



CENTRO UNIVERSITÁRIO DA GRANDE DOURADOS

WAGNER LIMA CHIMENEZ

APLICAÇÃO PARA DISPOSITIVOS MÓVEIS COM COMUNICAÇÃO PARA UMA BASE DE DADOS VIA INTERNET

Dourados
2010



CENTRO UNIVERSITÁRIO DA GRANDE DOURADOS

WAGNER LIMA CHIMENEZ. RGM: 122.591

APLICAÇÃO PARA DISPOSITIVOS MÓVEIS COM COMUNICAÇÃO PARA UMA BASE DE DADOS VIA INTERNET

Monografia apresentada ao Curso de Ciência da Computação da Faculdade de Ciências Exatas, como pré-requisito para obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Profa. Noiza Waltrick Trindade.

Dourados
2010

Resumo

O surgimento do aparelho celular causou uma revolução na maneira de se comunicar, trazendo inovações e tecnologias surpreendentes que por aí não pararam. Vários aplicativos foram desenvolvidos para esse dispositivo com a finalidade de trazer agilidade ao usuário em resolver certos problemas comuns. Dessa forma em meio a muitas inovações, ele se tornou uma peça fundamental no bolso de muitas pessoas. A linguagem *Java ME* foi criada para desenvolvimento de aplicativos para diversas plataformas de dispositivos móveis, destacando-se por ter uma maior portabilidade e acessibilidade. O projeto visa a implementação de uma ferramenta que seja capaz de realizar orçamentos e pedidos de venda no ramo de vidraçaria de maneira eficiente, utilizando-se dos recursos disponíveis na linguagem *Java ME* e no dispositivo móvel (aparelho celular). Para que a aplicação móvel possa realizar orçamentos e pedidos de vendas será desenvolvido uma aplicação que servirá de base para a aplicação móvel, ela estará em uma máquina servidor na qual o dispositivo móvel fará requisições para realização das operações necessárias. A comunicação entre as aplicações ocorrerá por meio de *Servlets* que funcionam como meio de tratar requisições e respostas vindas de aplicações remotas. Será utilizado um arquivo XML que conterá as informações do Banco de Dados da aplicação servidor. Para que a aplicação móvel possa se comunicar com a aplicação servidor o arquivo XML será acessado via HTTP, onde o *Servlet* vai tratar a requisição, a aplicação servidor responderá a solicitação e armazenará a resposta em um arquivo XML, a aplicação móvel se comunica via HTTP, faz a leitura do arquivo XML e realiza a venda.

Palavras chave: *celular, Java ME, Servlet, XML, HTTP* .

Lista de Figuras

2.1	Arquitetura J2ME	15
2.2	Ciclo de Vida de uma MIDlet	17
7.1	Utilitarios	27
7.2	Main	27
7.3	Principal	28
7.4	Pedidos	28
7.5	Cadastros	29
7.6	Consultas	30
7.7	DER - Diagrama de Entidade e Relacionamento	31
7.8	Tela Principal	32
7.9	Tela Cadastro de Clientes	32
7.10	Tela Cadastro de Produtos	33
7.11	Tela Cadastro de Projetos Aba 1	33
7.12	Tela Cadastro de Projetos Aba 2	34
7.13	Tela Orcamentos Aba 1	34
7.14	Tela Orcamentos Aba 2	34
7.15	Tela Orcamentos Aba 3	35

Lista de siglas

BTREE	<i>Arvore Binária</i>
CDC	<i>Connected Device Configuration</i>
CLDC	<i>Connected Limited Device Configuration</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
J2ME	<i>Java Micro Edition</i>
J2SE	<i>Java Standard Edition</i>
J2EE	<i>Java Enterprise Edition</i>
JAD	<i>Arquivo Java Application Descriptor</i>
JAR	<i>Arquivo Java Archive</i>
JVM	<i>Java Virtual Machine</i>
MGA	<i>Arquitetura Multi-Generacional</i>
MIDP	<i>Mobile Information Device Prole</i>
OEM	<i>Original Equipment Manufacturer</i>
OLTP	<i>On-Line Transaction Procesing</i>
PDA's	<i>Personal Digital Assistant</i>
SGDBR	<i>Sistema Gerenciador de Banco de Dados Relacional</i>
SQL	<i>Structured Query Language</i>
UDF's	<i>Funções Criadas pelo Usuário</i>
XML	<i>Extensible Markup Language</i>
WAN	<i>Wide Area Network</i>
W3C	<i>Word Wide Web Consortium</i>

Sumário

1	Introdução	7
1.1	Objetivo Geral	8
1.2	Objetivo Específico	8
1.3	Justificativa	8
1.4	Metodologia	9
1.5	Cronograma	10
1.6	Organização do Texto	11
2	<i>Java e Java ME</i>	12
2.1	Configuração (<i>Configuration</i>)	13
2.2	Configuração <i>CLDC</i> - <i>Connected Limited Device Configuration</i>	13
2.3	Perfil (<i>Profile</i>)	13
2.4	Máquinas Virtuais <i>Java</i>	14
2.5	Arquitetura <i>Java ME</i>	14
2.6	Arquitetura do Perfil <i>MIDP</i>	14
2.6.1	MIDlet	16
2.7	Arquivo <i>JAR</i> e Arquivo <i>JAD</i>	17
3	Sistema Gerenciador de Banco de Dados <i>Firebird</i>	18
4	<i>Servlet</i>	20
4.1	Funcionamento do <i>Servlet</i>	21

5	Linguagem de Marcação XML	23
5.1	Estrutura de um documento XML	23
6	Especificações do Projeto	25
7	Implementação - Aplicação <i>Desktop</i>	27
	Referências Bibliográficas	36

1 Introdução

No ano de 1990 a Internet começou a alcançar a população em geral com o desenvolvimento da *World Wide Web* pelo engenheiro Tim Bernes Lee, possibilitando a utilização de uma interface gráfica e a criação de sites mais dinâmicos e visualmente interessantes. A partir deste momento a *Internet* cresceu em ritmo acelerado. Muitos dizem, que foi a maior criação tecnológica depois da televisão na década de 1950.

A década de 1990 tornou-se a era de expansão da *Internet*. Para facilitar a navegação pela *Internet*, surgiram vários navegadores (*browsers*) como, por exemplo, o *Internet Explorer* da *Microsoft* e o *Netscape Navigator*. O surgimento acelerado de provedores de acesso e portais de serviços *on-line* contribuíram para este crescimento (SUAPESQUISA.COM,).

As empresas descobriram na *Internet* um excelente caminho para melhorar seus lucros e as vendas *on-line* dispararam, transformando a *Internet* em verdadeiros *shopping centers* virtuais.

Nos dias atuais, é impossível pensar no mundo sem a *Internet*. Ela tomou parte dos lares de pessoas do mundo todo. Estar conectado a rede mundial passou a ser uma necessidade de extrema importância.

Estamos vivendo em um mundo onde a tecnologia nos surpreende a cada momento. A *Internet* que até então só existia para computadores passou a ganhar espaço também em aparelhos celulares, contribuindo para que as vendas e a popularização do dispositivo móvel se expandisse.

Atualmente são poucas as pessoas que não possuem um aparelho e a cada dia são desenvolvidas novas funcionalidades e novos recursos fazendo que o mesmo se torne cada vez mais presente na sociedade.

O aparelho celular possui grande importância no mercado de trabalho possibilitando que às empresas interajam com seus funcionários ou até mesmo que seus fun-

cionários possam manter contatos para futuras negociações, geralmente *on-line* utilizando-se dessa tecnologia como ferramenta estratégica de seus produtos.

A utilização de uma ferramenta que possa realizar o trabalho de coleta de dados de um determinado cliente, a consulta de informações de um determinado produto, ou ainda prestação de serviços oferecida pela empresa, diretamente de um aparelho celular seria uma importante funcionalidade que agilizaria o processo de atendimento e o andamento das negociações.

