UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

GERADOR DE APLICATIVOS ORACLE PL/SQL WEB BASEADO NA ESTRUTURA DAS TABELAS DO BANCO DE DADOS ORACLE

WAGNER DA SILVA

BLUMENAU
2007

2007/2-13

WAGNER DA SILVA

GERADOR DE APLICATIVOS ORACLE PL/SQL WEB BASEADO NA ESTRUTURA DAS TABELAS DO BANCO DE DADOS 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 Sistemas de Informação — Bacharelado.

Prof. Alexander Roberto Valdameri, Mestre - Orientador

BLUMENAU
2007

2007/2-13

GERADOR DE APLICATIVOS ORACLE PL/SQL WEB BASEADO NA ESTRUTURA DAS TABELAS DO BANCO DE DADOS ORACLE

Por

WAGNER DA SILVA

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

Presidente:	Prof. Alexander Roberto Valdameri, Mestre – Orientador, FURB
Membro:	Prof. Everaldo Artur Grahl, Mestre – FURB
Membro:	Prof. Oscar Dalfovo, Doutor – FURB

Blumenau, 06 de dezembro de 2007

Dedico este trabalho a minha namorada que me ajudou em todos os momentos.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça, e por permitir que eu alcance meus objetivos.

A minha família, por te me apoiado e não esquecendo também por ter baixado o volume das televisões em quase todos os dias deste semestre que estive fazendo este trabalho.

A minha namorada Jaqueline, pela ajuda desde a elaboração da proposta até a entrega deste volume final, pelas perdas dos finais de semanas na frente do computador junto comigo, meu muito obrigado!

Ao meu orientador, Alexander Roberto Valdameri, por ter me apoiado na idéia da elaboração deste trabalho e por suas criticas e sugestões de melhorias.

“Existem duas regras para o sucesso: 1) Nunca digas tudo que sabes”.

Roger H. Lincoln

RESUMO

Com o crescente avanço da internet, a utilização de sistemas *on-line* fica cada vez mais comum em todo o mundo. Muitas empresas optam por terem seus sistemas na internet para que seus funcionários possam utilizá-lo independentemente do local que estejam. Desta forma, as empresas fabricantes de software precisam evoluir o processo de desenvolvimento de software e, ao mesmo tempo, manter um padrão de qualidade.
Este trabalho apresenta uma solução para agilizar o processo de desenvolvimento de software para internet, com a construção de uma ferramenta de geração de aplicativos em linguagem Oracle PL/SQL Web. O objetivo principal desta ferramenta é de agilizar o dia a dia dos programadores web, gerando os aplicativos de cadastros básicos automaticamente. A ferramenta permite que sejam feitas configurações de validações para os formulários, bem como, que sejam alteradas características de apresentação do aplicativo que será gerado automaticamente.

Palavras-chave: Internet. Geração de aplicativos. Oracle PL/SQL Web.

ABSTRACT

With the increasing advance of the Internet, the use of online systems is increasingly common throughout the world. Many companies choose to have their systems on the internet so that their employees can use it regardless of the place they are. Thus, manufacturers of software companies need to evolve the process of developing software and at the same time maintain a standard of quality.
This work presents a solution to expedite the process of developing software for Internet, with the construction of a tool to generate applications in language Oracle PL / SQL Web. The primary purpose of this tool is to expedite the day to day lives of web developers, generating the applications of basic entries automatically. The tool allows settings validations are made to the forms and, to be altered characteristics of submission of the application that will be generated automatically.

Key-words: Internet. Generate applications. Oracle PL/SQL Web.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

ANSI - *American National Standard Institute*

API - *Application Programming Interface*

CSS - *Cascading Style Sheet*

DAD – *Database Access Descriptor*

DCL – *Data Control Language*

DDL – *Data Definition Language*

DML – *Data Manipulation Language*

FMB – *Form Module*

FMX – *Form Compiled*

HTML – *Hyper Text Markup Language*

IDE - *Integrated Development Environment*

JSP – *Java Server Page*

LOV – *List Of Values*

MMB – *Menu Module*

MMX – *Menu Compile*

PLL – *Package Library*

PL/SQL – *Programming Language for SQL*

RDBMS – *Relational Database Management System*

RF – Requisito Funcional

RNF – Requisito Não Funcional

RSI – *Relational Software Incorporated*

SEQUEL – *Structured English Query Language*

SGBD – Sistema de Gerenciamento de Bando de Dados

SQL – *Structured Query Language*

TCC – Trabalho de Conclusão de Curso

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Com o passar dos anos, o gerenciamento das informações de uma empresa enfatizou a necessidade da criação de sistemas especializados para cada área da empresa, tais como comercial, financeira, produção, entre outras. Os sistemas comunicam-se fazendo trocas de dados para que toda a empresa fique interligada, de modo que uma área da empresa saiba como a outra área se encontra em relação a prazos, valores, produções, etc.

Mesmo tendo toda estas ligações entre as áreas da empresa também foi necessário à utilização da internet. Venetianer (1996), acredita que “a internet é a mais recente demonstração da infinita capacidade dos seres humanos para desenvolver novas tecnologias, penetrarem nas profundezas do desconhecido, explorarem o inimaginável”. Nos tempos atuais a internet já está sendo algo comum na vida de grande parte dos indivíduos. As grandes empresas para armazenar e gerenciar suas informações no computador utiliza-se de uma tecnologia baseada em bancos de dados.

> Em essência, um banco de dados é apenas um sistema computadorizado de armazenamento de registros. O banco de dados pode, ele próprio, ser visto como o equivalente eletrônico de um armário de arquivamento. Em outras palavras é um repositório ou recipiente para uma coleção de arquivos de dados computadorizados. (DATE, 2000, p. 4).

No ano de 1970, houve uma mudança muito importante na história dos bancos de dados, que até está data eram somente hierárquicos ou de redes, passou a ser desenvolvida a nova filosofia de bancos de dados relacionais. Nesta mesma época a IBM desenvolveu uma nova linguagem para trabalhar com este novo tipo de banco de dados, denominada *Structured English Query Language* (SEQUEL) e depois batizada somente por *Structured Query Language* (SQL). Nos dias atuais está linguagem é a que predomina, sendo aceita como um padrão para banco de dados relacionais (FERNANDES, 2000).

> A SQL (*Structured Query Language*) é uma linguagem para interface com banco de dados relacionais, isto é, todos os usuários e programas que desejarem realizar alguma tarefa no banco de dados devem fornecer comandos escritos nesta linguagem. (FERNANDES, 2000, p.32).

Utilizando a SQL, os usuários do banco de dados poderão executar diversas operações como por exemplo: acrescentar novos arquivos; inserir novos dados em arquivos existentes; buscar dados de arquivos existentes; alterar dados em arquivos existentes; eliminar dados de arquivos existentes; remover arquivos existentes do banco de dados.

O problema existente hoje na geração de aplicações PL/SQL Web está na demora, por parte da programação, para criação do código da aplicação e nos detalhes de apresentação dos

dados na tela como tipos de campos, ordem de apresentação dos campos, etc. Também é limitado o número de programadores que conhecem a linguagem PL/SQL Web e ainda que tenham domínio da linguagem JavaScript e HTML, o que torna o seu desenvolvimento mais complexo ainda.

O sistema desenvolvido neste trabalho trata da construção de um aplicativo que faz a criação do código de um sistema de cadastro automaticamente, baseado na estrutura de uma tabela do banco de dados Oracle. Desta forma permitirá aumentar a produtividade na construção de aplicativos e não necessariamente exigindo do programador o domínio na linguagem PL/SQL Web.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um aplicativo em Oracle Forms 6i, que gere programas de cadastros em PL/SQL Web, *package* compilável baseado na estrutura de uma tabela do banco de dados Oracle, pronto para ser utilizada pelo usuário final do sistema, pessoa que irá utilizar o aplicativo.

Os objetivos específicos do trabalho são:

a) permitir ao usuário a criação de aplicativos em PL/SQL web;
b) gerar automaticamente os aplicativos que foram configurados pelo usuário, podendo estes ter as funcionalidades de inserção, alteração, exclusão e consulta aos dados da tabela.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está dividido em quatro capítulos: Introdução (Capítulo 1), Fundamentação Teórica (Capítulo 2), Desenvolvimento da Ferramenta (Capítulo 3) e Conclusões (Capítulo 4).

No Capítulo 1, é apresentada uma introdução do que será o trabalho, definindo seus objetivos, apresentada a metodologia de desenvolvimento e a estrutura do trabalho.

No Capítulo 2, é feita toda a fundamentação conceitual do trabalho e está dividida em sete itens principais: banco de dados, modelos de dados, linguagem SQL, HTML, gerador de código, Oracle Form Builder e PL/SQL e trabalhos correlatos.

No Capítulo 3, é apresentado a estrutura e o desenvolvimento da ferramenta.

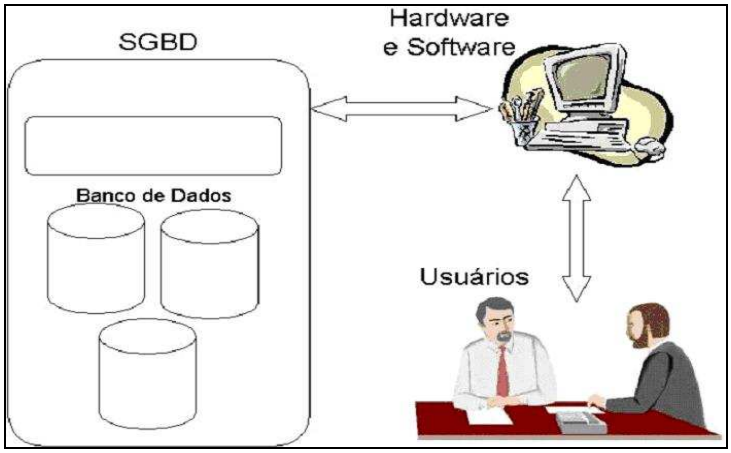
Finalmente, no Capítulo 4, são apresentadas as conclusões e recomendações sobre trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capitulo é apresentado o tema Banco de Dados descrevendo seu funcionamento e suas principais características, abordagem sobre Modelos de Dados, Linguagem SQL e HTML, geradores de código e a descrição das ferramentas utilizadas para implementação deste Trabalho de Conclusão de Curso (TCC), onde são relatados os temas Oracle Form Builder e Oracle PL/SQL. Por fim, são descritos os trabalhos correlatos pesquisados na Universidade Regional de Blumenau que possuem alguma semelhança com este trabalho.

2.1 BANCO DE DADOS

Segundo Date (2000, p.4), um sistema de banco de dados é basicamente um sistema computadorizado de armazenamento de registros; isto é, um sistema computadorizado cujo propósito geral é armazenar informações e permitir ao usuário buscar e atualizar estas informações quando solicitado. As informações em questão podem ser quaisquer coisas que tenha significado para o individuo ou a organização a que o sistema deve servir, em outras palavras, tudo que seja necessário para auxiliar no processo geral de tomada de decisões de negócios desse individuo ou dessa organização. O sistema de banco de dados envolve quatro componentes principais: Banco de Dados, *hardware*, *software* e usuários. A seguir estes componentes são apresentados na figura 1 que mostra o relacionamento entre os envolvidos.



Fonte: Rezende (2007).
Figura 1 - Componentes de um Sistema de Banco de Dados

Segundo Oliveira (2000, p.3), o banco de dados da empresa Oracle teve sua história começando com o Dr. Ted Codd que em 1970 anunciou para o mundo o seu Modelo de Dados Relacional. Os melhores protótipos de Modelo de Dados Relacional feitos por Codd foram o *System* R e o Ingress, sendo que o primeiro era um sistema desenvolvido pelo laboratório de pesquisas San Jose da IBM e o outro por uma equipe liderada por Michael Stonebraker da Universidade de Berkeley na Califórnia.

Um marco importante na época de 1970 foi à publicação no jornal de novas técnicas e dúvidas, sendo que o *System* R introduziu o SQL, a linguagem dos Bancos de Dados Relacionais que é referenciada como “padrão universal”. No tópico 2.3 deste trabalho este tema é detalhado.

Segundo Oliveira (2000, p.3), na Califórnia quatro analistas de sistemas (Bob Miner, Ed Oates, Bruce Scott e Larry Ellison) após a leitura de trabalhos de Codd sobre o Ingress e o *System* R tiveram a brilhante idéia de fazer uma versão comercial do *System* R, a qual em 1977 gerou a *Software Development Laboratories*.

Em 1979, o nome da companhia muda para *Relational Software Incorporated* (RSI) e neste momento foi criada a primeira versão comercial do Oracle o Oracle Version 2. O primeiro usuário Oracle foi a Base da Força Área de Wright Patterson, em novembro de 1979.

Em 1983 o Oracle Version 3 é lançado o primeiro RDBMS a rodar em Mainframes e PC’s, sendo o sistema mais portável do mundo, neste mesmo ano o nome da empresa mudou de RSI para *Oracle Corporation*.

Em 1997 a Oracle lança o Oracle 8 um sistema gerenciador de Banco de Dados Objeto-Relacional que comporta até 512 Petabytes de informação.

Em 1999 surge o Oracle 8i que integra soluções de internet com Banco de Dados, que inclui significativas extensões orientadas a objetos que rodam direto no servidor como a PL/SQL.

Conforme Oracle Magazine (2007, p.27), em 2007 a Oracle lança uma versão gratuita do seu banco de dados chamada Oracle 10G *Express Edition* a qual possui todas as funcionalidades e controles de um banco pago da Oracle sendo que só permite até 4 Gigabytes de informação.

2.2 MODELOS DE DADOS

De acordo com Date (2000, p.13), um modelo de dados é uma definição abstrata, autônoma e lógica dos objetos, operadores e outros elementos que, juntos constituem a máquina abstrata com a qual os usuários interagem. Os objetos permitem modelar a estrutura de dados. Os operadores permitem modelar seu comportamento.

Segundo Oliveira (2000, p.4), o Modelo de Dados Relacional foi descrito pela primeira vez pelo matemático *E. F. Codd* , em um artigo de junho de 1970 entitulado “A Relational Model of Data for Large Shared Data Banks” (“Um Modelo Relacional de Daods para Grandes Bancos de Dados”).

Os modelos mais utilizados na época eram os hierárquicos, de rede e estruturas de dados de arquivos planos, estes modelos apresentavam muita complexidade na manipulação e exigem um alto grau de manutenção das aplicações.

O Modelo Relacional foi tão amplamente aceito no mercado, que proporcionou o advento dos programas chamado Sistemas de Gerenciamento de Banco de Dados Relacionais, ou RDBMS.

Segundo Date (2000, p.15), as principais funções de um RDBMS são:

a) flexibilidade: facilidade na definição da estrutura de armazenamento e na manipulação de dados;
b) independência entre Dados e Programas: são os responsáveis por executar as tarefas solicitadas, o programador se preocupa apenas com a solicitação e manipulação dos dados;
c) integridade de Dados: por ter embasamento matemático, tornou o Modelo Relacional preciso e consistente com os dados, eliminando a redundância desnecessária de informações.

Em um modelo de dados tem-se a definição das estruturas das tabelas envolvidas no sistema abordado, gerando assim, uma fácil visualização de tudo que irá acontecer no sistema e qual a origem e destino dos dados da aplicação.

2.3 LINGUAGEM SQL

Conforme Garcia, Ullman e Widom (2001, p.16), a linguagem de banco de dados SQL tem um grande número de recursos, inclusive instruções que consultam e modificam o banco de dados. A modificação do banco de dados é feita através de três comandos, chamados INSERT, DELETE e UPDATE. Para a consulta do banco de dados é utilizado o comando SELECT.

Segundo Oliveira (2000, p.8), quando os Bancos de Dados Relacionais estavam sendo desenvolvidos, foram criadas linguagens destinadas à sua manipulação. Após, vários modelos e versões surgirem no mercado em 1986 o *American National Standard Institute* (ANSI) publicou um padrão SQL que se estabeleceu como linguagem padrão de Banco de Dados Relacional.

