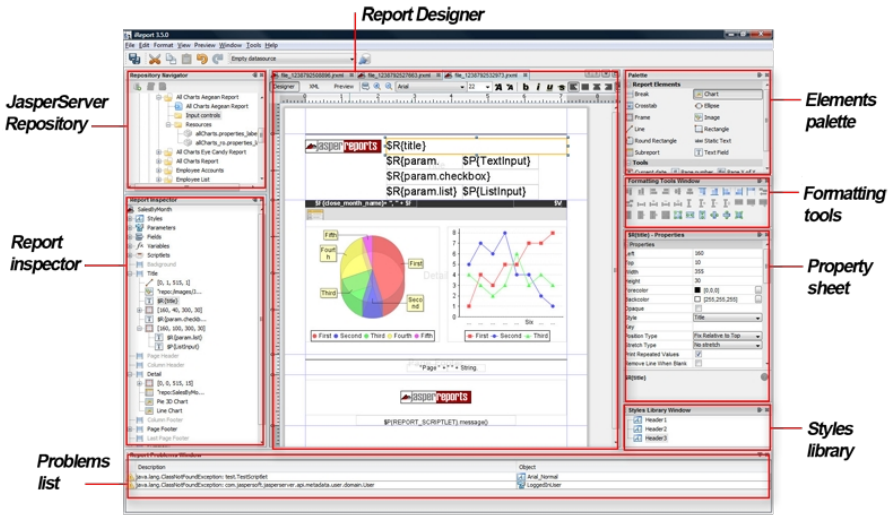

Figura 12: iReport Designer
Fonte: www.community.jaspersoft.com

2.9.1 Ciclo de vida do Report

> “Quando se cria um relatório com iReport Designer você está criando um arquivo JRXML <exemplo.jrxml>, que é um documento XML que contém a definição do *layout* do relatório. O layout é completamente desenhado de uma forma visual, de modo que pode-se ignorar a estrutura real do arquivo JRXML. Antes de executar um relatório, o JRXML deve ser compilado (Figura 13) em um objeto binário chamado arquivo Jasper <exemplo.jasper>. Esta compilação é feito por razões de desempenho e performance” (JASPERSOFT COMMUNITY, 2000, tradução nossa).

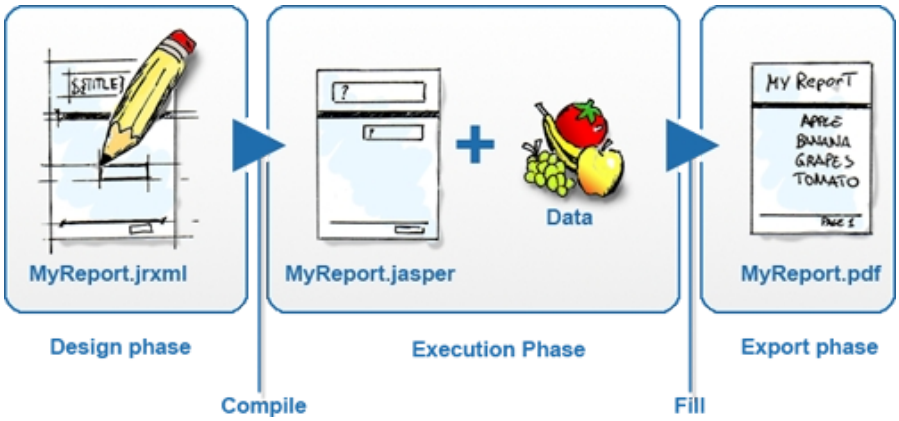

Figura 13: Compilação/Execução Report
Fonte: www.community.jaspersoft.com

“Uma aplicação necessita do arquivo Jasper e uma fonte de dados para o JasperReports a fim de executar os relatórios” (JASPERSOFT COMMUNITY, 2000, tradução nossa).

2.9.2 Flexibilidade

Com o arquivo JASPER compilado, basta preenche-lo com as informações necessárias para que o relatório saia como desejado.

> “Há diversos tipos de fonte de dados, é possível preencher um arquivo Jasper a partir de uma consulta SQL (*Structured Query Language* - Linguagem de Consulta Estruturada), um arquivo XML, um arquivo CSV (*Comma-Separated Values*), uma

> consulta HQL (*Hibernate Query Language*), uma coleção de *JavaBeans* entre outros" (JASPERSOFT COMMUNITY, 2000, tradução nossa).

2.10 ECLIPSE

Eclipse é um IDE (*Integrated Development Environment* - Ambiente Integrado de Desenvolvimento) para desenvolvimento Java, porém suporta várias outras linguagens a partir de *plugins* como C, C++, PHP, ColdFusion, Python, Scala e plataforma Android. Ele foi feito em Java e segue o modelo *open source* de desenvolvimento de software. (ECLIPSE, 2014, tradução nossa).

Hoje, o Eclipse é o IDE Java mais utilizado no mundo. Possui como características marcantes o uso da *SWT* e não do *Swing* como biblioteca gráfica, a forte orientação ao desenvolvimento baseado em *plugins* e o amplo suporte ao desenvolvedor com centenas de *plugins* que procuram atender as diferentes necessidades de diferentes programadores (ECLIPSE, 2014, tradução nossa).

Todo sistema, programa, software, seja lá como é chamado, começou como um projeto (independente da linguagem de programação utilizada). Este projeto precisa ser desenvolvido em cima de uma ferramenta: como um IDE, um editor de texto comum e etc. O ADempiere (além de ser um sistema de gestão empresarial) é um projeto desenvolvido em Eclipse e em linguagem Java. Em outras palavras, para se criar novas funções ou alterar as que já existem do ADempiere (via código), é necessário ter a ferramenta IDE Eclipse para que se possa abrir o código fonte do projeto.

2.10.1 Controle de Versão

Um sistema de controle de versões é um software que funciona utilizando o conceito de repositório. O repositório é um arquivo (ou local, dependendo do sistema

utilizado) onde todo o histórico de evolução do projeto é armazenado, para cada item que é monitorado pelo sistema.

Um projeto, sem um programa de controle de versão, ficará salvo localmente na máquina do usuário. O grande problema neste caso é na hora de compartilhar o código desenvolvido com outras pessoas.

Há empresa que utilizam do compartilhamento de pasta pública para desenvolver projetos. Em caso de pequenos projetos pode funcionar, mas em sistemas grandes (como um ERP por exemplo) isso raramente seria usado. A solução é utilizar uma ferramenta que controle as versões do programa, permitindo que todos os desenvolvedores sempre tenham a versão mais atual do projeto sem nenhum problema.

Os programas de controle de versão funcionam da seguinte maneira: o desenvolvedor não trabalha diretamente no repositório. Ao invés disso ele usa uma área que possui uma cópia de todos os arquivos que são monitorados pelo sistema de controle de versões (Figura 14). Nesta área, que podemos chamar de "área de trabalho", é que fazemos as alterações nos arquivos e é nesta mesma área que o sistema monitorará os arquivos para verificar alterações (DIAS, 2011).

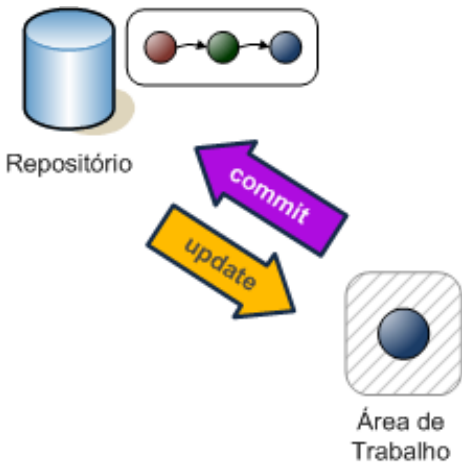

Figura 14: Controle de versão
Fonte: Autor

O monitoramento do sistema de controle de versões se dá através de duas operações básicas: *commit* e *update*. O *update* é feito para atualizar a área de trabalho com a versão mais recente dos arquivos do repositório enquanto o *commit* é feito para enviar as alterações na área de trabalho para o repositório. A sintaxe de uso dessas duas operações varia de sistema para sistema, mas a ideia é essencialmente esta. Outras operações como carregar uma versão anterior de um arquivo ou verificar as alterações em um arquivo podem estar disponíveis (DIAS, 2011).

Uma ferramenta de controle de versão é muito útil, em diversos aspectos, tanto para projetos pessoais pequenos e simples como também para grandes projetos comerciais. Entre os mais comuns encontram-se: *Soluções livres* **-** CVS, Mercurial, GIT e SVN; Soluções comerciais - SourceSafe, PVCS (Serena) e ClearCase.

2.10.2 BitBucket

BitBucket é um serviço de hospedagem de projetos controlados através do Mercurial e GIT. É similar ao GitHub (que utiliza GIT, somente). Possui serviço gratuito e comercial, e com ele é possível hospedar projetos para uso publico ou privado. O BitBucket permite hospedar projetos de diferentes plataformas como Java, C, HTML/CSS entre outros.

Pegaremos como exemplo o projeto ADempiere, e imaginaremos que este já foi importando para o Eclipse, ou seja, o código fonte já está no IDE. O interessante é deixar esse projeto em algum repositório e ir controlando sua versão conforme as modificações. Para isso, primeiro é preciso criar uma conta em um serviço de hospedagem como o BitBucket por exemplo. Após, é necessário criar um repositório dentro dessa conta criada, pois é através desse repositório criado onde será enviado e salvo o projeto sempre que o usuário desejar.