Podendo serem recebidas e enviadas no dispositivo móvel via *Internet* até o banco de dados da empresa as informações trariam uma forma dinâmica de atender o cliente sem ter que deixá-lo esperando por uma resposta, ou simplesmente fazendo com que o mesmo se sinta mais próximo facilitando a negociação.

Utilizando-se dessa nova ferramenta o funcionário da empresa poderia no momento da abordagem, estabelecer de forma clara e objetiva os valores das prestações e os serviços a serem realizados.

1.1 Objetivo Geral

Desenvolver uma aplicação para dispositivos móveis e uma aplicação para computadores *Desktop* no ramo de vidraçaria, que sejam capazes de efetuar orçamentos e vendas de produtos oferecidos pela empresa.

A aplicação móvel funcionará como cliente e a aplicação *Desktop* atuará como servidor do Banco de Dados onde fornecerá informações necessárias à aplicação cliente.

1.2 Objetivo Específico

Fazer com que a aplicação móvel se comunique com a aplicação *Desktop* na troca de informações necessárias para a realização dos orçamentos e vendas da empresa.

Fazer com que a utilização das aplicações possam ser úteis contribuindo para negociações no ramo comercial (vidraçaria).

1.3 Justificativa

A necessidade de melhor atender o cliente e satisfazê-lo tem sido algo muito importante no ramo comercial, principalmente quando se trata de uma negociação que provenha fazer com que o mesmo possa se sentir bem e estar interessado em adquirir uma prestação de serviço ou um produto da empresa.

O mercado de trabalho exige que as empresas se dediquem o máximo para conquistar seus clientes, a concorrência não está mais voltada somente nos produtos, mas sim na qualidade do atendimento.

A inovação e versatilidade são qualidades imprescindíveis para a sustentação de um produto no competitivo mercado da era global.

Uma ferramenta que possa trazer agilidade na negociação e melhor atendimento ao cliente é fundamental para que as empresas possam adquirir maior sucesso em suas vendas.

Uma aplicação voltada para realização de orçamentos e vendas onde se possam atender os clientes no local que se encontram e também auxiliar no trabalho de vendedores externos de uma empresa pode contribuir para uma maior competitividade no ramo comercial.

1.4 Metodologia

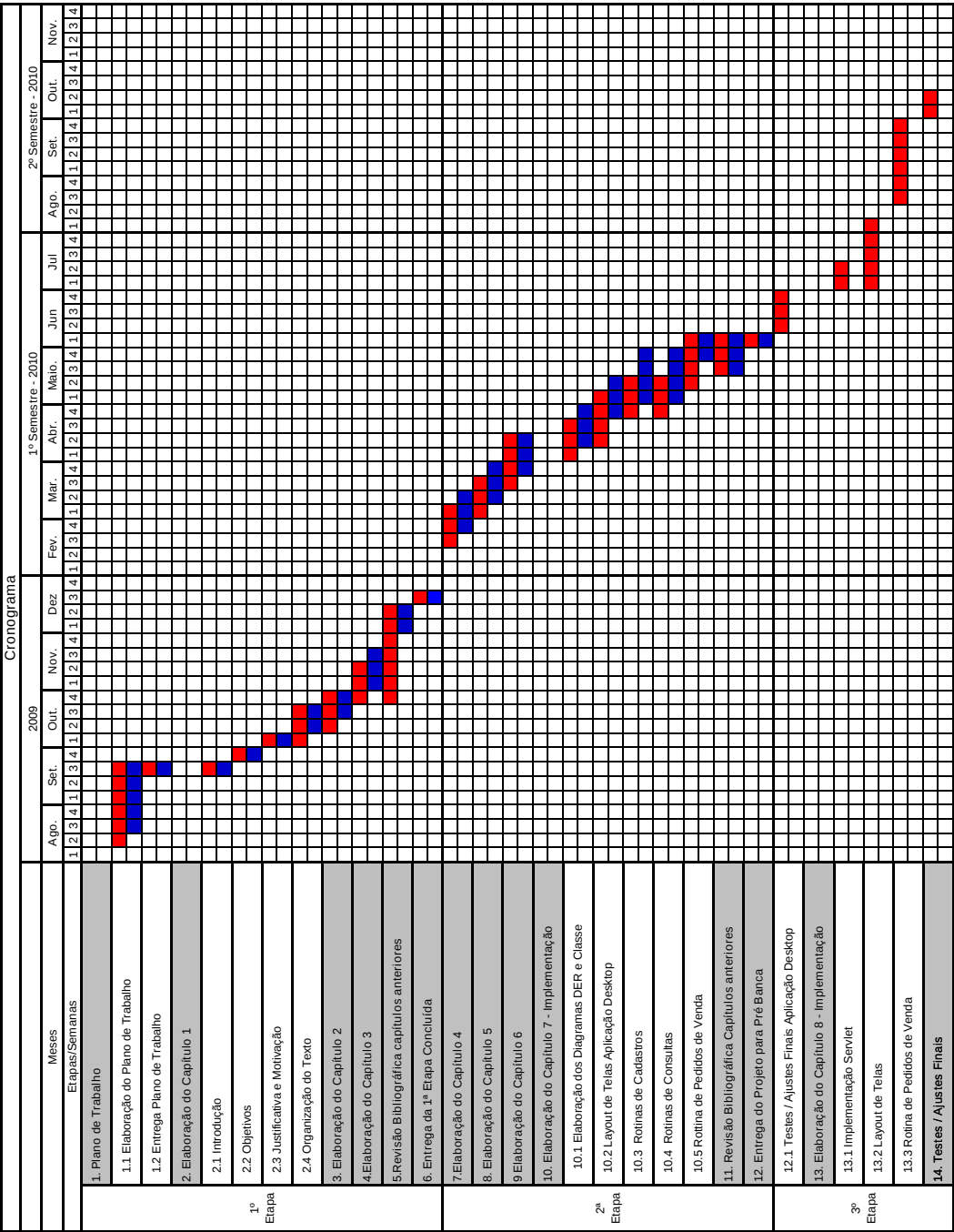
A pesquisa a ser realizada fará um estudo da linguagem *Java ME* utilizada para programação voltada para dispositivos móveis, e Java para *Desktop*.

Também será estudado um gerenciador de banco de dados *Firebird* para armazenar as informações necessárias referentes as prestações de serviços oferecidas pela empresa.

As aplicações a serem desenvolvidas serão voltadas para o ramo de vidraçarias, com a funcionalidade de efetuar orçamentos e pedidos de vendas.

Será estudado o funcionamento de *Servlets* e arquivos XML, no qual serão responsáveis pela comunicação entre as aplicações.

1.5 Cronograma



Prazos	Entrega do plano de trabalho	26/09/2009
	Entrega da primeira etapa	09/11/2009
	Entrega da segunda etapa	01/06/2010
	Entrega da terceira etapa	Out. / Nov.

■ Previsto
■ Realizado

1.6 Organização do Texto

O texto foi organizado em sete capítulos.

O primeiro capítulo apresentando uma introdução com objetivo geral e específico, justificativa, metodologia, e o cronograma a ser realizado durante o projeto.

O segundo capítulo as linguagem que serão utilizadas *Java* e *Java ME*.

O terceiro capítulo trata sobre o Banco de Dados a ser utilizado na aplicação.

O quarto capítulo fala sobre *Servlets*, e seu funcionamento.

O quinto capítulo trata sobre arquivo XML e sua estrutura.

O sexto capítulo especifica detalhadamente as escolhas do projeto.

E o sétimo capítulo a implementação da aplicação *Desktop*.

2 *Java e Java ME*

Java é uma linguagem de programação desenvolvida pela *Sun Microsystems*, lançada no mercado em 1995, tendo como base a orientação a objetos e portabilidade.

É uma linguagem puramente orientada a objetos, seus programas podem ser executados em qualquer plataforma operacional. É uma linguagem compilada e interpretada. sua compilação produz um código intermediário, em *bytecodes*, que é executado em uma máquina virtual *JVM (Java Virtual Machine)* implementada por *software* e associada à plataforma. Seu código de programação é interpretado garantindo maior velocidade de desenvolvimento e portabilidade, possuindo um alto desempenho (PUGA; RISSETTI, 2009).

