UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE SOFTWARE PARA DEMONSTRAR O APROVEITAMENTO DE APLICAÇÕES CLIENTE/SERVIDOR PARA AMBIENTE WEB UTILIZANDO O APPLICATION WEB SERVER DA ORACLE

GIOVANI LUEBKE

BLUMENAU
2003

2003/2-19

GIOVANI LUEBKE

PROTÓTIPO DE SOFTWARE PARA DEMONSTRAR O APROVEITAMENTO DE APLICAÇÕES CLIENTE/SERVIDOR PARA AMBIENTE WEB UTILIZANDO O APPLICATION WEB SERVER DA ORACLE

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Marcelo José Ferrari - Orientador

BLUMENAU
2003

2003/2-19

PROTÓTIPO DE SOFTWARE PARA DEMONSTRAR O APROVEITAMENTO DE APLICAÇÕES CLIENTE/SERVIDOR PARA AMBIENTE WEB UTILIZANDO O APPLICATION WEB SERVER DA ORACLE

Por

GIOVANI LUEBKE

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Marcelo José Ferrari – Orientador, FURB
Membro:	Prof. Marcel Hugo, FURB
Membro:	Prof. Alexander R. Valdameri, FURB

Blumenau, 9 de dezembro de 2003

A família está para o homem como o alicerce está para a casa. Esses anos foram momentos de descoberta, de conquista, de sonhos e de crescimento. Porém, também houve renúncias. Minha ausência em tantos dias ou situações importantes na vida de uma família, gerou tristeza, angústia e saudades. Trago o orgulho e a energia moral de dedicar o resultado do esforço, do trabalho, à minha família.

A natureza deu-nos duas orelhas e uma só boca, para nos advertir que se impõe mais ouvir do que falar.

Zenon de Citium

AGRADECIMENTOS

Agradeço a Deus, grande mestre, pelo dom da vida e por conceder-me um espírito de fortaleza, sabedoria e coragem para atingir minha meta, guiando-me e apoiando-me nesse longo caminho.

À minha família pelo amor presente, certo e possível; pela cumplicidade, pelos simples sorrisos que transmitiam compreensão e pelas palavras de apoio que serviam como sustentáculo psicológico para tantos momentos difíceis que se encontravam, minha eterna gratidão.

Assim como, à minha noiva Daniela que me proporcionou sempre um amor doce, transmissor de paz serena para as horas de total ansiedade e vivente nos mais necessários instantes no decorrer desses anos.

Ao meu orientador e professor Marcelo J. Ferrari que me mostrou o rumo a seguir, me incentivou e desde o princípio acreditou na possibilidade da realização dessa pesquisa, agradeço; assim como a todos os professores do curso de Ciências da Computação, mediadores de conhecimento, que se dedicaram a transmitir seus saberes, deixando marcas, fazendo, assim, parte da minha história de acadêmico.

A todos os companheiros que comigo caminharam, doando seus seres na luta incansável, partilhando experiências e realizando descobertas e, em especial, ao amigo Francisco P. Marinho que com muito esforço, responsabilidade e dedicação, contribuiu de forma significativa para o desenvolver desse trabalho, meu muito obrigado.

RESUMO

Através do Forms da Oracle pode-se aproveitar uma estrutura cliente/servidor para utilização no ambiente *Web* sem a necessidade de iniciar todo o desenvolvimento. Apresenta-se também os recursos principais e as características do Application Web Server através do Forms e quais as diferenças e cuidados na implementação de um software que vai receber dados através de um *browser*. São mostrados o contexto da reutilização de aplicações e um protótipo que exemplifica a conversão de um sistema implementado em Forms 6 para Forms 9i.

Palavras chaves: Forms; Application Web Server; Browser.

ABSTRACT

Through the Forms of Oracle you may use a structure client/server to use on environment web without being necessary to begin all development. Also presents the main resources and the features of the application web server through the Forms which the differences and cares in the implementation of a software which will receive information through a browser. The context of reuse of the applications and a prototype to exemplify a conversion and migration of a system implementing in Forms 6 to Forms 9i.

Key-Words: Forms; Application Web Server; Browser.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Com todos os caminhos levando aos ambientes integrados e às informações e decisões sendo analisadas através da Internet, fica evidente a grande necessidade dos sistemas serem obrigados também a convergir ao mesmo rumo, conforme destaca Corbellini (1998). Um dos grandes problemas encontrados pelas empresas é a migração de suas aplicações utilizadas internamente para ambientes externos, obrigando-as repassar informações de elevada importância para alimentar tomada de decisões que garantam resultados satisfatórios.

McClure (2001) afirma que a reutilização é um atributo importante do software do qual componentes de um sistema devem ser incorporados em outro sistema com a mesma performance. O reuso é a concepção básica ou a idéia de aproveitar uma parte previamente desenvolvida que apresenta função lógica ou estrutura de dados, antes de desenvolver a construção de um novo sistema ou aplicação. É uma prática poderosa que pode resgatar significantes melhorias na produtividade e qualidade, assim como, baixar substancialmente a manutenção e custos do desenvolvimento do software. Poucas empresas têm sido capazes de capitalizar os benefícios que a reutilização oferece.

Justino (1999) enfatiza em seu trabalho as discussões sobre os problemas da produção de software que ocorrem desde 1969, data da qual originou-se a expressão “crise do software”, onde já se pesquisava formas de melhorias no desenvolvimento e manutenção dos programas das empresas.

Como a internet tornou-se nos últimos tempos o maior e mais eficiente meio para o compartilhamento de informações, aguarda-se também, que torne uma grande impulsionadora do reuso das aplicações das organizações e desenvolvedores em geral. Corbellini (1998) menciona que a evolução da internet e das tecnologias a ela agregadas proporciona a distribuição de aplicações e serviços.

As empresas interessadas em explorar estes novos empreendimentos têm a necessidade de aproveitar suas estruturas cliente/servidor, integrando-as e interagindo-as com a rede internacional de computadores, conforme comenta Ferrari (2001).

As aplicações devem combinar potencialidade, heterogeneidade, compatibilidade, confiabilidade e segurança do ambiente cliente/servidor e a facilidade de uso do *browser*.

Segundo Ferrari (2001), existem várias possibilidades para acesso a bancos de dados através da *web*, entretanto, cabe a empresa adotar a melhor estratégia.

Diante dessas afirmações, verifica-se a necessidade de estudos de metodologias, técnicas e ferramentas que auxiliem na migração e reutilização para o desenvolvimento de soluções que atendam as necessidades crescentes do mercado.

O trabalho atribui o interesse em analisar através do ferramental da Oracle possibilidades de evoluir na utilização de aplicações já prontas e disponibilizá-las para o ambiente da internet sem ter que iniciar todo o ciclo de um novo desenvolvimento.

1.1 OBJETIVOS

O trabalho tem como objetivo principal especificar e implementar um protótipo de software, a fim de demonstrar o aproveitamento de um aplicativo com estrutura cliente/servidor, disponibilizando-o para o ambiente *web* utilizando o *Application Web Server* da Oracle.

Os objetivos específicos são:

a) demonstrar com essa migração uma forma de reutilizar aplicações sem ter que iniciar todo um novo desenvolvimento;
b) criar um protótipo de software cliente/servidor para cadastro de fotos de produtos com suas respectivas descrições a fim de facilitar a pesquisa por determinado item de produto;
c) apresentar o processo de geração e suas adaptações para ambiente *web* através do *Application Web Server* da Oracle;
d) avaliar as adequações, vantagens e desvantagens da migração, exemplificando-o nesse específico caso de estudo.

1.2 ORGANIZAÇÃO DO TRABALHO

A seguir estão descritos comentários de cada capítulo apresentado no trabalho.

O primeiro capítulo demonstra uma síntese da origem do trabalho, seu objetivo, relevância e organização.

O segundo capítulo apresenta algumas formas e métodos de como pode ser reutilizada uma aplicação.

O terceiro capítulo demonstra conceitos sobre banco de dados, ambiente cliente/servidor, aplicações e ambiente *web*.

O quarto capítulo apresenta fundamentações em servidores de aplicações na *web*, o *Application Web Server* da Oracle, a utilização do Forms e suas características principais para implementação na *web*.

No quinto capítulo são demonstradas as características do protótipo do sistema, assim como sua análise, projeto e telas.

O sexto capítulo relata a conclusão do trabalho, descrições das dificuldades encontradas e sugestões para novas pesquisas.

2 MIGRAÇÃO DE APLICAÇÕES NO CONTEXTO DA REUTILIZAÇÃO DE SOFTWARE

Com o baixo custo dos computadores e o crescente avanço nos desenvolvimentos de aplicações para suprir necessidades das organizações, os softwares tornaram-se complexos e variados.

Segundo Justino (1999), a complexidade torna a produção lenta com tendências a erros, e as variedades das aplicações acarretam em precipitação, baixando a qualidade, tempo de análise e testes.

A expressão crise do software enfatizava as discussões da produção de software desde 1969, onde já se pesquisava formas de melhorias no desenvolvimento e manutenção dos programas das empresas. Segundo Justino (1999) problemas de baixa produtividade, falta de qualidade, custos elevado e longo tempo empregado são problemas que ainda persistem.

Conforme Surdi (1997), Justino (1999) e McClure (2001) existem algumas estratégias que podem ser adotadas para permitir o melhoramento da produtividade e a qualidade no processo de produção de software:

a) automatizar ao máximo o processo de produção;
b) delegar a produção de software a sistemas especialistas;
c) reutilizar softwares existentes tanto quanto possível;
d) incluir a reutilização de componentes no ciclo de vida do software.

Uma das melhores formas de melhorar a produtividade, qualidade, confiabilidade, reduzir dos custos e amenizar os problemas resultantes da complexidade é a introdução da reutilização no processo de desenvolvimento do software conforme comenta Hamann (1999).

Em seu trabalho Ramos (1998) define a reutilização de software como sendo uma técnica utilizada para reaproveitar partes de um sistema que foi construído e estão disponíveis, na construção de um novo software.

O conceito de reutilização de software abrange não somente o código do programa, mas vai desde as especificações de projeto até a documentação.

Um programa de reutilização requer inúmeros esforços e gastos, pois os obstáculos são vários para as empresas conforme comenta Justino (1999).

2.1 BENEFÍCIOS DA REUTILIZAÇÃO

Os dois principais benefícios que a reutilização oferece às empresas são:

a) diminuição do tempo;
b) redução do custo.

Porém muitos outros benefícios podem ser destacados utilizando os modos de reuso de software segundo Justino (1999):

a) o aumento da produtividade é automaticamente alcançado, pois não é preciso implementar novamente;
b) a qualidade é superada, como os componentes são de outros sistemas e já tiveram seus testes e correções não estão sujeitos a erros;
c) o custo é substancialmente reduzido, com aumento da produtividade e a superação da qualidade reduz-se o tempo com implementação, manutenção além das otimizações nos processos de documentações e testes;
d) permite o compartilhamento de conhecimentos entre grupos de desenvolvimento.

2.2 EMPECILHOS DA REUTILIZAÇÃO

A implantação de uma metodologia para reutilização de não é tão simples. A reutilização conforme Hamann (1999) é utilizada na maioria informalmente.