Em seguida, deve-se baixar um programa de controle de versão (GIT ou Mercurial por exemplo) e configurar no IDE para que o projeto seja enviado ao repositório a partir do software de controle. A configuração do repositório para envio normalmente exige o URL (endereço do serviço de hospedagem, ex: https://bitbucket.org/usuario/nomeProjeto), usuário e senha cadastrados no site.

Por fim, basta enviar o projeto (fazer *commit*) para o repositório toda vez que alguma alteração for realizada. A cada *commit*, uma nova versão é feita, e o melhor de tudo isso é: a possibilidade de voltar a alguma versão anterior e/ou pegar a mais atual sempre que estiver disponível. Resumindo, o BitBubket é um repositório para projetos de softwares que utilizam como ferramentas de controle de versão o Mercurial ou GIT.

2.11 ERP

Um ERP (*Enterprise Resource Planning* - Planejamento dos Recursos Empresariais) é um software que integra todos os dados e processos de uma organização, pois é uma plataforma desenvolvida para integrar os diversos departamentos de uma empresa possibilitando a automação e armazenamento de todas as informações de negócios, além de possibilitar um fluxo de informações único, contínuo e consistente (ALECRIM, 2013).

> “Imagine que você tenha uma empresa que conta com vários sistemas, um para lidar com as contas a pagar, um para gerar folhas de pagamento, um para controlar vendas, um para gerenciar impostos, um para analisar metas e desempenho, entre outros. Em vez de existir um ou mais softwares isolados para cada departamento da companhia, o ERP integra todos os departamentos/setores em um único sistema, com isso a empresa passa a ter menos fornecedores de software, o que diminui custos com licenças, suporte técnico, servidores, treinamento, entre outros” (ALECRIM, 2013).

Integrar todos os departamentos, ou pelo menos os mais importantes, torna a comunicação interna menos custosa e fácil, essa é a função do ERP. (ALECRIM, 2013).

Figura 15: ERP
Fonte: www.ammc.com.br

> Segundo Miltello (1999), “o ERP controla a empresa, manuseando e processando suas informações. Todos os processos são documentados e contabilizados, gerando regras de negócio bem definidas e permitindo maior controle sobre alguns pontos vulneráveis do negócio, como a administração de custos, controle fiscal e estoques. A adoção desses sistemas põe fim aos vários sistemas que funcionavam de forma isolada na empresa, com informações redundantes e não confiáveis”.

2.11.1 Funcionalidade

> “Cada módulo contempla uma área da empresa e sua integração permite entender os processos que evolvem a operacionalidade do negócio, servindo de apoio à tomada de decisões de todos os setores e quebrando barreiras impostas pelas estruturas departamentais” (GASPAR, 2012).

Cada setor “conversa amigavelmente” com outro, pois não há sistema isolado/separado e os dados são acessíveis a todos, não havendo mais problema de passar uma informação a um departamento para ser processado por um software específico, correndo o risco dos dados chegarem corrompidos ou até mesmo errados.

2.11.2 Mercado e Customização

Os sistemas ERP encontrados no mercado são compostos por uma estrutura básica *default* que permite ser customizada em função das necessidades das empresas. O custo e a rapidez com que essas customizações e parametrizações podem ser desenvolvidas e implementadas criam um grande diferencial entre as empresas desenvolvedoras (GASPAR, 2012).

2.11.3 Implementação de Sistemas ERP

> "Cada empresa, em face de suas atividades e de suas estratégias operacionais, possui necessidades distintas das outras, portanto, sistemas ERP só serão funcionais se ao menos as características mais importantes da companhia forem levadas em conta" (ALECRIM, 2013).

Um sistema que trabalha com transporte de mercadorias tem necessidades diferentes de outra que trabalha com vendas. Por isso, um ERP não é algo único que serve para todos os negócios, assim como não é algo que pode ser baixado da internet, instalado e está pronto para ser utilizado! Enfim, como é possível perceber, cada companhia precisa contar com um sistema de gestão que se adapte a ela (ALECRIM, 2013).

É muito importante à empresa analisar as soluções de ERP existentes no mercado. A empresa cliente (essencialmente ou não) pode contar com uma equipe de T.I. (Tecnologia da Informação) capaz de realizar a análise da companhia, senão, procurar por uma consultoria (ALECRIM, 2013).

2.11.4 Tempo de Implantação

> “Sistemas ERP não começam a funcionar da noite para o dia. Os provedores das soluções precisam de tempo para adaptar o software às atividades da empresa, sem contar que necessitam considerar a infraestrutura, os recursos de segurança, testes, treinamento de pessoal, integração entre departamentos, migração de sistemas entre outros e etc. Além disso, a implementação geralmente ocorre por etapas, de forma que determinados módulos do sistema sejam instalados somente depois de este processo já ter ocorrido com outros. Portanto, a implementação de um ERP pode consumir vários meses” (ALECRIM, 2013).

2.11.5 Vantagens e Desvantagens do ERP

Pode-se destacar as seguintes vantagens do ERP: ajudar na comunicação interna, agilizar a execução de processos internos, diminuir a quantidade de processos internos, evitar erros humanos (cálculos de tributos e pagamentos, por exemplo), ajudar na tomada de decisões, auxiliar na elaboração de estratégias operacionais, diminuir o tempo de entrega do produto ou serviço ao cliente, ajudar a lidar com grandes volumes de informação, evitar trabalho duplicado, fazer com que a empresa se adapte melhor a mudanças no mercado entre outros (ALECRIM, 2013).

Em um sistema ERP, pode destacar as seguintes desvantagens: alto custo com customização e implementação; implementação demorada (uma solução de ERP não fica pronta da noite para o dia), risco de prejuízo financeiro ou de desempenho com erros inesperados do sistema, possíveis problemas com suporte e manutenção caso o fornecedor do software seja vendido ou encerre suas atividades, dependência (dificulta as as atividades da empresa quando o sistema fica, por algum motivo, indisponível), adaptação e treinamento dos funcionários pode demorar mais tempo que o esperado, resistência ao novo, com as atualizações e acréscimos de módulos o sistema excessivamente ficará mais complexo entre outros (ALECRIM, 2013).

3 ADEMPIERE

> Segundo o site do ADEMPIERELBR (2014), “o ADempiere é um software livre de Gestão Empresarial do tipo ERP, ou seja, ele permite integrar informações de diversos departamentos de uma empresa, podendo ser implantado em empresas de todos os tamanhos”.

O ADempiere é um software O*pen Source* (código livre) desenvolvido em linguagem Java, e esta é uma das suas grandes vantagens: seu uso é gratuito. Utiliza o PostgreSQL ou o Oracle como SGBDR. Dependendo da versão do PostgreSQL, é necessário instalar o PL/Java para o funcionamento do software.

A Figura 16 demostra os principais recursos que o ADempiere traz nativamente. Como pode ser analisado ele cobre os principais setores: Administração Financeira, CRM (Gestão de Clientes e Fornecedores), Logística (Relatórios, Análise de Performance), Recursos Humanos e uma parte para Configurações.

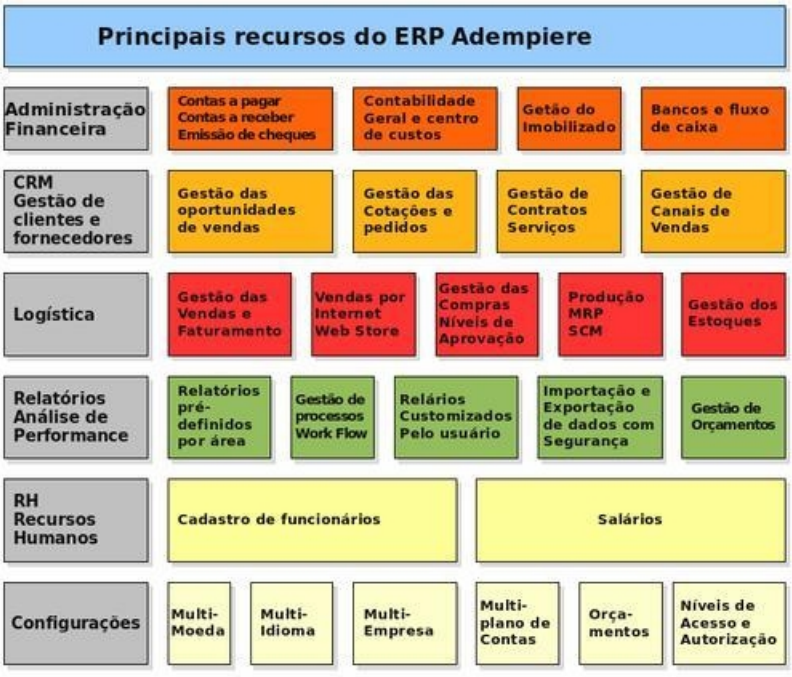

Figura 16: Recursos Adempiere
Fonte: www.adempierelbr.wikispaces.com

> “Segundo Paula (2013), no ADempiere a organização é feita através de processos, e não através de módulos, como é comum na maioria das outras soluções de ERP. O sistema é uma ferramenta única, totalmente integrada, que fornece uma visão única e centralizada do negócio, permitindo a construção de indicadores de desempenho” como na Figura 17.

