

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

GERENCIADOR DE HOMOLOGAÇÃO DE VERSÕES DO
SISTEMA SILOS NA BUNGE ALIMENTOS

ANDRÉ FERNANDO SPENGLER

BLUMENAU
2006

2006/1-02

ANDRÉ FERNANDO SPENGLER

**GERENCIADOR DE HOMOLOGAÇÃO DE VERSÕES DO
SISTEMA SILOS DA BUNGE ALIMENTOS**

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 Ciências
da Computação — Bacharelado.

Prof. Wison Pedro Carli , Orientador

**BLUMENAU
2006**

2006/1-02

GERENCIADOR DE HOMOLOGAÇÃO DE VERSÕES DO SISTEMA SILOS NA BUNGE ALIMENTOS

Por

ANDRÉ FERNANDO SPENGLER

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

Presidente: _____
Prof. Wilson Pedro Carli, Titulação – Orientador, FURB

Membro: _____
Prof. Everaldo Artur Grahl – FURB

Membro: _____
Prof. Marcel Hugo – FURB

Blumenau, 08 de junho de 2006

Dedico este trabalho à minha querida mãe Irene, por sempre estar ao meu lado nos momentos mais difíceis da minha vida.

AGRADECIMENTOS

Agradeço imensamente aos meus pais, Mário Francioso Spengler (in memoriam) e Irene Schmitt Spengler, que nunca mediram esforços para me dar três coisas: amor, disciplina e educação formal. Tudo que tenho e sou hoje, devo a eles. A meu irmão, Mário, meu companheiro de jornada nessa vida, muito obrigado pelo eterno carinho e apoio.

À minha noiva Fernanda, que com uma paciência incrível tolerou todos os momentos que subtraí de nosso convívio para executar esse trabalho e ainda assim me impulsionava para frente.

Ao meu orientador Prof. Wilson Pedro Carli pela atenção, amizade e profissionalismo com que me guiou nessa empreitada.

Às empresas Consei Consultoria em Informática e Bunge Alimentos S.A., pelo incentivo dado a este trabalho, pelo acesso aos dados e pela confiança.

Quem, de três milênios, não é capaz de se dar conta, vive na ignorância, na sombra, à mercê dos dias, do tempo.

Johann Wolfgang von Goethe

RESUMO

A ferramenta desenvolvida neste trabalho implanta um controle de reserva de servidores de testes baseado nas versões dos pacotes de classes existentes em cada servidor, conciliando-os com as dependências do novo *build* , ou seja, nova versão da aplicação a ser homologada. Dessa forma, o aplicativo auxilia a atividade de homologação de versões dos sistema silos da Bunge Alimentos uma vez que o testador, passa a poder liberar seus *builds* para homologação nos servidores de testes sem preocupar-se com a alocação de servidor para seus *builds* e com dependências de pacotes requisitadas pelo novo *build*.

Palavras-chave: *Build*. Homologação.

ABSTRACT

The tool developed in this work implants a control of reserve tests servers based in the existing packages in each server, conciliating them with the dependences of new build to be homologated, in other words, new version of the application to be ratified. Of this form, the software assists the activity of homologation of versions Bunge Alimentos system Silos version a time that the tester, passes it to be able to liberate its builds for homologation in the tests servers without being worried about the allocation of server for its builds and with dependences of packages requested for new build.

Key-words: Build. Homologation.

LISTA DE ILUSTRAÇÕES

Figura 1 – Processo de originação de grãos	14
Figura 2 – Arquitetura do Sistema.....	15
Figura 3 – Estrutura das atividades da GCS	16
Figura 4 – Número de versão do <i>build</i>	21
Figura 5 - Listagem dos pacotes e suas respectivas versões em servidor de teste	21
Figura 6 – Diagrama de Atividades para o Sistema Atual	27
Figura 7 - Diagrama de Atividades para o Sistema Desenvolvido.....	29
Figura 8 - Visão da fábrica de software.....	32
Figura 9 – Visão do analista de sistemas.....	32
Figura 10 – Visão do administrador do sistema	32
Quadro 1 – Caso de uso “Manter <i>build</i> para Homologação”	33
Quadro 2 – Caso de uso “Reservar Servidor”	34
Quadro 3 – Caso de uso “Atualizar situação de <i>build</i> ”	35
Quadro 3 – Caso de uso “Manter fábrica de software”	36
Quadro 4 – Caso de uso “Manter analistas Bunge”.....	37
Quadro 5 – Caso de uso “Manter servidores de testes silos”	37
Figura 11 – Modelo Físico de dados	39
Figura 12 – Tela inicial do sistema.....	43
Figura 13 – Manter cadastro de <i>build</i>	44
Figura 14 – Reserva de Servidor	45
Figura 15 – Lista de builds pendentes de avaliação e com reservas de servidor.....	46
Figura 16 – Lista de <i>builds</i> pendentes de avaliação e com reservas de servidor.....	46
Quadro 6 – Características de cada trabalho	48