Surdi (1997) e Barbosa (2002) descrevem questões que dificultam a reutilização do software:

a) a questão do que é reutilizável;
b) dependência de linguagens de programação;
c) a dificuldade em compreender os requerimentos de interface de um componente, seja por documentação insuficiente ou devido a uma excessiva complexidade;
d) entender os efeitos da alteração;
e) a necessidade de modificações extensa e trabalhosa, a dependência de software exterior, pode conduzir à eliminação do componente no desenvolvimento de determinada aplicação;
f) um componente pode custar 25% mais para desenvolver que componentes não projetados para reuso;

g) os ganhos de produtividade não são imediatos e requerem uma série de mudanças técnicas e gerencias que implicam em custos.

2.3 MECANISMOS, FORMAS OU MODOS DE REUTILIZAÇÃO

Em seu trabalho Justino (1999) descreve que qualquer informação necessária ao processo de criação de software pode ser reusada para a criação de um novo software. Para Ambler (1998) alcançar os reais benefícios da reutilização, principalmente em orientação a objetos, deve-se entender as diferentes espécies de reutilizações, onde e como aplicá-las.

Isto leva a concluir que existem diversas formas de reutilização. Yourdon (1995) classifica as formas de reutilização em quatro níveis: reuso de código, reuso de dados, reuso de projetos físicos e reuso de especificações. Yourdon (1995) cita ainda que a reutilização pode também ser abordada principalmente por uma biblioteca de componentes e técnicas de desenvolvimento orientadas a objetos. Já Ambler (1998) descreve algumas formas de reuso na orientação a objetos: reuso de código, de herança, de modelos *(templates)* componente, *framework*, artefatos, padrão e componente de domínio.

2.3.1 REUTILIZAÇÃO DE CÓDIGO

Segundo Ambler (1998), a forma mais conhecida de reutilização é a de aproveitar códigos. Refere-se à reutilização de código fonte dentro de seções de uma aplicação e potencialmente através de múltiplas aplicações. É alcançada compartilhando-se classes, funções e rotinas comuns. O que acontece na realidade, conforme descreve Ambler (1998), é que na maioria das vezes a reutilização de código se faz copiando e depois modificando código existente. Esta é geralmente a única forma de reutilização praticada pelos desenvolvedores.

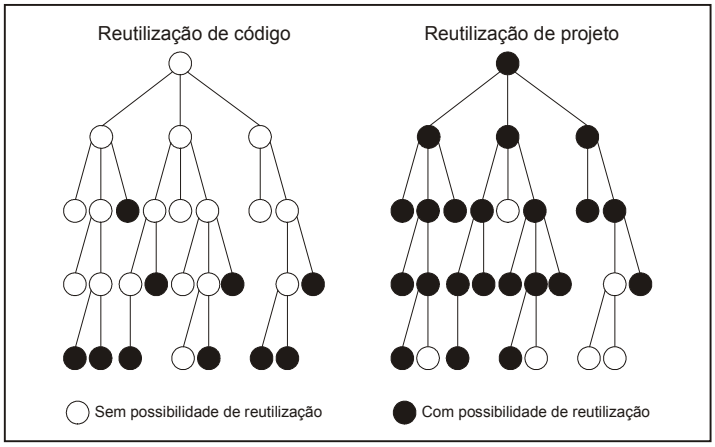
Outro aspecto importante é ter o acesso ao código fonte, quando existe a necessidade de modificação pode ser prontamente realizado. Ao mesmo tempo pode ser útil ou não, pois ao verificar o código, pode-se concluir, de modo geral demoradamente se é compensador ou não reutilizá-lo. A principal vantagem, segundo Ambler (1998) de reutilizar código é que ela reduz a quantidade real de código, diminuindo os custos tanto do desenvolvimento quanto da manutenção. A desvantagem é que o escopo do efeito é limitado à programação e geralmente aumenta o acoplamento dentro da aplicação.

Justino (1999) descreve em seu trabalho que o reuso ao nível de código é limitado à linguagem de que está se desenvolvendo, o sistema operacional e a aplicação. Esse aproveitamento de código geralmente precisa ser alterado e isso facilita a inserção de erros e a falta de controle. O importuno causado com as alterações nos códigos faz com que as empresas utilizem pouco essa técnica de desenvolvimento de sistemas.

2.3.2 REUTILIZAÇÃO DE PROJETOS FÍSICOS

O projeto físico e a especificação são fases do desenvolvimento que são difíceis e consomem muito tempo, por isso, é que com a reutilização alcança-se melhores resultados.

A figura 1 demonstrada por Yourdon (1995), apresenta a diferença entre a reutilização de código, que ocorre somente nos níveis mais inferiores do desenvolvimento, e a reutilização do projeto, que abrange vários níveis desde a fase inicial do plano.



Fonte: adaptado de Yourdon (1995)

FIGURA 1 – Reuso de código *versus* reutilização de projeto

2.3.3 REUTILIZAÇÃO DE ESPECIFICAÇÕES

Segundo Justino (1999) a reutilização de especificações compreende a reutilização de modelos de especificação, como diagramas de fluxo de dados, diagramas de entidade – relacionamento, diagramas de transição de estado, etc. Isso é possível por causa do repositório de dados das ferramentas CASE, onde estão mantidos.

A reutilização de especificações tem grande consideração nas formas de reuso de software, pois permite eliminar esforços envolvidos em projetar, codificar e testar uma implementação conforme descreve Yourdon (1995).

2.3.4 BIBLIOTECA DE COMPONENTES

São exemplos comuns de componentes, partes de um programa como código fonte de subrotinas, as classes no caso de programação orientada a objeto, diagramas, telas, etc.

As bibliotecas de componentes permitem que as empresas organizem partes de softwares evitando problemas na administração das mesmas.

2.3.5 TÉCNICAS DE DESENVOLVIMENTO ORIENTADAS A OBJETOS

Segundo Justino (1999), as técnicas de desenvolvimento orientadas a objetos são oriundas das linguagens que permitem essa metodologia, oferecendo alguns benefícios como, por exemplo, o encapsulamento de dados e funções. Dessa forma o encapsulamento esconde dados e funções que não são do interesse do programador e que são necessários apenas para o componente. A herança é uma das características principais dessa metodologia, onde um componente herda todos as funcionalidades de outro, além de poder facilmente acrescentar ou alterar funções a fim de adaptar.

3 SISTEMA DE BANCO DE DADOS

Segundo Silberschatz (1999), o conjunto formado por um banco de dados mais as aplicações que manipulam o mesmo é chamado de sistema de banco de dados.

Um banco de dados pode ser criado e mantido por um conjunto de aplicações desenvolvidas especialmente para esta tarefa ou por um sistema gerenciador de banco de dados (SGBD). Um SGBD permite aos usuários criarem e manipularem bancos de dados de propósito geral.

Uma característica importante da abordagem banco de dados é que o SGBD mantém não somente os dados mas também a forma como os mesmos são armazenados, contendo uma descrição completa do banco de dados. Estas informações são armazenadas no catálogo do SGBD, o qual contém informações como a estrutura de cada arquivo, o tipo e o formato de armazenamento de cada tipo de dado, restrições, etc.

3.1 ARQUITETURA DE SISTEMAS DE BANCO DE DADOS

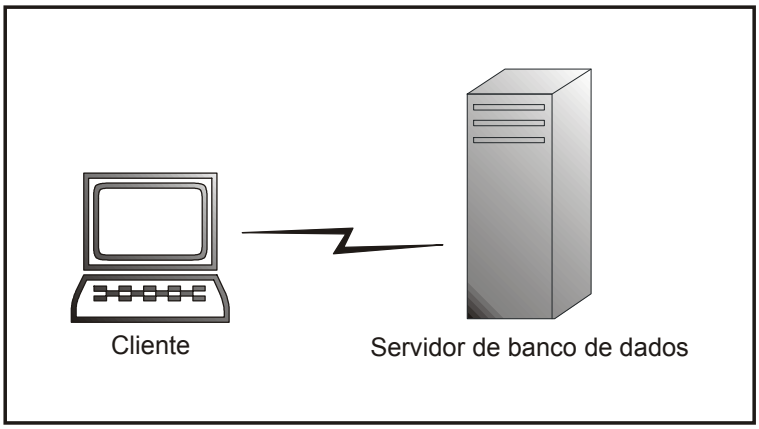
Segundo Silberschatz (1999) a arquitetura de um sistema de banco dados é fortemente influenciada pelo sistema básico computacional sobre o qual o sistema de banco de dados é executado. Silberschatz (1999) descreve as características que influenciam na arquitetura do banco de dados segundo:

a) rede de computadores permite que algumas tarefas sejam executadas no servidor e outras no cliente. Essa divisão de trabalho tem levado ao desenvolvimento de sistemas de banco de dados cliente/servidor;
b) processamento paralelo permite que atividades do sistema sejam realizadas com maior rapidez. Consultas podem ser processadas aproveitando o paralelismo oferecido pelo sistema operacional. Essa necessidade tem levado ao desenvolvimento de sistemas de banco de dados paralelos;
c) distribuição de dados tem se desenvolvido para administrar dados distribuídos geograficamente. A distribuição de dados pelos nós da rede ou pelos diversos departamentos de uma organização permitem que esses residam onde são gerados ou mais utilizados. Manter cópias em diferentes nós permitem as organizações manter operações em seus bancos de dados mesmo quando um nó for afetado.

3.2 AMBIENTE CLIENTE/SERVIDOR

Com o objetivo de demonstrar a evolução das aplicações em ambiente cliente/servidor é apresentado apenas conceitos básicos dessa arquitetura.

Segundo Zacker (2000), o conceito cliente/servidor descreve um sistema computacional no qual o processamento efetivo necessário para efetuar uma determinada tarefa é dividido entre um computador principal e uma estação de trabalho individual. Os dois são conectados por um meio de transmissão. A função primária de cada um em relação ao outro pode ser expressa em termos simples: o cliente solicita serviços do servidor e este responde fornecendo os serviços. Em vez da funcionalidade completa para vários usuários ser responsabilidade de um único computador, como ocorre na situação terminal e *mainframe*, os recursos de processamento do servidor e de todas as máquinas clientes são combinados, tornando-se mais eficiente. A figura 2 ilustra o ambiente cliente/servidor conhecido também como ambiente de duas camadas.



Fonte: adaptado de Fernandes (2002)
FIGURA 2 – Esquema de um ambiente de duas camadas

Isto representa uma evolução da tecnologia em três etapas, conforme descreve Teixeira (2000): *time-sharing*; *resource-sharing*; e, *client / server*.

Na tecnologia *time-sharing* todo o processamento era feito em um único processador, como nos *mainframes* em geral. Na *resource-sharing*, várias unidades processadoras compartilhavam recursos de uma outra unidade central, como em arquivos armazenados em um *mainframe*. Já na tecnologia *client-server*, há processamento tanto nas unidades de clientes quanto na unidade provedor de recursos que é o servidor.