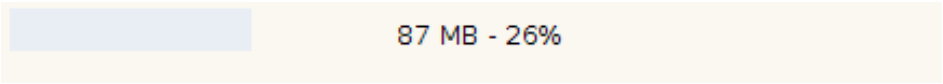

Figura 17: Indicador de Desempenho
Fonte: Autor

3.1 PONTOS POSITIVOS E NEGATIVOS

Apesar do ADempiere ser um excelente *framework* e software de gestão empresarial, alguns pontos positivos e negativos podem ser ressaltados sendo eles:

- **(Ponto Positivo)** - “Possui uma série de funcionalidades que envolvem todos os processos de uma empresa, que podem ser facilmente personalizados para uma melhor aderência ao detalhes específicos de cada empresa” (PAULA, 2013).
- **(Ponto Positivo) -** Possui um completo projeto de localização para a legislação brasileira, chamado ADempiereLBR, o qual contempla itens como NF-e, SPED, GIA, Boleto Bancário e outros itens fundamentais para empresas interessadas em utilizar um ERP *Open Source* no Brasil (PAULA, 2013).
- **(Ponto Negativo) - “**Para usuários iniciantes a interface pode não ser muito “amigável” e pouco intuitiva” (PAULA, 2013).

3.2 FUNCIONAMENTO

> “O ADempiere é um sistema baseado na arquitetura cliente-servidor. Um computador irá atuar como servidor do programa e outros computadores atuarão como clientes do sistema. Os usuários incluem, alteram e consultam as informações através dos clientes. Por ser escrito em Java, um servidor ADempiere

> pode rodar em Windows, Linux e Mac OS X. As informações são armazenadas e processadas no servidor, que, por sua vez, pode atender a muitos clientes ao mesmo tempo. O computador servidor também pode ser usado como cliente em pequenas empresas" (ADEMPIERELBR, 2014).

3.3 FERRAMENTAS DE INSTALAÇÃO NO SERVIDOR

> "O servidor ADempiere requer a instalação do servidor de banco de dados PostgreSQL (ou Oracle), da linguagem Java, e do módulo PL/Java, que permite à linguagem Java acessar o banco de dados PostgreSQL. A execução do sistema ADempiere se dá através do servidor de aplicação JBoss, que é instalado junto com o ADempiere" (ADEMPIERELBR, 2014).

Caso desejar, a customização LBR (configuração do ADempiere para as necessidades brasileiras) é implementada ao final da instalação, depois de todos os componentes de software descritos já estarem instalados. Obs.: Não é necessário o PLJAVA se a versão do PostgreSQL for a 8.4 ou superior (ADEMPIERELBR, 2014).

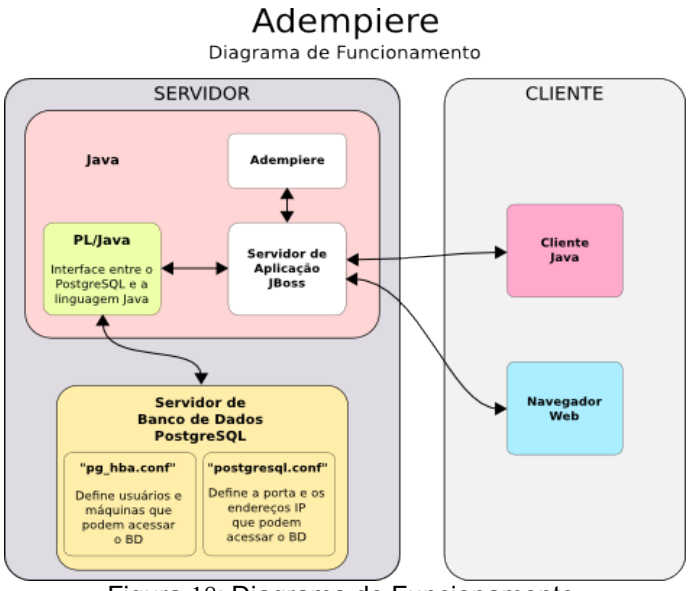

Figura 18: Diagrama de Funcionamento
Fonte: www.adempierelbr.wikispaces.com

3.4 REQUISITOS PARA INSTALAÇÃO DO ADEMPIERE

A instalação do ADempiere requer os seguintes componentes/requisitos: Java, PostgreSQL 8.3 ou mais novo, PL/Java (não é necessário se a versão do PostgreSQL for a 8.4 ou superior), e do Software ERP Adempiere.

3.4.1 Instalação Passo a Passo no Ubuntu

A instalação a seguir se dará no sistema operacional Ubuntu (independente da versão). Primeiramente, deve-se baixar o JKD 6, PostgreSQL 8.4 e o ERP ADempiere 4.3.2 ou 3.6.0. Lembrando que não será necessário baixar nem instalar o PL/Java. Após baixar as ferramentas necessárias, deixe os programas baixados em uma pasta no *home* ou onde achar melhor.

3.4.1.1 Configuração e Instalação do JDK

Pela console (terminal), localize a pasta onde está os arquivos baixados e após mover o JDK para pasta *opt* (com o comando: *sudo cp jdk-6u45-linux-x64.bin /opt*), faça os seguintes passos:

1. Ir no diretório *opt*: *cd /opt*
2. Deixe o JDK executável : *sudo chmod +x jdk-6u45-linux-x64.bin*
3. Execute o JDK: *sudo ./jdk-6u45-linux-x64.bin*
4. Segure ENTER até surgir a mensagem: *Do you agree to the above license terms? [yes or no]*
5. Digite yes e tecle ENTER para o JDK ser instalado.

3.4.1.2 Configuração e Instalação do PostgreSQL

Pela console (terminal), localize a pasta onde está os arquivos baixados e deixe o programa do PostgreSQL executável (com o comando: *sudo chmod +x postgresql-8.4.17-1-linux-x64.run*), em seguida faça os seguintes passos:

1. Execute o PostgreSQL: *sudo ./postgresql-8.4.17-1-linux-x64.run*
2. Na instalação coloque - Senha: *postgres*. Confirmar senha: *postgres*.
3. O diretório que ele sugerir para salvar será o *opt* assim como o *bin*. Deixe como está e apenas siga a instalação. Ao terminar clique para finalizar desmarcando o *CheckBox*.

3.4.1.3 Configuração das Variáveis de Ambiente

É necessário configurar duas variáveis para o funcionamento do ADempiere. Pela console (terminal), vá ao diretório *etc* (comando: *cd /etc*) e faça o seguinte processo:

1. Execute o comando de edição de arquivo: *sudo nano /etc/environment*
2. Quando o arquivo se abrir na console, em *PATH* coloque o endereço do *bin* do JDK e do PostgreSQL: */opt/PostgreSQL/8.4/bin:/opt/jdk1.6.0_45/bin:*
3. Lembrar que cada diretório deve ser colocado após dois ponto no último diretório inserido.
4. Em seguida, adicionar as seguintes linhas no arquivo:
    1. *JAVA_HOME=”/opt/jdk1.6.0_45/”*
    2. *ADEMPIERE_HOME="/Adempiere"*
5. Aperta Ctrl + X e depois Y ou S para salvar. Tecle Ctrl + M para sair.

3.4.1.4 Configurando Permissões de Acesso no PostgreSQL

É necessário configurar algumas permissões de acessos do PostgreSQL para que este inicie junto com o S.O. como serviço, então pela console (terminal), execute o comando "*sudo passwd postgres",* e depois faça os seguintes passos:

1. Coloque a senha de *root* do sistema operacional, em seguida tecle ENTER. Após será solicitada a nova senha UNIX. Informe da seguinte forma: Senha: *postgres*. Confirmar senha: *postgres*
2. Em seguida, execute: *su – postgres*
3. Coloque a senha do PostgreSQL para executar comandos pela console como usuário PostgreSQL.
4. Após, entre no diretório data do PostgreSQL: *cd /opt/PostgreSQL/8.4/data*
5. Execute o comando de edição de arquivo: *nano pg_hba.conf*
6. Após deixar o arquivo como na Figura 19, aperte Ctrl + X e depois Y ou S para salvar. Tecle Ctrl + M para sair.