A SQL apresenta uma série de comandos que permite a definição dos dados, chamada DDL. Como exemplos de comandos da classe DDL, têm-se os comandos *Create, Alter, Drop* e *Rename*, conforme detalhados no Quadro 1.

Comando	**Função**
CREATE	Utilizado para criar um novo objeto no banco de dados, como exemplo: tabelas, índices e visões.
ALTER	Utilizado para alterar um objeto no banco de dados, como exemplo: colunas de uma tabela e chaves de uma tabela.
DROP	Utilizado para remover um objeto do banco de dados, como exemplo: remover uma tabela, remover um índice, remover uma coluna ou mesmo remover as chaves de uma tabela.
RENAME	Utilizado para alterar o nome de um objeto do banco de dados, como exemplo: nome de colunas ou tabelas.

Fonte: Pesquisa direta.

Quadro 1 – Comandos DDL

Os comandos da série DML são destinados a consultas, inserções, exclusões e alterações em um ou mais registros de uma ou mais tabelas de maneira simultânea. Como exemplo de comandos da classe DML, têm-se os comandos *Select, Insert , Update, Delete, Commit* e *Rollback*, conforme detalhados no Quadro 2.

Comando	**Função**
SELECT	É o principal comando da linguagem, como ele é possível recuperar dados de uma tabela ou de uma visão.
INSERT	Insere linhas em uma tabela.
UPDATE	Altera o conteúdo de colunas de uma tabela.
DELETE	Exclui linhas de uma tabela.
COMMIT	Efetiva todas as transações realizadas e não pendentes de efetivação da sessão para o banco de dados. Após este comando ser executado outras sessões passam a visualizar os dados inseridos ou manipulados nesta sessão.
ROLLBACK	Desfaz todas as modificações efetuadas na sessão desde o último commit. Qualquer dado inserido ou manipulado que esteja ainda não efetivado pelo uso do comando *commit* será excluído da sessão, não tendo mais como recupera-lo.

Fonte: Pesquisa direta.

Quadro 2 – Comandos DML

A DCL que é uma subclasse de comandos DML, dispõe de comandos de controle como *Grant* e *Revoke,* conforme detalhados no Quadro 3.

Comando	**Função**
GRANT	O propósito do comando é ceder privilégios. Tem-se dois tipos de privilégio: sobre os objetos e de sistemas, que autorizam determinadas ações dos usuários no banco de dados.
REVOKE	Retira privilégios previamente fornecidos, tanto de sistemas quanto de objetos.

Fonte: Pesquisa direta.

Quadro 3 – Comandos DCL

Outra característica interessante na linguagem SQL é a capacidade que dispõe de cancelar uma série de atualizações ou de efetivar, depois de iniciar uma seqüência de atualizações. Os comandos *Commit* e *Rollback* são responsáveis por estas facilidades.

Um ponto importante é que a linguagem SQL consegue implementar estas soluções, somente pelo fato de estar baseada em Banco de Dados Relacional que garantem por si mesmo a integridade das relações existentes entre as tabelas e seus índices.

2.4 HTML

Segundo Venetianer (1996), HTML é a acrossemia de *Hyper Text Markup Language* (Linguagem de Anotação de Hipertexto). É uma linguagem de programação muito simples, utilizada para criar documento hipertexto, que pode ser portada de uma plataforma computacional para outra. Isto significa que você pode escrever códigos-fonte HTML sem se preocupar em qual computador e por qual sistema operacional este documento será visualizado.

HTML é uma linguagem interpretada. O interpretador é o *browser* que converte os comandos anotados na representação gráfica de objetos estruturados (textos e imagens). Seus principais recursos são: a manipulação de *strings* e de imagens. No caso das *strings*, a HTML permite manipular seus atributos (negrito, itálico, corpo da fonte, espaçamento interparagrafar) e sua posição relativa na página (diagramação). No caso de imagens é possível determinar apenas sua posição. A HTML permite definir também âncoras e *links* de hipertexto e hiperimagem.

Segundo Ramalho (2001), um programa HTML possui três partes básicas: a estrutura principal, o cabeçalho e o corpo do programa, é composta basicamente por títulos, textos, parágrafos, imagens e *links*, que são responsáveis pela exibição de outras paginas na tela.

2.5 GERADOR DE CÓDIGO

Segundo Aho, Ullman e Ravi (1995, p. 237), o código de saída para um gerador de código precisa ser correto e de alta qualidade. Matematicamente o problema de se gerar um código ótimo não pode ser solucionado. Na verdade, existe uma preocupação com as técnicas heurísticas que geram um código bom, não necessariamente ótimo. A escolha dos métodos heurísticos é importante no desenvolvimento de geradores de código.

Segundo Herrington (2003, p.93), para o desenvolvimento de um gerador de código, o ideal seria realizar os seguintes passos que também são representados na figura 2:

a) construir um código de teste: o primeiro passo é construir um protótipo manualmente de como será a saída do código gerado, sendo em que na maioria dos

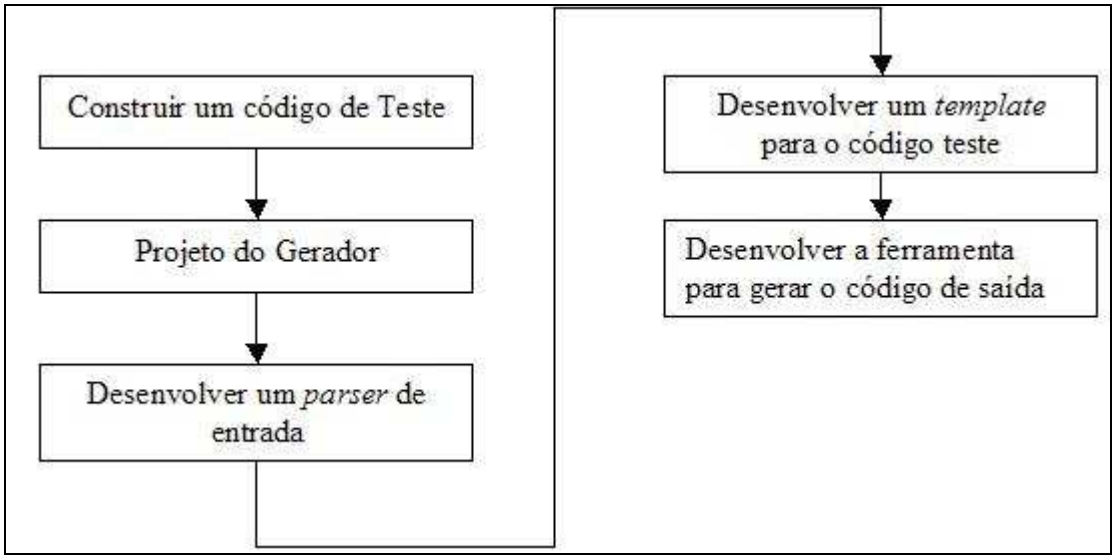
casos já se tem este código em mãos sendo ele parte da aplicação que já é utilizada;

b) projeto do gerador: uma vez que possui a saída, deverá determinar como será construído o gerador que criará o código de teste como saída. O mais importante é a especificação dos dados requeridos. Tendo esses dados especificados deverá ser definido um arquivo de entrada;

c) desenvolver um *parser* de entrada: o primeiro passo para a implementação é definir um arquivo *parser*.

d) desenvolver template para código de teste: com os arquivos de entrada já definidos, poderão ser criados *templates* para gerar os arquivos de saída;

e) desenvolver o código de saída: o último passo é escrever o código que percorre os arquivos de entrada definidos nas templates e gerar os arquivos de saídas. No final do desenvolvimento você pode testar com o código teste definido no inicio do processo.



Fonte: Adaptado de Herrington (2003, p.93).

Figura 2 – Passos para o desenvolvimento de um gerador de código

As técnicas de geração de código trazem muitos benefícios para todos os níveis da engenharia de software. Sendo que Herrington (2003, p.15) relaciona alguns destes benefícios:

a) qualidade: a maior parte do código escrito manualmente tende a ser inconsistente. O código gerado através de templates cria um código base consistente, e quando os templates são alterados e o gerador é executado, os erros são reparados ou o código é improvisado aplicando consistência em toda a base do código;

b) consistência: o código que é construído por um gerador de código é consistente no projeto das APIs e na nomeação de variáveis. Isto resulta em uma interface fácil de entender e de usar;
c) mais tempo para o projeto: a programação para a geração de código de um projeto é diferente de um projeto gerado manualmente. Quando alterações são feitas na API do *framework*, grande quantidade de código deve ser reescrita para usar a API apropriada, com a geração de código os desenvolvedores podem reescrever apenas os templates e executar o gerador para reparar as alterações;
d) decisões do projeto não previstas: gerador de código usa definições de arquivos abstratos para especificar o projeto do código a ser gerado. Esses arquivos são muito curtos e mais específicos que o código resultante.

Quanto precisa especificar, desenvolver ou atualizar um gerador de código segundo Herrington (2003, p.25) precisa seguir um conjunto de regras citadas:

a) respeitar o código escrito manualmente: deve-se respeitar e detestar o código escrito manualmente. Deve-se respeitar porque há casos especiais integrados e quando o código escrito manualmente é substituído preciso certificar-se que os casos especiais estão sendo tratados. Deve detestar por que tempo é extremamente valioso para desperdiçar com tarefas repetitivas;
b) escrever o primeiro código: tem-se a obrigação de entender o seu *framework* para depois gerar o código. É ideal escrever o primeiro código manualmente e depois usar este código para gerar os templates do gerador;
c) controle do código fonte: é importante ter um controle do código fonte para o sucesso de um gerador de código. Se o gerador trabalha diretamente com implementações de arquivos que contém algum código escrito manualmente, certifique-se que este código esteja consistente;
d) decidir sobre a linguagem de implementação: as ferramentas usadas para construir o gerador não precisam ser as mesmas usadas para escrever a aplicação. O problema que o gerador está tentando resolver é completamente diferente do problema que está sendo resolvido pela aplicação. Por essa razão o gerador deverá ser visto como um projeto independente;
e) integrar o gerador ao processo de desenvolvimento: o gerador é uma ferramenta para ser usada por desenvolvedores, assim deve ser adaptada com o processo de desenvolvimento. Se for apropriado, pode ser intergrado como uma IDE;

f) incluir avisos: o gerador deverá sempre mostrar avisos sobre o código que está sendo gerado para que as pessoas não modificarem o código. Se elas modificarem o código e rodar novamente, o gerador perderá suas revisões. O fato de que elas estão usando a ferramenta é um grande passo;
g) tornar amigável: apenas porque o gerador é uma ferramenta para programadores não significa que não precisa ser amigável. O gerador deverá informar ao coordenador o que está fazendo, e quais arquivos estão sendo alterados ou criados. A ferramenta que é difícil de usar pode ser ignorada e todo o esforço usado para o desenvolvimento é desperdiçado;
h) incluir documentação: boa documentação é essencial para um gerador. A documentação deve ser completa e destacando os pontos chaves: o que o gerador faz, como é instalado, como é executado e quais arquivos são aceitos;
i) manter em mente que a geração de código é uma cultura: ensinando os colegas através da documentação, seminários e reuniões individuais é importante para o sucesso do gerador. Necessita-se quebrar com aqueles interesses e dúvidas e enfatizar o gerador foi projetado para trazer benefícios;
j) manter o gerador: a menos que o gerador seja provisório, ele precisa ser mantido em longo prazo. Se o gerador controla grande parte do código, é importante que o coordenador mantêm alguma parte do código. No orçamento deverá incluir o tempo dedicado e dinheiro para manter e atualizar o gerador.

Existe muita diferença entre o código gerado através de um gerador e um código escrito manualmente. Segundo Herrington (2003, p. 6) a geração de código manualmente é inviável em sistemas muito grandes, além de consumir muito o tempo do desenvolvedor, não é possível garantir um código de qualidade. Herrington (2003, p. 6) faz uma comparação entre o código escrito manualmente e o gerado através de um gerador e que está descrita a seguir.

No código gerado através do gerador:

a) a qualidade do código é consistente através de todas as entidades;
b) quando mais mudanças são requeridas, o gerador é executado gerando novos templates;
c) os erros dentro das classes são reparados mudando os templates. Quando o gerador é executado todas as classes são reparadas automaticamente;
d) as unidades de testes são criticas. São usados textos diferenciados para garantir que o gerador está criando as implementações apropriadas. Pode-se criar e gerar

unidades de testes clássicas como se fosse gerada manualmente;

e) a estrutura e a lógica de negócios são armazenas em uma base de dados, o gerador e os templates podem ser alterados para construir códigos para uma linguagem diferente ou framework;
f) a estrutura e as entidades são criadas ao mesmo tempo pelo mesmo mecanismo. A sincronização é automática. Se há um problema de sincronização de falha no gerador, a edição é feita facilmente reparando e executando novamente o gerador.

No código gerado manualmente:

a) há muito copiar e colar código podendo gerar um código inconsistente. O código é inconsistente através das entidades;
b) as alterações são feitas em cada entidade uma por uma;
c) quando um problema afeta mais de uma classe, deverá ser resolvido manualmente um por um através das entidades. Alguma alteração feita na base de dados às vezes resolve o problema;
d) cada classe tem uma unidade de teste correspondente de ajustes dentro da unidade de testes do *framework*;
e) uma camada de compatibilidade é requerida de baixo código para movê-lo para uma estrutura diferente.

Algumas das orientações apresentadas por Herrington foram seguidas neste trabalho, conforme a necessidade e a adequação as ferramentas e linguagens de programação utilizadas para o desenvolvimento da ferramenta.

2.6 ORACLE FORM BUILDER E PL/SQL

Com base nas técnicas e regras apontadas no último capítulo deste trabalho para ferramentas de geração de código, é utilizada a ferramenta Oracle Form Builder e a linguagem de programação da Oracle PL/SQL. As principais características da ferramenta bem como da linguagem de programação são brevemente detalhadas abaixo, sendo que, para um estudo mais aprofundado devem ser adquiridos manuais e apostilas específicas do assunto abordado.

De acordo com Fernandes (2000, p. 676), a ferramenta Form Builder é capaz de construir três tipos de módulos principais diferentes:

a) módulo *form*: Consiste da aplicação *online*, contendo lógicas de atualização, telas, botões, itens, etc. O código fonte de um módulo *Form* possui a extensão FMB e o executável FMX;
b) módulo menu: Consiste de um conjunto de submenus e lógicas para acionamento de diversos módulos *Form* ou de submenus, execução de aplicações PL/SQL e *batch*, acionamento de gráficos, etc. O código fonte de um módulo Menu possui a extensão MMB e o executável MMX;
c) módulo PL/SQL *library*: Consiste de um conjunto de programas PL/SQL que pode ser compartilhado por diversas aplicações *Form* (ou *Report*) que executam num ambiente. Possui apenas um tipo de arquivo cuja extensão é PLL.

Um quarto tipo de arquivo pode ser gerado com esta ferramenta:

a) *object library*: Consistem em um conjunto de objetos internos do *Form* que podem ser definidos uma única vez e utilizados em outras aplicações. Esta biblioteca de objetos auxilia o desenvolvedor na criação de padrões.

> A PL/SQL é uma linguagem procedural da Oracle que estende a SQL com comandos que permitem a criação de procedimentos de programação. Com ela, podemos usar comandos de SQL *Data Manipulation Language* (DML) para manipular os dados da base de dados Oracle e estabelecer fluxos de controle para processar estes dados. (FERNANDES, 2000, p. 280).

A linguagem permite a declaração de constantes, variáveis, subprogramas (procedimentos e funções), que favorecem a estruturação de códigos, e possui mecanismos para controle de erros de execução. Incorpora os novos conceitos de objetos, encapsulamento e, ainda, permite a interface com rotinas escritas em outras linguagens.

De acordo com Oliveira (2000, p. 49), o PL/SQL permite a utilização direta de comandos SQL dentro do código e oferece também um mecanismo eficiente de tratamento de erros. O código PL/SQL utilizado para interagir com o Banco de Dados, também é armazenado no próprio Banco de Dados. É a única linguagem que oferece uma interface de acesso nativo ao Oracle e que esta situada dentro do próprio ambiente Oracle.