LISTA DE SIGLAS

AJAX - Asynchronous Javascript and XML

GCS – Gerência de configuração

HTML – Hiper Text Markup Language

J2EE - Java 2 Enterprise Edition

JAR – Java Archive

RAD - Rapid Application Development

SIS – Curso de Sistemas de Informação – Bacharelado

XML – Extensible and Markup Language

SUMÁRIO

1 INTRODUÇÃO.....	11
1.1 OBJETIVOS DO TRABALHO	12
1.2 ESTRUTURA DO TRABALHO	13
2 FUNDAMENTAÇÃO TEÓRICA	14
2.1 SISTEMAS SILOS – VISÃO GERAL	14
2.2 GERÊNCIA DE CONFIGURAÇÃO.....	15
2.3 MANUTENÇÃO DE SOFTWARE.....	18
2.4 O PROCESSO DE MANUTENÇÃO DO SISTEMA SILOS	19
2.4.1 BUILD	20
2.5 TESTES DE SOFTWARE	22
2.6 TESTES PARA HOMOLOGAÇÃO DO SISTEMA SILOS	22
2.7 TRABALHOS CORRELATOS	23
3 DESENVOLVIMENTO DO TRABALHO.....	25
3.1 SISTEMA ATUAL	25
3.2 SISTEMA DESENVOLVIDO	27
3.3 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO.....	29
3.4 REQUISITOS NÃO FUNCIONAIS	30
3.5 ESPECIFICAÇÃO	31
3.5.1 Casos de Uso do projeto.....	31
3.5.2 Descrição dos casos de uso	33
3.5.3 Modelo de Dados	38
3.6 IMPLEMENTAÇÃO	39
3.6.1 Técnicas e ferramentas utilizadas.....	40
3.6.1.1 Genexus	40
3.6.1.2 Arquitetura J2EE	41
3.6.2 Operacionalidade da implementação	42
3.7 RESULTADOS E DISCUSSÃO	47
4 CONCLUSÕES	49
4.1 EXTENSÕES	50
REFERÊNCIAS BIBLIOGRÁFICAS	51
APÊNDICE A – Código Gerado pela ferramenta Genexus	52

1 INTRODUÇÃO

A Bunge Alimentos está presente no Brasil desde 1905 onde tornou-se a mais importante empresa na industrialização de soja e trigo e líder na comercialização de grãos como soja, trigo, milho, sorgo, girassol e semente de algodão. É também a maior exportadora brasileira no agronegócio está presente em 16 estados brasileiros, com unidades industriais; de armazenamento; moinhos; centro de distribuição, escritórios; e terminais portuários. Sua sede fica em Gaspar, Santa Catarina. O faturamento anual gira em torno de R\$ 15 bilhões e emprega, diretamente, mais de 7.000 pessoas. Compra de mais de 30 mil produtores rurais um volume em torno de 15 milhões de toneladas de soja, além de trigo, milho e caroço de algodão e se relaciona regularmente com clientes em quase 30 países (BUNGE ALIMENTOS, 2006).

Para gerenciar a aquisição de grãos Bunge Alimentos possui uma robusta solução para automatização de seus negócios em suas filiais de recebimento e armazenagem de grãos. Trata-se do Sistema Integrado de Gestão - Silos (ERP-Silos), projetado para atender as demandas peculiares da empresa nos silos.

Por tratar-se de um sistema amplo, moldado às necessidades da empresa e considerando a complexidade das leis e regras fiscais envolvidas nesse processo, principalmente as legislações estaduais e internacionais, há uma grande demanda por manutenções de ordem preventiva, corretiva e principalmente evolutiva, uma vez que a própria empresa encontra-se em constante mudança. O departamento de desenvolvimento de sistemas é responsável pela execução das customizações necessárias solicitadas pelos usuários do sistema.