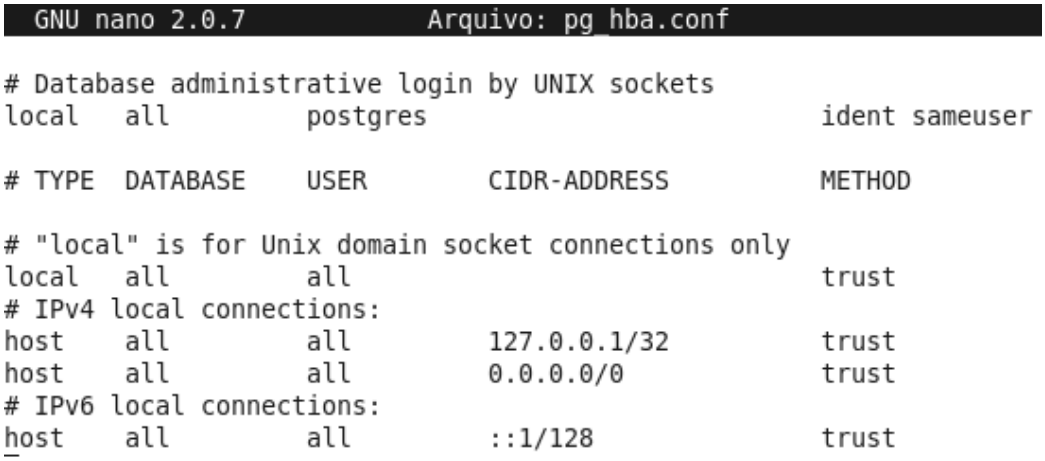

Figura 19: Alteração do Arquivo pg_hba.conf
Fonte: Autor

7. Dê um *exit* no terminal: *exit*
8. Agora logado como usuário comum do sistema, execute: *sudo nano /etc/ld.so.conf*
9. Insira as seguintes linhas:
    1. */opt/jdk1.6.0_45/jre/lib/amd64*
    2. */opt/jdk1.6.0_45/jre/lib/amd64/native_threads*
    3. */opt/jdk1.6.0_45/jre/lib/amd64/server*
10. Aperte Ctrl + X e depois Y ou S para salvar. Tecle Ctrl + M para sair.
11. Em seguida, execute os seguintes comandos:
    1. *sudo ldconfig*
    2. *cd /etc/init.d*
12. Reinicie o PostgreSQL: *sudo ./postgresql-8.4 restart*

3.4.1.5 Servidor e Role de Permissão no pgAdminIII

Quando o PostgreSQL é instalado na máquina, um SGBDR chamado pgAdminIII é instalado junto. Localize o pgAdminIII e abra-o.

Em seguida, clique duas vezes em PostgreSQL 8.4 (*localhost*) e digite a senha de instalação do postgres (no caso postgres) e marque para gravar a senha.

Na tela seguinte, marque para não surgir mais a mensagem e siga em frente clicando em “Adicionar Conexão a um Servidor” como na Figura 20.

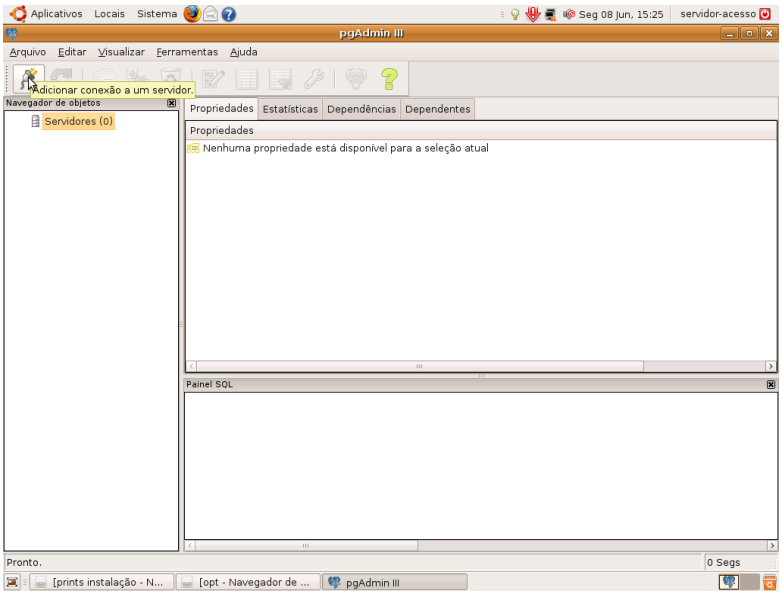

Figura 20: Adicionando Nova Conexão de Servidor
Fonte: Autor

Após clicar em “Adicionar Conexão a um Servidor”, deixe as configurações como está na Figura 21 (a senha é postgres) e aperte o botão OK.

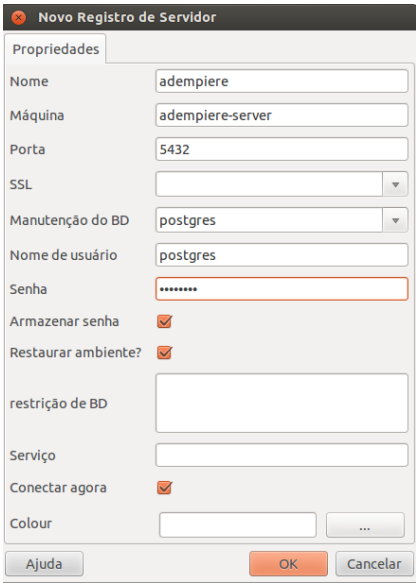

Figura 21: Configuração do Servidor
Fonte: Autor

É preciso criar o Role Adempiere, então clique com o direito em cima de Login Role e aperte “*New Login Role*” (Figura 22). Preencha as informações na janela que se abrir com: Nome do Role: adempiere, Senha: adempiere. Na aba “*Role privileges*”, habilite todos os privilégios clicando em todos os *CheckBox* e aperte o botão OK.

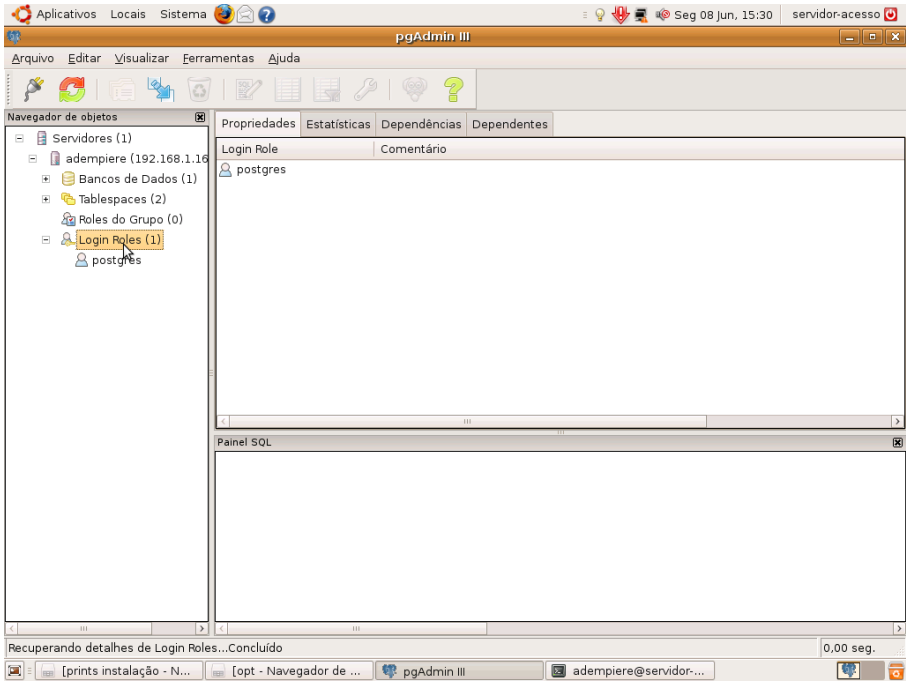

Figura 22: Criando Login Role
Fonte: Autor

3.4.1.6 Criando Base de Dados e Host

É necessário criar um banco de dados chamado adempiere, pois este será utilizado quando for preciso levantar a base de mesmo nome através de um programa do próprio ADempiere.

Para isso, clique com o botão direito do mouse em Banco de Dados e selecione a opção “Novo Banco de Dados”. Preencha os campos como na imagem Figura 23.

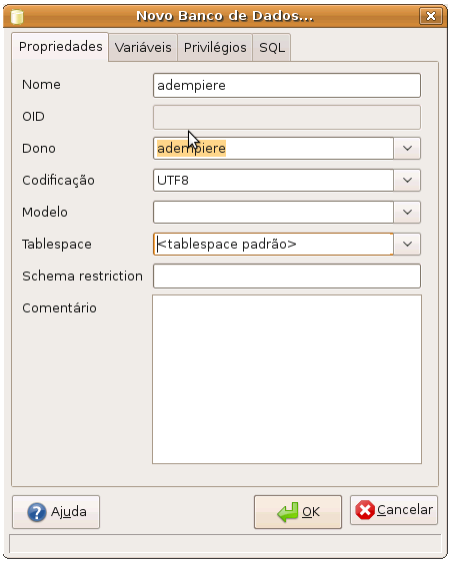

Figura 23: Novo Banco de Dados
Fonte: Autor

Abra a console (terminal), execute o comando "*sudo nano /etc/hosts"* e faça os seguintes passos:

1. Insira a seguir a seguinte linha abaixo do último IP mostrado no arquivo: *127.0.0.1 adempiere-server*
2. Aperte Ctrl + X e depois Y ou S para salvar. Tecle Ctrl+M para sair.