Os módulos de código PL/SQL são divididos em quatro categorias:

a) *package* (pacote): Uma coleção de *procedures* e *functions* que tem duas partes. Uma composta por uma lista com as disponíveis no pacote, e outra, uma parte de implementação, que consiste no código fonte de cada uma delas;
b) *procedure* (procedimentos): Uma série de comandos que aceitam e retornam zero ou mais variáveis;

c) *function* (funções): Uma série de comandos que aceitam zero ou mais variáveis que retornam um valor;

d) *trigger* (gatilho): Uma série de comando PL/SQL que é associada a uma tabela no Banco de Dados. O gatilho sempre é executado quando um evento ocorre sobre a tabela-alvo (uma operação *select*, *insert*, *update* ou *delete*).

A seguir no quadro 4 é apresentado um exemplo das categorias de código PL/SQL *package*, *procedure* e *functions*:

```
-- Especificação da Package com o nome de PCK_EXEMPLO
CREATE OR REPLACE PACKAGE PCK_EXEMPLO IS

    -- Especificação de procedimento com o nome de PRINCIPAL
    PROCEDURE PRINCIPAL;
    -- Especificação da função com o nome de RETORNA_ZERO
    FUNCTION RETORNA_ZERO RETURN NUMBER;

END PCK_VEICULO;
-- Fim da especificação da Package com o nome de PCK_EXEMPLO

-- Corpo da Especificação da Package com o nome de PCK_EXEMPLO
CREATE OR REPLACE PACKAGE BODY PCK_EXEMPLO IS

    -- Corpo do Procedimento com o nome de PRINCIPAL
    PROCEDURE PRINCIPAL IS
    BEGIN
      -- Bloco de código.
      NULL;
    END PRINCIPAL;
    -- Fim do Procedimento com o nome de PRINCIPAL

    -- Corpo da Função com o nome de RETORNA_ZERO
    FUNCTION RETORNA_ZERO RETURN NUMBER IS
    BEGIN
      -- Bloco de código.
      RETURN 0;
    END RETORNA_ZERO;
    -- Fim do Corpo da Função com o nome de RETORNA_ZERO

END PCK_EXEMPLO;
-- Fim do corpo da Especificação da Package com o nome de PCK_EXEMPLO
```

Fonte: Pesquisa direta.

Quadro 4 – Código PL/SQL

A seguir no quadro 5 é apresentado um exemplo da categoria de código PL/SQL *trigger*:

```
-- Declaração da trigger com nome de EXEMPLO que dispara
-- sempre que for inserido, alterado ou excluído um registro
-- da tabela com nome de TABELA_EXEMPLO
CREATE OR REPLACE TRIGGER EXEMPLO
    BEFORE INSERT OR UPDATE OR DELETE ON TABELA_EXEMPLO
    FOR EACH ROW
BEGIN
  -- Ação que será feita sempre que disparada a trigger
  NULL;
END EXEMPLO;
-- Fim da trigger com nome de EXEMPLO
```

Fonte: Pesquisa direta.

Quadro 5 – Código PL/SQL Trigger

2.7 TRABALHOS CORRELATOS

Abaixo são apresentados trabalhos de conclusão de curso pesquisados na Universidade Regional de Blumenau que possuem um resultado final diferenciado deste trabalho mas que utilizam ferramentas e técnicas similares às utilizadas neste trabalho.

Em Coelho (2006) foi desenvolvida uma ferramenta para geração de código html baseado no dicionário de dados de um banco de dados, bem como a geração de scripts para fazerem a manipulação dos dados. A ferramenta desenvolvida por Coelho é feita em linguagem Java gerando os códigos das aplicações também para Java, enquanto que neste trabalho a ferramenta é feita em Oracle Forms 6i gerando as aplicações em Oracle PL/SQL Web.

Em Schvepe (2006) foi desenvolvida uma ferramenta para migração de aplicações em Oracle Forms 6i para código Java. A relação existente entre o proposto por Schvepe e o proposto neste trabalho são a geração de código para uma outra linguagem de programação, que no caso de Schvepe era gerar código Java e no caso deste trabalho foi a geração de código PL/SQL Web.

A empresa Oracle disponibiliza junto com o pacote de ferramentas de PL/SQL um conjunto próprio para trabalhar com geração de código para web que chamam de Oracle Toolkit, que é um pacote de funções que fazem a criação do código HTML da página, sendo

que para cada *tag* em HTML tem-se uma função correspondente no Oracle Toolkit. A utilização deste pacote se faz necessário somente no caso de se querer padronizar o código das aplicações em PL/SQL Web no padrão pré-definido pela Oracle, no entanto se pode utilizar diretamente as *tag* HTML, visto que são muito mais simples e conhecidas.

Em Dolla (2001) foi desenvolvida uma ferramenta que faz a indentação de um código fonte em PL/SQL, gera a documentação em pontos chaves do código fonte e demonstra avisos de determinadas construções não recomendadas de programação. A relação do trabalho feito por Dolla e o proposto neste trabalho são a utilização do banco de dados Oracle e a utilização da linguagem PL/SQL. Outra grande diferença é que no trabalho de Dolla é utilizado um código fonte PL/SQL já existente, ou seja, o sistema não gerou este código enquanto que neste trabalho é gerado todo o código fonte automaticamente para cada novo cadastro que se deseja gerar pelo aplicativo.

Em Menin (2005) foi desenvolvida uma ferramenta de geração de código em linguagem JSP utilizando banco de dados MySQL. A relação entre o trabalho feito por Menin e o proposto neste trabalho é que ambos são geradores de código, que permitem fazer inclusões, exclusões, alterações e consultas no banco de dados. Mas a diferença está nas linguagens de geração que enquanto Menin gera em JSP, neste trabalho a geração é feita para PL/SQL Web, e o banco de dados também muda, sendo que neste trabalho é utilizado Oracle ao invés de MySQL.

3 DESENVOLVIMENTO DA FERRAMENTA

Neste capítulo é apresentado o desenvolvimento, através do levantamento dos requisitos, especificação, implementação do gerador de aplicativos em Oracle PL/SQL Web e também é demonstrado passo a passo a criação de uma aplicação.

3.1 VISÃO GERAL DA FERRAMENTA

O problema consiste em ler uma tabela da base de dados Oracle, permitir a parametrização e configuração da apresentação através de um aplicativo feito em Oracle Forms 6i e por final gerar um aplicativo em Oracle PL/SQL Web pronto para ser utilizado através do navegador de internet.

A seguir é detalhado cada passo para a solução do problema que está sendo apresentado na figura 3:

a) no passo 1, no banco de dados Oracle já deve estar criada uma tabela com a estrutura que se deseja para o aplicativo final em PL/SQL Web;

b) no passo 2, neste exemplo tem-se uma tabela denominada CARRO cuja estrutura é bem simples, possuindo apenas os atributos CD_PLACA com o objetivo de armazenar o código da placa do veículo, NM_VEICULO com o objetivo de armazenar o nome do veículo e NM_MARCA com o objetivo de armazenar o nome da marca do veículo;

c) no passo 3, a ferramenta em Oracle Forms 6i, após ser informado o nome da tabela que se deseja para o aplicativo em PL/SQL Web, automaticamente faz a leitura da sua estrutura de colunas do banco de dados e apresenta no bloco de configurações de apresentação da tela, onde será possível dizer qual o tipo de campo que deve ser apresentado para cada coluna lida da base de dados, se o campo será visível na tela ou não, se o campo terá lista de valores para auxiliar o seu preenchimento ou não. Por fim, é pressionado o botão visualizar da ferramenta, o qual fará a geração da package com o código do aplicativo pronto no diretório escolhido e em seguida fará a compilação no banco de dados, estando criada no banco de dados é feita

uma requisição desta package através de um navegador de internet que apresentará o aplicativo em html;

d) no passo 4, o aplicativo em PL/SQL Web está pronto para ser utilizado pelo usuário final permitindo todas as operações que foram concedidas no momento da configuração na ferramenta em oracle Forms 6i, sendo assim, o usuário poderá inserir novos registros na tabela CARRO, atualizar os registros já inseridos, remover os registros que não se deseja mais e até mesmo pesquisar um determinado registro.

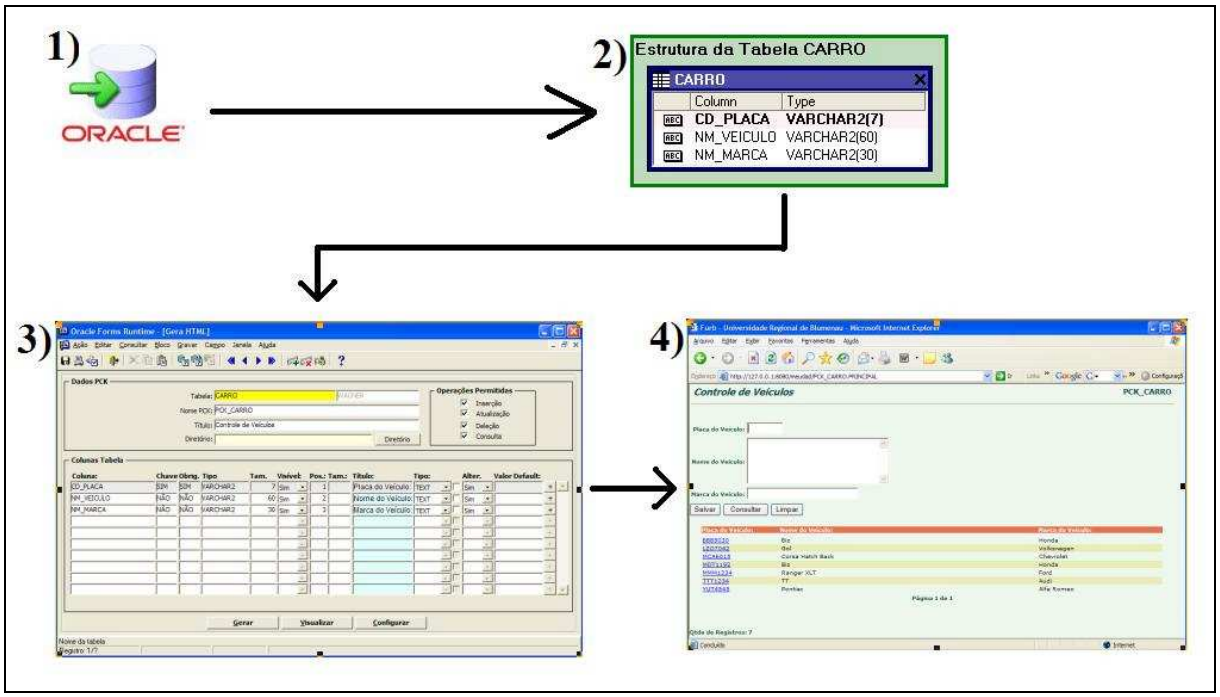


Figura 3 – Passo a passo da resolução do problema

3.2 REQUISITOS DO SISTEMA

Com o levantamento dos requisitos é possível identificar quais são os passos a serem tomados para conseguir chegar à solução desejada. A seguir são relacionados os Requisitos Funcionais (RF) e os Requisitos Não Funcionais (RNF) da ferramenta:

a) permitir ao desenvolvedor a escolha de uma das tabelas já existentes no banco de dados na qual a aplicação fará a manipulação dos dados (RF);

b) deverá possibilitar ao desenvolvedor a personalização dos tipos de campos que serão apresentados na aplicação gerada (RF);
c) possibilitar ao desenvolvedor a personalização de listas de valores (RF);
d) possibilitar ao desenvolvedor a personalização das validações em Java Script, para validação dos formulários gerados pela aplicação (RF);
e) possibilitar ao desenvolvedor a personalização da ordem de apresentação dos campos na tela (RF);
f) possibilitar a geração de um aplicativo Oracle PL/SQL Web pronto para utilização do usuário final (RF);
g) ser desenvolvida em ambiente de programação Oracle Forms 6i (RNF);
h) gerar os aplicativos em Oracle PL/SQL Web (RNF);
i) utilizar o banco de dados Oracle 10g Express Edition (RNF).

3.3 ESPECIFICAÇÃO

A especificação da ferramenta foi feita utilizando diagramas da UML através da ferramenta Enterprise Architect para o desenvolvimento dos casos de uso e a utilização da ferramenta PL/SQL Developer para criação do modelo de dados relacional.

3.3.1 Diagrama de Caso de Uso

É representado na Figura 4 o diagrama de caso de uso do desenvolvedor dos aplicativos PL/SQL Web.

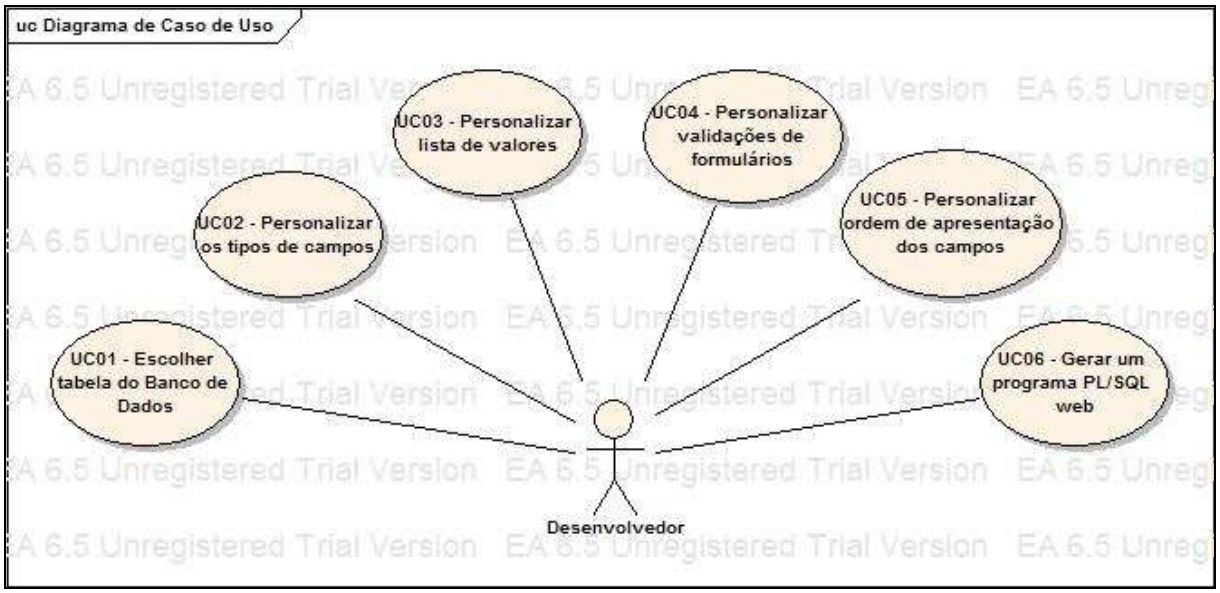


Figura 4 - Diagrama de caso de uso do desenvolvedor

A seguir é apresentada a especificação para cada caso de uso:

a) UC01 – Escolher tabela de Banco de Dados: o desenvolvedor deve selecionar o nome da tabela que será utilizada para sua aplicação em uma lista de valores onde são apresentadas todas as tabelas existentes no banco de dados no qual se está conectado;
b) UC02 – Personalizar os tipos de campos: o desenvolvedor configura os tipos de campos que devem ser apresentados na tela do aplicativo web (*Text*, *Select*, *CheckBox*, *RadioButton*, *Hidden*), estes tipos são formatos válidos em HTML que são utilizados para geração dos aplicativos gerados por esta ferramenta;
c) UC03 – Personalizar lista de valores: o desenvolvedor cria consultas a base de dados através de instruções de seleção de dados descritas na linguagem *SQL*, que selecionam o código e a descrição dos valores válidos para o campo no qual se está personalizando. No aplicativo gerado estes dados devem ser apresentados em uma tela que permita a escolha de somente um registro;
d) UC04 – Personalizar validações de formulários: o desenvolvedor define as funções de validação que devem ser chamadas em JavaScript para consistir se os valores digitados nos formulários são realmente do tipo de dados esperado pela aplicação, para os campos que irão receber números inteiros, números reais ou datas;
e) UC05 – Personalizar ordem de apresentação dos campos: o desenvolvedor escolhe a ordem de apresentação dos campos no formulário informando para cada campo do aplicativo um número que represente sua ordem de apresentação, com isto, o

sistema faz ordenação crescente dos valores digitados pelo desenvolvedor e apresenta os campos no aplicativo nesta mesma ordem sendo um por linha para os valores distintos e na mesma linha para os valores iguais;

f) UC06 – Gerar um programa PL/SQL Web: o desenvolvedor faz a geração do aplicativo PL/SQL Web, chamando automaticamente a compilação da package gerada com o código do aplicativo no banco de dados e sua execução através do navegador de internet.