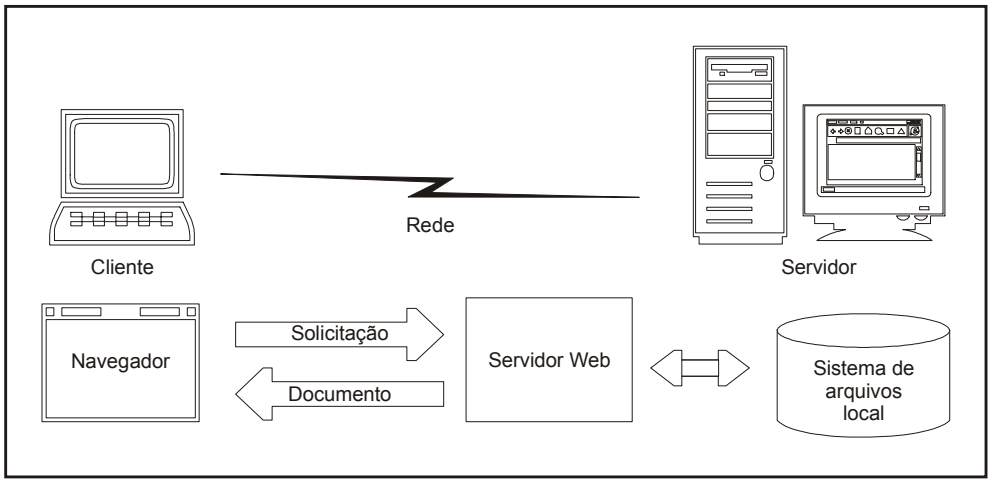
3.3 APLICAÇÕES EM AMBIENTE WEB

Segundo Conallen (2003) as aplicações *web* evoluíram de *sites* ou sistemas *web*. Os primeiros *sites* criados por Tim Berners-Lee enquanto estava na CERN (the European Laboratory for Particle Physics), formavam um sistema hipermídia distribuído que permitia aos pesquisadores acessos a documentos e informações publicados por outros pesquisadores da mesma área diretamente de seus computadores. Os acessos aos documentos eram visualizados com um software denominado navegador (*browser*), uma aplicação que era executada no computador cliente.

Conforme Conallen (2003), com um navegador, o usuário pode solicitar documentos de outro computador e visualizá-lo. Para isso, o usuário deve iniciar o navegador e digitar o nome do computador *host* onde ele pode ser encontrado. O navegador envia uma solicitação do documento para o computador *host*. A solicitação é tratada por um software aplicação denominada servidor *web*, uma aplicação que normalmente é executada como um serviço, que monitora as atividades da rede em uma porta específica, em geral, a porta 80, através da qual o navegador envia uma solicitação especialmente formatada de uma página *web* para o servidor *web*. O servidor *web* recebe a solicitação, localiza o documento no seu sistema de arquivos local e o envia para o navegador, conforme ilustra a figura 4.

Conforme Conallen (2003) esse sistema *web* é um sistema de hipermídia porque seus recursos são ligados uns aos outros. O termo *web* vem da visão do sistema como um conjunto de nós com *links* de interconexão. Vendo por esse lado, ele parece com a teia de uma aranha. Os *links* proporcionam um meio de navegar pelos recursos do sistema. A maioria deles conecta documentos de texto, mas o sistema pode ser usado também para distribuir áudio, vídeo e dados personalizados. Os *links* facilitam a navegação para outros documentos. O usuário só precisa clicar em um *link* do documento e o navegador interpreta isso como uma solicitação para carregar o documento ou recurso referenciado no seu local.

Uma aplicação *web* é desenvolvida e estendida a partir de um sistema *web* para adicionar funcionalidades de negócios. Nos seus termos mais simples, uma aplicação *web* é um sistema que permite a seus usuários executar a regra do negócio com um navegador *web*.



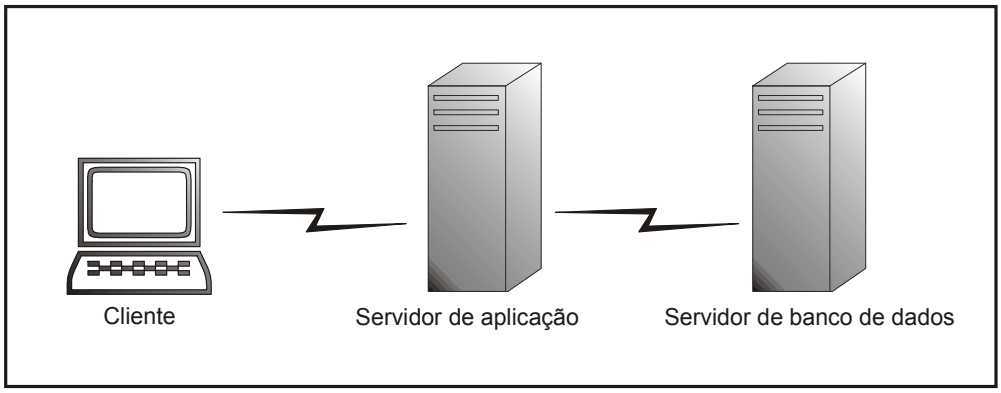
Fonte: (Conallen, 2003)

FIGURA 3 – O sistema *web* básico

3.4 SERVIDORES DE APLICAÇÕES

Servidores de aplicações (*application servers*), segundo Godoi (2002), são softwares que fornecem a infraestrutura de serviços para a execução de aplicações distribuídas. Os servidores de aplicação são executados em servidores e são acessados pelos clientes através de uma conexão de rede. As vantagens dos servidores de aplicação em relação ao modelo cliente/servidor residem nos serviços implementados por eles e disponíveis aos desenvolvedores, fazendo com que eles possam concentrar a maior parte do tempo no desenvolvimento da lógica de negócio. Em geral estes serviços diminuem a complexidade do desenvolvimento, controlam o fluxo de dados, incrementam a performance, gerenciam a segurança. O servidor de aplicação predispõe a utilização da arquitetura chamada de três camadas ou n camadas, que permite um melhor aproveitamento das características de cada componente (servidor de banco de dados, servidor de aplicação e cliente). A primeira camada, chamada *front-end*, usualmente é *browser*, que serve para apresentação e algumas validações. A segunda camada é a aplicação sendo executada no servidor de aplicação. A terceira camada é o servidor de banco de dados.

Os servidores de aplicação priorizam o compartilhamento de componentes e aplicações, fazendo assim com que seja mais fácil o desenvolvimento, manutenção e gerenciamento de sistemas complexos.



Fonte: adaptado de Fernandes (2002)

FIGURA 4 – Esquema de um ambiente de três camadas

3.4.1 CARACTERÍSTICAS DOS SERVIDORES DE APLICAÇÃO

Além das características já citadas, Godoi (2002) comenta sobre outros serviços que também estão disponíveis nos servidores de aplicação:

a) tolerância a falhas através de políticas para recuperação e distribuição de componentes em clones dos servidores;
b) balanceamento de carga através da análise da carga nos servidores, permite a distribuição de clientes de forma a maximizar a utilização dos recursos disponíveis;
c) gerenciamento dos componentes através de ferramentas para a manipulação de serviços, tais como, gerenciamento de sessão, notificação, distribuição da lógica de negócios;
d) garantia da integridade na transação em ambientes distribuídos;
e) gerenciamento de vários servidores de aplicação através de um único sistema gráfico;
f) segurança da aplicação.

Existem várias implementações de servidores de aplicação, em sua maioria implementados na plataforma Java, como exemplos podem-se citar:

a) IBM webSphere Application Server;
b) Oracle9i Application Server;
c) BEA webLogic;
d) SUN iPlanet;
e) Jboss.

Outras implementações existem em outras plataformas, como o Apple Web Objects que roda em MacOS e o Zope Application Server que roda sobre a linguagem Python.

Em geral, os servidores de aplicação rodam em vários sistemas operacionais, como Solaris, Linux e Windows, o que permite que seja possível o desenvolvimento em uma plataforma e sua publicação para produção em outra.

4 ORACLE 9I APPLICATION SERVER

Segundo Godoi (2002), o Oracle 9iAS, assim como é chamado possui um conjunto de tecnologias de integração e colaboração, para aplicativos e processos de negócios.

4.1 SEGURANÇA

O Oracle 9iAS possui mecanismos de segurança que gerenciam o acesso dos usuários à camada intermediária. Com o suporte do protocolo SSL (*Secure Socket Layer*), os dados ficam protegidos durante sua transferência do *browser* para o servidor de aplicação e quando trafegam entre o servidor de aplicação e o banco de dados.

Com o fornecimento de autenticação unificada, implementada no Oracle Login Server os usuários podem acessar todos os aplicativos da *web* autorizados com um único *login*. As credenciais de *login* dos usuários podem ser gerenciadas centralmente com qualquer serviço de diretório compatível com LDAP v3, como o Oracle Internet Directory, conforme descreve Godoi (2002).

4.2 FORMS DEVELOPER R6I

O Forms Developer corresponde a um dos ambientes de produtividade da Oracle – RAD (*Rapid Application Development*), são assim chamados porque permitem construir rapidamente aplicações a partir das definições do banco de dados e podem dar manutenção à base de dados. Estas aplicações tanto podem ser implementadas em ambiente cliente/servidor quanto em uma arquitetura de três camadas ou na Internet, (Fernandes, 2002).

Várias ferramentas fazem parte do pacote de desenvolvimentos Forms Developer, dentre elas o trabalho apresenta o Form Builder 6i, que é utilizado para demonstração da geração do protótipo deste trabalho.

Faz-se também referência a nova versão, Form Builder 9i para ilustrar a produção do aplicativo diretamente a um *browser*.

O Forms *runtime* é o aplicativo para execução dos Forms, menus e bibliotecas PL/SQL em aplicações desenvolvidas.

4.3 O AMBIENTE DE DUAS CAMADAS

Conforme descreve Fernandes (2002) em um ambiente cliente/servidor o Forms *runtime* é executado na máquina do cliente e a atualização dos dados, no ambiente onde se encontra o servidor com o Forms e o banco de dados.

A instalação do Forms *runtime* pode ser feita localmente ou na rede, da mesma forma que as aplicações, porém toda a interface com o usuário assim como todas as lógicas presentes no programa são executadas na máquina do cliente. A exceção se dá por conta das *triggers* e *stored procedures* presentes no banco de dados, ambiente servidor, conforme descreve (Fernandes 2002).

Segundo Fernandes (2002), os equipamentos dos clientes devem ter capacidade de processamento e memória compatíveis com a tarefa de execução das aplicações localmente. O único servidor com capacidade de processamento significativo é o servidor de banco de dados.

4.4 O AMBIENTE DE TRÊS CAMADAS

Segundo Fernandes (2002) os servidores de aplicações na *web* funcionam em um ambiente de três camadas. Ambiente cliente, basicamente, possui um *browser*. Dois ambientes servidores: O primeiro constituído do banco de dados e o segundo, de um servidor de aplicações, onde está instalado o executável do Forms.

Em um ambiente *web*, os serviços de *runtime* do Forms e toda lógica das aplicações são instalados no servidor de aplicação e não mais nas máquinas cliente. Os *triggers* do Forms são processados no servidor de aplicação, enquanto que a interface ocorre no ambiente cliente.

Neste esquema os equipamentos dos clientes não necessitam de uma grande capacidade de processamento. Esta necessidade será transferida para o servidor de aplicação, que não precisa ser constituído de uma única máquina, pode ser um gerenciador, do tipo *pool*, que deverá ser capaz de atender aos requisitos dos clientes (Fernandes 2002).

4.5 FORMS SERVER DA ORACLE

Segundo Fernandes (2002), para atender a esta arquitetura de três camadas, a Oracle desenvolveu um conjunto de programas ou serviços chamados de Forms Services. Através desses recursos pode-se utilizar aplicações desenvolvidas para um ambiente de duas camadas em um ambiente de três camadas como a internet.