3.4.1.7 Início da Instalação do ADempiere

Nesta etapa sera iniciada a instalação do ERP ADempiere. De início, extraia o adempiere do arquivo zipado que foi baixado, renomeia-o para "Adempiere" e mova a pasta para o *opt* através da console (terminal) com o comando "*sudo mv Adempiere/ /opt/"* e faça os seguintes passos:

1. Entre na pasta Adempiere que foi movida para o *opt*: *cd /opt/Adempiere*
2. Caso tenha problemas em acessar a pasta por permissões do *root*, execute: *sudo chmod 777 /opt/Adempiere/*

3. Deixe todos os arquivos .sh executáveis: *sudo chmod +x *.sh*
4. Crie um link apontando para o Adempiere localizado no *opt* com o comando: *sudo ln -s /opt/Adempiere/ /Adempiere*
5. Um link será criado no diretório "/" (barra) do sistema operacional. Isso é feito para facilitar o acesso ao diretório do Adempiere. O comando pode ser traduzido como: *sudo ln -s nomeDoDiretorio nomeDoLink*
6. Com o link criado acesse o diretório Adempiere: *cd /Adempiere*
7. Em seguida, execute o arquivo *RUN_setup.sh* que irá inicializar a instalação com comando: *sudo ./RUN_setup.sh*
8. Deixe todas as configurações como na Figura 24 (a senha é adempiere)

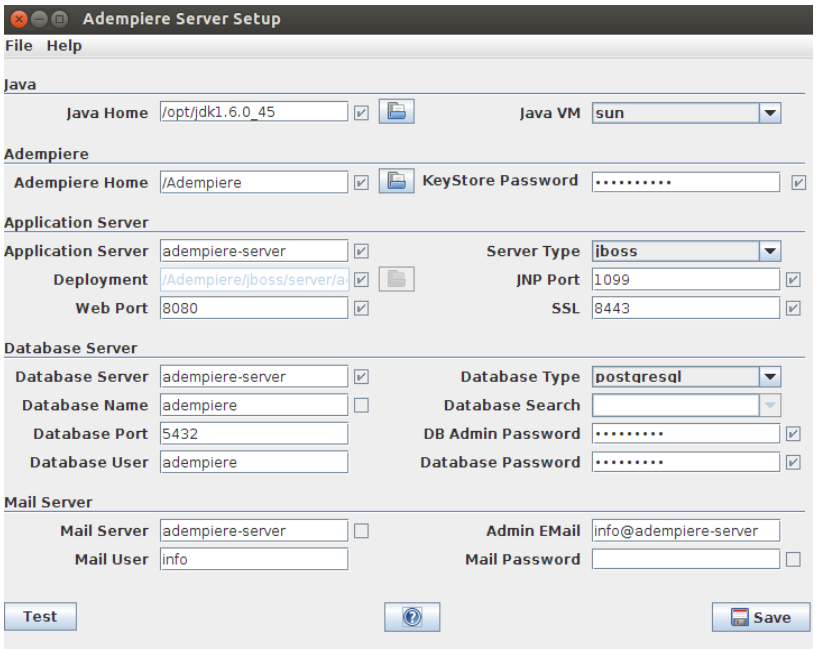

Figura 24: Adempiere Server Setup
Fonte: Autor

Assim que preencher os campos como na Figura 24, clique em "*Test*". As informações inseridas agora serão validadas e vários ícones de "certinho" vão sendo marcados no lado de cada campo. Quando terminar a validação, clique em "*Save*".

Na tela seguinte aceite os termos e dê OK na próxima mensagem que se abrir. Com isso a instalação será iniciada.

É normal que uma console (terminal) seja aberta demonstrando a evolução da instalação. Não deve-se fechar a console até que a instalação seja concluída.

3.4.1.7 Importando Base de Dados

Ao final da instalação, feche o terminal e abra-o novamente. Dê acesso ao usuário comum do sistema ao diretório Adempiere localizado no *opt* com o comando "*sudo chown usuario diretorioDoAdempiere*" (ex: sudo chown elaine /opt/Adempiere/ -R) e em seguida faça os seguintes processos:

1. Entre no diretório utils do Adempiere: *cd /Adempiere/utils/*
2. Execute o arquivo *Run_ImportAdempiere.sh* para importar a base dados com o comando "*./RUN_ImportAdempiere.sh"*. Lembrar que é preciso ter criado a base adempiere para que funcione corretamente o processo. Aperte ENTER e espere o processo de importação da base de dados finalizar.

3.4.1.9 Execução e Configuração

Ao chegar aqui, significa que todos os passos anteriores foram executados com sucesso e para finalizar a instalação, abra a console e vá no diretório principal do Ademperie com o comando "*cd /opt/Adempiere"* e faça os seguintes processos:

1. Execute o arquivo RUN_Adempiere.sh com o comando: *./RUN_Adempiere.sh*
2. A tela de login aparecerá como na Figura 25.

Figura 25: ADempiere Login
Fonte: Autor

É preciso configurar a base de dados no ADempiere. A base foi levantada mas o software não sabe que base é essa ou se existe, então para isso, clique no campo do Servidor para que uma tela se abra como a de Figura 26.

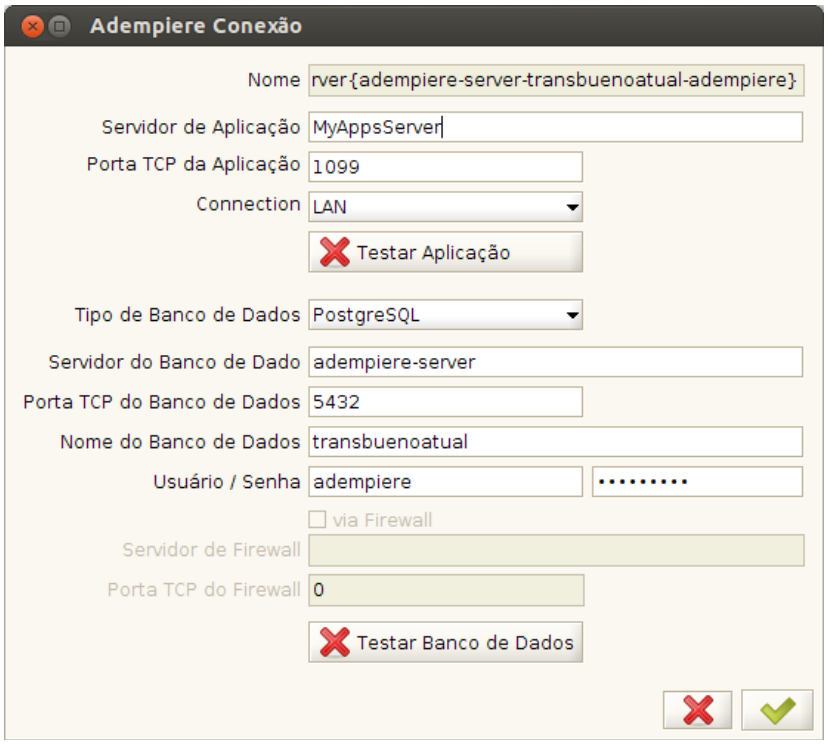

Figura 26: ADempiere Conexão
Fonte: Autor

No campo “Tipo de Banco de Dados” escolha a opção PostgresSQL. No campo “Servidor de Banco de Dados” coloque o nome da máquina (no caso “adempiere-server” configurado no *host*) e no campo “Nome do Banco de Dados” coloque “adempiere”.

Para finalizar, clique no botão “Testar Banco de Dados”, se o teste for executado com sucesso, será mudado o ícone de “X” para um “Certinho”, o que quer dizer que o ADempiere foi instalado com sucesso e está pronto para ser utilizado.

3.4.2 Requisitos Mínimos do Computador

Os Requisitos do sistema ADempiere varia consideravelmente com a complexidade do empreendimento a ser apoiado.

No lado do servidor, as configurações e ferramentas mínimas básicas para o funcionamento do ADempiere são:

- Sistemas Operacionais - *Windows*: 2000, XP *, Vista ou Windows 7; *Linux*: Suse*, Red Hat *, CentOS, Debian / Ubuntu ou FreeBSD; *Unix*: OpenSolaris; *MAC:* OSX.
- Hardware (processadores) - Intel, AMD Opteron, Sparc, Power ou Itanium.
- Banco de Dados (Databases) - Oracle 10g release 2 (Express, Standard e Enterprise Editions), PostgreSQL 8.3 (ou superior).
- Stack Obrigatório - Java 2 Plataform Standard Edition 5.0 ou superior, Jboss e Apache-Ant 1.6 ou superior.
- Tecnologias utilizadas - JSP, Servlets, EJB, SQL/SQLJ, XML, HTML/CSS e PDF.

No lado do cliente, as configurações e ferramentas mínimas básicas para o funcionamento do ADempiere são:

- Aplicação WEB (*Browsers*) - Firefox 2.0 (ou superior), Google Chrome, Safari 2 (ou superior), Internet Explorer 7.0 (ou superior) ou Java 2 Plataform S. E. 5.0 (ou superior).
- Para a aplicação *Desktop* - Java 2 Plataform Standard Edition 5.0 ou superior.
- Sistemas Operacionais - *Microsoft Windows:* 2000, XP *, Vista; *Linux:* Suse *, Red Hat *, CentOS, Debian / Ubuntu, FreeBSD; *Unix:* OpenSolaris; *MAC:* OSX

3.4.3 Backup e Restore da Base de Dados

O ADempiere fornece um arquivo de *script* para realizar a exportação do banco de dados . A exportação apenas será realizada caso a base tenha sido levantada pelo menos uma vez e o nome dela for adempiere. O programa se localiza no diretório *utils* do ADempiere e se chama *RUN_DBExport* (*sh para Linux ou *bat para Windows).