A linguagem *Java* possui um ambiente de desenvolvimento e um ambiente de aplicativos, ela surgiu baseada na linguagem C e tem o C++ em comum (LEMAY; L.PERKINS, 1996).

O desenvolvimento de uma aplicação *Java* passa por uma sequência de etapas, que vai da edição do programa à compilação, à carga das classes, à verificação e à execução. Ela se divide em *J2SE (Java Standard Edition)*, *J2ME (Java Micro Edition)* e *J2EE (Java Enterprise Edition)*.

A linguagem *J2ME (Java Micro Edition)* é utilizada para desenvolvimento de *softwares* para sistemas e aplicações embarcadas, que funcionam a partir de um dispositivo específico como celulares e *PDA's*.

Ela se destaca como sendo uma linguagem atrativa devido a sua portabilidade e acessibilidade, com aplicações que funcionam em diversas plataformas de diferentes empresas de celulares (MUCHOW *et al.*, 2004).

O *J2ME* é dividido em Configurações e Perfis.

2.1 Configuração (*Configuration*)

Define a máquina virtual e um conjunto de características básicas.

Uma configuração define os recursos da linguagem *Java* e as bibliotecas *Java* básicas da (*JVM Java Virtual Machine*) para essa configuração em particular.

A linha divisória de aplicação de uma configuração é, de modo geral, baseada na memória, no vídeo, na conectividade de rede e no poder de processamento disponível em um dispositivo.

Existem dois tipos de configurações básicas: *CLDC* e *CDC*.

A configuração *CLDC* (*Connected Limited Device Configuration*) especifica um ambiente *Java* para telefones celulares, *paggers* e *PDA's*.

A configuração *CDC* (*Connected Device Configuration*) especifica um ambiente *Java* para TV Digital, dispositivo sem fio de alto nível e sistemas automotivos.

No desenvolvimento da aplicação para o dispositivo móvel será utilizada a configuração *CLDC*.

2.2 Configuração *CLDC* - *Connected Limited Device Configuration*

A *CLDC* define um conjunto de classes (bibliotecas) *Java* de programação de aplicações e uma máquina virtual para dispositivos de recursos limitados, como telefones celulares, *paggers* e assistentes pessoais digitais *mainstream*.

A plataforma *Java ME* também especifica um número de perfis que definem um conjunto de *API's* de nível superior que define uma aplicação.

O *CLDC* pode ser combinado com o *MIDP* (*Mobile Information Device Profile*) para fornecer um completo ambiente de aplicações *Java* para desenvolvimento de aplicativos para rodar em dispositivos com memória, poder de processamento e capacidades gráficas limitadas.

2.3 Perfil (*Profile*)

Conjunto de *API's* que complementam uma configuração para prover funcionalidades para um determinado dispositivo.

Um perfil é uma extensão de uma configuração fornecendo as bibliotecas para um desenvolvedor escrever aplicativos para um tipo em particular de dispositivo.

2.4 Máquinas Virtuais *Java*

A máquina virtual *Java* é responsável em transformar os arquivos de classe de um código fonte em código de máquina para que a plataforma que está executando a *JVM* possa executar a aplicação desenvolvida. Ela também é responsável em fornecer segurança, alocar e não alocar memória e gerenciar linhas de execução.

Na configuração *CDC* a máquina virtual tem a mesma especificação do *J2SE* (*Java Standard Edition*) projetada para execução em máquinas simples de computadores pessoais e estações de trabalho.

Na configuração *CLDC* a *Sun Microsystems* desenvolveu o que é chamado de implementação de referência de uma máquina virtual, conhecida como *KVM* (máquina virtual K) projetada para atender aos requisitos da configuração *CLDC*.

2.5 Arquitetura *Java ME*

A arquitetura genérica de uma aplicação *Java ME* conforme Figura 2.1 é formada por um Sistema Operacional hospedeiro como base, seguido de uma máquina virtual que se encaixe na configuração *CDC* ou *CLDC*. Logo após as bibliotecas básicas *CLDC* ou *CDC* são as próximas na hierarquia. E por fim encontram-se os perfis que são projetados para fornecer um kit de ferramentas para escrever aplicativos para uma família de dispositivos em particular.

2.6 Arquitetura do Perfil *MIDP*

A arquitetura de um perfil *MIDP* inicia-se basicamente no hardware que é o nível mais inferior de suas camadas seguido pelo sistema operacional nativo, que seria um

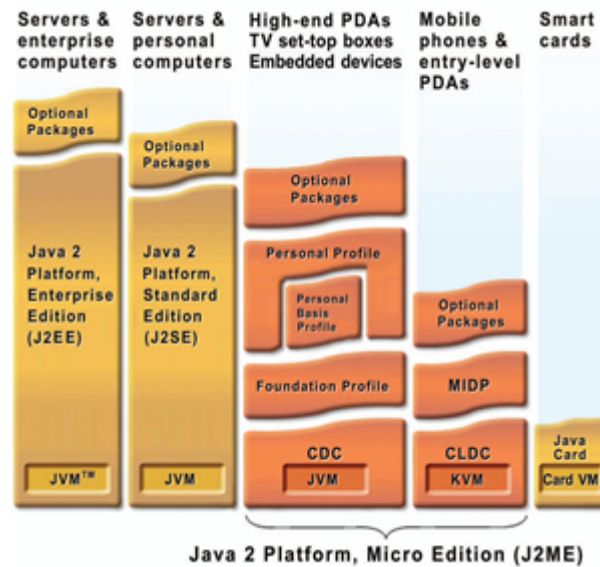


Figura 2.1: Arquitetura J2ME

programa de configuração que vem instalado no dispositivo, permitindo que você configure o tipo de toque, o volume, a data e a hora.

Uma camada acima encontra-se a configuração CLDC que é instalada no sistema operacional nativo e é a base do MIDP, finalizando com os aplicativos MIDP.

No mesmo nível que se encontra o perfil e os aplicativos encontram-se também as classes específicas de OEM (*Original Equipment Manufacturer*, fabricante de equipamento original) que são fornecidos pelo fabricante do dispositivo, podendo incluir opções de responder à chamadas recebidas, ou de pesquisar entradas em uma agenda telefônica.

As classes OEM são classes não definidas pelo MIDP e podem ser utilizadas para funcionalidades específicas para determinado aparelho, o que significa que elas podem ou não ser portáteis para outros MIDs.

O MIDP é a definição de uma arquitetura e APIs associadas necessárias para prover um ambiente de desenvolvimento aberto para MIDs (*Mobile Information Devices*). O MIDP foi feito para rodar em cima da configuração CLDC.

Segundo (UFGRS, 2009) um MID deve possuir as seguintes características mínimas de hardware (além daquelas que são requeridas pela configuração CLDC):

- *Display:*

Tamanho da tela: 96x54;

Profundidade: 1 bit;

Formato do pixel (proporção de aspecto): 1:1;

- *Input:*

“*One handed keyboard*” ou

“*Two handed keyboard*” ou

Touch Screen

- *Memória:*

128 *Kbytes* para os componentes MIDP;

8 *Kbytes* para dados das aplicações;

32 *Kbytes* para o *JAVA* runtime;

- *Rede:*

Duplex, sem fio, possivelmente intermitente e com largura de banda limitada.

2.6.1 MIDlet

Uma MIDlet é um aplicativo *Java* projetado para ser executado em um dispositivo móvel. Tem como classes básicas do *Java* a configuração CLDC e o perfil MIDP, constituindo em um ou mais aplicativos empacotados utilizando um arquivo JAR(*Jar Archive*).