Conforme descreve Fernandes (2002) o Forms Services possui basicamente três componentes:

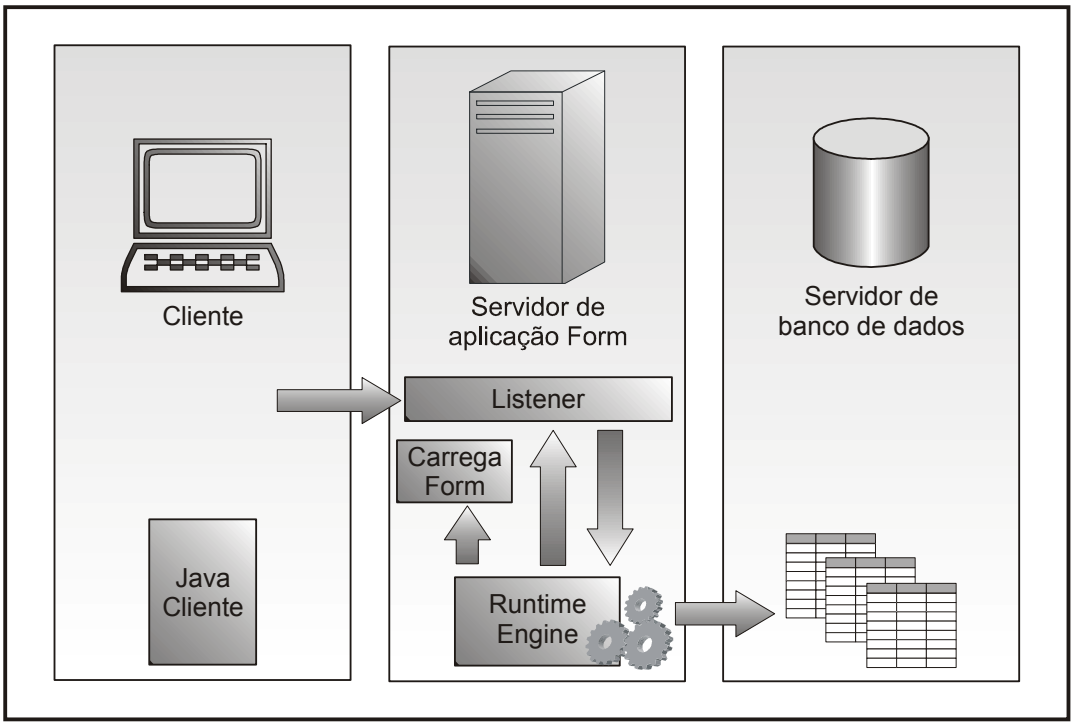
a) Forms *applet*: este componente é executado no ambiente cliente e tem por finalidade realizar a interface entre o cliente (*browser*) e o Forms *runtime*. Este componente pode ser carregado na máquina do cliente dinamicamente, exemplo é quando abre-se uma sessão Forms via *web*. Em um ambiente controlado em uma intranet, por exemplo, pode-se previamente instalar este componente para agilizar o processo de inicialização;
b) Forms *listener*: este componente é executado no servidor de aplicação. Funciona como um ouvidor aguardando que solicitações sejam feitas para estabelecer a ligação entre o cliente e o *runtime*. O *listener* é responsável por iniciar o processo *runtime* ou de manter um *pool* quando chega uma solicitação, para que a conexão com o cliente fique mais rápida;
c) Forms *runtime engine*: este componente também é executado no servidor de aplicação. Tem a finalidade de processar a lógica das aplicações criadas e estabelecer conexão direta com o banco de dados. Age como um servidor recebendo e respondendo às requisições do cliente quando está em comunicação com um cliente, e quando está em comunicação com o servidor de banco de dados age como um cliente solicitando informações.

4.5.1 FLUXO DE PROCESSAMENTO DO FORMS SERVER

O processamento, segundo Fernandes (2002), começa quando utiliza-se um *browser* e aciona-se a URL (*Universal Resource Locator*), da qual indica que uma aplicação Forms deve ser executada. Em resposta a esta solicitação uma página HTML é enviada ao *browser* que também pode ser o arquivo Java que contém a Forms *applet*. Logo a *applet* é iniciada e captura os parâmetros informados, por exemplo, o *login* para enviar ao servidor, indicando que a aplicação Forms deve ser iniciada.

A Forms *applet* envia uma requisição ao *listener* que corresponde a uma porta específica na máquina servidora de aplicação, juntamente com os parâmetros capturados da tela. O *listener* estabelece a conexão entre o cliente e o Forms *server*.

A partir dessa conexão, a comunicação passa a ser feita, diretamente, entre a Forms *applet* e o Forms *runtime* e entre o Forms *runtime* e o banco de dados. Então o *listener*fica disponível aguardando novas requisições de conexão, conforme ilustra a figura 5.



Fonte: adaptado de Godoi (2002)

FIGURA 5 - Fluxo de processamento do Forms *server*

4.5.2 A CONEXÃO DO FORMS SERVICES

O Form Services pode usar três métodos de conexão conforme descreve Fernandes (2002):

a) sockets: uma conexão sockets utiliza uma interface padrão TCP/IP (*Transmission Control / Internet Protocol*). Através de uma requisição a uma porta específica de uma URL com um *browser*, por exemplo, http://www.xxx.com.br:90, o *listener* recebe a requisição e estabelece a conexão. Contudo, as máquinas cliente e servidora devem estar aptas a se comunicarem uma com a outra diretamente na rede. Não é possível o uso de *proxy* no servidor neste modo. O *proxy* é uma característica de segurança invisível ao cliente, que tem a finalidade de impedir acessos não autorizados ao servidor. Se o servidor e o cliente estiverem separados

por uma rede não segura, como é o caso da internet, o uso deste método pode acarretar riscos no que se refere à segurança;

b) HTTP: utiliza HTTP *socket* para a conexão. O *listener*espera por uma conexão HTTP em vez de uma conexão proprietária. A comunicação entre o Forms *Services* e o cliente é encapsulada em pacotes HTTP. Este tipo de comunicação pode ser utilizado em *site* que usam *Firewall*, (proteções utilizadas na Internet). A utilização de *proxy* é completamente transparente para o cliente;

c) HTTPS: forma de comunicação que utiliza o mesmo mecanismo do HTTP, com a adição de regras de segurança SSL (*secure sockets layer*). O Form Services pode utilizar SSL como um protocolo transparente com o objetivo de fornecer privacidade (com criptografia das mensagens), integridade (impede que as mensagens sejam modificadas) e autenticação (a máquina cliente tem condições de verificar se o servidor é quem diz ser). Diversos certificados gerados por companhias além da Oracle estão homologados para uso com o Form Services.

4.5.3 ELEMENTOS DA CONFIGURAÇÃO DO BROWSER

Conforme descreve Fernandes (2002), os usuários poderão utilizar aplicações Oracle Forms na *web* utilizando as seguintes configurações do *browser*:

a) Internet Explorer 5 com o Native JVM (*Java Virtual Machine*) - fornecido pela Oracle através de um arquivo específico, o f60all.cab para permitir que o Forms Java Applet execute dentro do Internet Explorer 5;

b) Oracle Jinitiator *plug-in* usando Netscape Navigator ou Internet Explorer - em vez de utilizar a JVM padrão do *browser*, este *plug-in* define uma JVM no cliente e executará como um *plug-in* para o Netscape e como um componente ActiveX para o Explorer. São fornecidos dois arquivos, o f60all.jar e o f60all_jinit.jar que agrupam as classes necessárias ao serviço no ambiente cliente. Uma vez no cliente são armazenados em memória para uso futuro;

c) Applet Viewer - é um componente JDK (*Java Developer Kit*) que as máquinas cliente podem utilizar para se comunicar com as aplicações Forms. Nesta forma de comunicação o uso de protocolo HTTPS não é permitido.

4.5.4 QUANTIDADE DE SERVIDORES

Segundo Fernandes (2002), a capacidade de balanceamento presente no Form Services permite a otimização dos recursos de hardware. Quando se atinge o limite da capacidade na máquina servidora de aplicação pode-se optar por adicionar mais máquinas ao *pool*, balanceando a execução das aplicações pelas diversas máquinas.

A critério de informação básica sobre a implementação da execução das aplicações Forms, pode-se utilizar tanto o componente Forms Servlet ou Forms CGI. Ambos permitem o balanceamento entre as máquinas. Em ambas as formas utilizam-se o arquivo de configuração formsweb.cfg para especificação dos parâmetros.

As principais diferenças se referem à velocidade de criação dos arquivos HTML, que com o uso de *servlets* é mais rápido, especialmente em situações de grande tráfego na rede. O Forms Servlet também detecta o tipo de *browser* do cliente e gera a página HTML mais adequada, determinando as *tags* corretas.

Em virtude do ambiente Internet, Intranet e Servlets corresponderem uma tecnologia mais atual, é provável que as futuras implementações da Oracle sejam desenvolvidas nesta tecnologia.

4.5.4.1 CONFIGURAÇÕES DO SERVIDOR WEB

A fim de demonstrar a execução de um aplicativo desenvolvido em Forms no ambiente *web*, é necessária a instalação de um servidor *web*. Para o estudo dos requisitos e particularidades do uso dessa ferramenta instalou-se o Forms *Server* localmente.

Na instalação do servidor *web* são criados alguns caminhos virtuais para a instalação dos arquivos de software do Forms. Os diretórios relativos aos softwares estão subordinados aos Oracle Home.

O quadro 1 apresenta estes caminhos virtuais com seus caminhos físicos correspondentes.

Caminho virtual	Diretório físico	Conteúdo
/forms60java/	c:\oraforms\forms60\java	Arquivos java do Forms
/dev60html/	c:\oraforms\tools\web60\html	HTMLs para execução dos Forms
/servlet/	c:\oraforms\forms60\java\oracle\forms\servlet	Servlets (executáveis)
/dev60cgi/	c:\oraforms\tools\web60\cgi	CGIS (executáveis)
/jinitiator/	c:\oraforms\jinit\	Jinitiator (para download)
/dev60temp/	c:\oraforms\tools\web60\temp\	Arquivos temporários do Forms

Fonte: adaptado de Fernandes (2002)

QUADRO 1 – Caminhos virtuais e caminhos físicos

No caso do Oracle 9i os diretórios virtuais são especificados em um arquivo chamado 6iserver.conf.

4.5.5 VARIÁVEIS DE AMBIENTE

Segundo Fernandes (2002), as variáveis de ambiente são aquelas presentes no registrador do Windows. Essas variáveis são criadas no momento da instalação da ferramenta e são descritas adequadamente não necessitando de alterações.

No registrador do Windows, no nó hkey_local_machine\software\oracle encontram-se os subnós com o nome Home0, Home1, etc. A quantidade dependerá do número de produtos instalados em Oracle Homes diferentes. O conjunto destas variáveis contém parâmetros importantes para configurações de ambiente, conforme segue:

a) forms60_path - neste parâmetro informa-se todas as possíveis localizações dos arquivos .fmx, .mmx e .pll. Diversos diretórios podem ser incluídos na lista;
b) forms60_output - indica-se neste parâmetro o diretório físico onde serão armazenados os relatórios gerados;
c) forms60_timeout - especifica-se a quantidade de tempo em minutos em que o processo Forms Services é terminado quando não ocorrerem solicitações dos clientes;
d) forms60_mapping - nesta variável indica-se o diretório virtual relativo ao diretório físico especificado na variável forms60_output;
e) forms60_repformat - especifica-se nesta variável o formato que o relatório gerado através de uma chamada à rotina run_product no Forms terá. Os valores válidos são HTML e PDF;
f) forms60_message_encryption - esta variável pode ser criada, por padrão as mensagens são criptografadas usando criptografia de 40 bits;

g) forms60_wallet - utiliza-se esta variável para fazer a implementação do modo de comunicação HTTPS. O modo *wallet* suporta o certificado usado para a comunicação segura com o servidor. No trabalho sugerido não houve configuração do ambiente servidor, utilizou-se a comunicação via *socket*;
h) Forms60_https_negotiate_down - esta variável é usada em comunicações segura HTTPS para indicar ao servidor se deve ou não aceitar chaves de criptografia oriundas do ambiente cliente com chave de criptografia inferior a 128 bits. Essa variável precisa ser alterada se no ambiente servidor estiver utilizando chave de 128 bits e deseja-se estabelecer comunicação com clientes que usem criptografia de 40 bits.