O processo de desenvolvimento de software na Bunge Alimentos acontece sempre em parceria com empresas prestadoras de serviço em desenvolvimento de sistemas. A partir de uma documentação contendo todos os requisitos funcionais e não funcionais a empresa

contratada implementa a customização solicitada e entrega uma versão do sistema para ser homologada.

Com alta demanda de projetos paralelos gera-se uma grande quantidade de versões entregues pelas fábricas de software para serem validadas. Estas versões geralmente possuem dependências entre si, mesmo quando se tratam de módulos diferentes do sistema.

Segundo Martins (2006), um plano de gerenciamento de configuração, voltado à área de desenvolvimento de software fornece meios para identificar, controlar e seguir as diferentes versões de cada item do trabalho.

O processo de validação de novas versões (*builds*) requer um controle da disponibilidade de servidores de testes e gerenciamento dos pacotes liberados nesses servidores para não ocorrer sobreposição de versões. Desta forma, verificou-se a necessidade de uma ferramenta que auxiliasse o processo de validação de novas versões no que se refere à gerência de servidores e pacotes liberados.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho foi desenvolver uma ferramenta para gerenciar a homologação e implantação das versões liberadas pelas fábricas de software. Os objetivos específicos do trabalho são:

- a) entender o processo atual de homologação de *builds*;
- b) aprimorar o processo de homologação de *builds*;
- c) dominar a ferramenta Genexus para desenvolvimento para *web* na plataforma Java J2EE;

1.2 ESTRUTURA DO TRABALHO

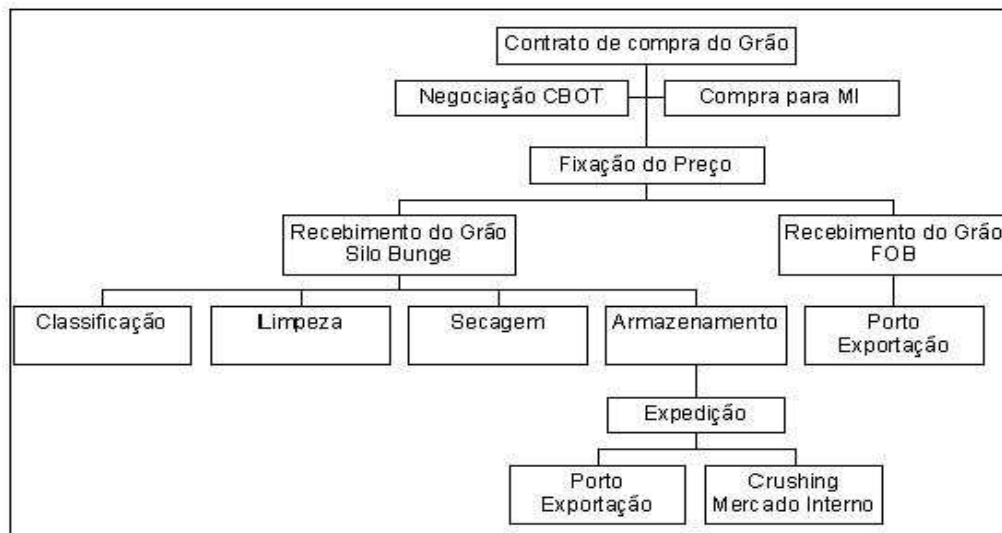
A estrutura deste trabalho foi dividida em quatro capítulos. No primeiro capítulo, encontra-se uma introdução e os objetivos a serem alcançados com o desenvolvimento deste trabalho. No segundo capítulo é apresentada a fundamentação teórica para os itens este trabalho, contextualizando o sistema silos, gerência de configuração, manutenção de software, manutenção do sistema silos, *build*, testes de software e testes para homologação do sistema silos. O terceiro capítulo aborda o desenvolvimento do trabalho, apresentando os requisitos, especificação e artefatos gerados durante a especificação do projeto, ferramentas utilizadas na especificação, desenvolvimento e execução do protótipo, resultados e problemas encontrados durante o desenvolvimento. Finalmente, o quarto capítulo trata das considerações finais sobre o trabalho e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O presente capítulo tem por objetivo tratar do ambiente de desenvolvimento de sistemas da Bunge Alimentos para o sistema Silos. Inicialmente é apresentada uma visão geral do sistema silos e processo de manutenção do sistema silos com ênfase no fluxo de homologação de novas versões do sistema. Por fim, são enfatizadas as técnicas e ferramentas relevantes envolvidas na implementação deste trabalho e trabalhos correlatos ao tema.