Uma MIDlet é formada por:

- *Ambiente de Execução* que é compartilhado por todas as MIDlets que estão na mesma MIDlet suite, onde qualquer MIDlet pode interagir com outra que esteja no mesmo pacote.
- *Empacotamento da MIDlet suite* onde uma ou mais MIDlets podem ser empacotadas num único arquivo JAR, que contém as classes compartilhadas e os arquivos de recursos utilizados pelas MIDlets, além de um manifesto descrevendo seu conteúdo.
- *Descritor de Aplicação* sendo utilizado para gerenciar uma MIDlet e é usado pela própria MIDlet para atributos de configuração específica. O descritor permite que seja verificado que a MIDlet é adequada ao aparelho antes de se carregar todo o arquivo JAR da MIDlet suite. Ele também permite que parâmetros sejam passados para as MIDlets sem modificar os arquivos JAR.

- *Ciclo de Vida de Aplicação:* Uma MIDlet possui três estados que correspondem aos seu respectivo ciclo de vida, podendo ser ativo, pausado, destruído, conforme Figura 2.2.

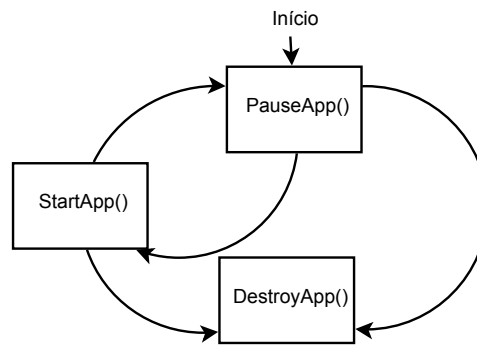


Figura 2.2: Ciclo de Vida de uma MIDlet

2.7 Arquivo *JAR* e Arquivo *JAD*

JAR (Arquivo *Java Archive*) é um arquivo onde se encontram todos os arquivos que foram necessários para ser feito uma aplicação, como por exemplo: arquivos de classes *Java*, imagens e dados do aplicativo.

Além dos arquivos de classe e recurso, um JAR contém um arquivo conhecido como manifesto. O arquivo de manifesto é armazenado como parte do próprio arquivo JAR e nele contém vários atributos como: nome do conjunto de MIDlets, número de versão da Midlet, quem desenvolveu a MIDlet, qual configuração da linguagem exigida pela MIDlet, e dentre outros mais que especificam tudo que foi utilizado no desenvolvimento da aplicação.

JAD (*Arquivo Java Application Descriptor*) é um arquivo que se encontra dentro de um arquivo JAR, onde são armazenadas todas as informações sobre as MIDlets.

3 Sistema Gerenciador de Banco de Dados *Firebird*

Firebird é um Sistema Gerenciador de Banco de Dados Relacional (SGBDR) SQL, cliente/servidor compacto e poderoso que pode ser executado em uma diversidade de plataformas de sistemas operacionais servidores e clientes, incluindo Windows, Linux e diversas outras plataformas UNIX, tais como FreeBSD e Mac OS X. É um SGBDR de capacidade industrial que possui um alto nível de compatibilidade com os padrões SQL (BORRIE, 2006).

O *Firebird* é um *software* projetado especialmente para uso local e em redes WAN. Seu núcleo consiste em dois programas de *software* principais: o servidor de banco de dados, que é executado em um computador hospedeiro de rede, e a biblioteca cliente, através da qual os usuários em estações de trabalho remotas se conectam e se comunicam com os bancos de dados gerenciados pelo servidor.

O *Firebird* fornece segurança de acesso de usuários ao servidor através de senhas criptografadas e IDs de usuários. Como qualquer servidor de banco de dados, ele baseia-se em acesso físico adequado a rede e na existência de segurança de sistemas de arquivos. O *Firebird* pode armazenar dados criptografados mas, exceto pela criptografia de senhas, não fornece capacidade de criptografar os próprios dados.

O *Firebird* é construído em uma arquitetura multi-geracional (MGA), que provê um dispositivo *versioning* único que assegura alta disponibilidade para usuários que suportam decisões (*support-decision*). Servidores de banco de dados suportam tradicionalmente o modelo de interação de banco de dados *On-Line Transaction Processing* (OLTP), caracterizado por um grande volume de transações simples e curtas. Enquanto o dispositivo *versioning* do *Firebird* suporta essas transações curtas, do tipo OLTP, o *Firebird* gerencia ao mesmo tempo transações de longa duração do tipo *support-decision* (MAGNO, 2009).

O SGBDR permite a criação de funções (UDFs) que provêem meios de aumentar as capacidades analíticas do *Firebird* através da criação de funções habituais de negócios.

UDFs são códigos reutilizáveis que asseguram a integridade e a confiabilidade dos dados. As UDFs podem ser usadas para chamar aplicações externas ao banco de dados (VAZ, 2009).

O *Firebird* permite também a junção de múltiplos bancos de dados, possui sua distribuição livre, é um SGDBR *open-source* e *free*.

O Gerenciamento e reaproveitamento de espaço é feito de maneira que à medida que os dados são inseridos no banco, os arquivos do banco de dados crescem no disco. Partindo desta premissa, será lógico pensar que quando um volume de informações é removida (apagada) do banco de dados, o mesmo deveria diminuir de tamanho, no entanto não é isso que acontece. O *Firebird* sabe que a alocação de espaço em disco é muito mais onerosa à performance do que o reaproveitamento do espaço já alocado com novas informações, sendo assim ele prefere deixar o banco com o mesmo tamanho e reutilizar os espaços marcados como “lixo” para armazenar novas informações.

É possível a realização de *backups* e *restores*. Os *backups* podem ser feitos sem a necessidade de acesso exclusivo ao banco. contendo uma imagem consistente do bando de dados no momento em que o processo foi iniciado, ou seja, independente do tempo decorrido ele vai conter a imagem exata do banco de dados no momento em que ele foi iniciado.

Os dados armazenados em um *backup* não incluem os dados dos índices definidos no Banco de Dados, fazendo com que geralmente os arquivos fiquem bem menores que o arquivo do banco de dados original (CANTU, 2005).

Quando ocorre o processo de *restore* o banco é recriado, bem como todos os seus índices ativos. O *Firebird* utiliza o conceito de árvores BTREE na sua estrutura de índices. Quando o índice é recriado, essa árvore fica balanceada permitindo uma maior eficiência quando utilizados em consultas SQL. Também durante um *restore*, todas as informações dispensáveis (marcadas como lixo) são dispensadas fazendo com que geralmente o banco de dados gerado após um processo de *backup/restore* tenha um tamanho menor do que o arquivo original.

4 *Servlet*

Servlets são responsáveis por tratar requisições e respostas entre aplicações, normalmente são utilizadas em aplicações *Web* juntamente com *WebServices* (Web Services é um termo utilizado atualmente para designar um conjunto de protocolos e regras que, juntos, permitem a realização de chamadas de procedimentos remotos através da *Internet*). Um *servlet*, por exemplo, pode receber dados de uma aplicação *Java SE* por meio de uma requisição HTTP, processar os dados, atualizar a base de dados de uma empresa, e gerar alguma resposta dinamicamente para outra aplicação que fez a requisição.

Servlets não apresentam nenhum tipo de interface gráfica, eles são controlados por um serviço de rede podendo ser um servidor Web. São independentes do servidor utilizado e de seu sistema operacional, o que quer dizer que apesar de o *servlet* estar escrito em *Java*, a aplicação pode estar escrita em qualquer linguagem de programação, obtendo-se o mesmo resultado que seria obtido em *Java*.

Podem atuar como enlace entre o cliente e uma ou várias bases de dados em arquiteturas cliente-servidor de três camadas caso a base de dados esteja em um servidor diferente, ele também pode tratar múltiplas requisições concorrentes, e pode sincronizá-las, permitindo que o mesmo possa suportar sistemas com conferências *on-line*.

O pacote *java.servlet* proporciona classes e interfaces para escrever *servlets*, ela define cinco métodos.

- `service()`: É responsável pelo tratamento de todas as requisições dos clientes.
- `init()`: Responsável pela inicialização do Servlet.
- `getServletConfig()`: Retorna o objeto `ServletConfig` passado pelo método `init()`.
- `destroy()`: Destrói o Servlet, efetuando operações de limpeza da estrutura de dados como a desalocação de recursos do sistema que estejam sendo utilizados.
- `getServletInfo()`: Retorna uma string com informações sobre o Servlet.