3.3.2 Diagrama de Atividades

O diagrama de atividades da ferramenta, é representado na figura 5.

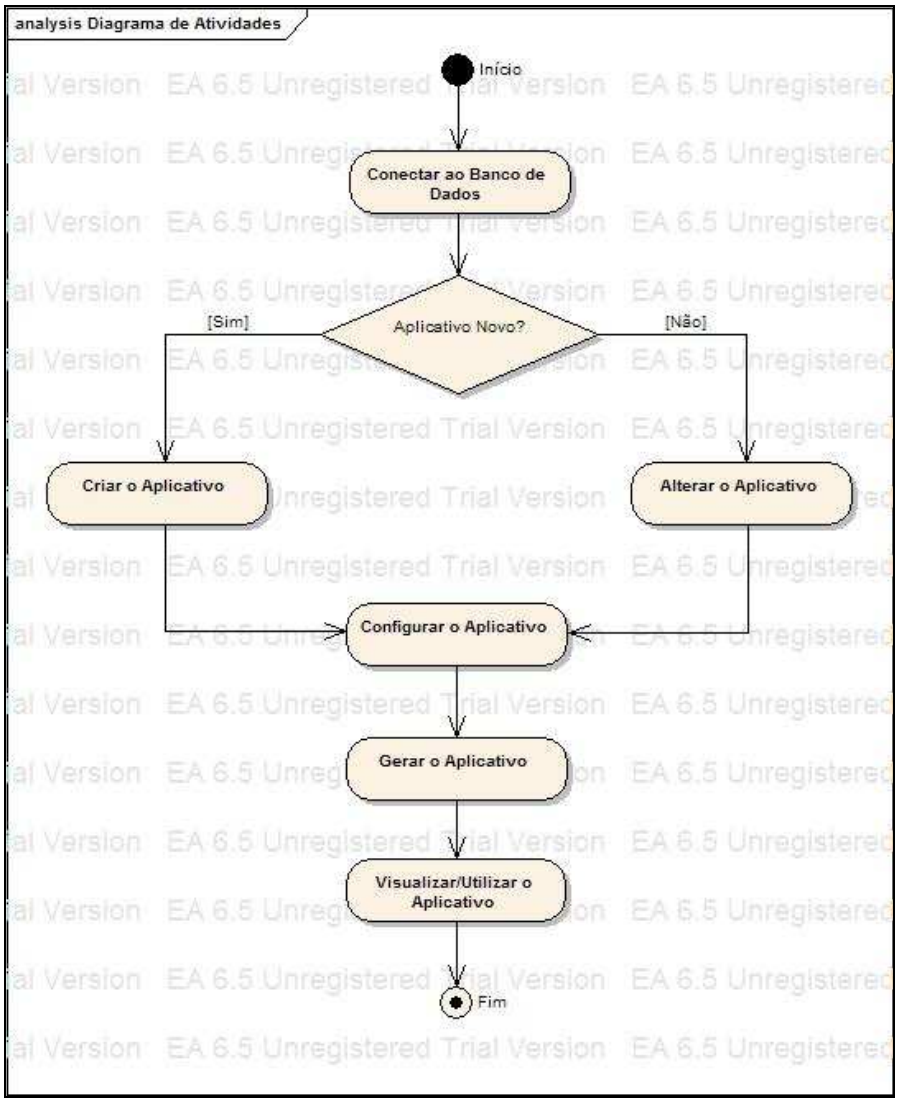


Figura 5 - Diagrama de Atividades

3.3.3 Modelo de Dados Relacional

O modelo de dados relacional, das entidades utilizadas pela ferramenta, está representado na figura 6. Este modelo foi construído na ferramenta PL/SQL Developer cujo fabricante é a empresa AllroundAutomations

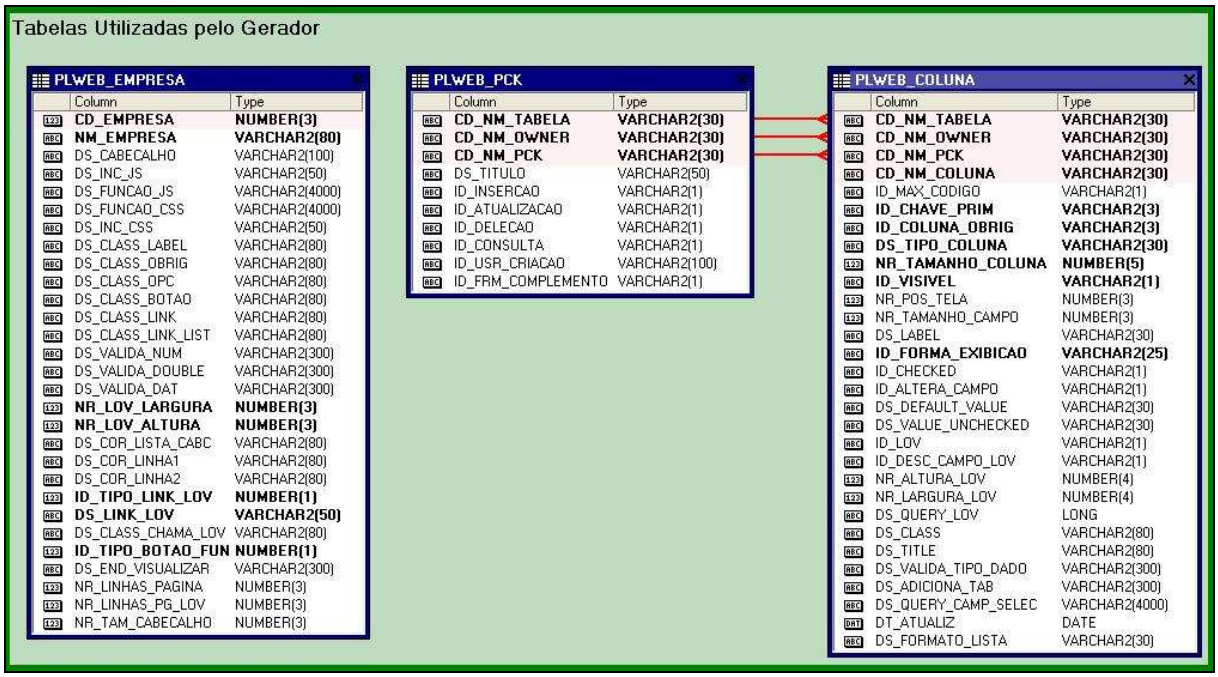


Figura 6 – Modelo de dados relacional

A seguir é apresentada uma breve descrição das entidades utilizadas para o desenvolvimento da ferramenta:

a) plweb_pck: entidade responsável por armazenar as informações dos aplicativos gerados pela ferramenta;
b) plweb_coluna: entidade responsável por armazenar as informações de cada campo dos aplicativos gerados pela ferramenta;
c) plweb_empresa: entidade responsável por armazenar as informações de configuração por empresa dos aplicativos gerados pela ferramenta.

3.3.4 Dicionário de Dados

O dicionário de dados de cada entidade do modelo de dados relacional encontra-se nas tabelas de 1 a 3. Onde mencionado o tipo de dado VARCHAR2 entendesse que é um campo

que armazena caracteres, onde mencionado o tipo de dado NUMBER entendesse que é um campo que armazena somente números e quando mencionado tipo de dado DATE entendesse que é um campo que armazena uma data, composta de século, dia, mês, ano, hora, minuto e segundo.

Tabela 1 – Dicionário de Dados da Entidade PLWEB_PCK

Nome	**Descrição**	**Tipo**
CD_NM_TABELA	Nome da tabela base da package.	VARCHAR2
CD_NM_OWNER	Nome do usuário que criou a package.	VARCHAR2
CD_NM_PCK	Nome da package no banco de dados.	VARCHAR2
DS_TITULO	Título da tela gerada pela ferramenta.	VARCHAR2
ID_INSERCAO	Indica se deve permitir inserção de registro na tela gerada pela ferramenta.	VARCHAR2
ID_ATUALIZACAO	Indica se deve permitir atualização de registro na tela gerada pela ferramenta.	VARCHAR2
ID_DELECAO	Indica se deve permitir deleção de registro na tela gerada pela ferramenta.	VARCHAR2
ID_CONSULTA	Indica se deve permitir consulta de registro na tela gerada pela ferramenta.	VARCHAR2
ID_USR_CRIACAO	Indica o usuário responsável pela criação da package.	VARCHAR2
ID_FRM_COMPLEMENTO	Indica se deve gerar frames (html) complementares na tela gerada pela package.	VARCHAR2

Tabela 2 – Dicionário de Dados da Entidade PLWEB_COLUNA

Nome	**Descrição**	**Tipo**
CD_NM_TABELA	Nome da tabela base da package.	VARCHAR2
CD_NM_OWNER	Nome do usuário que criou a package.	VARCHAR2
CD_NM_PCK	Nome da package no banco de dados.	VARCHAR2
CD_NM_COLUNA	Nome da coluna no banco de dados.	VARCHAR2
ID_MAX_CODIGO	Indica se o valor do código deve ser gerado automaticamente com valor máxima do banco de dados nesta coluna + 1.	VARCHAR2
ID_CHAVE_PRIM	Indica se a coluna é chave primaria no banco de dados.	VARCHAR2
ID_COLUNA_OBRIG	Indica se o preenchimento da coluna é obrigatório no banco de dados.	VARCHAR2
DS_TIPO_COLUNA	Descrição do tipo da coluna no banco de dados.	VARCHAR2
NR_TAMANHO_COLUNA	Número de caracteres permitidos para a coluna no banco de dados.	NUMBER
ID_VISIVEL	Indica se a coluna é visível na tela.	VARCHAR2
NR_POS_TELA	Número que determina a posição em que o campo deve aparecer na tela.	NUMBER
NR_TAMANHO_CAMPO	Número de caracteres que devera aparecer para este campo na tela (width).	NUMBER
DS_LABEL	Descrição do título do campo que deve aparecer na tela.	VARCHAR2

Nome	Descrição	Tipo
ID_FORMA_EXIBICAO	Indica o tipo de campo html que deve aparecer na tela.	VARCHAR2
ID_CHECKED	Indica se o campo é para vir marcado quando o tipo for checkbox ou radio-button.	VARCHAR2
ID_ALTERA_CAMPO	Indica se o campo pode ser alterado na tela.	VARCHAR2
DS_DEFAULT_VALUE	Descrição do valor inicial do campo, pode ser numérico ou alfanumérico.	VARCHAR2
DS_VALUE_UNCHECKED	Descrição do valor para o campo do tipo checkbox quando não selecionado.	VARCHAR2
ID_LOV	Indica se o campo possui lista de valores para facilitar seu preenchimento.	VARCHAR2
ID_DESC_CAMPO_LOV	Indica se o campo de descrição para o valor exibido na lista de valores será criado.	VARCHAR2
NR_ALTURA_LOV	Número que determina a altura da janela da lista de valores.	NUMBER
NR_LARGURA_LOV	Número que determina a largura da janela da lista de valores.	NUMBER
DS_QUERY_LOV	Descrição da instrução de seleção (comando select) que apresenta os dados da lista de valores.	VARCHAR2
DS_CLASS	Descrição da classe específica que deve ser utilizada pelo campo (css).	VARCHAR2
DS_TITLE	Descrição que aparece quando o mouse do usuário ficar parado sobre o campo na tela.	VARCHAR2
DS_VALIDA_TIPO_DADO	Descrição da validação que será chamada em javascript quando o campo perder o cursor.	VARCHAR2
DS_ADICIONA_TAB	Descrição das tags que devem ser acrescentadas diretamente no fonte (propriedade html tag).	VARCHAR2
DS_QUERY_CAMP_SELEC	Descrição da instrução de seleção (comando select) que apresenta os dados para o campo do tipo select.	VARCHAR2
DT_ATUALIZ	Data da última atualização desta coluna na base de dados pela ferramenta.	DATE
DS_FORMATO_LISTA	Descrição do formato de apresentação que a coluna deve ter ao ser listada na tela de consulta (ex: 990.00 ou dd/mm/yyyy).	VARCHAR2

Tabela 3 – Dicionário de Dados da Entidade PLWEB_EMPRESA

Nome	Descrição	Tipo
CD_EMPRESA	Código da empresa.	NUMBER
NM_EMPRESA	Nome da empresa.	VARCHAR2
DS_CABECALHO	Descrição da rotina que deve ser chamada para geração do cabeçalho (pode ser código PL/SQL) os valores substituídos são :DS_TITULO pelo título informado para tela e :DS_PCK pelo nome dado para package.	VARCHAR2

Nome	Descrição	Tipo
DS_INC_JS	Descrição do caminho onde está o arquivo das funções JavaScript (JS).	VARCHAR2
DS_FUNCAO_JS	Descrição de códigos de funções javascritps colocados diretamente na ferramenta que não encontrasse no arquivo.	VARCHAR2
DS_FUNCAO_CSS	Descrição de códigos de funções css colocados diretamente na ferramenta que não encontrasse no arquivo.	VARCHAR2
DS_INC_CSS	Descrição do caminho onde está o arquivo das configurações de estilos de campos (CSS).	VARCHAR2
DS_CLASS_LABEL	Descrição do nome da classe em css que deve ser utilizada para os campos do tipo label.	VARCHAR2
DS_CLASS_OBRIG	Descrição do nome da classe em css que deve ser utilizada para os campos obrigatórios.	VARCHAR2
DS_CLASS_OPC	Descrição do nome da classe em css que deve ser utilizada para os campos opcionais.	VARCHAR2
DS_CLASS_BOTAO	Descrição do nome da classe em css que deve ser utilizada para os campos do tipo botão.	VARCHAR2
DS_CLASS_LINK	Descrição do nome da classe em css que deve ser utilizada para os links.	VARCHAR2
DS_CLASS_LINK_LIST	Descrição do nome da classe em css que deve ser utilizada para os links das listas de consultas.	VARCHAR2
DS_VALIDA_NUM	Descrição do nome da função em javascript que deve ser utilizada para validação de campos numéricos.	VARCHAR2
DS_VALIDA_DOUBLE	Descrição do nome da função em javascript que deve ser utilizada para validação de campos numéricos com decimais.	VARCHAR2
DS_VALIDA_DAT	Descrição do nome da função em javascript que deve ser utilizada para validação de campos de datas.	VARCHAR2
NR_LOV_LARGURA	Número que determina a largura da janela da lista de valores.	NUMBER
NR_LOV_ALTURA	Número que determina a altura da janela da lista de valores.	NUMBER
DS_COR_LISTA_CABC	Descrição do nome da classe em css dos cabeçalhos das listas de consulta.	VARCHAR2
DS_COR_LINHA1	Descrição do nome da classe das linhas impares de uma lista de consulta.	VARCHAR2
DS_COR_LINHA2	Descrição do nome da classe das linhas pares de uma lista de consulta.	VARCHAR2
ID_TIPO_LINK_LOV	Descrição do tipo do link para fazer a chamada da lista de valores do campo pode ser uma imagem, texto ou botão.	NUMBER
DS_LINK_LOV	Descrição do conteúdo que será mostrado para fazer a chamada da lista de valores do campo.	VARCHAR2

Nome	Descrição	Tipo
DS_CLASS_CHAMA_LOV	Descrição do nome da classe onde chama a lista de valores.	VARCHAR2
ID_TIPO_BOTAO_FUNC	Indica o tipo da chamada de uma função que pode ser texto ou botão.	NUMBER
DS_END_VISUALIZAR	Descrição do endereço na internet para poder visualizar a package, geralmente o caminho que foi definido no DAD, propriedade Oracle.	VARCHAR2
NR_LINHAS_PAGINA	Número de linhas exibidas por página de consulta.	NUMBER
NR_LINHAS_PG_LOV	Número de linhas exibidas por página de consulta de uma lista de valores.	NUMBER
NR_TAM_CABECALHO	Número do tamanho em pixel ocupado pelo cabeçalho.	NUMBER

3.4 IMPLEMENTAÇÃO

A seguir são apresentadas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.4.1 Técnicas e ferramentas utilizadas

O desenvolvimento da ferramenta foi feitos através do software Oracle Forms 6i, que foi criado pela empresa Oracle e que permite a criação de aplicativos com acesso nativo aos bancos de dados também da empresa Oracle.