2.1 SISTEMAS SILOS – VISÃO GERAL

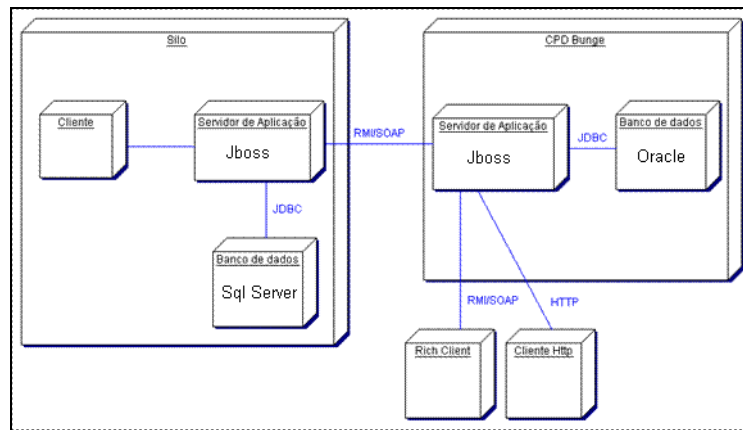
Segundo Heusi (2005), o Sistema de Gestão Integrada – Silos contempla todos os processos envolvendo a área de originação, que é o departamento responsável pela aquisição de grãos. Conforme a Figura 1, o processo se inicia na contratação e fixação do grão e se encerra na entrega do produto, seja no porto para exportação ou em uma fábrica da Bunge para processamento (Divisão *Crushing*).



fonte: Heusi (2005)

Figura 1 – Processo de originação de grãos

O sistema de silos foi desenvolvido utilizando-se uma arquitetura que gere a precária infra-estrutura de comunicações existente nos silos. Conforme a Figura 2, o ambiente nos silos é composto basicamente de um servidor onde se encontra instalado o servidor de aplicações e um banco de dados, e diversos clientes conectados a esse servidor de aplicações. Os clientes trabalham conectados localmente ao servidor de aplicações (*Jboss*), que possui um mecanismo de fila que armazena uma cópia de todas as transações executadas no banco de dados e conforme a disponibilidade do *link* replica esses comandos para a matriz da Bunge, onde são executados da mesma forma que localmente. Com isso, disponibilizou-se um ambiente que gera todas as transações para a matriz conforme a disponibilidade do *link*.



fonte: Heusi(2005)

Figura 2 – Arquitetura do Sistema

2.2 GERÊNCIA DE CONFIGURAÇÃO

A GCS pode ser definida como o processo através do qual todos os componentes de um sistema de informação (hardware e *software*), as suas inter-relações, a sua documentação e mesmo os processos de trabalho associados são administrados, principalmente no que se refere às parametrizações para o bom funcionamento do conjunto e as alterações que se tornam necessárias (PRESSMAN, 1995).

O principal objetivo da GCS é possibilitar o controle das mudanças de forma eficiente, contribuindo para a melhoria do processo de desenvolvimento. Segundo Oliveira et al (2001):

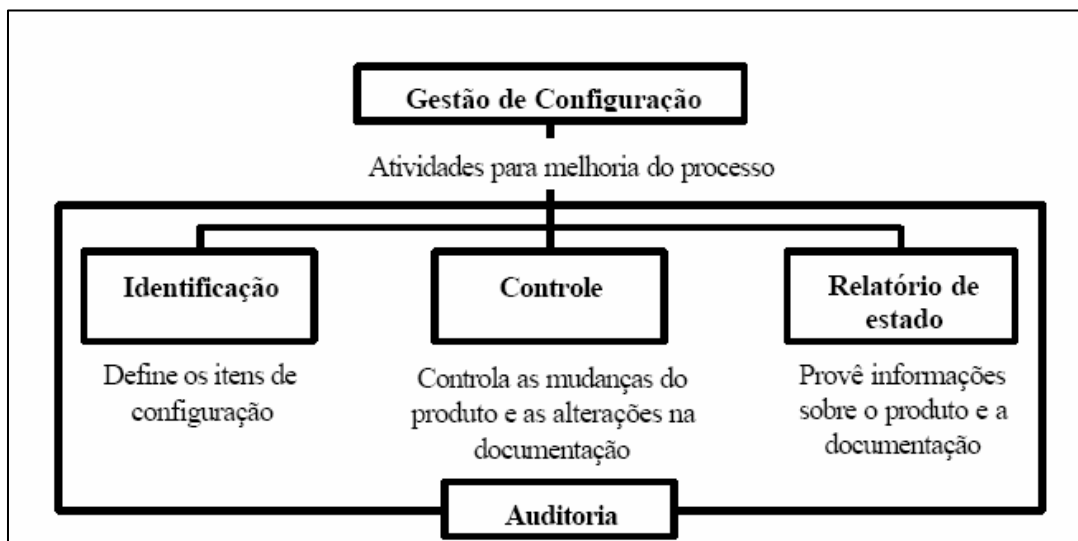
A Gerência de Configuração de *Software* propõe-se a minimizar os problemas surgidos durante o ciclo de vida do *software* através de um controle sistemático sobre as modificações. Não é objetivo da GCS coibir as modificações, pelo contrário, é facilitá-las. Procura-se apenas cuidar que elas ocorram de forma sistemática e controlada para evitar, ou minimizar, as suas decorrências negativas.