Quando um *servlet* acessa uma chamada de um cliente, recebe dois objetos: *ServletRequest* e *ServletResponse*. O *ServletRequest* envia a comunicação do cliente para o servidor, e o *ServletResponse* responde a uma chamada do cliente, enviando do *servlet* para o cliente informações requisitadas.

ServletRequest permite ao *servlet* obter informações como nomes dos parâmetros remetidos pelo cliente, protocolo, nome da máquina remota e nome do servidor que recebeu a solicitação. Permite também uma *stream* de entrada chamada *ServletInputStream* para obter os dados dos cliente que utilizam os serviços como POST e PUT, do protocolo HTTP.

ServletResponse proporciona métodos para responder o cliente, e uma *stream* de saída chamada *ServletOutputStream*, e um *writer* pelo qual o *servlet* pode devolver dados ao cliente.

A classe *HttpServlet* possui métodos que são requeridos pelo método *service* da classe *HttpServlet* para processar requisições baseadas no protocolo HTTP. São eles: *doGet*, *doPost*, *doPut*, *doDelete*, *doHead*, *doOptions*, *doTrace*.

4.1 Funcionamento do *Servlet*

Para que um *servlet* possa ser executado ele precisa de um servidor pra realizar a comunicação entre as aplicações. Um servidor bastante conhecido e popularmente utilizado é o *Tomcat*, por ser gratuito e fácil de adquirir.

O *Tomcat* é um servidor *container* onde são instalados *Servlets* para tratar as requisições que o servidor receber.

Quando um *servlet* é carregado pela primeira vez o método *init()* é chamado pelo servidor HTTP e mantido na memória entre requisições enquanto o *servlet* estiver sendo executado. O método *init()* é responsável por inicializar o *servlet* passando como argumento um *ServletConfig* onde possui configurações do *servlet*.

Após inicializado o *servlet* pode receber ou enviar requisições de outras aplicações que solicitam serviço ao mesmo, através do método *service()*. O método *service()* lê a requisição e produz uma mensagem de resposta através de seus dois parâmetros. Um objeto *ServletRequest* com os dados do cliente que é passado e um *ServletResponse* que representa a resposta do *servlet* de volta ao cliente.

Há duas maneiras do cliente enviar a informação para um *servlet*. A primeira é

enviar os valores dos parâmetros e a segunda é enviar a informação através do `InputStream` que é passado dentro do objeto `ServletRequest`.

O servidor espera para chamar o método `destroy()` até que sejam completadas todas as chamadas de serviço, ou que tenha se passado uma certa quantidade de tempo, ou seja, o método pode ser chamado enquanto alguns métodos como o `service()` possam estar ainda em execução.

5 Linguagem de Marcação XML

Linguagem de marcação são linguagens onde a sintaxe é baseada em marcas (*tags*). Sua vantagem é que são razoavelmente simples e de fácil entendimento. Elas não são linguagens de programação, são uma forma de se ter um texto e marcá-lo, identificando alguma coisa, ou estruturando o mesmo (FILHO,).

XML(*Extensible Markup Language*) foi criado pela W3C(Word Wide Web Consortium) em meados dos anos 1998. O objetivo do XML era ser uma linguagem boa para utilização na *Internet*, suportar grande variedade de aplicações, busca por multi-plataforma, compatível com SGML(Linguagem que deu origem ao HTML), ser fácil de ser criada e escrever programas que processassem documentos XML (MENEZES,).

Ela pode ser aplicada para guardar dados de uma pesquisa no Banco de Dados, guardar dados simples como configurações de software, guardar textos ou qualquer dado que precisamos de forma estruturada. O objetivo de aplicações utilizarem XML é a integração de sistemas e tecnologias diversas.

5.1 Estrutura de um documento XML

Um documento escrito em XML possibilita seu utilizador a possibilidade de definir as suas próprias marcas à medida e conforme as suas necessidades. O documento deve seguir um conjunto de regras que fazem que o mesmo obtenha como resultado um documento que respeite as regras gramaticais definidas permitindo seu processamento mais eficiente do que o HTML.

Um documento XML é composto por:

- Declaração XML;
- Instruções de processamento;
- Comentários;

- Elemento raiz;
- Elementos filhos;
- Elementos vazios;
- Atributos.

```
1  <!-- Declaração XML-->
2  <?xml version= "1.0"?>
3
4  <!-- Instruções de processamento-->
5  <!-- Cria uma instrução de processamento e identifica uma folha de stilo para um documento XML
   -->
6  <?xml-stylesheet type= "text/xsl" href= "stylesheet.xsl"?>
7
8  <!-- Elemento Raiz -->
9  <!-- Em um documento XML só pode existir apenas um elemento raiz e dentro dele pode existir
   elementos filhos -->
10 <peessoa>
11
12 <!-- Elementos Filhos -->
13 <nome>Wagner Lima Chimenez</nome>
14 <cidade>Dourados</cidade>
15 <idade>21 anos</idade>
16
17 <!-- Elemento vazio imagem -->
18 <!-- Atributo nome -->
19 <!-- São elementos que não têm conteúdo próprio, eles possuem atributos para armazenar dados
   sobre o elemento -->
20 <imagem nome="foto.jpg" w="528" h="349"/>
21
22 </peessoa>
```

6 Especificações do Projeto

Para realização do projeto foram escolhidas:

A linguagem de programação para o desenvolvimento da aplicação móvel como sendo *Java ME* (*Java Micro Edition*), porque ela suporta a criação de aplicativos para dispositivos móveis e hoje a maioria dos celulares e *smartphones* já estão sendo vendidos com suporte a essa linguagem.

A linguagem de programação para o desenvolvimento da aplicação *Desktop* como sendo *Java SE* (*Java Standard Edition*) por possuir portabilidade entre sistemas operacionais e contribuindo para sua execução em diferentes plataformas.

O gerenciador de banco de dados para efetuar consultas e manipulações de registros como sendo o *Firebird*, onde o mesmo tem suporte a várias plataformas como Windows, Linux, UNIX, FreeBSD e Mac OS X e possui um alto nível de compatibilidade com os padrões SQL.

Servlets que possuem função de realizar comunicação entre as aplicações e o Banco de Dados na requisição e resposta a informações necessárias a cada uma delas.

O documento XML como forma padronizada de texto para facilitar a leitura e escrita das aplicações sobre registros do Banco de Dados, não limitando que futuramente possam ser desenvolvidas outras ferramentas com a mesma finalidade sem ter a necessidade de que sejam escritas nas linguagens escolhidas acima.

A aplicação *Desktop* irá compor um sistema comercial no ramo de vidraçaria de uma determinada empresa fazendo a emissão de pedidos de venda referentes a produtos oferecidos.

A aplicação móvel ou aplicação cliente será capaz de realizar orçamentos dos produtos que a aplicação *Desktop* conter, contribuindo na realização de vendas.

A comunicação entre essas duas aplicações será realizada de forma que o sistema

comercial *Desktop* com Banco de Dados Firebird instalado em uma máquina servidor forneça o acesso ao banco para a aplicação móvel através de um *Servlet* que irá receber uma requisição de dados vinda da aplicação cliente, e irá tratar essa requisição respondendo através de um arquivo XML que será interpretado por ambas as aplicações (DEV MEDIA,).

O dispositivo móvel por meio de qualquer tecnologia de *Internet* poderá acessar o banco para colher as informações necessárias no momento da venda, e as informações poderão serem armazenadas no aparelho por um período de tempo evitando que o mesmo fique conectado a *Internet* constantemente.

Após a realização do orçamento o dispositivo móvel pode se conectar a *Internet* novamente para gravar as informações no banco de dados da empresa.

7 Implementação - Aplicação *Desktop*

Para a realização da implementação foi feito um estudo dos diagramas orientados a objetos que compõem a aplicação e suas correspondentes telas:

Diagrama de Classes, Figura 7.1, Figura 7.2, Figura 7.3, Figura 7.4, Figura 7.5, Figura 7.6.