Ao executar o *script* pela console (terminal) será criado dois arquivos no mesmo diretório *utils*: um com extensão *jar e outro com extensão *dmp. No caso, o nome do arquivo de backup será nomeado de ExpDat automaticamente.

Além do backup com o *RUN_DBExport*, o ADempiere também traz outro *script* chamado *RUN_DBRestore* (*sh para Linux ou *bat para Windows) que faz a restauração de um banco de dados

Para a restauração de uma base de dados, é necessário que um arquivo de backup com extensão *dmp esteja no diretório *data* do ADempiere. Com o arquivo no diretório *data*, basta executar o programa *RUN_DBRestore* através da console.

O banco de dados adempiere no PostgreSQL será apagado e recriado com a nova base colocada no diretório *data*. O nome do arquivo de backup deve ser o original, ou seja, ExpDat.dmp.

4 MANIFESTO ELETRÔNICO DE DOCUMENTOS FISCAIS

> Segundo o site oficial do Manifesto Eletrônico (2013), "o MDF-e tem como objetivo a implantação de um modelo nacional de documento fiscal eletrônico que venha substituir a sistemática atual de emissão do documento em papel, com validade jurídica garantida pela assinatura digital do emitente, simplificando as obrigações acessórias dos contribuintes e permitindo, ao mesmo tempo, o acompanhamento em tempo real das operações comerciais pelo Fisco".

4.1 OBRIGATORIEDADE E FINALIDADE DO MANIFESTO

> "O MDF-e deverá ser emitido por empresas prestadoras de serviço de transporte para prestações com mais de um conhecimento de transporte ou pelas demais empresas nas operações, cujo transporte seja realizado em veículos próprios, arrendados, ou mediante contratação de transportador autônomo de cargas, com mais de uma nota fiscal" (MDFE, 2013).

"A finalidade do MDF-e é agilizar o registro em lote de documentos fiscais em trânsito e identificar a unidade de carga utilizada e demais características do transporte" (MDFE, 2013).

"A autorização de uso do MDF-e implicará em registro posterior dos eventos, nos documentos fiscais eletrônicos nele relacionados" (MDFE, 2013).

Um dos objetivos do MDF-e é incentivar o uso do relacionamento eletrônico de clientes, mais conhecido como *Business-to-Business* (B2B). Então espera-se que tal relacionamento seja impulsionado pela utilização de padrões abertos de comunicação pela Internet e pela segurança trazida pela certificação digital (MDFE, 2013).

4.2 DESCRIÇÃO SIMPLIFICADA DO MODELO OPERACIONAL

> “A empresa emissora do MDF-e gerará um arquivo eletrônico (XML) contendo as informações do veículo de carga, condutor, previsão de itinerário, valor e peso da carga e documentos fiscais, o qual deverá ser assinado digitalmente, de maneira a garantir a integridade dos dados e a autoria do emissor” (MDFE, 2013, p.9).

“O arquivo eletrônico do MDF-e, será transmitido pela Internet, para o ambiente autorizador, que fará uma validação do arquivo e devolverá uma mensagem eletrônica com o resultado da validação, podendo ser rejeição ou autorização” (MDFE , 2013, p.9).

Apenas poderá iniciar o transporte quando o resultado da resposta for igual a autorização de uso.

Figura 27: Operacionalidade do MDF-e
Fonte: www.set.rn.gov.br

4.3 FUNCIONAMENTO TÉCNICO NO ADEMPIERE

Quando o usuário clicar para emitir o MDF-e, um processo é executado para que a emissão seja realizada com sucesso, e este processo funciona basicamente da seguinte maneira:

1. Primeiramente se cria o XML do MDF-e baseado nas informações inseridas no MSE e contidas nos CTRCs vinculados a esse MSE.
2. Assim que o XML está pronto para ser enviado, algumas informações do mesmo são salvas no banco de dados PostgreSQL.
3. Após, esse XML precisa ser enviado para o Web Service da Sefaz, então uma comunicação é estabelecida entre a máquina cliente (máquina que vai enviar o XML) e o servidor (máquina da Sefaz). A comunicação é realizada através do protocolo SOAP e o XML é enviado via HTTP.
4. Em seguida, o Web Service receber o XML, verifica e valida as informações do mesmo e devolve uma resposta XML ao cliente da mesma forma que lhe foi enviado.
5. Por fim, o cliente recebe o XML de resposta, verifica se o XML enviado foi Autorizado ou Rejeitado e atualiza a base de dados de acordo com a resposta.

O processo de cancelamento, consulta e encerramento funcionam da mesma maneira, será apenas diferente a estrutura e os dados que vão no XML de envio e resposta. Quanto a impressão do Documento Auxiliar do Manifesto Eletrônico de Documentos Fiscais (DAMDFE), este funciona da seguinte forma:

1. O usuário após clicar para imprimir o MDF-e, um processo em iReport é chamado.
2. As informações que são listadas no Report são as mesmas salvas na base de dados.
3. Assim que o relatório estiver “alimentado” com as informações, a janela do Report é aberta ao usuário, podendo este imprimir, exportar salvar entre outras ações.

Para acompanhar o transporte das mercadorias deverá ser impresso, em papel, o documento auxiliar do MDF-e – DAMDFE.

4.4 EVENTOS DO MANIFESTO

A operação não termina com a emissão do manifesto e a mensagem de "Autorização de Uso" como resposta. Existem três processos (eventos) que podem e até devem ser executados assim que o documento digital é válido pela Sefaz.

4.4.1 Evento de Encerramento do Manifesto

"A empresa emitente deverá encerrar o MDF-e no final do percurso. Se houver MDF-e pendente de encerramento não será possível autorizar novo MDF-e para o mesmo par de UF carregamento, descarregamento e mesmo veículo" (MDFE, 2013, p. 10).

"Se no decorrer do transporte houver qualquer alteração nas informações do MDF-e (veículos, carga, documentação, motorista, etc.), este deverá ser encerrado e ser emitido um novo MDF-e com a nova configuração" (MDFE, 2013, p. 10).

> "Entende-se como encerramento do MDF-e o ato de informar ao fisco, através de Web Service de registro de eventos o fim de sua vigência, que poderá ocorrer pelo término do trajeto acobertado ou pela alteração das informações do MDF-e através da emissão de um novo" (MDFE, 2013, p. 10).

"O Ambiente Autorizador será o repositório nacional de todos os MDF-e emitidos e disponibilizará os documentos para as Secretarias de Fazenda das Unidades Federadas, RFB e SUFRAMA" (MDFE, 2013, p. 10).

"O sistema MDF-e implementa o conceito de "evento", que é o registro de uma ação ou situação relacionada com o manifesto, que ocorreu após a autorização de uso, como o registro de um cancelamento, encerramento e consulta" (MDFE, 2013, p. 10).

4.4.2 Evento de Cancelamento do Manifesto

"Após ter o seu uso autorizado pela SEFAZ, um MDF-e não poderá sofrer qualquer alteração, pois qualquer modificação no seu conteúdo invalida a sua assinatura digital" (MDFE, 2013).

> "O emitente poderá antes de iniciada a prestação de serviço de transporte, efetuar o cancelamento do MDF-e, por meio da geração de um arquivo XML específico para isso. Da mesma forma que foi realizada a emissão de um MDF-e, o pedido de cancelamento de um MDF-e também deverá ser autorizado pelo Ambiente Autorizador através do sistema de registro de evento" (MDFE, 2013).

> "Somente poderá ser cancelado um MDF-e que tenha sido previamente autorizado o seu uso pelo Fisco e desde que não tenha ainda ocorrido o fato gerador, ou seja, em regra, ainda não tenha ocorrido o início do transporte. O prazo atual para o cancelamento do MDF-e é de 24 horas. Passadas as 24 horas, o manifesto apenas poderá ser encerrado" (MDFE, 2013).

4.5 MODELS ESSENCIAIS

O MDF-e necessita de outros documentos e conhecimentos para que qualquer processo do mesmo (emissão, cancelamento e etc) seja realizado. Com base nisso, o ADempiere deverá possuir os *models* necessários para que a implementação e o desenvolvimento do MDF-e se torne possível. Os documentos fiscais e *models* necessários são: MSE, CT-e, CTRC e NF-e.

4.5.1 MSE (Manifesto de Serviço Expresso)

O MSE é um módulo desenvolvido para as empresas transportadoras, e tem como finalidade ser “mais” do que um manifesto comum como:

1. Fechamento de transporte de cargas.
2. Controle de viagens executadas pelos caminhões e pagamento de motoristas terceirizados.
3. Análise de custo do frete e da viagem e confrontá-las. Ex: se o frete cobre o serviço executado e despesas e etc.
4. Um MSE também funciona como um totalizador de valores dos CTRCs vinculados a ele.

O manifesto é um documento utilizado por empresas transportadoras de cargas, onde são relacionados todos os conhecimentos de transporte emitidos em uma operação de transporte de carga fracionada (onde em um mesmo caminhão há mercadorias para diversos destinatários).