Conforme SILVA et al. (2005, p.4):

Para a implantação da GCS não existe nenhum tipo de roteiro formalizado, que possa ser aplicado em todos os tipos de organização, uma vez que cada uma delas apresenta cenários diversos umas das outras, tais como necessidades diferentes, além de variados tipos de recursos humanos e computacionais.

Segundo Pressman (1995) quando não se controla as mudanças, elas controlam a organização, indicando que os projetos devem possuir mecanismos que possibilitem suas modificações, baseado na premissa de que as mudanças são inevitáveis e a complexidade tende a aumentar à medida que mais pessoas são envolvidas.

Tradicionalmente, a gestão de configuração é aplicada através de quatro atividades principais, conforme a Figura 3.



Fonte: Pressman (1995)

Figura 3 – Estrutura das atividades da GCS

A seguir, cada uma das principais atividades da GCS são detalhadas, baseadas em Pressman (1995):

- a) identificação: consiste em selecionar os itens de configuração (*software configuration items* - SCIs) e armazenar suas características físicas e funcionais. Os SCIs são as informações que são criadas como parte do processo de engenharia de *software*. Qualquer documento importante para o desenvolvimento deve tornar-se um item de configuração, mas os itens mais comuns são: Especificação do Sistema, Plano de Projeto de *Software*, Especificação dos Requisitos de *Software*, Manual Preliminar do Usuário, Especificação do Projeto, Listagem do código fonte, Plano e Procedimento de testes, entre outros;
- b) controle: uma das partes mais críticas da gestão de configuração. O controle consiste na implementação de mecanismos que regulem as modificações sob a alçada da gestão de configuração. Deve ser levado em conta a natureza da modificação, a identificação de novos elementos envolvidos e o efeito da alteração nos demais elementos do sistema. O mecanismo de controle deve evitar a ocorrência de incompatibilidade;
- c) auditoria: processo realizado para verificar se o produto desenvolvido está de acordo com o que foi definido nas especificações ou em outra documentação contratual. Existem duas vertentes essenciais:
1. PCA - *Physical Configuration Audit*: Verificação física da localização do hardware, *software* ou documentação;
 2. FCA – *Functional Configuration Audit*: Verificação de que as práticas anteriormente definidas estão sendo corretamente realizadas;
- d) relatório de estado: consiste no registro e relato do estado da configuração completa do sistema em momentos definidos. Os relatórios devem ser capazes de responder às seguintes perguntas: O que aconteceu? Quem fez? Quando aconteceu? O que mais será afetado?

Para Oliveira et al (2001), no Brasil, o assunto ainda é pouco conhecido e não existem no mercado ferramentas nacionais de destaque para o controle das atividades automatizáveis da GCS. Atualmente, percebe-se um interesse crescente por GCS, o que se deve, em grande parte, à busca por certificações de qualidade, como ISO 9000, ou nível de maturidade, de acordo com o CMM, mas também devido às necessidades colocadas pela competição do mercado que, cada vez mais, exige qualidade.

De acordo com Oliveira et al (2001), verifica-se maior incidência do uso de GCS em empresas multinacionais ou de grande porte. Além disso, deve-se observar também que o aumento de formalismo altera a rotina de trabalho da organização como um todo, gerando uma série de procedimentos adicionais de trabalho que devem ser seguidos. Esta situação torna-se mais grave quando se observa pela ótica da realidade nacional, na qual boa parte dos produtos de *software* é desenvolvida por pequenas empresas.