DER - Diagrama de Entidade e Relacionamento, Figura 7.7

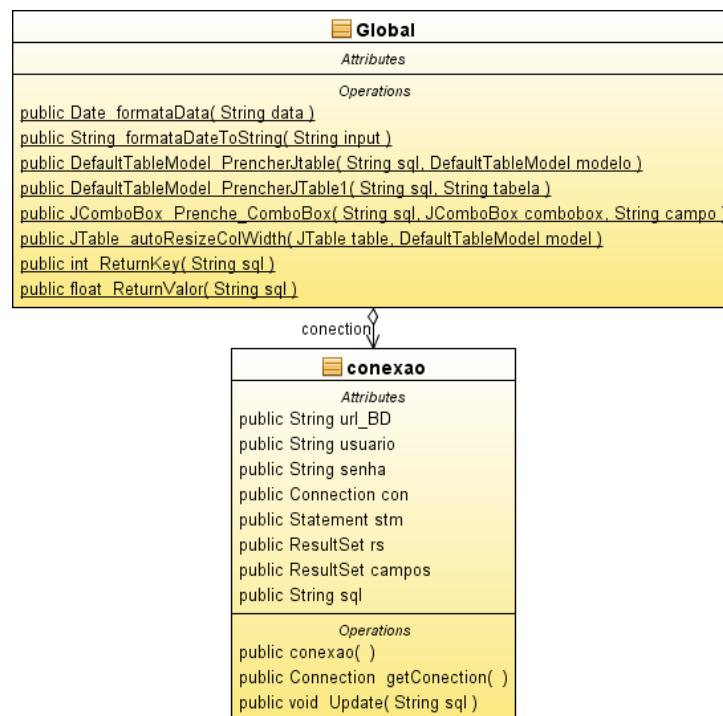


Figura 7.1: Utilitarios

Figura 7.2: Main

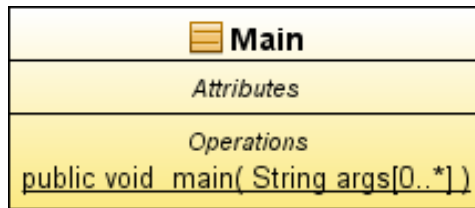
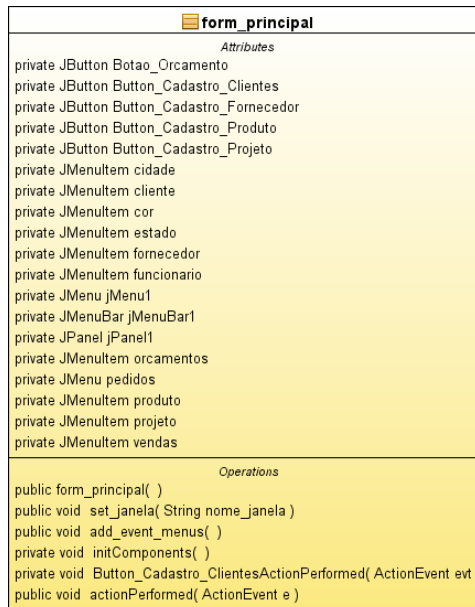