Ao criar um novo documento MSE no ADempiere, o usuário vincula CTRCs ao MSE, e automaticamente o software faz a totalização de alguns valores dos CTRCs salvando-os no próprio documento MSE, como: total do peso frete, total da prestação e etc. O MDF-e é criado a partir deste documento. De modo bem simples e genérico, o MDF-e é a forma emitida eletronicamente do MSE (assim como a NF-e é a versão emitida eletronicamente de uma NF).

4.5.2 CTRC (Conhecimento de Transporte Rodoviário de Cargas)

O CTRC é um documento fiscal emitido pelas transportadoras de cargas para cobrir e dar conhecimento ao cliente, motorista e para a própria empresa, em caso de fiscalização, das mercadorias entre a localidade de origem e o destinatário da carga. Além

disso, para a transportadora em si, também tem utilidade de nota fiscal, sendo utilizado para contabilizar as receitas e efetivar o faturamento (RECEITA, 2009).

O conhecimento de transporte é um impresso fiscal em papel (como a nota fiscal de produtos ou serviços). O conhecimento está relacionado com a atividade de transporte da empresa e conforme determina as leis de cada estado, podendo ser: rodoviário, aéreo, ferroviário, fluvial e multimodal.

4.5.3 CT-e (Conhecimento de Transporte Eletrônico)

> "Documento de existência apenas digital, emitido e armazenado eletronicamente, com o intuito de documentar, para fins fiscais, uma prestação de serviço de transporte de cargas realizada por qualquer modal (Rodoviário, Aéreo, Ferroviário, Aquaviário e Dutoviário). Sua validade jurídica é garantida pela assinatura digital do emitente (garantia de autoria e de integridade) e pela recepção e autorização de uso, pelo Fisco" (CTE, 2012).

O CT-e substitui diversos documentos utilizados pelos modais para cobertura de suas respectivas prestações de serviços, sendo um deles o CTRC (Conhecimento de Trasporte Rodoviário de Carga).

4.5.4 NF-e (Nota Fiscal Eletrônica)

> "Documento de existência apenas digital, emitido e armazenado eletronicamente, com o intuito de documentar, para fins fiscais, uma operação de circulação de mercadorias ou uma prestação de serviços, ocorrida entre as partes. Sua validade jurídica é garantida pela assinatura digital do remetente (garantia de autoria e de integridade) e a Autorização de uso fornecida pelo Fisco, antes da ocorrência do fato gerador" (NFE, 2005).

4.5.5 Hierarquia dos Documentos e Manifestos

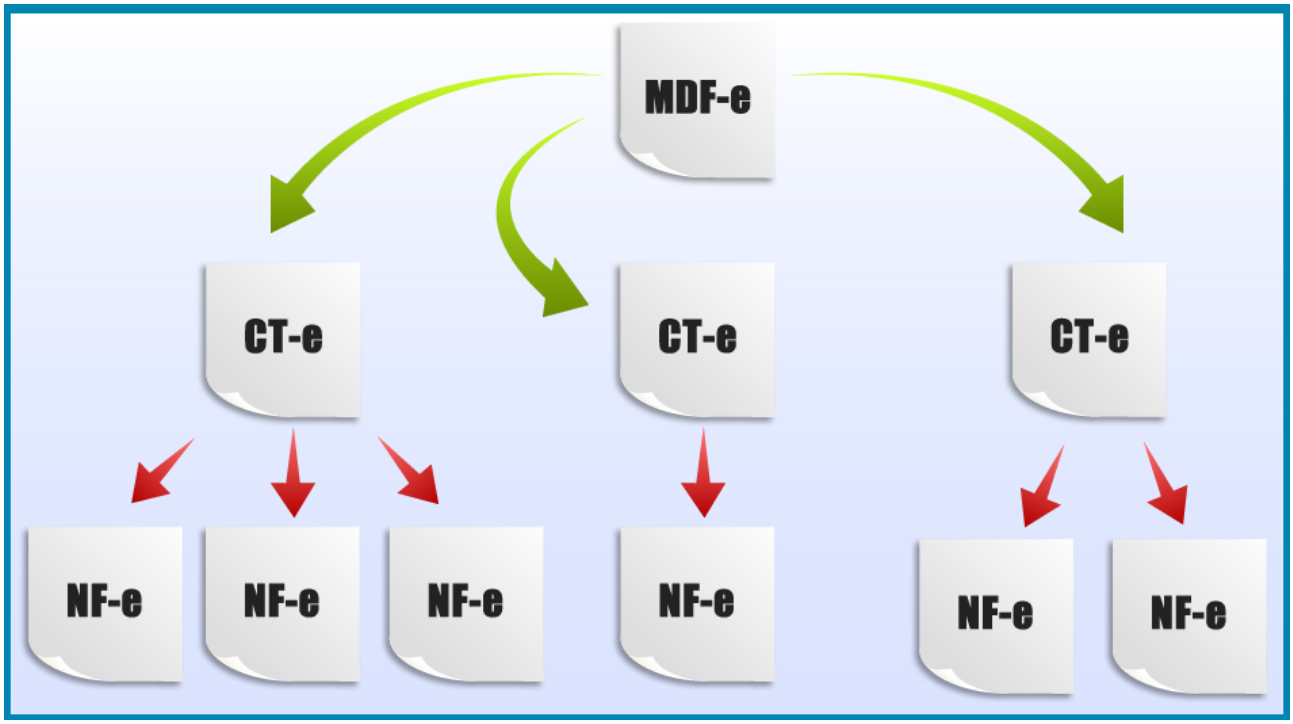

Figura 28: Hierarquia
Fonte: Autor

A Figura 28 pode ser explicada da seguinte maneira: um documento MDF-e é uma viagem de caminhão e este pode ter vários CT-es. Cada CT-e é uma entrega a ser realizada e este pode possuir diversas NF-es. Cada NF-e é uma documentação fiscal de circulação de mercadoria ou prestação de serviço. Em outras palavras, para cada viagem (MDF-e), há várias entregas (CT-e) de diversas NF-es.

Ao fiscalizar o MDF-e através de seu código de barras através de um leitor, o fiscal saberá qual ou quais CT-es estão vinculados e suas respectivas NF-es.

5 ANÁLISE DE RESULTADOS

O MDF-e proporciona benefícios a todos os envolvidos na prestação do serviço de transporte. Os beneficiados com este documento eletrônico são: emitente do documento, sociedade, contabilista e fisco.

5.1 EMITENTES DO MDF-E

Como o documento é emitido eletronicamente, há uma redução nos custos de impressão do mesmo. O MDF-e contempla a impressão de apenas um documento físico (papel), denominado DAMDFE, cuja função é apenas acompanhar o transporte e consequentemente informar o trânsito dos documentos da carga. Com base nisso, também pode-se dizer que há redução nos custos de armazenagem de documentos fiscais (Figura 29). Atualmente, os documentos em papel devem ser mantidos para apresentação futura ao fisco pelo prazo decadencial. Guardar os documentos em papel exige espaço e envolve toda a logística necessária para a recuperação do documento. O custo de arquivamento do documento digital é muito menor do que o custo do arquivamento físico (MDFE, 2013).

“Para o GED (Gerenciamento Eletrônico de Documentos), o MDF-e é um documento eletrônico que não requer a digitalização do original em papel.” (MDF-E, 2013).

NÚMERO DE ARMAZENAGENS EM PAPEL DOS MANIFESTOS NÃO EMITIDOS ELETRONICAMENTE		
Nº MANIFESTOS EMITIDOS POR DIA	Nº MANIFESTOS EMITIDOS POR MÊS	Nº MANIFESTOS EMITIDOS FINAL 5 ANOS
100	2000	120000

NÚMERO DE ARMAZENAGENS EM PAPEL DOS MANIFESTOS EMITIDOS ELETRONICAMENTE		
Nº MANIFESTOS EMITIDOS POR DIA	Nº MANIFESTOS EMITIDOS POR MÊS	Nº MANIFESTOS EMITIDOS FINAL 5 ANOS
0	0	0

Figura 29: Armazenagem dos Manifestos
Fonte: Autor

Não há necessidade da impressão em papel do CT-e quando o caminhoneiro sair para estrada, pois o MDF-e já vincula estes a ele. Em outras palavras, quando ocorrer a fiscalização no meio do trajeto, basta o documento DAMDFE ser lido com um *scanner* de código de barras para o fiscal saber na hora quais conhecimentos de transporte fazem parte daquela viagem.

Outra vantagem para os caminhoneiros é a redução de paradas em Postos Fiscais de Fronteira, pois os processos de fiscalização de mercadorias em trânsito foi simplificado com manifesto eletrônico (MDFE, 2013).

5.2 SOCIEDADE

Assim como já mencionado, um dos objetivos do MDF-e é incentivar o comércio eletrônico e o uso de novas tecnologias, com isso padronizando o relacionamentos eletrônicos entre empresas e gerando oportunidades de negócios e empregos na prestação de serviços ligados ao manifesto (MDFE, 2013).

O MDF-e traz também como vantagem à sociedade, a redução do consumo de papel (impacto positivo em termos ecológicos para todos).

5.3 CONTABILISTAS

Os contabilistas podem contar com novas oportunidades de serviços e consultoria ligados ao MDF-e (MDFE, 2013).

Como o MDF-e é um documento eletrônico e seu armazenamento é digital e não físico, o trabalho dos contabilistas é automatizado e agilizado.