Conforme SILVA et al. (2003, p.6):

a implantação de GCS e sua utilização em uma organização não é uma tarefa fácil, mas deve-se manter em foco que as mudanças ocorridas com a implantação trazem muitos benefícios para a qualidade do *software* produzido, melhorando a satisfação do cliente, diminuindo os retrabalhos e, principalmente, facilitando as manutenções e alterações no *software* durante seu ciclo de vida. A mesma possibilita ainda que mudanças simples possam ser feitas em *softwares* complexos sem que ocorra perda de qualidade e funcionalidade, e que mudanças complexas possam ser feitas de forma pontual e gradativa, de maneira que estas mudanças se tornem simples e ocorram sem dificuldades. Isto faz com que uma organização que adote a GCS torne-se flexível e competitiva no mercado, pois produz *software* com qualidade e flexibilidade durante todo o ciclo de vida do mesmo.

2.3 MANUTENÇÃO DE SOFTWARE

De acordo com Pressman (1995), pode-se definir a manutenção descrevendo quatro atividades que são levadas a efeito depois que um programa é liberado para uso:

- a) manutenção corretiva: O processo que inclui diagnóstico e a correção de um ou

mais erros . Não é razoável presumir que a atividade de testes de software descobrirá todos os erros latentes num grande sistema de software. Durante o uso de qualquer programa grande, erros ocorrerão e serão relatados ao desenvolvedor;

- b) manutenção adaptativa: atividade que modifica o software para que ele tenha uma interface adequada com um ambiente mutante. Ou seja, surgem novas gerações de hardware, novos sistemas operacionais, periféricos e outros elementos de sistema e o sistema é modificado para suportar tais evoluções;
- c) manutenção perfectiva: atividade responsável pela inclusão de novas capacidades, modificação em funções existentes geralmente requisitadas por usuários. Este tipo de manutenção é responsável pela maior parte do esforço despendido em manutenção de software;
- d) manutenção preventiva: atividade de modificação de software para melhorar a confiabilidade ou manutenibilidade futura, ou para oferecer uma base melhor para futuras ampliações.

2.4 O PROCESSO DE MANUTENÇÃO DO SISTEMA SILOS

O Sistema Silos gera uma grande demanda por manutenções de ordem corretiva, adaptativa, preventiva. e principalmente perfectiva, uma vez que a própria empresa encontra-se em constante mudança. O departamento de desenvolvimento de sistemas é responsável pela execução das manutenções necessárias solicitadas pelos usuários do sistema (HEUSI, 2005).

Por opção estratégica, todas atividades de atendimento e correções do sistema silos (Suporte Silos) são desempenhadas por uma empresa parceira. Apenas a homologação das correções efetuadas são efetuadas por recursos da Bunge Alimentos. Semanalmente o suporte silos envia ao departamento de desenvolvimento de sistemas uma nova versão do sistema

silos intitulada de *build* semanal contendo um pacote de correções reivindicadas por usuários dos silos. Esta nova versão é validada por analistas de sistemas responsáveis por cada módulo alterado.

Paralelamente, a equipe de desenvolvimento de sistemas da Bunge Alimentos desenvolve uma grande quantidade de projetos para atender novas necessidades dos usuários do Sistema Silos. De acordo com a metodologia da empresa, para cada alteração ou adição de funcionalidade é aberta uma solicitação de serviço, onde devem ser documentado o escopo da alteração desejada e informações necessárias para implementação da mesma. Esta ficará sob a responsabilidade do analista de sistemas e deverá ser encaminhada para uma fábrica de software para implementação. A fábrica de software deverá no prazo pré-estipulado disponibilizar um *build* da aplicação para homologação pelo responsável do projeto na Bunge Alimentos.

2.4.1 BUILD

Na convenção de termos relacionados à gerência de configuração da Bunge Alimentos, um *build* é um conjunto de pacotes de classes Java e outros arquivos de configuração agrupados em um arquivo compactado para liberação nos silos. Por sua vez, os pacotes contém um conjunto classes de Java, geralmente agrupadas por funcionalidade. Cada pacote é identificado com o número da versão do *build*. Este número identificador fica localizado no arquivo *manifest.inf* e é compactado junto com as classes Java formando um arquivo com a extensão “.jar”.