Figura 7.3: Principa



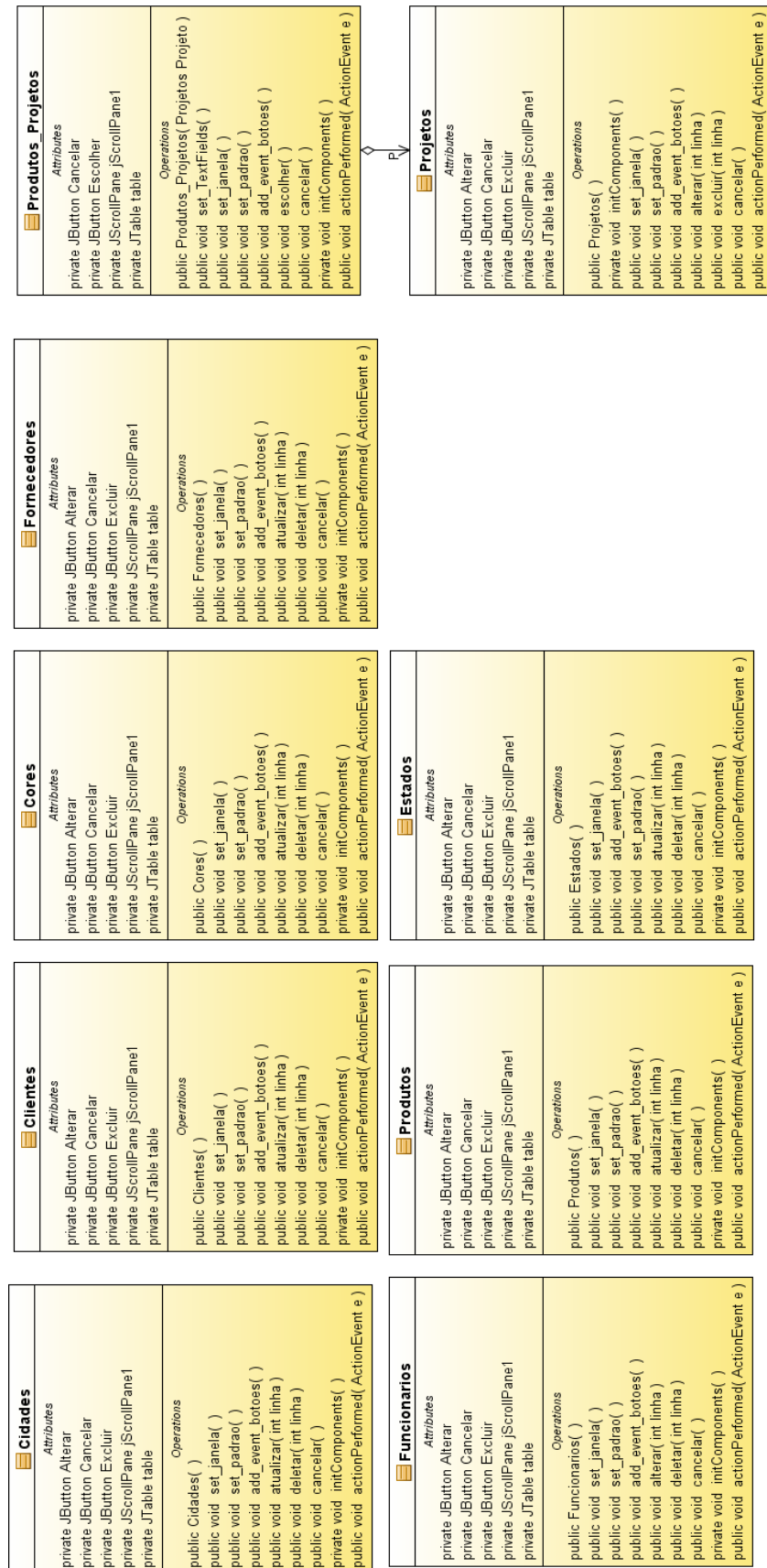


Figura 7.6: Consultas

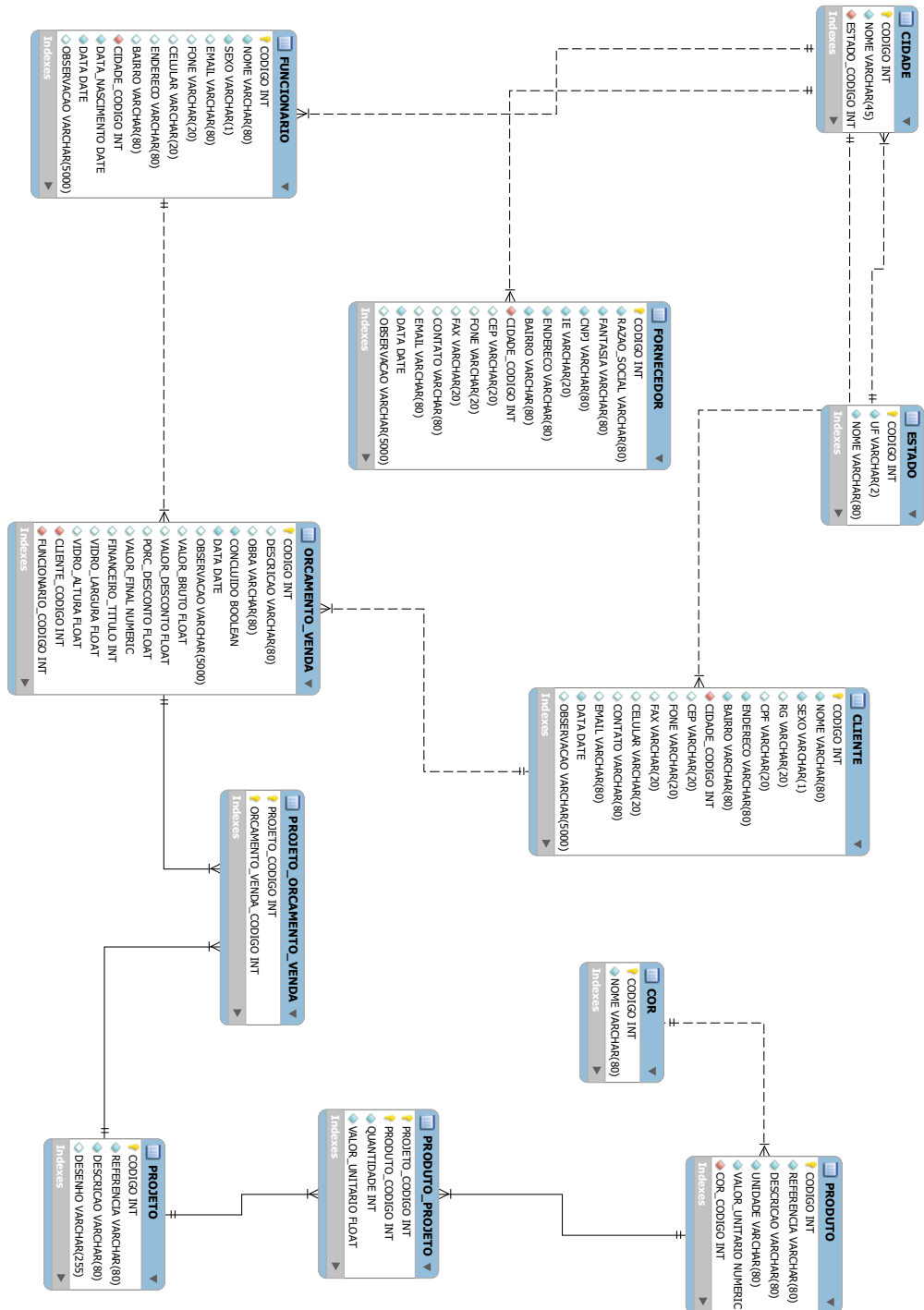


Figura 7.7: DER - Diagrama de Entidade e Relacionamento

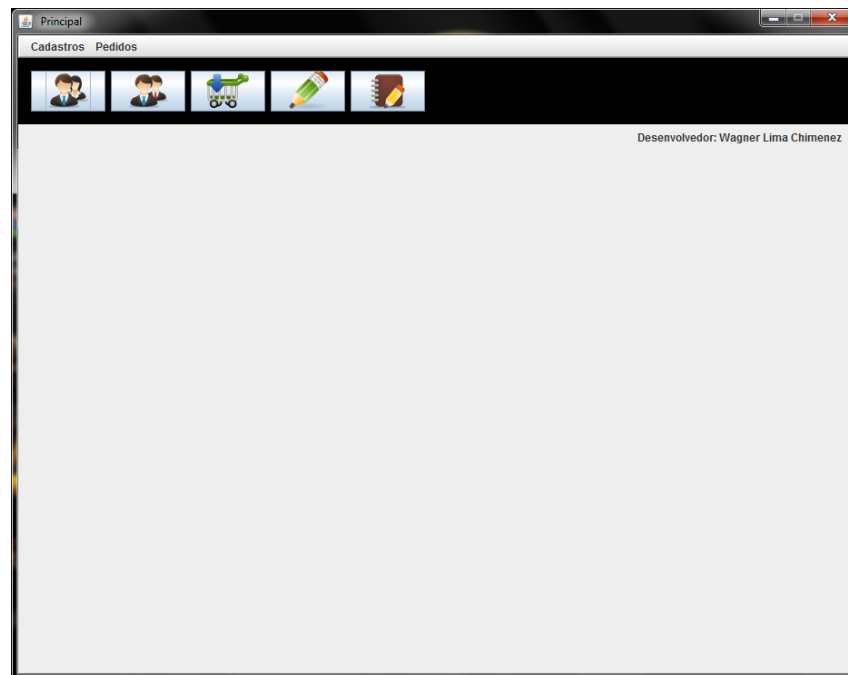


Figura 7.8: Tela Principal: Esta tela da entrada a todas as funcionalidades do sistema, é através dela que são realizados, cadastros, consultas ao Banco de Dados, e pedidos e orçamentos de venda.

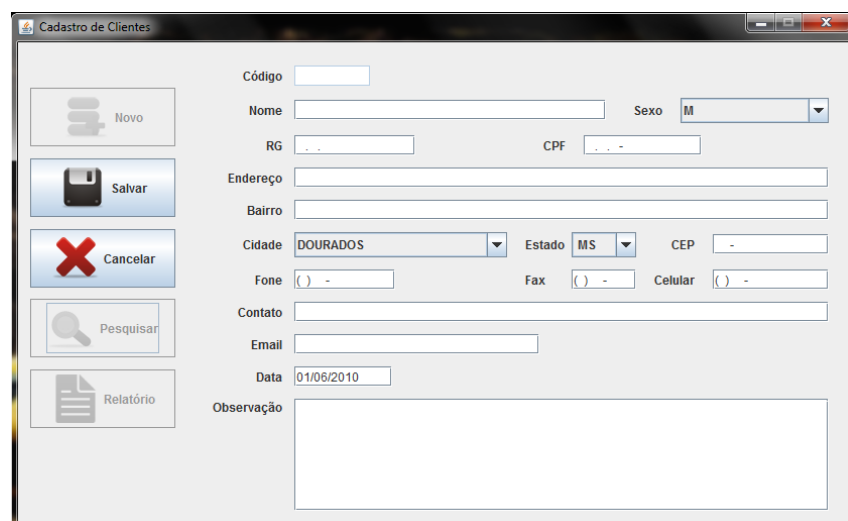
A screenshot of the 'Cadastro de Clientes' window. The window has a title bar with the text 'Cadastro de Clientes' and standard Windows window controls. On the left side, there is a vertical sidebar with five buttons: 'Novo' (with a plus icon), 'Salvar' (with a floppy disk icon), 'Cancelar' (with a red X icon), 'Pesquisar' (with a magnifying glass icon), and 'Relatório' (with a document icon). The main area of the window contains a form with the following fields: 'Código' (text box), 'Nome' (text box), 'Sexo' (dropdown menu with 'M' selected), 'RG' (text box), 'CPF' (text box), 'Endereço' (text box), 'Bairro' (text box), 'Cidade' (dropdown menu with 'DOURADOS' selected), 'Estado' (dropdown menu with 'MS' selected), 'CEP' (text box), 'Fone' (text box with a parenthesis icon), 'Fax' (text box with a parenthesis icon), 'Celular' (text box with a parenthesis icon), 'Contato' (text box), 'Email' (text box), 'Data' (text box with '01/06/2010' entered), and 'Observação' (a large text area).

Figura 7.9: Tela Cadastro de Clientes: Esta tela permite o preenchimento do formulário de cadastro de clientes, onde as devidas informações coletadas são necessárias para realização de pedidos e orçamentos de venda. Assim como esta as demais telas de cadastro preenchem informações necessárias para a realização de vendas.