O banco de dados utilizado pela ferramenta foi o Oracle 10g *Express Edition*, que é distribuído gratuitamente pela empresa Oracle para estudantes e também para desenvolvedores de pequeno porte, pois somente é permitido até 4 GigaBytes de armazenamento.

Como navegador de internet foi utilizado o Internet Explorer da Microsoft, o qual permitiu os testes dos aplicativos gerados pela ferramenta.

3.4.2 Arquitetura

A seguir são apresentadas as telas de cada com uma breve explicação de suas funcionalidades.

Ao abrir a ferramenta pode-se optar pela escolha de um aplicativo já gerado anteriormente ou por se fazer um novo aplicativo, na figura 8 está representada a tela inicial do sistema.

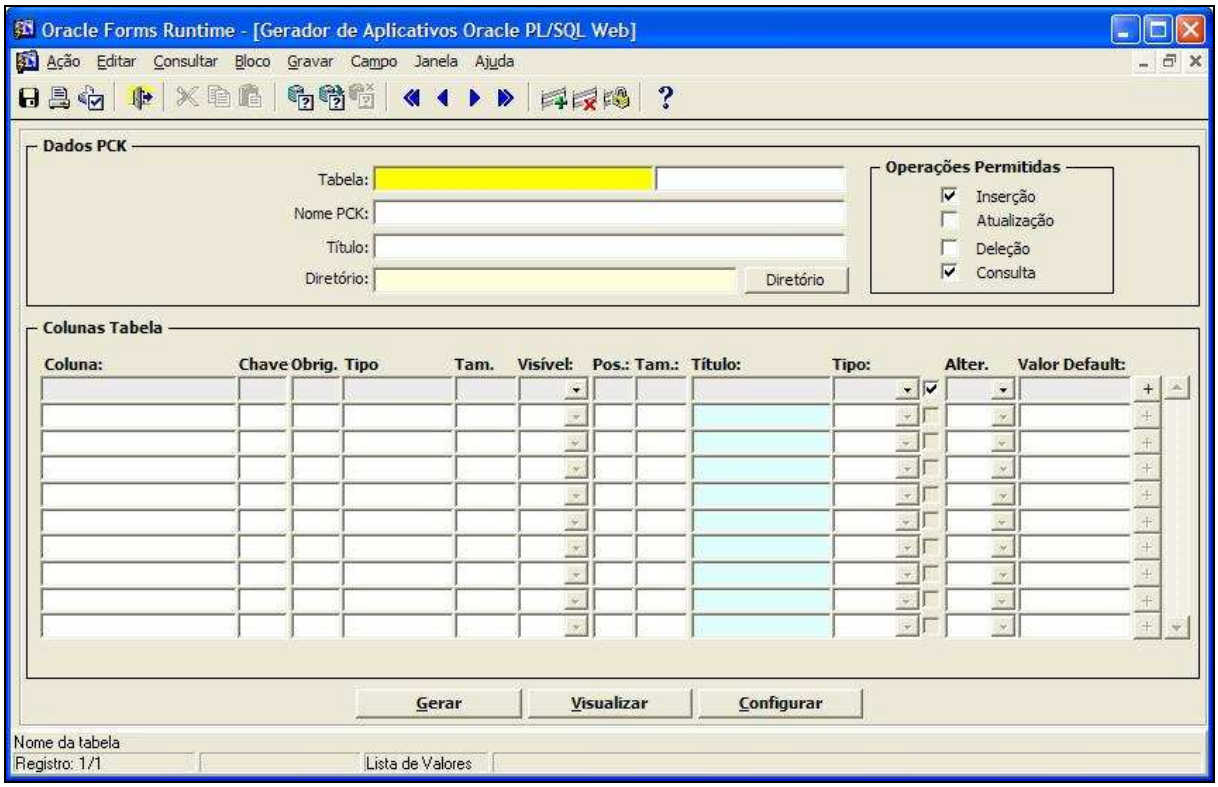


Figura 8 – Tela inicial do sistema

Neste ponto deve-ser informar qual o nome da **tabela** do banco de dados que será utilizada pelo aplicativo, podendo esta ser escolhida através da lista de valores disponível para facilitar o preenchimento do campo, conforme apresentado na figura 9 ou informando o nome da tabela diretamente no campo. Após ter informado o nome da tabela o sistema automaticamente irá sugerir o **nome da *package*** do aplicativo no banco de dados, caso seja necessário poderá ser informado outro nome diferente do sugerido. Em seguida, o **título** do aplicativo deve ser informado, este título será apresentado no cabeçalho do aplicativo gerado. O campo **diretório** é opcional caso seja informado o sistema faz a geração do arquivo da *package* no diretório informado, caso contrário, o sistema utilizará automaticamente o diretório definido como padrão “C:\Temp”, deve-se selecionar quais as **operações permitidas**

para o aplicativo que se está gerando, podendo estas ser inserção, atualização, deleção e consulta.

Ao final deve-se pressionar a tecla “F10” para salvar as informações para que a próxima etapa de configuração de apresentação do aplicativo possa ser iniciada, neste instante a tela apresentada pelo sistema será semelhante à apresentada na figura 10.

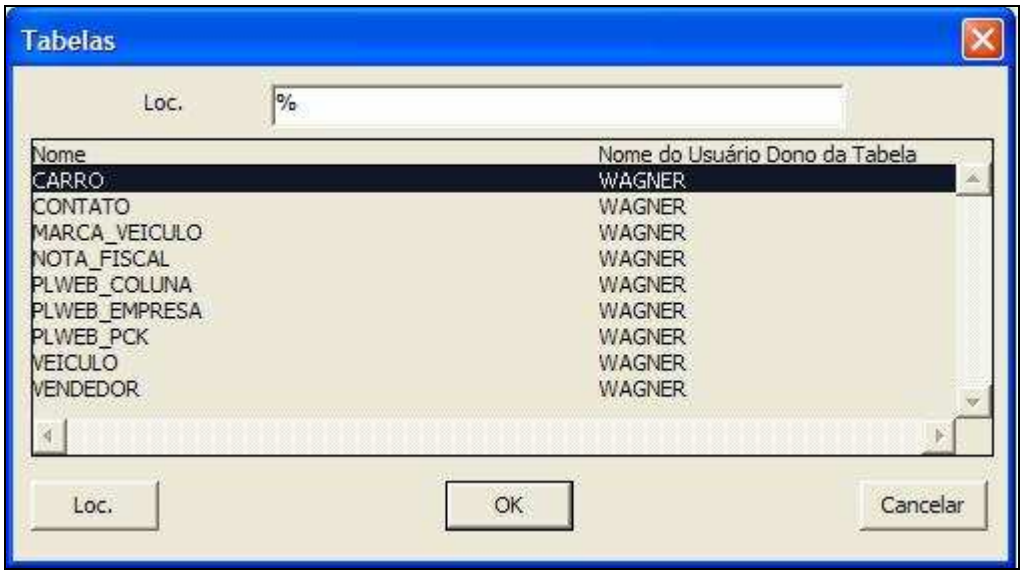


Figura 9 – Lista de valores das tabelas

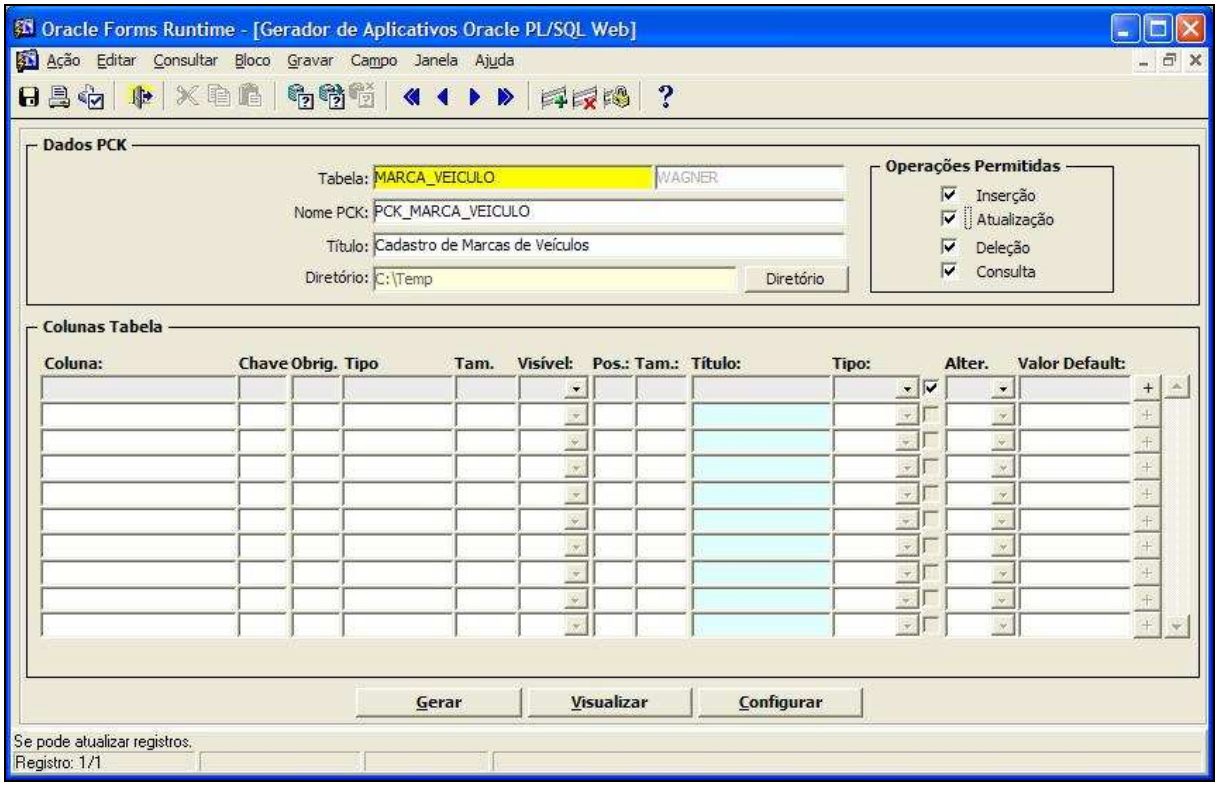


Figura 10 – Fazendo um novo aplicativo

No quadro **colunas tabela** são apresentadas todas as colunas pertencentes à tabela

informada para o aplicativo, o sistema executa uma consulta SQL para recuperar da base de dados Oracle as informações para serem listadas neste bloco da ferramenta, a consulta SQL e os dados recuperados para a tabela MARCA_VEICULO utilizada como exemplo, estão sendo apresentados na figura 11. Ainda neste mesmo quadro, são feitas as configurações para todos os campos recuperados para a tabela do aplicativo, as configurações determinam se o campo será **visível** na tela, **posição** do campo na tela, **tamanho** do campo, **título** do campo, indicador se permite **alteração**, valor ***default*** (inicial ou padrão) para o campo e **tipo** do campo podendo este ser definido como: *Text, Select, CheckBox, RadioButton*, *Hidden*, estes tipos são formatos válidos em HTML, é demonstrado na figura 12 estes tipos de campos.

Ao ser pressionado o botão "+", apresentado no final do quadro de **colunas tabela**, o sistema apresenta a tela de **configurações adicionais** de apresentação para a coluna que se estiver posicionado no bloco. Nesta tela, pode ser definida uma lista de valores para a coluna, para isto deve ser marcado o campo **criar lov** e definido qual a consulta SQL que deve ser feita para recuperar os dados a serem apresentados na lista de valores. Como regra deve ser sempre informada uma consulta que recupere o código e a descrição do que se pretende apresentar na lista de valores, a descrição que será selecionada pela consulta pode ser retornada para a tela do aplicativo se escolhida a opção **incluir descrição do campo**.

Ainda na tela de configurações adicionais, apresentada na figura 13, podem ser configuradas todas as opções listadas a seguir:

a) a altura e a largura da lista de valores;
b) a class (classe) do arquivo de estilos css para o campo;
c) a mensagem que deve aparecer quando o foco do mouse ficar parado sobre o campo no aplicativo (campo título);
d) a função em JavaScript para validar o campo (campo valida campo);
e) as tags HTML caso se deseje fazer alguma restrição específica em JavaScript para o campo;
f) o valor para o campo do tipo checkbox quando desmarcado (campo unchecked);
g) o formato em que o campo deve ser apresentado na lista de consulta, sendo este um formato de apresentação válido no banco de dados Oracle (por exemplo número: 990D00 ou data: DD/MM/YYYY);
h) caso o campo for um seqüencial pode ser definido que o valor atribuído para ele será sempre o valor máximo da coluna da tabela mais um (campo max +1).

```
SELECT tab.column_name                              Coluna
      ,decode(tab.nullable,'Y','NÃO','SIM')         Obrigatório
      ,tab.data_type                                "Tipo de Dado"
      ,NVL(DECODE(tab.data_type
                 ,'NUMBER', tab.data_precision
                          , tab.char_length),5) Tamanho
  FROM all_tab_columns tab
 WHERE tab.owner       = 'WAGNER'
   AND tab.table_name  = 'MARCA_VEICULO'
```

Coluna	Obrigatório	Tipo de Dado	Tamanho
CD_MARCA_VEICULO	SIM	NUMBER	3
NM_MARCA	SIM	VARCHAR2	30
ID_USUARIO	SIM	VARCHAR2	12
DT_ATUALIZACAO	SIM	DATE	0

Figura 11 – Recupera campos da tabela

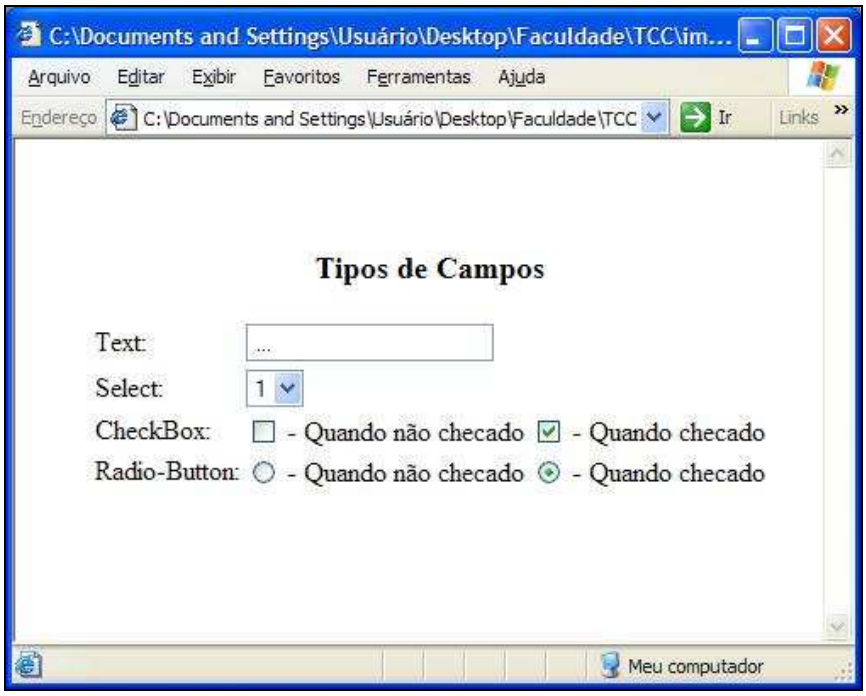


Figura 12- Tipos de campos válidos para o aplicativo

Figura 13 – Configurações adicionais

A ser pressionado o botão **configurar,** a tela de configurações por empresa é apresentada conforme figura 16, nesta tela são realizadas as configurações gerais para a aplicação definindo para isto um código e nome de empresa, podem ser feitas diversas configurações diferentes.