Conforme figura 4, a numeração do *builds* é definida por quatro dezenas e opcionalmente uma expressão que identifica o tipo de *build*. Cada dezena possui um significado, sendo que a primeira representa o número principal da versão, a segunda o

número da liberação, a terceira é incrementada a cada alteração de escopo para aquela liberação e a quarta é incrementada cada vez que a liberação é rejeitada. Opcionalmente pode ser incluído uma expressão para identificar uma determinada peculiaridade da versão.

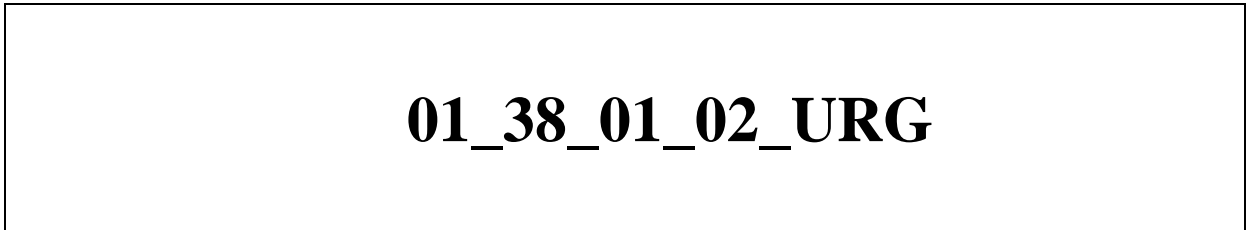


Figura 4 – Número de versão do *build*

Ao liberar um novo *build*, uma nova numeração é gerada conforme regras expostas. Para não sobrecarregar o tráfego de rede no momento da liberação para os silos, o *build* geralmente contém apenas os pacotes que possuem as classes Java alteradas.

A figura 5 mostra a listagem dos pacotes contidos no diretório do servidor de aplicações instalado em um dos computadores de testes da Bunge Alimentos.

c:\opt\jboss\server\default\deploy\silos.ear			
Arquivo	Versão	Última Alteração	Tamanho
BMInvokerSession-generic.jar	01_03_52_01	10/05/2006 14:37:46	4,876 KB
bungewsclient.jar	01_39_16_02_PAE22	23/05/2006 09:23:54	14,247 KB
caps-server.jar	01_03_52_01	10/05/2006 14:37:46	24,334 KB
Conceitos.jar	01_39_16_02_PAE22	23/05/2006 09:23:54	223,036 KB
DAOInvokerSession-generic.jar	01_03_52_01	10/05/2006 14:37:46	2,278 KB
geral-common-tecl.jar	01_39_16_02_PAE22	23/05/2006 09:23:54	37,714 KB
geral-common.jar	01_03_52_01	10/05/2006 14:37:46	155,937 KB
geral-server.jar	01_03_52_01	10/05/2006 14:37:46	43,796 KB
infra-server.jar	01_03_52_01	10/05/2006 14:37:46	311,655 KB
infra-services.jar	01_03_52_01	10/05/2006 14:37:46	129,391 KB
InvokerSession-generic.jar	01_03_52_01	10/05/2006 14:37:46	2,266 KB
LabsEJB.jar	01_39_16_02_PAE22	23/05/2006 09:23:54	137,876 KB
LabsServer.jar	01_39_16_02_PAE22	23/05/2006 09:23:54	370,579 KB
ldss-common.jar	01_03_52_01	10/05/2006 14:37:46	299,651 KB
ldss-server.jar	01_03_52_01	10/05/2006 14:37:46	192,511 KB
less-server.jar	01_03_52_01	10/05/2006 14:37:46	112,478 KB
LOV.jar	01_39_16_02_PAE22	23/05/2006 09:23:54	19,511 KB
LovEJB.jar	01_39_16_02_PAE22	23/05/2006 09:23:54	43,25 KB
ReportLayouts.jar	01_39_01_02	27/04/2006 09:44:54	510,39 KB
reports-lavouts.jar	01_03_52_01	10/05/2006 14:37:46	256,456 KB

Figura 5 - Listagem dos pacotes e suas respectivas versões em servidor de teste

2.5 TESTES DE SOFTWARE

Para Pressman(1995), o objetivo principal do projeto de casos de teste é derivar um conjunto de testes que tenha uma alta probabilidade de revelar defeitos no software. Para atingir esse objetivo, duas categorias diferentes de técnicas de projeto de casos de teste são usadas: o teste de caixa branca e o teste de caixa preta.