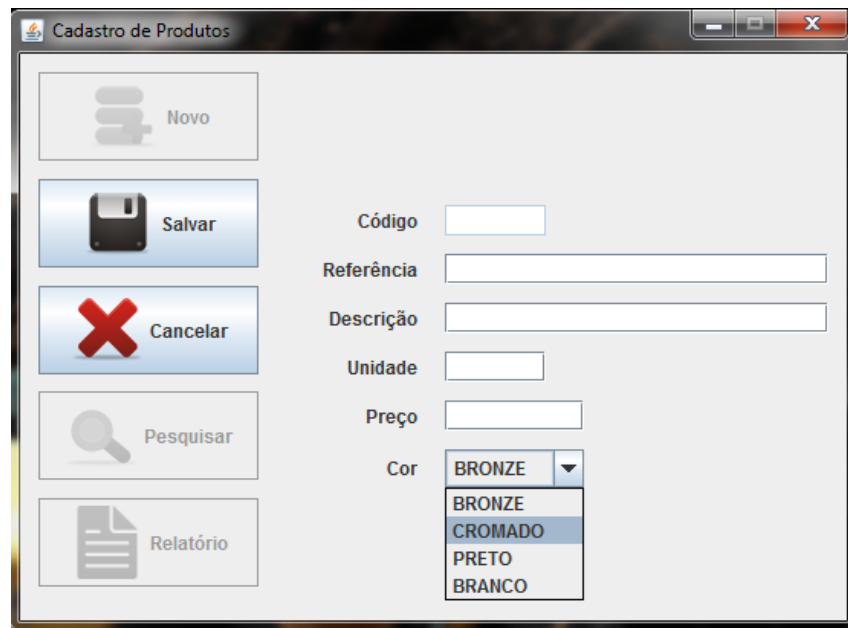


Figura 7.10: Tela Cadastro de Produtos: Esta tela possibilita o cadastramento de novos produtos oferecidos pela empresa, onde os mesmos são fundamentais para realização de pedidos e orçamentos de venda.

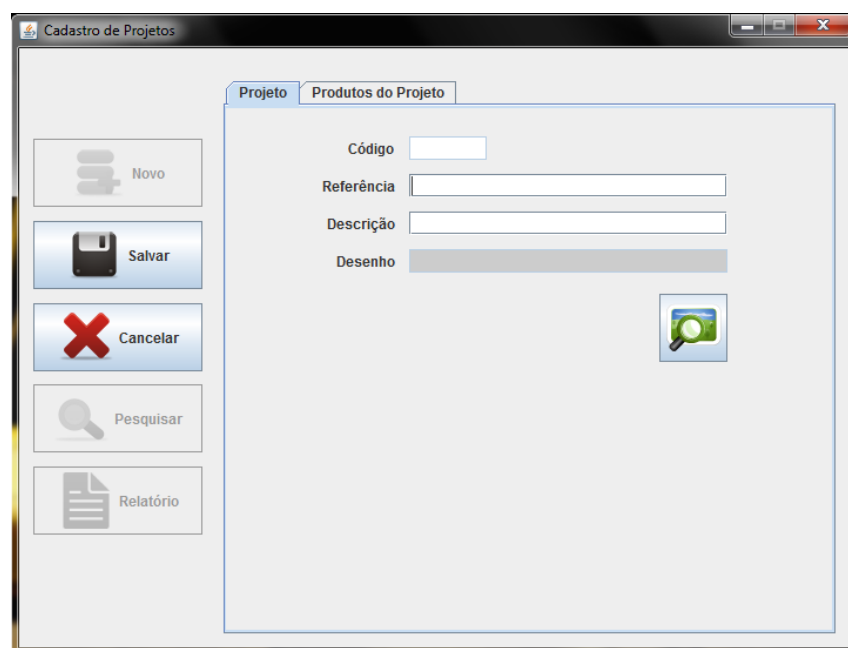


Figura 7.11: Tela Cadastro de Projetos Aba 1

Figura 7.12: Tela Cadastro de Projetos Aba 2: Esta tela possibilita o cadastro de projetos que a empresa oferece, onde o mesmo é composto por diversos produtos que o compõem. Ex: Um projeto cadastrado como PORTA DE ABRIR possui vários produtos que o compõem, como o vidro e suas ferragens.

Figura 7.13: Tela Orcamentos Aba 1

Figura 7.14: Tela Orcamentos Aba 2

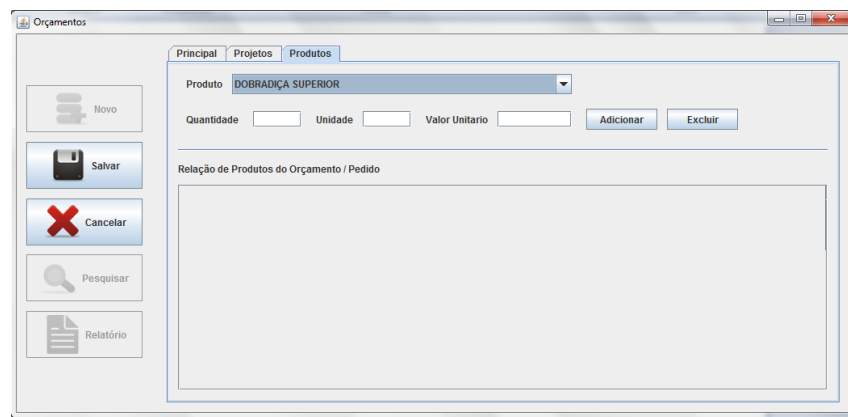


Figura 7.15: Tela Orcamentos Aba 3: Esta tela oferece a possibilidade de realização de orçamentos de produtos ou prestação de serviços que a empresa oferece, ela é dividida em três abas. A primeira é onde são coletadas as informações do pedido, vendedor, cliente e etc. Nela são visualizados os valores da venda. A segunda aba é reservada para os projetos que estarão no pedido, sua cor, e tamanho do vidro utilizado, e a terceira aba se destina aos produtos que o mesmo compõem.

Referências Bibliográfica

BORRIE, Helen. *Dominando Firebird: uma referência para desenvolvedores de banco de dados*. [S.l.]: Rio de Janeiro, 2006.

CANTU, Carlos H. *Firebird Essencial*. [S.l.]: Rio de Janeiro, 2005.

DEVMEDIA. *WebMobile 2 - Conexão HTTP com J2ME*. Disponível em: <<http://www.devmedia.com.br/articles/viewcomp.asp?comp=2936>>.

FILHO, Ademir Constantino. *Entendendo o XML*. Disponível em: <<http://www.guj.com.br/article.show.logic?id=19>>.

LEMAY, Laura; L.PERKINS, Charles. *Aprenda em 21 dias Java*. [S.l.]: Rio de Janeiro, 1996.

MAGNO, Alexandre. 20 motivos para sua empresa utilizar o firebird. p. 3, 2009. Disponível em: <www.comunidade-firebird.org/>.

MENEZES, José de. *Estrutura de um arquivo XML*. Disponível em: <<http://www.plugmasters.com.br/sys/materiais/68/1/Estrutura-de-um-arquivo-XML>>.

MUCHOW, Jow W. *et al. Core J2ME, tecnologia e MIDP*. [S.l.]: São Paulo, 2004.

PUGA, Sandra; RISSETTI, Gerson. *Lógica de Programação e estruturas de dados com aplicações em Java*. [S.l.]: São Paulo, 2009.

SUAPESQUISA.COM. *A História da Internet*. Disponível em: <www.suapesquisa.com/internet>.

UFGRS. *Connected, Limited Device Configuration e Mobile Information Device Profile*. 2009. Disponível em: <<http://www.inf.ufgrs.br/gppd/disc/inf01008/trabalhos/sem01-1/t2/pitoni/>>.

VAZ, Paulo. *Firebird & mysql - parceiros ou concorrentes?* 2009.