As opções de configuração por empresa estão dividas em quadros, conforme explicadas a seguir:

a) quadro Empresa: Definição de propriedades gerais para o aplicativo, definição do cabeçalho que deve ser uma *package* compilada no banco de dados Oracle que gera um cabeçalho para o aplicativo, definição do local do arquivo JS e do arquivo CSS para os aplicativos, definição do endereço http configurado no servidor para rodar as aplicações geradas, tamanho disponível para o cabeçalho (campo tamanho do cabeçalho) e o número de linhas exibidas por página quando o aplicativo tiver a operação de consulta (campo linhas por página). Existe também a possibilidade de escrever código JavaScript ou uma classe CSS diretamente no gerador, para isto, basta pressionar um dos respectivos botões denominados JS e CSS que uma caixa de texto será apresentada para que seja feita a escrita destas funções ou estilos, conforme apresentado nas figuras 14 e 15;

Figura 14 – Funções JavaScript

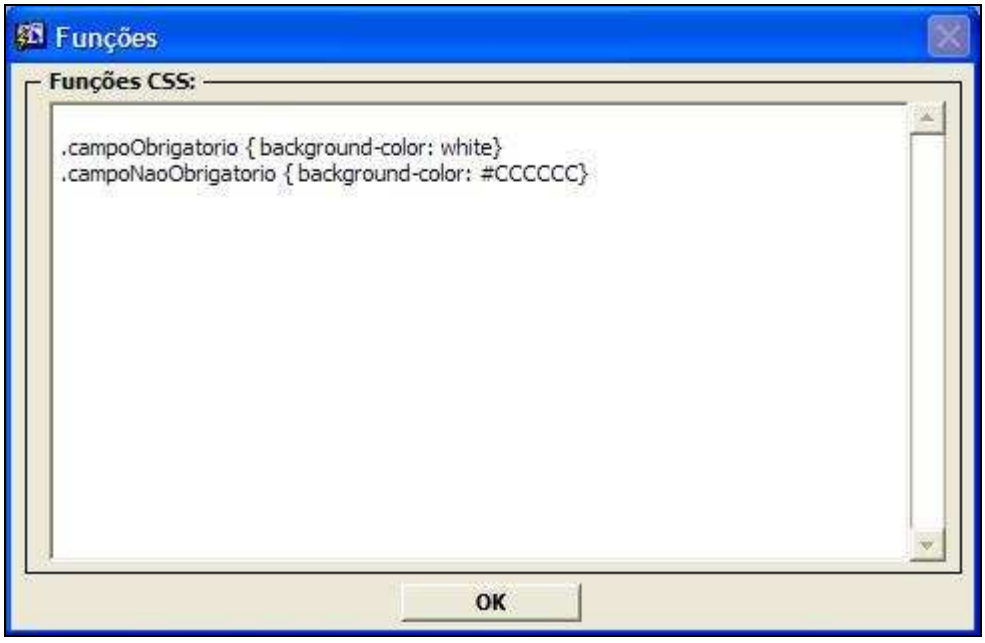


Figura 15 – Funções CSS

b) quadro Validação: Definição das validações em JavaScript que devem ser chamadas para validar os campos do tipo número inteiro, número real e data;

c) quadro Class: Definição das classes css que devem ser empregadas para campos do tipo campo obrigatório, campo opcional, label, cabeçalho de lista, linha impar da lista de consultas, linha par da lista de consultas, botão, link e link de lista de consultas;

d) quadro LOV: Definição das propriedades para as lista de valores, sendo definidas tipo chamada LOV que determina se será uma imagem, link ou botão que fará a

chamada para a lista de valores do campo, a largura e a altura da lista de valores, a class css para a lista de valores e o número de linhas que devem ser apresentados por página da lista de valores;

e) quadro Chamada de Função: Definição do tipo de chamada das funções de salvar, excluir, consultar, limpar e novo que pode ser através de botão ou link.

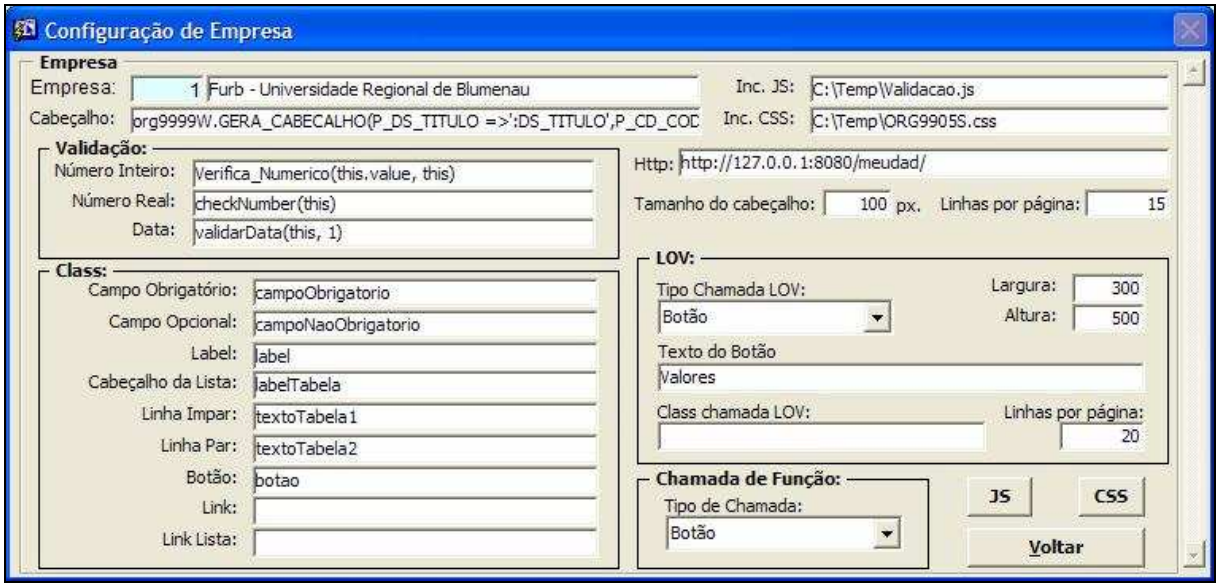


Figura 16 – Configurações por empresa

Após todos os campos serem configurados para o aplicativo, a figura 17 representa a tela com todos os campos configurados. Ao ser pressionado o botão **gerar** o sistema faz a leitura de todos os dados informados para o aplicativo bem como de todas as configurações de apresentações, com isto, o sistema irá gerar um arquivo com extensão "pck" no **diretório** escolhido na ferramenta. Na figura 18 é apresentada parte do código da ferramenta que gera o arquivo no diretório escolhido pelo usuário. É importante descrever que para a montagem do arquivo para o aplicativo o sistema vai concatenando textos na ordem correta para geração de uma package Oracle, por exemplo, para ser criado um campo de texto no aplicativo gerado o sistema irá fazer a concatenação da seqüência de caracteres que esta sendo apresentada no quadro 6.

```
<input type=TEXT name=CD_MARCA_VEICULO size=5>
```

Fonte: Pesquisa direta.

Quadro 6 – Caracteres para criação de campo de texto

Onde, o texto "*type*" representa o **tipo** de campo configura na tela, o texto "*name*" representa o nome da coluna vinda do banco de dados e o texto "*size*" representa o **tamanho** definido para a coluna. Esta mesma seqüência fica se repetindo para todas as colunas

existentes para a tabela da aplicação, alternando o tipo do campo, nome, tamanho e demais propriedades de campo que são configuráveis.

Ao ser pressionado o botão **visualizar**, o sistema faz a geração da package no diretório escolhido pelo usuário, compila a rotina no banco de dados e por fim faz a chamada da rotina no endereço **http** de execução do aplicativo configurado na ferramenta, permitindo assim que o usuário utilize o aplicativo.

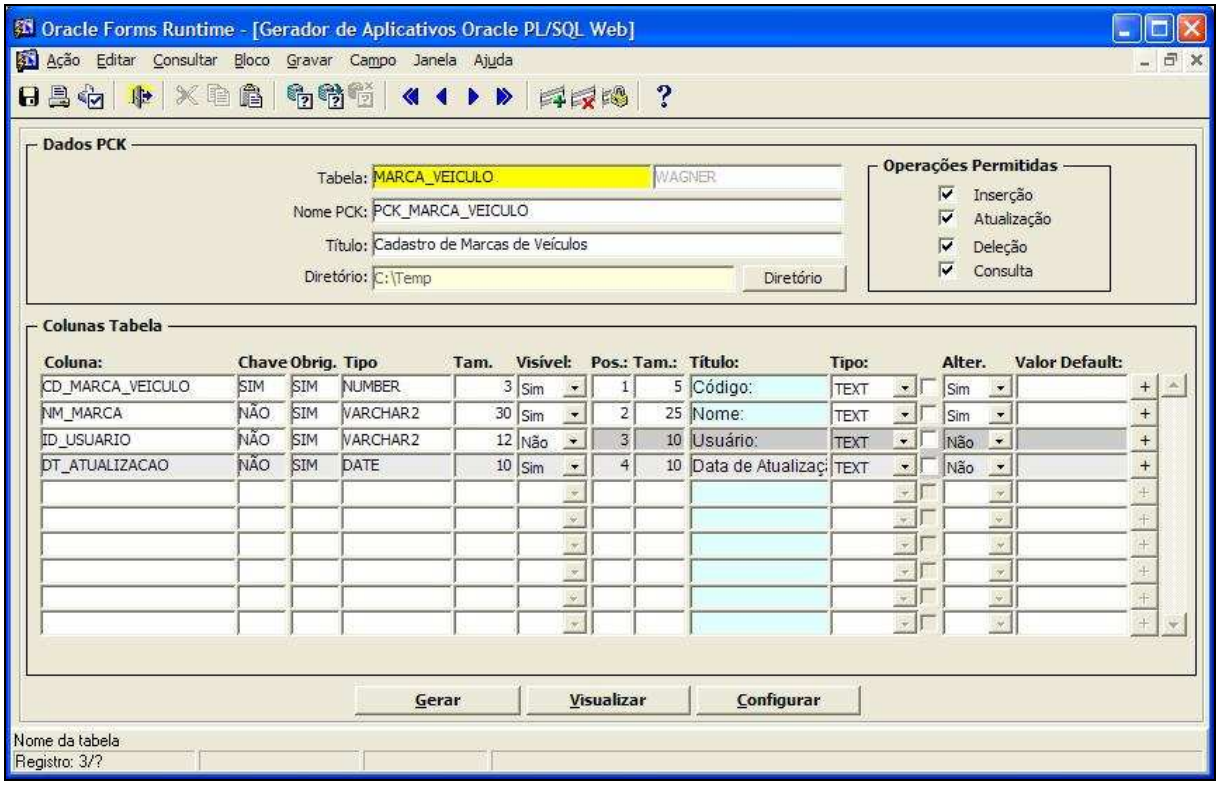


Figura 17 – Posição final da ferramenta antes da geração do aplicativo

```
PROCEDURE prc_arq_open(p_nm_arquivo IN VARCHAR2
                      ,p_id_modo    IN VARCHAR2 DEFAULT 'w') IS
BEGIN

  p_local.w_arquivo_tmp := text_io.fopen(p_nm_arquivo||'.pck',p_id_modo);

EXCEPTION
  WHEN form_trigger_failure THEN
    RAISE form_trigger_failure;
  WHEN OTHERS THEN
    alerta('Erro ao criar o arquivo: '||SQLERRM,'E');
END;
```

Figura 18 – Código PL/SQL de geração do arquivo

3.4.3 Operacionalidade da implementação

Para demonstrar a operacionalidade da ferramenta será utilizado o contexto de uma revendedora de veículos. Esta revendedora pretende manter no banco de dados o controle das Marcas de Veículos, os Veículos, os Vendedores e para cada venda feita a necessidade da emissão de um comprovante simples com o vendedor, veículo e valor da venda denominado no modelo de nota fiscal, conforme o modelo de dados relacional apresentado a seguir na figura 19.

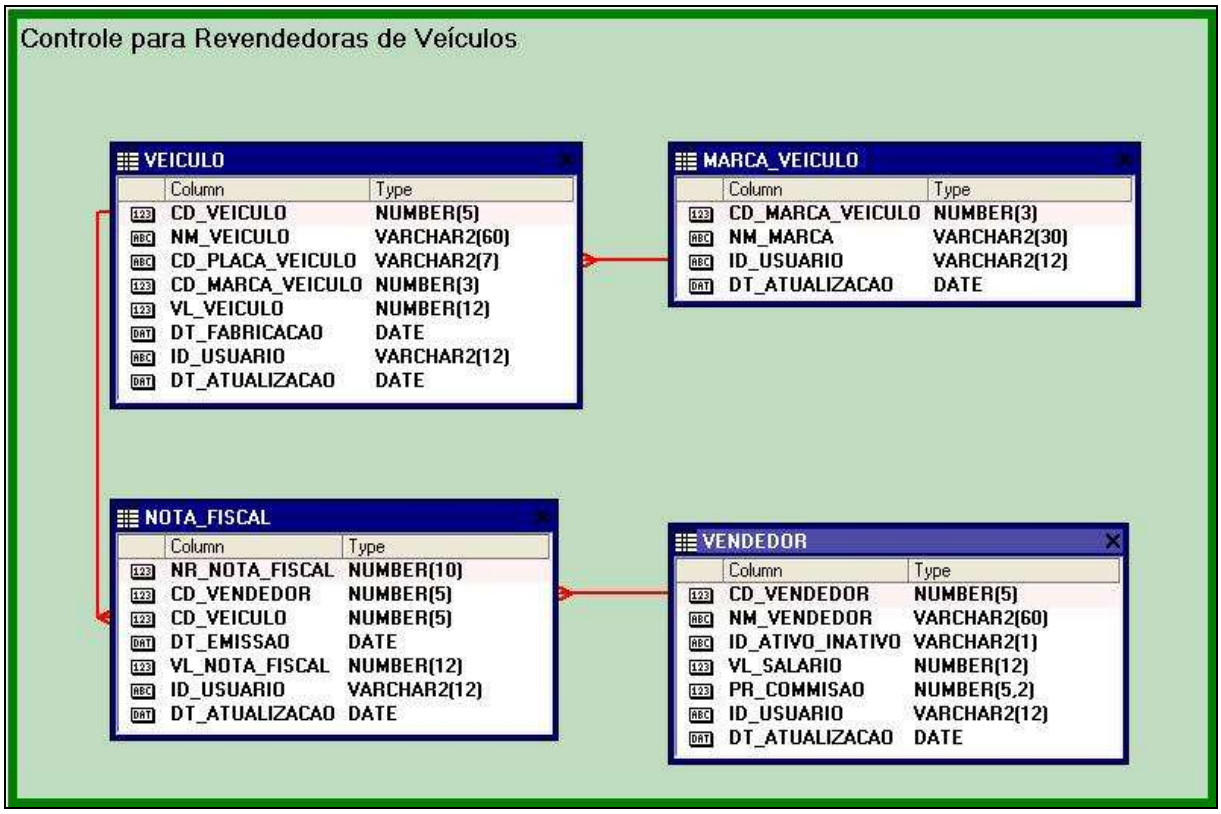


Figura 19 – Modelo de dados para revendedora de veículos

Neste contexto, será demonstrada a criação do aplicativo para manter as informações do Veículo, sendo que para o restante das funções os mesmos passos devem ser seguidos.