4.5.6 PARÂMETROS PARA O SERVIÇO DE INICIALIZAÇÃO DO FORMS SERVICE

A seguir descreve-se alguns parâmetros que são usados para a inicialização do Forms Service:

a) port - determina a porta na qual o processo servidor é iniciado, este número deve ser igual ao configurado no arquivo HTML para solicitação da execução de uma aplicação;
b) mode - determina o modo como os serviços do Form serão executados. As formas válidas são Socket, HTTP ou HTTPS;
c) pool - determina o número de conexões adicionais que devem ser disponibilizadas para usuário subseqüentes;
d) log - determina que seja gerado um arquivo de log para acompanhamento das atividades.

Esses parâmetros estão presentes nas ferramentas administrativas do Windows.

4.5.7 CUSTOMIZANDO OS ARQUIVOS DE CONFIGURAÇÃO

No momento que um usuário no ambiente cliente estabelece comunicação pela primeira vez com a aplicação, o arquivo base.html é lido pelo Forms Servlet ou CGI. Todas as variáveis com valores do tipo %nome da variável% são atualizadas com valores corretos obtidos do arquivo formsweb.cfg, conforme descreve (Fernandes, 2002).

O arquivo formsweb.cfg contém um conjunto de parâmetros utilizados tanto por Servlet quanto por CGI.

Todos os arquivos de configuração e os arquivos HTM utilizados pelo Forms podem ser customizados. Apresenta-se neste trabalho apenas comentários, detalhes específicos de configuração podem ser encontrados em Fernandes (2002).

4.6 CONSIDERAÇÕES SOBRE APLICAÇÕES FORMS NA WEB

Qualquer tipo de aplicação para a *web* e não apenas as do tipo Forms devem ser analisadas e discutidas anteriormente. A seguir cita-se algumas considerações:

a) avaliar os fatores que venham a afetar a performance da aplicação, tais como freqüentes idas ao banco de dados para a obtenção de informações ou apresentação de grande quantidade de informação ao usuário;
b) cada vez que uma imagem é apresentada, ela deve ser buscada do servidor de aplicação ou extraída do banco de dados;
c) como num ambiente cliente/servidor a otimização das consultas é importante, num ambiente *web* esta preocupação deve ser ainda maior.

Alem disto, a seqüência de tópicos a seguir se refere especificamente a aplicações Forms, conforme descreve Fernandes (2002):

a) uma das sugestões da Oracle é que o usuário crie seu próprio arquivo HTML base. Com isso pode-se diminuir os parâmetros do arquivo formsweb.cfg e modificar este HTML incluindo textos, imagens pequenas, título da janela, etc;
b) outra sugestão é de ter uma página HTML funcionando como menu principal para os diversos outros sistemas. Dessa forma ter-se-ia somente um endereço de entrada. Isso elimina a necessidade de distribuir a URL de cada um dos sistemas;
c) seguem dois aspectos sobre a preocupação com o tráfego na rede:
   - *triggers* de *mouse* – *triggers* do tipo *when-mouse-click*, *when-mouse-doubleclick*, *when-mouse-down* e *when-mouse-up* deve, se possível, ser eliminada uma vez que freqüência de uso destes *triggers* poderá promover uma queda na performance,
   - *timers* – sugere-se eliminar ou diminuir o número de vezes em que é disparado. A implementação de JavaBeans pode realizar o mesmo tipo de controle, porém não requer a intervenção do Forms Service,

d) a redução do número de itens imagens ou imagens de fundo, tanto para Forms quanto para Reports, é outro item que deve ser analisado. Uma vez que elas são baixadas no cliente a cada vez que forem apresentadas em uma tela da aplicação;

e) a padronização de fontes é um aspecto a ser analisado uma vez que nem todas as fontes estão disponíveis em todas as plataformas. Deve-se verificar anteriormente as fontes padrões que estão disponíveis em diferentes plataformas para que se possa estabelecer um padrão e minimizar a necessidade de mudanças imprevistas. Quando a fonte definida não é encontrada na plataforma o Forms tenta usar uma similar. Durante a execução o Forms Services mapeia as fontes do Forms em sua equivalente Java. A Java, então, determina a fonte associada para a plataforma de destino;

f) recomenda-se que os objetos similares botões, *text items*, etc fiquem juntos no *Object Navigator*. Essa ordenação de objetos faz com que as propriedades enviadas para o cliente para apresentação da aplicação possam utilizar um algoritmo de compressão de dados chamado de *message diff-ing*, que diminui de forma eficiente a quantidade de informação trafegada. Caso haja a necessidade de modificar propriedades de diversos itens dinamicamente, deve-se ordenar estas modificações pela propriedade e não pelo item. Em relação ainda às propriedades, quanto mais similares elas forem dos itens mais eficiente se torna o algoritmo *message diff-ing*. Desta forma sugere-se:
   - utilizar os valores padrões das propriedades e modificar somente aqueles atributos indispensáveis ao objeto,
   - utilizar *smart classes* para descrever grupos de objetos,
   - determinar o número mínimo de atributos visuais,

g) a propriedade *prompt* deve ser sempre preenchida em vez de usar um *boilerplate* de texto;

h) todos os *boilerplates* são carregados na fase de inicialização da aplicação. Os *boilerplates* retângulos e linhas são mais otimizados dos que os arcos, círculos e polígonos, portanto torna-se interessante o uso destes;

i) um *event bundle* é enviado ao servidor cada vez que um evento de navegação termina, portanto é interessante realizar a menor quantidade possível de navegações. Deve-se tentar preencher o conteúdo dos campos dos valores *default*, para conseguir um ganho na performance. A aplicação deve estimular o usuário à

rapidamente encerrá-la, o que fará com que todos os demais eventos associados à navegação sejam disparados em conjunto, enviando um único *event bundle*;

j) deve-se tentar reduzir o tempo necessário para construção da tela inicial, tendo somente a quantidade indispensável, por exemplo, título, logo, *username* e *password*;

k) para que os objetos que não forem visíveis de imediato não sejam carregados, pode-se colocar na propriedade *raise on entry=yes* junto com *visible=no*. Isso pode ser especialmente útil para *canvas* do tipo *tab*, onde todos os elementos de todas as pastas são carregados quando esta se torna visível. Outra alternativa para diminuição do tempo de carga é dividir a *canvas tab* em diversas *canvas stacked* que são independentes umas das outras;

l) a utilização de *triggers* do tipo *when-validate-item*, também deve ser analisada e reduzida, uma vez que ele é processado pelo Forms Services. Deve-se avaliar o custo e benefício de substituir a funcionalidade *default* dos itens presentes no ambiente cliente com componentes Java. Desta forma a validação do item seria enviada para o cliente pois estaria contida no item;

m) sugere-se que, grandes aplicações sejam divididas em pequenas, a fim de que o tempo de apresentação de cada tela seja reduzido uma vez que nem todos os objetos precisarão ser inicializados.

4.7 RESTRIÇÕES PARA APLICAÇÕES FORMS NA WEB

Determinadas características habituais no desenvolvimento de aplicações Forms podem não ser suportadas ou podem ocorrer de formas diferenciadas:

a) ActiveX, OCX, OLE, VBX – estas características não são suportadas uma vez que os aplicativos são executados no ambiente *server* e os clientes não podem visualizar os resultados;

b) *when-mouse-enter*, *when-mouse-move*, *when-mouse-leave* – estes *triggers* não são suportados uma vez que o nível de tráfego com o servidor de aplicação seria tão intenso que tornaria inviável a execução da aplicação;

c) Firewall – em caso de se desejar ter um ambiente seguro pode-se usar este recurso, portanto, deve-se usar o modo HTTP ou HTTPS;

d) Host, Ora-Ffi, User_exit – o uso destas funções se dará no servidor de aplicação, no entanto deve-se estar cientes de que se elas retornarem informações para o cliente,

deve-se modificar este uso, pois o usuário cliente ficará sem a possibilidade de apresentação;

e) Botões icônicos – o uso de imagens nos botões é factível, deste que o formato dos ícones seja GIF ou JPG. O formato .ICO não é suportado.

4.8 EXEMPLO DE UMA MIGRAÇÃO DE APLICAÇÃO

De acordo com Ribeiro (2003) a utilização da arquitetura em vias de conversão já usada em 1996, ainda hoje não é comum nos sistemas de informação. Um exemplo de utilização dessa migração é do Sistema de informação da faculdade de engenharia da universidade do Porto (SiFEUP).

O (SiFEUP) vem ao encontro deste trabalho na consolidação dos dados estruturados numa base de dados relacional, no caso Oracle 8 e o acesso à informação por navegadores *web* comuns do lado dos clientes, com recurso a um servidor de HTTP (*HyperText Transfer Protocol*) especializado na ligação a bases de dados. Utiliza-se o Oracle *Web Server* do lado do servidor, o qual também serve as páginas HTML (*Hyper Text Markup Language*) correspondentes à componente de informação não estruturada.

Em seu trabalho Ribeiro (2003), propondo atualizações e refinando o sistema existente, apresenta a utilização de ferramentas de desenvolvimento de aplicações como o Oracle Designer, que permitiram manter a coerência do software e aumentar a produtividade, pela sua capacidade de gerar aplicações de introdução de dados em Forms de maneira praticamente automática e de gerar primeiras versões de módulos para a *web*, que posteriormente necessitaram de verificações e complementos.

A tecnologia da *web* está predestinada, pela sua simplicidade, disponibilidade e versatilidade como interface preferencial para a navegação informal em documentos relacionados. Contudo, Ribeiro (2003) comenta que não é uma forma eficiente de fazer pesquisas em dados estruturados, pois os usuários apenas vêem uma parcela do potencial de partilha de recursos e de trabalho cooperativo. Os sistemas de bases de dados já atacaram, com sucesso.

A maior parte das páginas visíveis são dinâmicas, isto é, criadas no momento do pedido, a partir de informação recolhida diretamente da base de dados e de rotinas PL/SQL (*Procedure Language / Structured Query Language*) de formatação para HTML.

5 ANÁLISE E PROJETO DO SISTEMA DE INFORMAÇÕES DE PRODUTOS (SIP)

Para ilustrar a migração de sistemas cliente/servidor para o ambiente *web* apresenta-se um protótipo de sistema para gerenciar um banco de imagens, aproveitando-se de uma necessidade atual de uma empresa.

A análise do protótipo foi desenvolvida segundo o método da análise essencial.

5.1 ANÁLISE ESSENCIAL

Segundo Tonsig (2003) a análise essencial pode ser considerada um refinamento da análise estruturada. O problema existente é estudado, porém não é modelado: os esforços são concentrados na identificação das funcionalidades lógicas requeridas para o software que será criado e, a partir daí, cria-se um modelo essencial do software que será desenvolvido, não se incorporando as exigências físicas.

Conforme Tonsig (2003) a premissa básica na análise essencial é descrever o sistema de maneira independente de restrições tecnológicas, o que poderia antecipadamente impor alguma restrição à solução pensada. Deve-se considerar na confecção do modelo essencial a existência de uma tecnologia perfeita.