5.4 FISCO

O Fisco pode contar com melhoria no processo de controle fiscal possibilitando um melhor intercâmbio e compartilhamento de informações entre os fiscos e redução de custos no processo de controle dos manifestos capturados pela fiscalização de mercadorias em trânsito (MDFE, 2013).

Outra grande vantagem e benefício ao fisco (não menos importante) é com o aumento na confiabilidade da fiscalização do transporte de cargas. Como o MDF-e é assinado digitalmente, há integridade e autenticidade do documento, ou seja, a assinatura digital garante ao destinatário que o documento não foi alterado ao ser enviado (integridade) e ainda comprova a autoria do emitente (autenticidade). Documentos eletrônicos não assinados digitalmente têm as características de alterabilidade e fácil falsificação.

6 CONSIDERAÇÕES FINAIS

O objetivo geral do trabalho foi desenvolver um módulo para o ERP *Open Source* ADempiere, que possibilitasse as diversas operações relacionadas ao Manifesto Eletrônico de Documentos Fiscais (MDF-e), desde de uma emissão, à um cancelamento, encerramento entre outros (assim como exigidos pela legislação brasileira).

O módulo desenvolvido correspondeu perfeitamente as expectativas esperadas, satisfazendo as necessidades das empresas transportadoras e a obrigatoriedade da legislação do país. A utilização e operacionalidade dos processos implementados no sistema integrado (relacionados ao modelo desenvolvido do MDF-e) são extremamente fáceis.

Diversas dificuldades encontradas durante o desenvolvimento do projeto podem ser levantadas e ressaltadas, como: consumo do WebService da Sefaz, geração do impresso DAMDFE através da ferramenta iReport e as funcionalidades do próprio *framework* ADempiere para criação de janelas, abas, componentes e etc.

Todas as metologias, tecnologias e ferramentas utilizadas escolhidas, atenderam muito bem ao que foi planejado. A comunicação entre a máquina que envia o XML e o Web Service que a recebe, é muito rápida e segura. O processo de emissão costuma demorar entre 10 e 15 segundos e um cancelamento, encerramento, consulta e impressão não passam de 5 segundos.

Ao longo da criação deste trabalho surgiram novas possibilidades que não foram desenvolvidas, pois tornariam o trabalho muito extenso. Uma delas é a utilização do SGBD Oracle em relação a dados e informações com banco de dados, à fim de testar performance e desempenho de acesso e comunicação.

REFERÊNCIAS

ADEMPIERELBR. **O que é ADempiere**.
Disponível em: <http://adempierelbr.wikispaces.com/>
Acesso em: 17 abr de 2014

ALECRIM, Emerson. **O que é ERP (Enterprise Resource Planning)?**
Disponível em: <http://www.infowester.com/erp.php>
Acesso em: 12 abr de 2014

ALENCAR, Cristiano. **Protocolo de Rede**. Disponível em: <http://www.cristianoalencar.com.br/Aulas/Protocolos%20de%20Rede.pdf>. Acesso em: 06/04/2014

CAELUM: ensino e inovação. **O que é Java?**
Disponível em: <http://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java>.
Acesso em: 09 abr 2014

CARLOS, Morimoto. **Hardware**: o guia definitivo. São Paulo: Sul Editores, 2007. 848 p.

CERIBELLI, Cinthia. **Guia como se faz**: montagem, manutenção e instalação de computadores. Rio de Janeiro: Escala, 2008; 162 p.

CTE. **Conceito e Uso**. Disponível em: <http://www.cte.fazenda.gov.br/perguntasFrequentes.aspx?tipoConteudo=fYFuI10FiqM=>
Acesso em: 14 abr de 2014

DEITEL, Harvey M.. **Sistemas Operacionais**. São Paulo: Prentice Hall, 2005. 784 p.

DIAS, André Felipe. **Conceitos Básicos de Controle de Versão de Software**: centralizado e distribuído. Disponível em: <http://www.pronus.eng.br/artigos_tutoriais/gerencia_configuracao/conceitos_basicos_controle_versao_centralizado_e_distribuido.php>. Acesso em: 21 abr de 2014

ECLIPSE. **About the Eclipse Foundation**. Disponível em: <https://www.eclipse.org/org/>
Acesso em: 21 abr de 2014
FILHO, Antonio M. da Silva. **Arquitetura de Software**: desenvolvimento orientado para arquitetura. Rio de Janeiro: Campus, 2002. 240 p.

GASPAR, Heloisa. **O que é sistema ERP?**
Disponível em: <http://www.pwi.com.br/blog/o-que-e-sistema-erp/>.
Acesso em: 15 abr de 2014

JASPERSOFT COMMUNITY. **iReportDesigner Getting Started**. Disponível em: <http://community.jaspersoft.com/wiki/ireport-designer-getting-started>. Acesso em: 10 abr de 2014

MANZANO, José Augusto N. G..**Programação Assembly**: padrão ibm-pc 8086/8088. São Paulo: Érica, 2012. 336 p.

MCLAUGHLIN; POLLICE; WEST. **Use a Cabeça!**: Análise & Projeto Orientado ao Objeto. São Paulo: Alta Books, 2008. 472 p.

MDFE. **Projeto Manifesto Eletrônico de Documentos Fiscais**: Nota Técnica 2013/004. São Paulo, SP. Secretaria da Fazenda, 2013. 127 p. Disponível em: <https://www.fazenda.sp.gov.br/mdfe/MDFe_Nota_Tecnica_2013_0041.pdf> Acesso em: 19 abr de 2014.

MDFE PORTAL. **Sobre o MDF-e**: Apresentação do Projeto.
Disponível em: <https://mdfe-portal.sefaz.rs.gov.br/Site/Sobre>
em: 18 abr de 2014

MDFE PORTAL. **Perguntas Frequentes**: Apresentação do Projeto. Disponível em: <https://mdfe-portal.sefaz.rs.gov.br/Site/Faq> Acesso em: 18 abr de 2014

MENDES, Antonio. **Arquitetura de Software**: desenvolvimento orientado para arquitetura. Editora Campus. Rio de Janeiro - RJ, 2002.

MILTELLO, Katia. **Quem precisa de um ERP?** Info Exame, p. 140, mar. 1999.

NFE. **Conceito, uso\\\\\\\\\\\\\\ e obrigatoriedade da NF-e (26 questões)**. Disponível em: <http://www.nfe.fazenda.gov.br/portal/perguntasFrequentes.aspx?tipoConteudo=E4+tmY+ODf4=> Acesso em: 14 abr de 2014
ORACLE. **PL/SQL**.
Disponível em: <http://www.oracle.com/technetwork/database/features/plsql/index.html>.
Acesso em: 10 abr de 2014

PACIEVITCH, Yuri. **Software**. Disponível em: <http://www.infoescola.com/informatica/software/>. Acesso em: 02 abr 2014.
W3C. **Hypertext Transfer Protocol Protocol - HTTP/1.1**. Disponível em: <http://www.w3.org/Protocols/rfc2616/rfc2616.html>. Acesso em: 07 abr 2014

PAULA, Vinicus de. **ADempiere**.
Disponível em: <http://pt.slideshare.net/Gilvancdr/unitri-sistemas-de-informao-adempiere>
Acesso em: 17 abr de 2014

POSTGRESQL. **About**.
Disponível em: <http://www.postgresql.org/about/>.
Acesso em: 10 abr de 2014

RECEITA FEDERAL. **Conhecimento de Carga**: introdução. Disponível em: <http://www.receita.fazenda.gov.br/manuaisweb/importacao/topicos/entrega_de_documentos/conhecimento_de_carga/introducao.htm> Acesso em: 15 abr de 2014

RENAUD, Paulo E.. **Introdução aos Sistemas Cliente/Servidor**: guia prático para profissionais de sistemas. São Paulo: IBPI Press, 1994. 335 p.

SUSVIELA, Carlos Alberto Nunes. **Visual Basic**: versão 5. Rio Grande do Sul: Power Informática, 2000. 177 p.

W3C. **Web Services Activity**. Disponível em: <http://www.w3.org/2002/ws/>. Acesso em: 08 abr 2014

W3C. **HTML**: The Markup Language (an HTML language reference). Disponível em: <http://www.w3.org/TR/2013/NOTE-html-markup-20130528/>. Acesso em: 08 abr 2014

W3SCHOOLS. **XML Tutorial**. Disponível em: <http://www.w3schools.com/xml/default.asp>. Acesso em: 08 abr 2014

W3SCHOOLS. **SOAP Introduction**. Disponível em: <http://www.w3schools.com/webservices/ws_soap_intro.asp>. Acesso em: 08 abr 2014

GLOSSÁRIO

Carga fracionada - é quando precisa levar algo que ocupe apenas uma parte do espaço do caminhão. Exemplo: levar um fogão de SP para MG sem nenhuma restrição em levá-lo no mesmo caminhão que outros clientes.

Interoperabilidade - comunicação de forma transparente (ou o mais próximo disso) com outro sistema (semelhante ou não).

Open Source - software de utilização livre, cuja licença não é cobrada e cujo código fonte é disponibilizado, de forma gratuita, pelo autor.

Parametrização - permitir alterar muito profundamente e rapidamente o sistema, definindo novos fluxos de informações e/ou novos procedimentos, sem desenvolvimento suplementar.