Com a ferramenta de geração aberta, deve ser feita a escolha da tabela **VEICULO** na lista de valores que é apresentada no campo **Tabela**, informado um **Título** para o aplicativo que neste exemplo é utilizado "**Cadastro de Veículo**", devem ser selecionadas todas as operações que serão permitidas neste aplicativo e deve ser feita à gravação dos dados para a ferramenta, podendo esta ser feita pressionando a tecla **F10** ou pressionando a imagem de um disquete localizada no canto superior esquerdo da tela, esta imagem e a tela no momento antes

da gravação estão sendo apresentadas na figura 20.

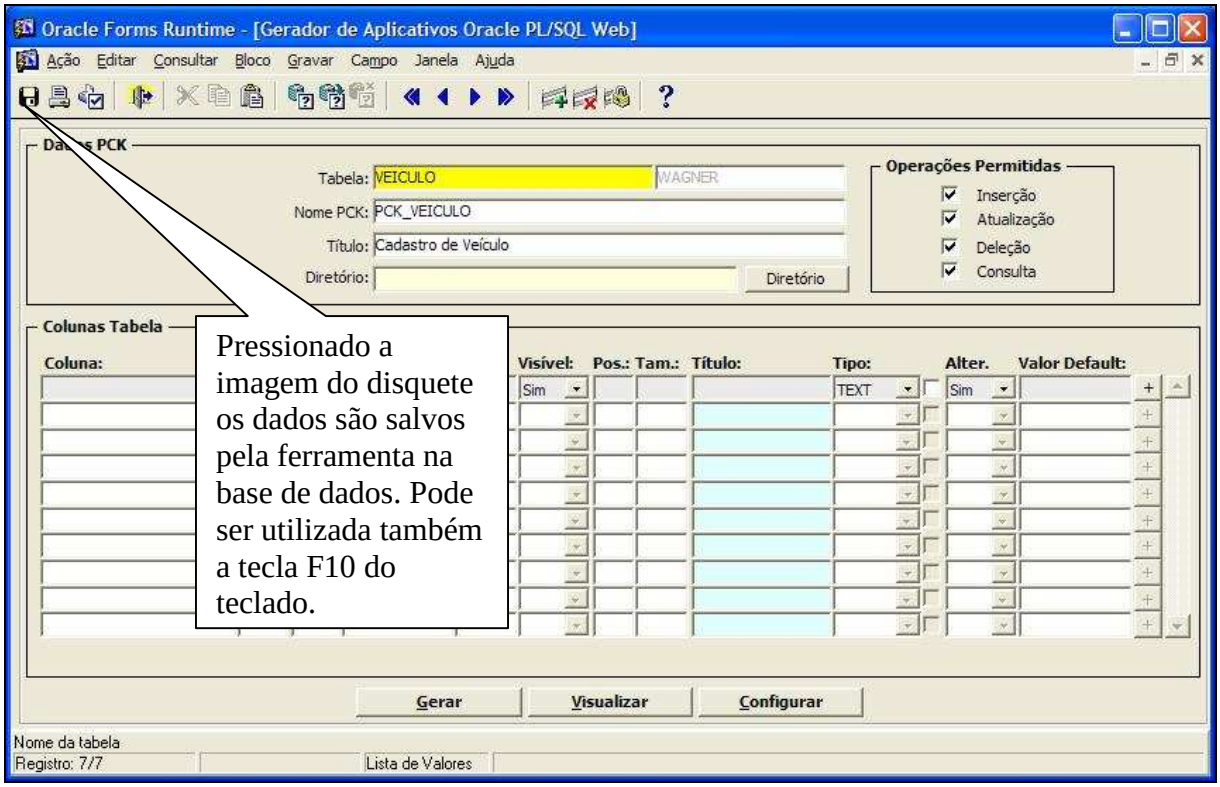


Figura 20 – Cadastro de veículo

Tendo preenchido as informações necessárias para a geração da package, agora devem ser feitas todas as configurações de apresentação dos campos, conforme a figura 21. Uma solução diferenciada utilizada nesta configuração foi à criação de um campo do tipo SELECT para a coluna CD_MARCA_VEICULO, como todas as marcas estarão cadastradas em uma outra tabela de banco de dados pode então ser preenchido esta lista de seleção com os dados existentes nesta tabela, a consulta SQL utilizada para popular esta lista de seleção esta sendo apresentada na figura 22.

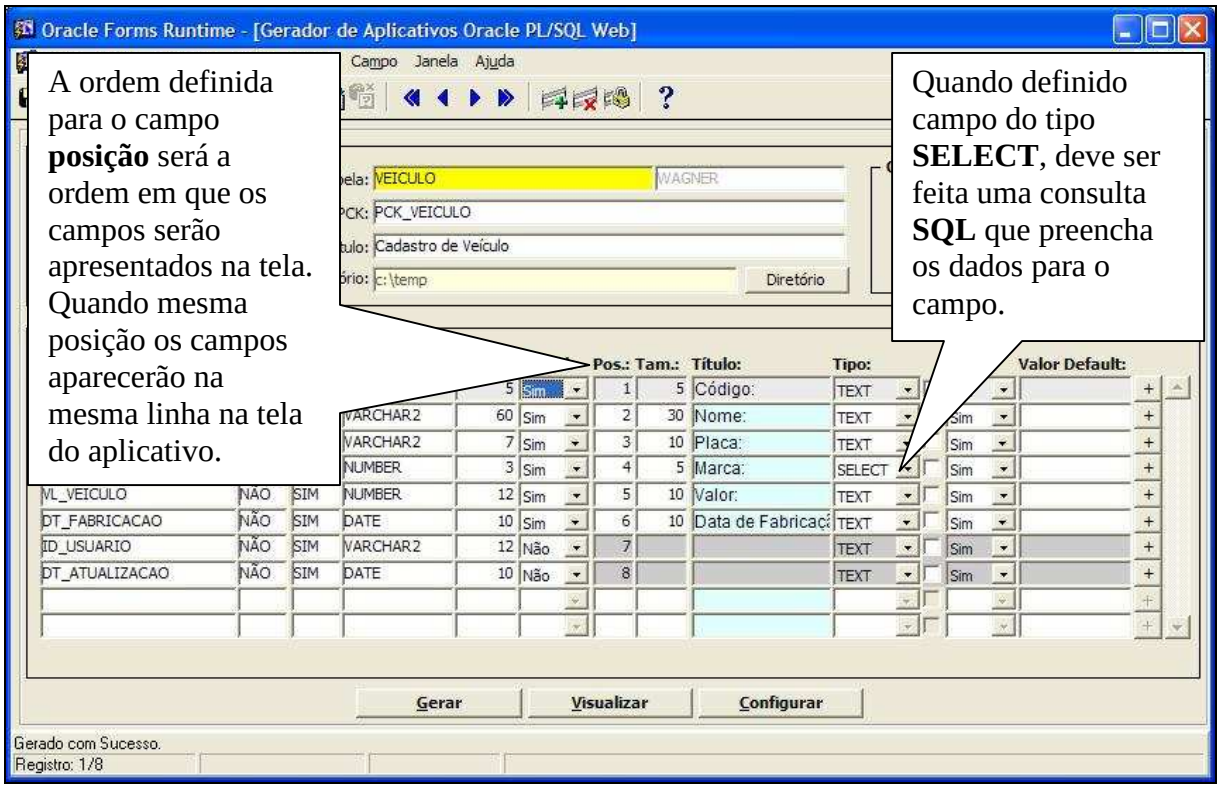


Figura 21 – Configuração de apresentação dos campos

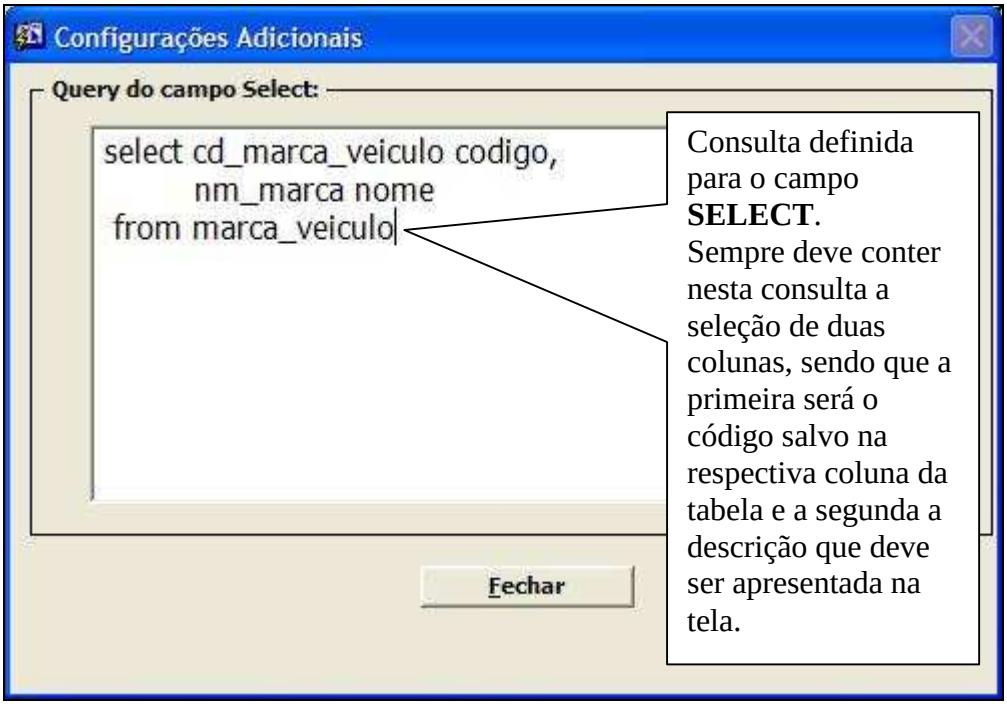


Figura 22 – Consulta do campo do tipo select

Para finalizar a criação do aplicativo, precisam ser feitas as configurações de geração gerais da ferramenta, conforme apresentado na figura 23. Nesta etapa devem ser feitas as definições das validações que serão chamadas em JavaScript para validar os valores

informados nos campos do aplicativo, bem como as classes que devem ser utilizadas do arquivo de estilos css para cada tipo de campo. A definição do endereço http em que o aplicativo deve rodar, também e definido nesta tela.

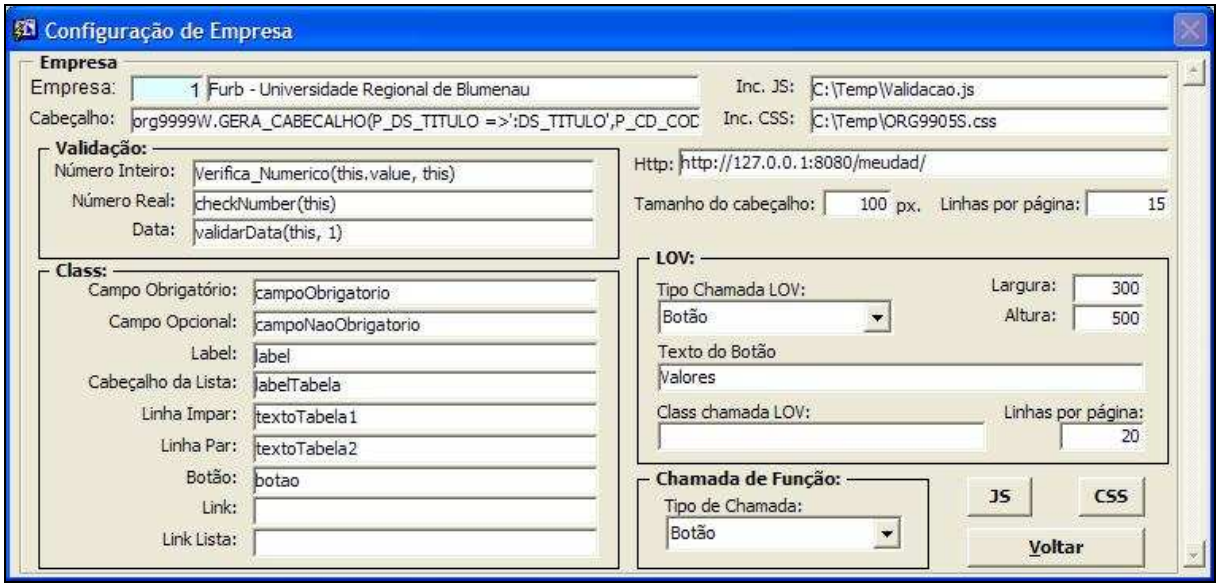


Figura 23 – Configurações gerais

Por fim, na tela principal da ferramenta é pressionado o botão Visualizar, que faz a geração do aplicativo PL/SQL Web, compila a package com o código do aplicativo gerado pela ferramenta no banco de dados e faz automaticamente a sua execução através do navegador de internet, conforme apresentado nas figuras 26, 27 e 28.

Caso o usuário precise somente gerar o código do aplicativo deve ser pressionado o botão Gerar que irá somente fazer a geração no diretório especificado pelo usuário no campo Diretório da ferramenta, caso este campo não seja informado o diretório padrão utilizado será em “C:\Temp”.

O aplicativo gerado pela ferramenta já possui algumas customizações que facilitam a utilização do usuário, a seguir são listadas essas facilidades:

a) qualquer campo da tela pode ser utilizado para filtrar os dados na realização de uma consulta, no aplicativo gerado como exemplo pode-se pesquisar, por exemplo, por uma marca de veículo especifica ou mesmo por sua data de fabricação;

b) ao informar um campo de data, automática o aplicativo ira colocar na seguinte máscara DD/MM/AAAA, caso a data informada não seja uma data válida o sistema também irá validar e a mensagem apresentada na figura 24 será apresentada, sendo que para isto o usuário deve ter feito anteriormente a configuração da função em JavaScript chamada para validar os campos de data na tela de configurações de empresa;

Figura 24 – Mensagem de data inválida

c) ao ser pressionado o botão Deletar do aplicativo a mensagem apresentada na figura 25 será apresentada para o usuário e somente se pressionado o botão OK o registro será apagado da base de dados, caso o botão Cancelar seja pressionado nada será feito;

Figura 25 – Mensagem de deleção de registro

d) a quantidade de registros apresentados na consulta do aplicativo sempre é mostrada na parte inferior esquerda da tela, conforme destacado na figura 28 e também no apêndice A com a figura 29.