5.1.1 DESCRIÇÃO DE COMO SÃO REALIZADAS AS OPERAÇÕES NO DEPARTAMENTO DE DESENVOLVIMENTO DE PRODUTOS

Atualmente existem cerca de quarenta mil fotos de produtos que foram desenvolvidos nos últimos seis anos. Muitas fotos estão armazenadas em pastas, outras em arquivos de aço. São um total de doze armários, divididos em tipos de produtos e qualidade conforme ilustra foto em anexo B. Cada foto está em um envelope de plástico, contendo um cartão datilografado com os dados do produto, artigo, tamanho, gramatura e cliente. As fotos são ordenadas pelo código do produto.

O gerenciamento e a administração são realizados por meio de uma estrutura organizada por um assistente de produtos que obedece ao fluxo funcional demonstrado na figura 6.

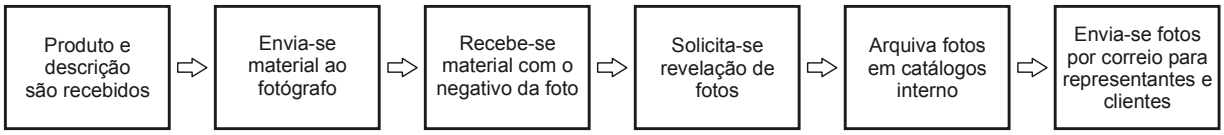


FIGURA 6 – Fluxo funcional

Da forma como se encontra o armazenamento das fotos, possibilita apenas encontrar uma foto caso conheça-se o número do código do produto. As demais informações tornam-se inativas, com isso os serviços são mais demorados, acarretando problemas nas pesquisas e impossibilitando uma emissão de dados referentes a todos produtos com fotos cadastradas.

5.1.2 DECLARAÇÃO DOS OBJETIVOS

O protótipo do aplicativo sugerido tem por objetivo agilizar os processos que ainda são realizados de forma inteiramente manual e agregando custos que não submetem mais na realidade dos negócios atuais.

5.1.3 CARACTERÍSTICAS DO SISTEMA

O software que se apresenta é chamado de Sistema de Informações de Produtos, SIP e será dotado de uma única base de dados, com o objetivo de centralizar todas as informações necessárias para o gerenciamento das imagens dos produtos. Outras informações além das descritas no cartão anexo as fotos serão adicionadas durante o cadastro do produto para facilitar a pesquisa de um determinado item.

5.1.4 DIAGRAMA DE CONTEXTO

Diagrama de contexto reflete graficamente a relação do sistema com o meio ambiente onde está inserido. Essa relação dá-se por meio do recebimento de estímulos do meio ambiente, os quais ativam processos, e estes, por sua vez geram respostas, (Tonsig 2003).

A figura 7 demonstra o diagrama de contexto do protótipo.

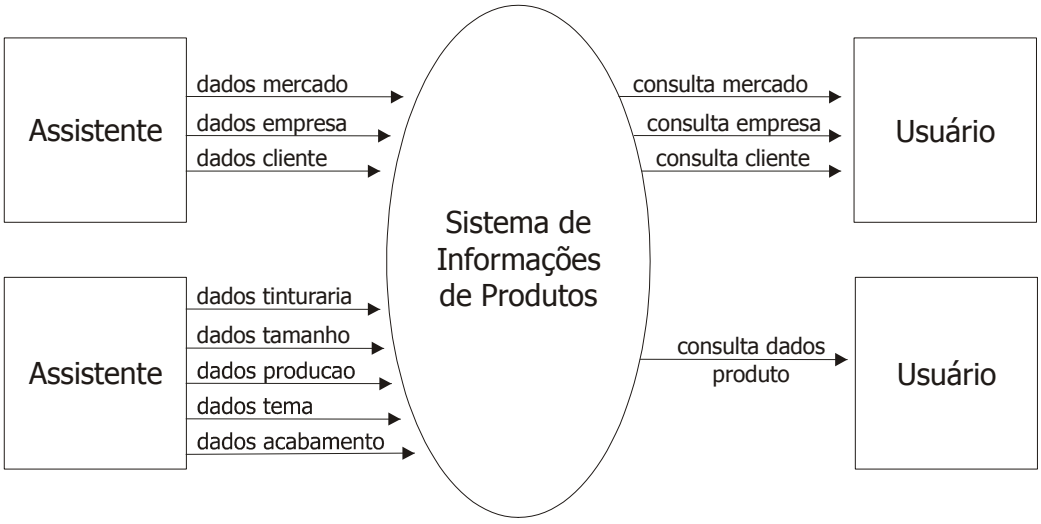


FIGURA 7 – Diagrama de contexto

5.1.5 LISTA DE EVENTOS

Segundo Tonsig (2003), a lista de eventos trata-se da especificação dos processos essenciais que o sistema terá. Tais processos serão ativados por estímulos, executam processamento e geram respostas. O quadro 2 apresenta a lista de eventos do protótipo.

01 - Assistente cadastra mercado.
02 - Assistente cadastra empresa.
03 - Assistente cadastra cliente.
04 - Assistente cadastra tinturaria.
05 - Assistente cadastra tamanho.
06 - Assistente cadastra tema.
07 - Assistente cadastra producao.
08 - Assistente cadastra acabamento.
09 - Usuário consulta mercado.
10 - Usuário consulta empresa.
11 - Usuário consulta cliente.
12 - Usuário consulta produto.

QUADRO 2 – Lista de eventos

5.1.6 DIAGRAMA ENTIDADE RELACIONAMENTO

Segundo Tonsig (2003), o DER auxilia o analista a mapear como os dados estão organizados e como eles se relacionam.

A figura 7 ilustra o DER do protótipo proposto. O DER foi criado a partir da ferramenta Power Designer assim como o *script* para criação das tabelas no Oracle 9i.

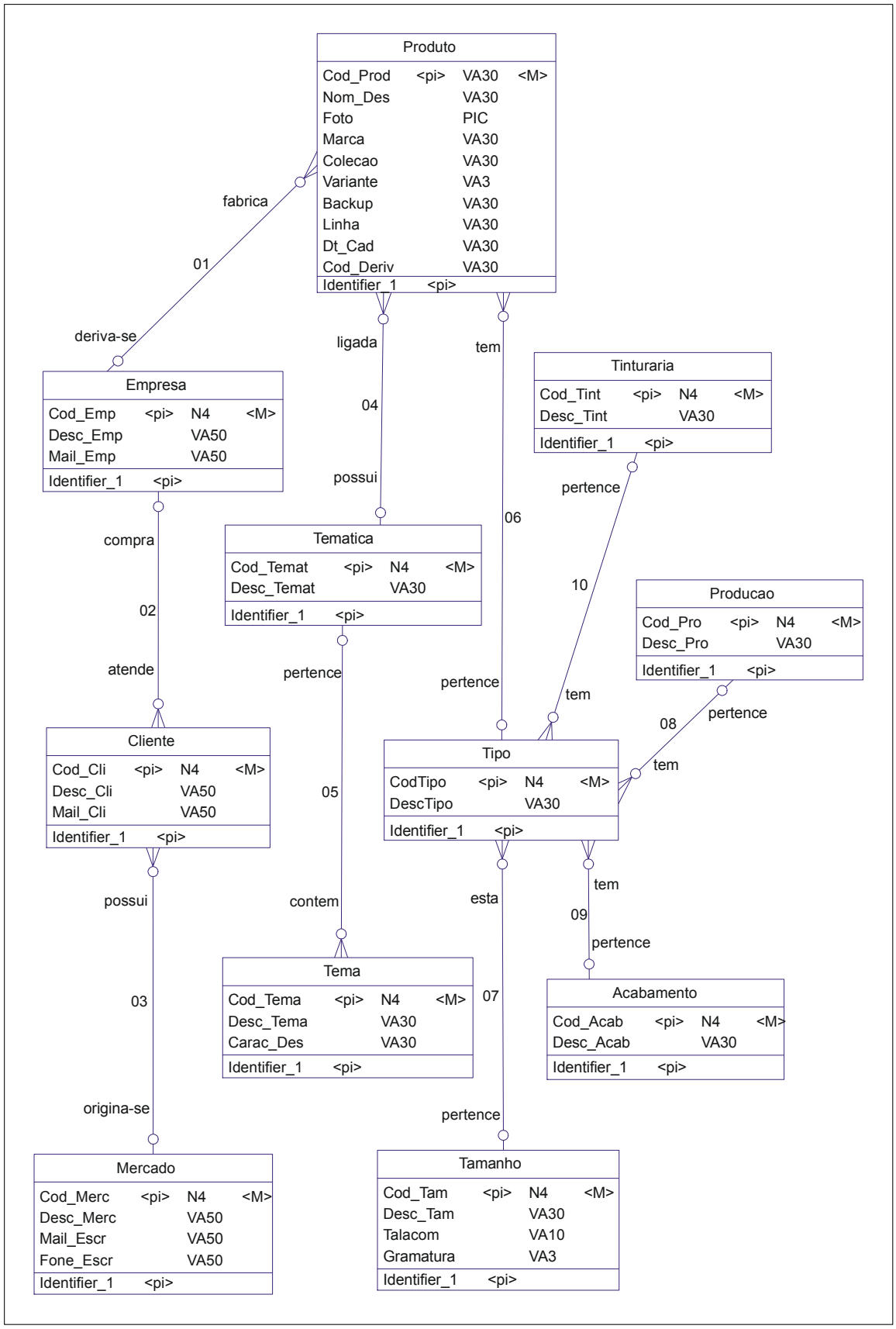


FIGURA 7 – DER do protótipo sugerido

5.1.7 IMPLEMENTAÇÃO

O protótipo foi implementado inteiramente no Forms 6i. A partir dos arquivos gerados nessa implementação, copiou-se os mesmos para um novo diretório. Neste caso, o diretório que recebeu os arquivos foi o diretório padrão oracle\developer\forms90 a fim de poder compilá-los na versão do Forms 9i. É necessária a compilação na nova versão para poder executar o sistema. É importante salientar que uma vez compilado no Forms 9i, a versão 6i não consegue executá-lo novamente.

Foi alterado no arquivo do Forms o caminho onde se encontra o arquivo de menu do sistema. Além dessa alteração não foi necessário nenhuma outra, nesse caso específico de estudo.

5.1.8 TELAS DO SISTEMA

O sistema contém telas de cadastros das informações dos produtos e de consultas dos mesmos. São apresentadas as principais telas geradas através do Forms para ambiente cliente/servidor e também as que foram geradas automaticamente para a *web* sobre o *browser* da Microsoft Internet Explorer. Em anexo A está descrito as teclas de atalho do Forms 6i e as do 9i.

FIGURA 8 – Tela inicial do sistema

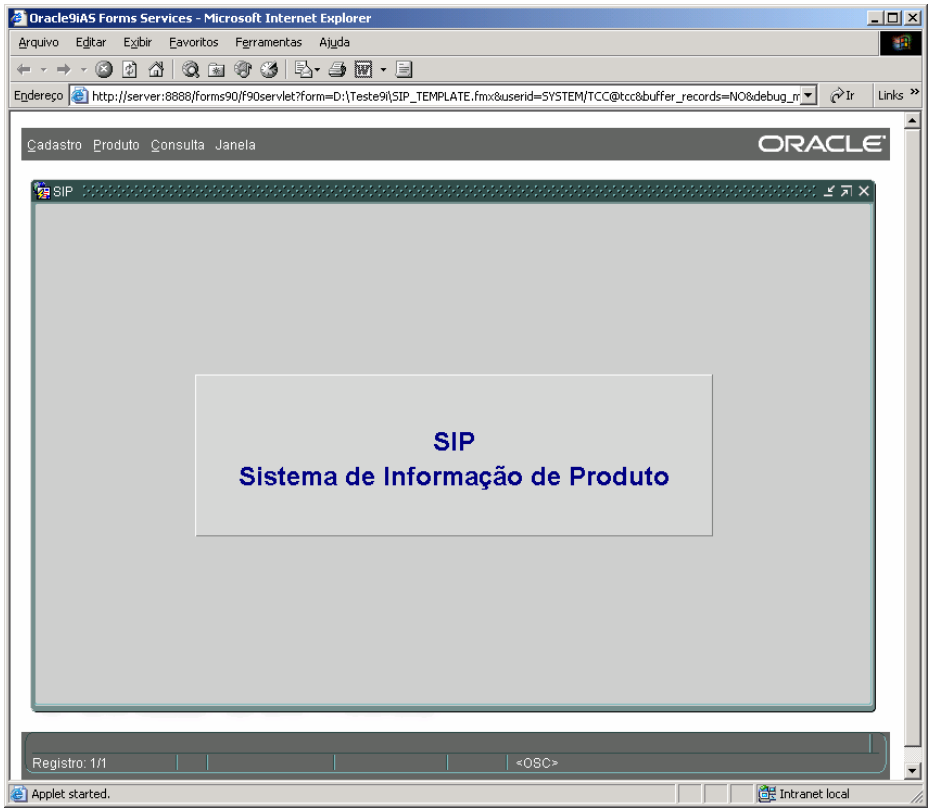


FIGURA 9 – Tela inicial do sistema gerada no Forms 9i

A figura 10 apresenta o menu de cadastro onde o usuário assistente irá cadastrar informações do cliente, mercado, empresa e itens do produto.

FIGURA 10 – Tela inicial do sistema ilustrando o menu cadastro

Na tela de cadastro do cliente, figura 11, está os dados de contato do cliente com seu respectivo mercado de atuação.

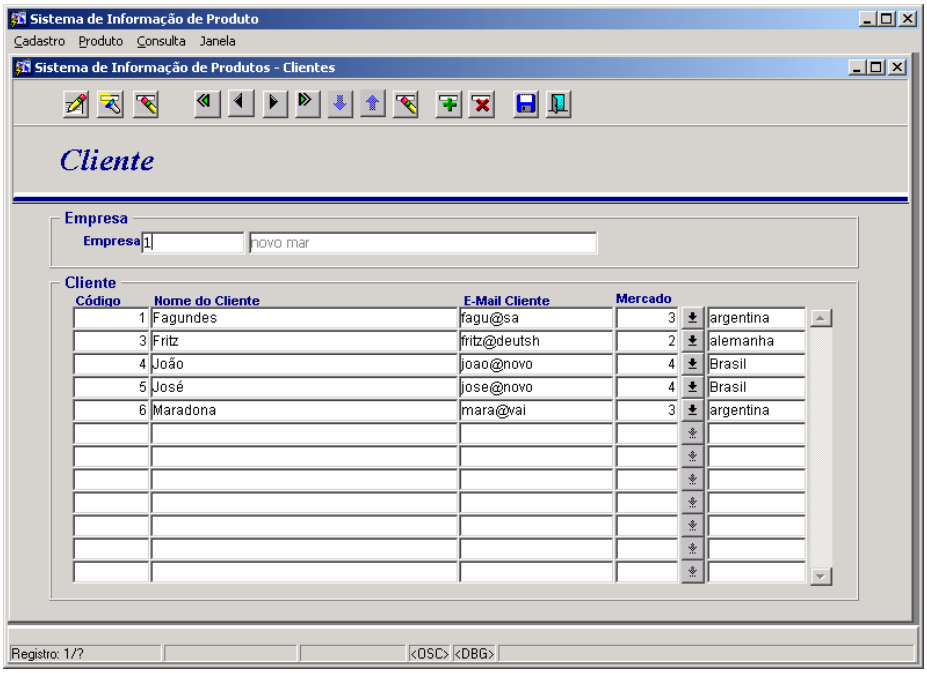


FIGURA 11 – Tela de cadastro de clientes

No menu cadastro, itens de produto, tipo de produto é cadastrado os tipos de produto com sua forma de estampa, tamanho e acabamento, conforme ilustra a figura 12.

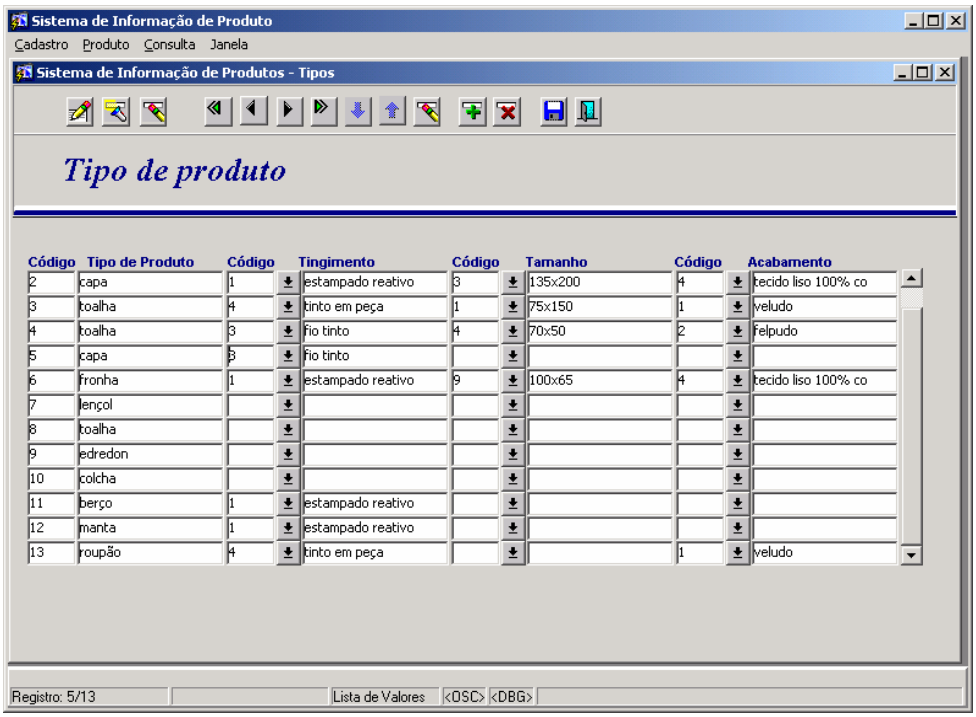


FIGURA 12 – Tela de cadastro de tipos de produtos

A figura 13 mostra a tela de tipo de produto gerada pelo Forms 9i.

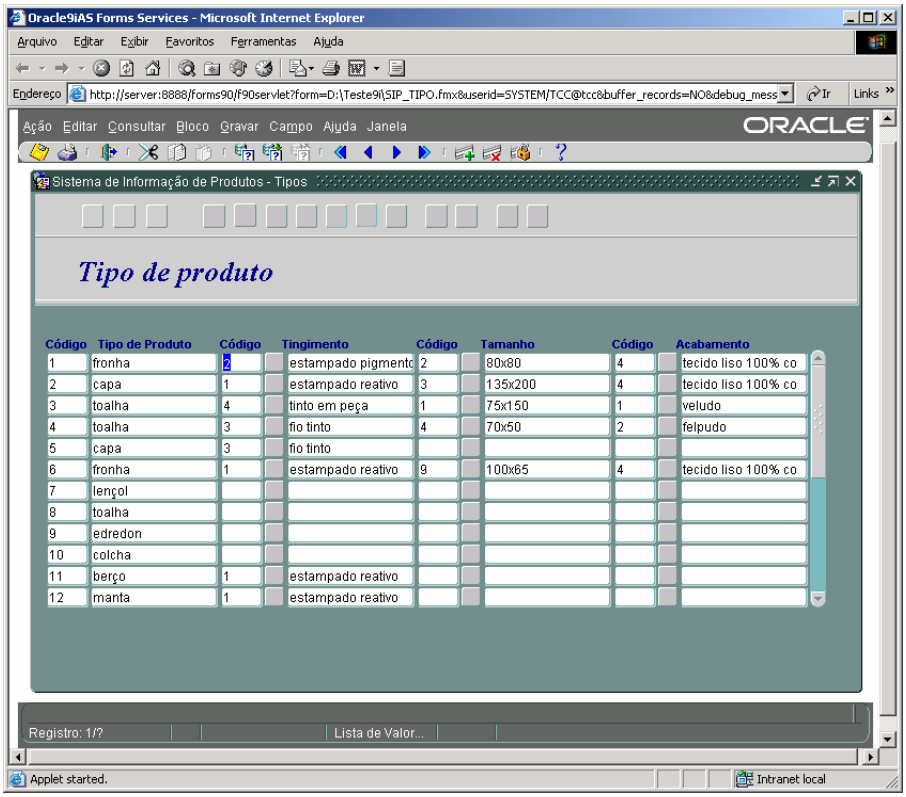


FIGURA 13 – Tela de cadastro de tipos de produtos sobre o Internet Explorer

No menu produtos, dados técnicos são informados todos os itens de produto assim com sua respectiva foto, conforme ilustra a figura 14.

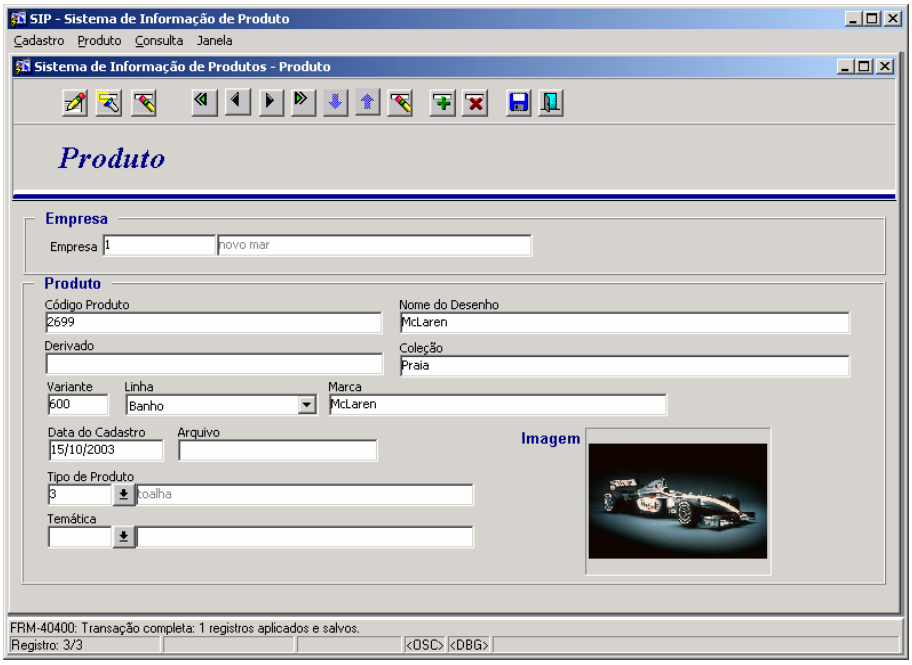


FIGURA 14 – Tela de cadastro dos dados técnicos do produto

A figura 15 demonstra a tela de dados do produto executando a partir do *browser*.