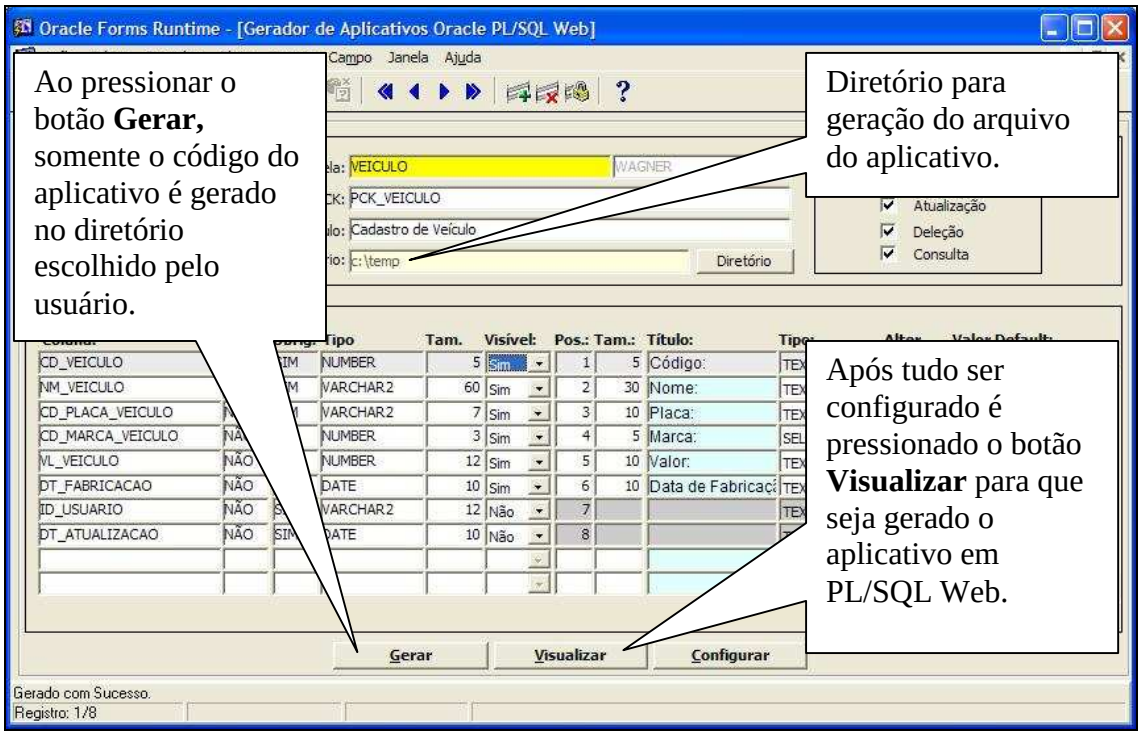


Figura 26 – Aplicativo de geração do PL/SQL Web

```
IF P_CD_VEICULO IS NOT NULL AND
   P_NM_VEICULO IS NOT NULL AND
   P_CD_PLACA_VEICULO IS NOT NULL AND
   P_CD_MARCA_VEICULO IS NOT NULL AND
   P_VL_VEICULO IS NOT NULL AND
   P_DT_FABRICACAO IS NOT NULL THE
  INSERT INTO VEICULO (
         CD_VEICULO
        ,NM_VEICULO
        ,CD_PLACA_VEICULO
        ,CD_MARCA_VEICULO
        ,VL_VEICULO
        ,DT_FABRICACAO)
      VALUES (w_max_CD_VEICULO
        ,P_NM_VEICULO
        ,P_CD_PLACA_VEICULO
        ,P_CD_MARCA_VEICULO
        ,P_VL_VEICULO
        ,to_date(P_DT_FABRICACAO,'dd/mm/rrrr'));
  COMMIT;
  ATUALIZ_HINT('Registro salvo com sucesso!');
ELSE
  ATUALIZ_HINT('Preencha todos os campos obrigatórios.');
END IF;
```

Trecho do código gerado automaticamente pelo aplicativo. No ponto indicado pela seta, pode ser visto a instrução de inserção na tabela denominada **veiculo**.

Figura 27 – Parte do código gerado pela aplicação

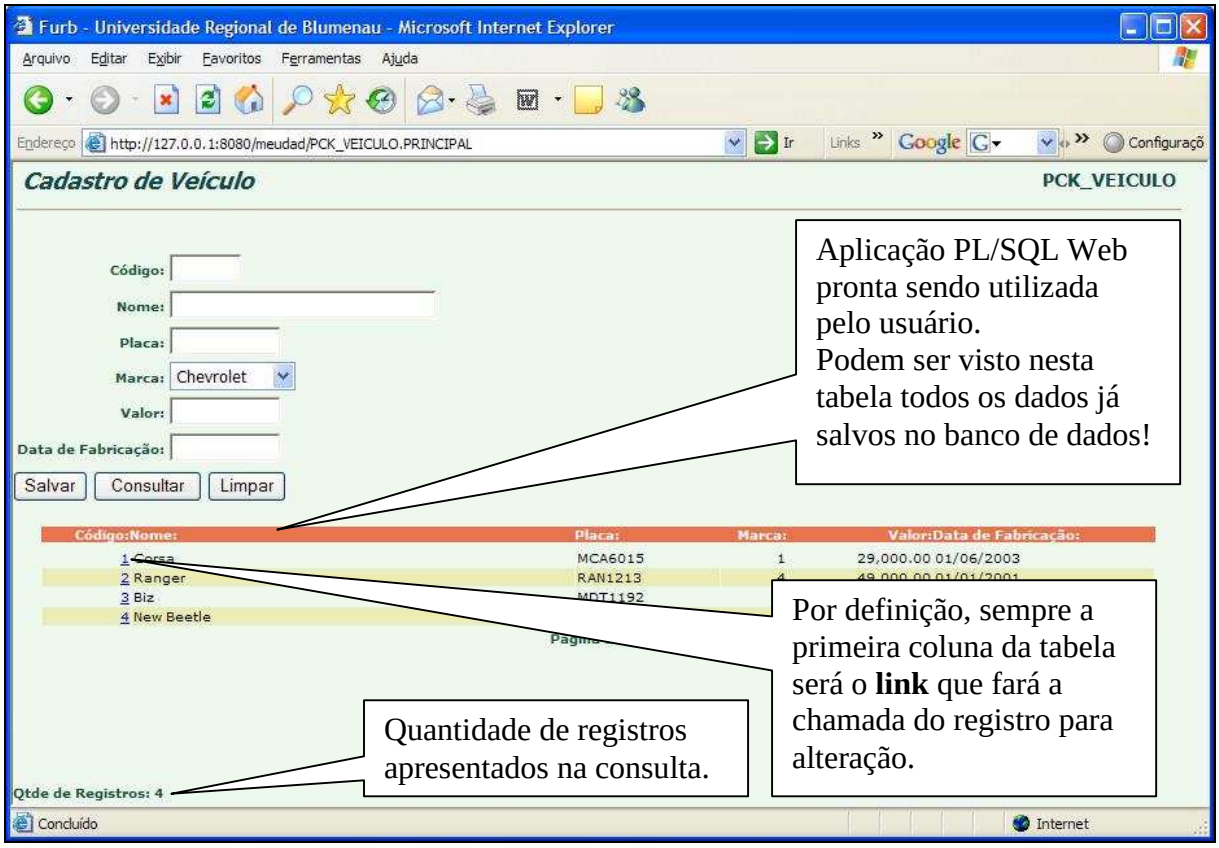


Figura 28 - Programa PL/SQL Web finalizado

3.5 RESULTADOS E DISCUSSÃO

A ferramenta foi concluída com sucesso, conseguindo fazer todos os requisitos propostos na realização deste trabalho. Como adicionais, foram feitos alguns facilitadores para a ferramenta conforme listados a seguir:

a) agilidade: a possibilidade da criação de lista de valores para auxiliar o preenchimento dos campos do aplicativo facilitou o uso do aplicativo e a integridade das informações;
b) ordenação: as consultas geradas podem ser ordenadas por qualquer coluna apresentada no aplicativo, bastando para isto clicar sobre o seu título;
c) apresentação: com a utilização de folhas de estilos css, as páginas geradas ficaram com estilos atuais e modernos e com isto, deixando a visualização do aplicativo completamente dinâmica e de fácil alteração.

Algumas das regras para a geração de código, propostas por Herrington, foram seguidas para facilitar o desenvolvimento da ferramenta e deixá-la rápida e fácil de usar. De um modo geral, a ferramenta realmente veio para ajudar as tarefas do dia a dia dos programas de PL/SQL Web, sendo que com sua utilização os códigos gerados serão padronizados, sem erros ou falhas comuns que ocorrem constantemente na área da programação.

4 CONCLUSÕES

A ferramenta construída permite a geração de aplicativos com as funcionalidades básicas de um sistema que são inclusão, exclusão, alteração e relatórios de consulta. Nas consultas é gerada automaticamente a opção de ordenação por qualquer coluna, bastando clicar sobre o título da coluna para que a pesquisa seja refeita e apresentada conforme a coluna escolhida.

Para atingir os objetivos específicos do trabalho foram feitos diversos levantamentos de como funciona a geração de aplicativos PL/SQL Web quando feitos de forma manual, ou seja, linha a linha do código programadas para cada novo aplicativo. Com este levantamento foram identificados padrões de codificações, que convertidos para linguagem de programação, deram a origem a ferramenta para geração de código PL/SQL Web de forma automatizada.

Verificou-se também que com a utilização de ferramentas para a geração de códigos de aplicações prontas para o uso é muito mais fácil de poder fazer uma aplicação consistente e objetiva, sendo que, uma vez feita à ferramenta da forma que se deseja não será necessário ter que ficar validando o código de cada aplicação gerada pelos programadores.

Com o uso deste gerador de aplicações PL/SQL Web o tempo gasto no desenvolvimento de aplicações foi reduzido em relação à forma tradicional de construção das mesmas. A padronização do código é outro fator favorável à utilização de um software de geração de código como o realizado neste trabalho.

Por fim, este trabalho mostrou todos os passos que devem ser seguidos no dia a dia de um profissional da área de sistema de informação. Sendo que para sua realização conhecimentos de banco de dados, modelagem de dados, arquitetura, administração, matemática e outras tantas que poderiam ser citadas foram empregadas de modo a conseguir atingir os objetivos finais do trabalho.

O cotidiano de um profissional da área de sistemas de informação realmente é as etapas necessárias para a realização deste trabalho de conclusão de curso, sendo necessário o levantamento sobre o assunto que se deseja trabalhar, sendo necessário o estudo aprofundado do tema também denominado de fundamentação, podendo assim, fazer o desenvolvimento do trabalho que foi pedido e por final entregar um produto funcional que agrade o cliente e que represente realmente a sua profissão e seu valor no mercado de trabalho.

4.1 EXTENSÕES

Para trabalhos futuros, há necessidade ainda de melhorar a apresentação da tela gerada para que se adapte para cada resolução de navegador escolhido pelo usuário. Também é preciso ainda a criação de um aplicativo que permita o usuário definir a forma e o tipo dos campos que irão aparecer na tela de uma maneira visual, ou seja, ambiente gráfico que mostre uma pré-visualização da página web antes de gerar o código da aplicação.

Um ponto que poderia ser implementado em trabalhos futuros é o de possibilitar a definição de regras de negócio dentro da própria ferramenta, com isto, as customizações pós-geração seriam praticamente desnecessárias.

REFERÊNCIAS BIBLIOGRÁFICAS

AHO, Alfred V.; ULLMAN Jeffrey D.; RAVI Sethi. **Compiladores, princípios, técnicas e ferramentas**. Rio de Janeiro: Guanabara Koogan, 1995.

COELHO, Luis Fernando. **Gerador de código HTML baseado em dicionário de dados utilizando banco de dados.** 2006. 52 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

DATE, C. J. **Introdução a sistemas de bancos de dados.** Rio De Janeiro : Campus, 2000. xxiii, 803p, il. Tradução de: An introduction to Database Systems - 7.ed. americana.

DOLLA, Dyckson Dyorgio. **Ferramenta de apoio a reestruturação de código fonte em linguagem PL-SQL baseado em padrões de legibilidade**. 2001. xi, 61p, il. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

FERNANDES, Lucia. **Oracle para desenvolvedores: Oracle developer, Oracle 8-8i, curso completo.** Rio de Janeiro : Axcel Books, 2000. xciii, 1758p, il.

GARCIA-MOLINA, Hector; ULLMAN, Jeffrey D; WIDOM, Jennifer. **Implementação de sistemas de bancos de dados.** Rio de Janeiro : Campus, 2001. 685p, il. Tradução de: Database system implementation.

HERRINGTON, Jack. **CodeGeneration.** Califórnia: Manning, 2003.

MENIN, Juliane. **Gerador de código JSP baseado em projeto de banco de dados MySQL**. 2005.70 f, il. Trabalho de conclusão de curso - Universidade Regional de Blumenau, Curso de Ciências da Computação, Blumenau, 2005. Disponível em: <http://www.bc.furb.br/docs/MO/2005/306325_1_1.pdf>. Acesso em: 19 nov. 2007.

OLIVEIRA, Wilson José de. **Oracle 8i & PL-SQL.** Florianópolis: Visual Books, 2000. 250p.

ORACLE MAGAZINE, Estados Unidos. v.XXI, n.4, p. 24 – 34, julho/agosto 2007.

RAMALHO, José Antônio A.(José Antônio Alves). **HTML 4:** teoria e prática. 5. ed. São Paulo : Berkeley, 2001. 163p, il.

REZENDE, Ricardo. **Conceitos Fundamentais de Banco de Dados**: Conceitos Básicos. Disponível em: http://www.devmedia.com.br/articles/viewcomp.asp?comp=1649. Acesso em: 19 nov. 2007.

SCHVEPE, Claudio. **Gerador de código Java a partir de arquivos do Oracle Forms 6i.** 2006. 15 f. Proposta de Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

VENETIANER, Tomas. **HTML:** desmistificando a linguagem da Internet. Sao Paulo : Makron Books, 1996. xxvi, 249p, il.

APÊNDICE A – Aplicativo utilizando outra folha de estilo CSS

A seguir na figura 29 é apresentada o aplicativo de cadastro de veículos utilizando outra folha de estilo CSS.

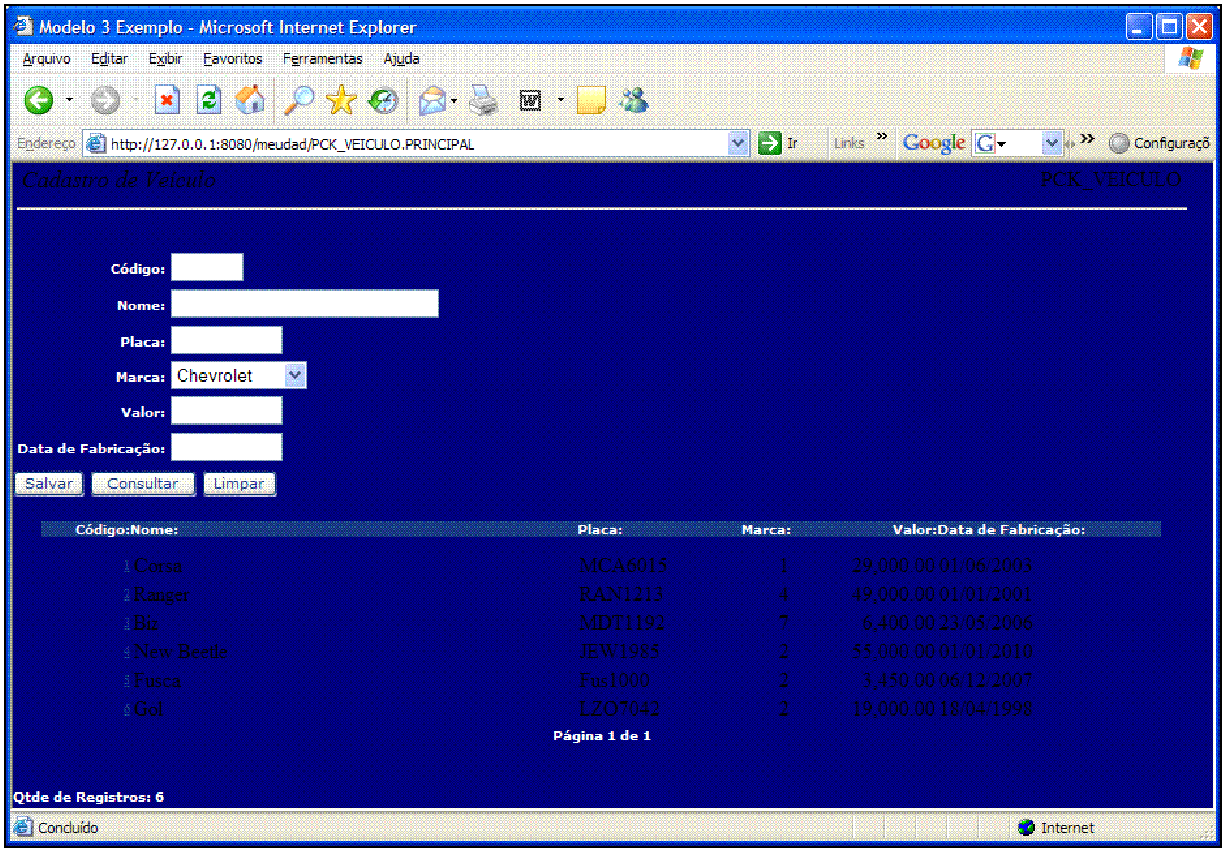


Figura 29 – Representação da mudança de estilo CSS