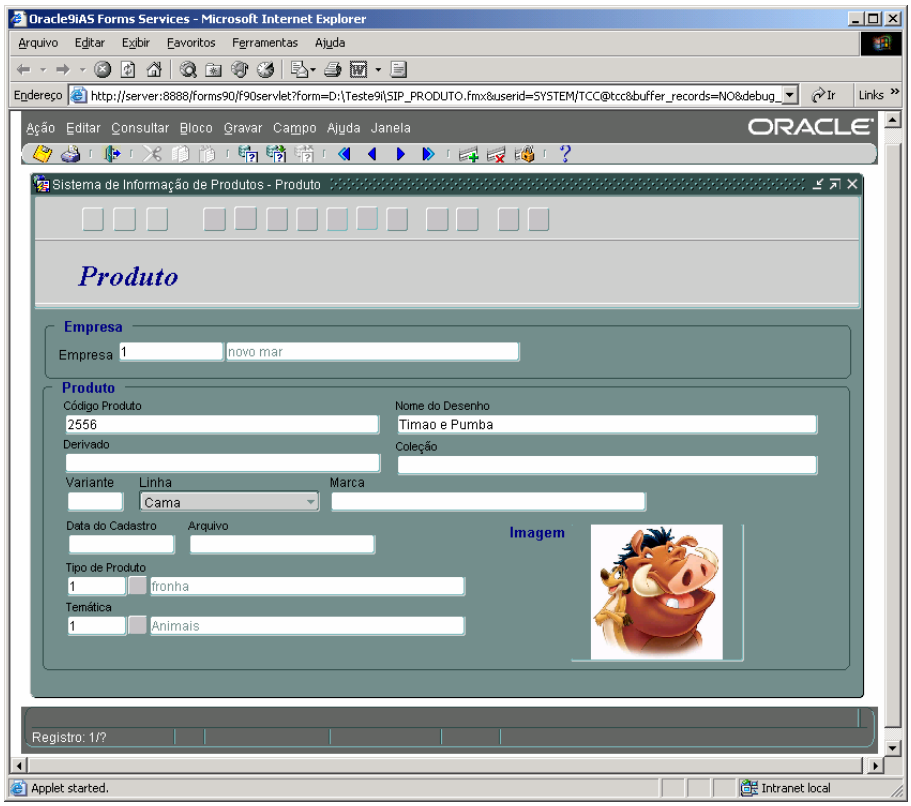


FIGURA 15 – Tela de cadastro dos dados técnicos do produto gerada no Forms 9i

6 CONCLUSÕES

Objetivando demonstrar o reaproveitamento de aplicações, migrando e convertendo partes de um desenvolvimento, nota-se a enorme dificuldade em organizar e delimitar todos os parâmetros para formar um conjunto e tomar proveito do maior grau possível.

Conforme apresentado no trabalho, verifica-se o grande custo que cresce ao redor dos instantes que se prepara para pensar e discutir como fazer algo já existente com ou sem padrão tomar um novo formato, sem ter que iniciar praticamente todo o processo. Com esses argumentos é que as empresas, como a do caso estudado deslizam em direções adversas tomando rumos sem perceber o destino.

Apresenta-se com esse trabalho uma forma de poder utilizar uma aplicação em Forms já desenvolvida para poder utilizá-la na Internet, da qual é a grande impulsionadora desse êxito da manipulação da informação na atualidade.

A utilização de um *web server* como comentado no trabalho vem de encontro com a necessidade de acelerar as informações e fazer com que não se perca tempo em transações de dados como vem acontecendo. A empresa Oracle coloca o *Application Web Server*, objeto da pesquisa proposta, como sugestão de controlar o fluxo de dados, incrementando performance, gerenciando segurança e possivelmente diminuindo a complexidade do desenvolvimento.

Como ferramenta de apoio no desenvolvimento da aplicação verificou-se o Forms para gerar automaticamente a aplicação desenvolvida para a *web*. Essa conversão pode-se afirmar que não é inteiramente automática se não for analisada com detalhes e não parcialmente, antes de praticar a geração do novo aplicativo. Detalhes como descritos no trabalho, além de performance de máquina, tráfego de rede, configurações de *browser*, deve-se apoiar em medidas econômicas de aproveitamento de processamento. Todo processo foi testado em uma máquina não colocada em disputa na rede e assim já houve considerável perda de eficiência.

Conforme apresentado no trabalho é importante analisar e discutir previamente qualquer tipo de aplicação para a *web* e não apenas as do tipo Forms. É importante salientar as facilidades de implementação e a forma como é mostrado ao usuário, o ambiente de desenvolvimento do Developer.

A pesquisa que possibilitou a realização do trabalho apresentado, resultou num conjunto de satisfações que possibilitaram uma abertura ao conhecimento de um ferramental conceituado no mercado, promissor e de grande abertura de oportunidades.

6.1 DIFICULDADES ENCONTRADAS DURANTE A REALIZAÇÃO DESSE TRABALHO

Foram encontradas algumas dificuldades nesse trabalho, principalmente pela pouca experiência em ambiente Oracle. Existem exigências na instalação do repositório para utilização da ferramenta Case Designer, geradora também de módulos para *web*, este acarretou em deficiência na performance e impossibilitou a utilização do mesmo. A performance também é prejudicada em virtude da chamada de *applets* Java para simular a navegação na Internet, ainda que o protótipo demonstrado não possui complexidade.

6.2 EXTENSÕES

Durante o desenvolvimento desse trabalho surgiram propostas que podem servir para novas pesquisas e trabalhos futuros, como por exemplo:

a) realização e análise comparativa entre servidores de aplicações de outros fabricantes;
b) análise entre ferramentas CASE para criação e geração de aplicações desde o modelo lógico, até a geração das telas para ambiente *web*.

REFERÊNCIAS BIBLIOGRÁFICAS

AMBLER, Scott; Uma visão realística da reutilização em orientação a objetos. [s.l], 1998. Disponível em: <http://www.profissionaloracle.com.br/index.php>. Acesso em: 25 ago. 2003.

BARBOSA, Ana Lucia Ribeiro e Paes; ELICINEI Soares dos Santos. **O que é reutilização?** Rio de Janeiro, 2002. Disponível em: <http://www.dcc.ufrj.br/~schneide/es/2002/1/g04/reuso/fes1.html>. Acesso em: 05 ago. 2003.

BROWN, Bradley D. **Oracle8i web development**. Califórnia: McGraw-Hill, 2000.

CORBELLINI, Humberto; OLIVEIRA, José Palazzo Moreira de. **Definição de uma metodologia para geração de aplicações e integração de banco de dados na web**. Porto Alegre, 1998. Disponível em: <http://www.inf.ufrgs.br/pos/semanaacademica/semana98/humberto.html>. Acesso em: 12 ago. 2003.

CONALLEN, Jim. **Desenvolvendo aplicações web com UML**. Rio de Janeiro: Campus, 2003.

FERNANDES, Lúcia. **ORACLE 9i para desenvolvedores** Rio de Janeiro: Axcel Books, 2002.

FERRARI, Marcelo José. Solução para ambiente web sem esforço de desenvolvimento. **Leonardo**: revista da Faculdade Asselvi. Indaial, v. 1, n. 2, p. 15-24, jan./jun. 2001.

GODOI, Robson. **Migração client / server para 9iDS**. [s.l], 2002. Disponível em: <http://www.guors.com.br/documentos.htm>. Acesso em: 21 set. 2003.

HAMANN, Kátia Simone. **Comparativo de esquemas de classificação de componentes reusáveis**. 1999. 52 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

JUSTINO, Gilvan. **Ferramenta de apoio ao processo de reutilização de especificação estruturada**. 1999. 82 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

KAMKE, Claudiomar. **Protótipo de sistema especialista para padronização de modelo de dados baseado no Oracle Designer 2000**. 2001. 69 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MCCLURE, Carma L. **Software reuse:** a standards based guide. IEEE Computer Society, 2001.

RAMOS, Débora Cristina Leira. **Ferramenta para gerenciamento de componentes reutilizáveis em Access**. 1998. 111 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

RIBEIRO, Lígia Maria; DAVID, Gabriel. **Justificação da candidatura**. Sistema de informação da faculdade de engenharia da universidade do Porto. Porto, [2003?]. Disponível em: <http://www.fe.up.pt/~gtd/publs/justificacao.doc>. Acesso em: 25 ago. 2003.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de banco de dados**. São Paulo: Makron Books, 1999.

SURDI, Luciano Rodrigo. **Proposta de ciclo de vida de software aplicando reuso**. 1997. 72 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

TEIXEIRA Filho, Jaime. **Ambiente cliente/servidor:** estratégias de migração, Rio de Janeiro, 2000. Disponível em: <http://www.informal.com.br/artigos/art013.htm>. Acesso em: 26 ago. 2003.

TONSIG, Sérgio Luiz. **Engenharia de software**: análise e projeto de sistemas. São Paulo: Futura, 2003.

YOURDON, Edward. **Declínio e queda dos analistas e dos programadores**. São Paulo: Makron Books, 1995.

ZACKER, Craig; DOYLE, Paul. **Redes de computadores: configuração, manutenção e expansão**. São Paulo: Makron Books, 2000.

ANEXO A – Diferença entre as teclas de atalho do Forms 6i com as do Forms 9i

Função	Teclas Forms 6i	Função	Teclas Forms 9i
Aceitar	F10	Atualizar registro	Ctrl+U
Ajuda	F1	Ajuda	Ctrl+H
Campo – item anterior	Shift+Ctrl+Tab	Bloco anterior	Shift+PageUp
Campo – item anterior	Shift+Tab	Campo anterior	Shift+Tab
Cancelar	Esc	Campo duplicado	Shift+F5
Contar reg. coincidentes	Shift+F2	Contar consulta	F12
Deletar para trás	Delete	Listar páginas de guia	F2
Deletar para trás	Backspace	Inserir registro	Ctrl+Down
Deletar registro	Shift+F6	Deletar registro	Ctrl+Up
Duplicar campo – item	F3	Limpar bloco	F7
Duplicar registro	F4	Duplicar registro	Shift+F6
Editar	Ctrl+E	Editar	Ctrl+E
Entrar com consulta	F7	Informar consulta	F11
Executar consulta	F8	Executar consulta	Ctrl+F11
Exibir erro	Shift+F1	Exibir erro	Shift+Ctrl+E
Imprimir	Shift+F8	Imprimir	Ctrl+P
Limpar bloqueio	Shift+F5	Menu bloco	Ctrl+B
Limpar campo – item	Ctrl+U	Limpar campo	F5
Limpar Form	Shift+F7	Limpar Form	F8
Limpar registro	Shift+F4	Limpar registro	F6
Lista de valores	F9	Lista de valores	Ctrl+L
Menu bloquear	F5	Mostrar teclas	Ctrl+K
Mostrar chaves	Ctrl+F1	Próximo bloco	Shift+PageDown
Nova definição de registros	Ctrl+>		
Novo registro	F6	Retornar	Return
Para baixo	Down	Rolar para baixo	PageDown
Para baixo	Ctrl+I	Para baixo	Down
Para cima	Up	Rolar para cima	PageUp
Para cima	Ctrl+P	Para cima	Up
Próxima chave primária	Shift+F3	Próxima chave primária	Shift+F7
Próximo campo item	Tab	Próximo campo	Tab
Próximo campo item	Ctrl+Tab	Próximo conjunto de reg.	Shift+F8
Próximo registro	Shift+Down	Próximo registro	Down
Registro anterior	Shift+Up	Registro anterior	Up
Sair	Ctrl+Q	Sair	F4
Voltar	Enter	Submeter a commit	Ctrl+S

ANEXO B – Forma de armazenamento atual das fotos de produtos