Os testes de caixa branca focalizam a estrutura de controle do programa. Os casos de teste são derivados para garantir que todas as instruções do programa tenham sido exercitadas pelo menos uma vez durante os testes e que todas as condições lógicas tenham sido exercidas.

Os testes de caixa preta são projetados para validar os requisitos funcionais, sem se preocupar com o funcionamento interno de um programa. As técnicas de teste de caixa preta concentram-se no domínio de informações do software, derivando os casos de teste ao dividir a entrada e a saída de uma maneira que proporcione uma satisfatória cobertura de teste (PRESSMAN, 1995).

2.6 TESTES PARA HOMOLOGAÇÃO DO SISTEMA SILOS

Segundo Vilas Boas (2003), o teste de software é um elemento crítico da garantia de qualidade de software e representa a última revisão da especificação, do projeto e da codificação. Por isso, o processo de teste deve ser executado durante todo o ciclo de vida do projeto do software. Do ponto de vista de custo, Vilas Boas (2003), afirma que aproximadamente 50% do tempo e mais de 50% do custo total de desenvolvimento de um produto de software é gasto em teste.

Na metodologia de desenvolvimento de sistemas para o sistema Silos da Bunge Alimentos, a etapa de testes possui duas fases distintas. A primeira ocorre na fábrica de

software, onde serão realizados os testes de caixa branca e caixa preta, com maior enfoque no primeiro. Esses testes seguem um roteiro específico, documentado em um caderno de testes. A segunda fase é denominada fase de homologação, e esta é executada pelo analista de sistema responsável pelo projeto. Nesta fase o analista deve realizar os testes de caixa preta e assim verificar se todos os requisitos foram contemplados. Um usuário do sistema geralmente participa dos testes nesta fase. Por fim, o analista deve fazer um teste de integração que envolva a replicação de dados entre silos e matriz.

2.7 TRABALHOS CORRELATOS

Pode-se citar como trabalho correlato o sistema “Controle de Versão”, utilizado para monitoração constante das versões do Sistema Silos na Bunge Alimentos. De acordo com o Heusi (2005), a ferramenta Controle de Versões identifica a versão do sistema instalado em cada silo e realiza uma comparação com um servidor base, na matriz na cidade de Gaspar, para verificar quais servidores estão desatualizados. No entanto, o objetivo da proposta aqui apresentada é controlar os servidores e os testes de homologação de novas versões do sistema Silos.

Outro trabalho relacionado intitula-se Sistemas de Informação Aplicado a Gestão de *Outsourcing* na TI da Bunge Alimentos. Segundo Schumacker (2005), este sistema controla atividades contratadas pelo departamento de desenvolvimento de sistemas na Bunge Alimentos. O sistema disponibiliza indicadores de produtividade, horas trabalhadas por projeto e outros relacionados a recursos terceirizados. A ferramenta aqui proposta não objetiva medir desempenho de pessoas. A proposta objetiva maximizar a utilização dos servidores, orientar os analistas no *deploy* da aplicação para os servidores de testes e incentivar a realização dos testes de homologação no horário programado.

Conseqüentemente, espera-se um aumento de produtividade dos recursos envolvidos nessas atividades. No entanto, esses recursos incluem funcionários da Bunge Alimentos e colaboradores contratados de empresas parceiras.

3 DESENVOLVIMENTO DO TRABALHO

Para o desenvolvimento do trabalho foram necessários um levantamento e análise dos requisitos que definem as características que o sistema deve ter. Os requisitos deste projeto foram levantados e definidos em uma reunião realizada na Bunge Alimentos, com a participação dos usuários principais, onde foram identificadas necessidades dos analistas de sistemas da Bunge em relação à aplicação do Gerenciador de Homologação. Uma especificação se fez necessária para expressar como se encontra o fluxo de homologação de *build* atualmente e como seria o fluxo desejável. A análise dos requisitos e suas especificações são tratadas a seguir

3.1 SISTEMA ATUAL

Os *builds* disponibilizados pelas fábricas de software e Suporte Silos, são testados independentemente, nos servidores de testes. Atualmente, a Bunge possui três servidores destinados para testes do Sistema Silos e um servidor exclusivo para testes do módulo de replicação de dados. Embora, teoricamente mais de um *build* possa ser aplicado em um mesmo servidor, desde que os pacotes de classes desses não coincidam, na prática, este procedimento é evitado. Isso é causado devido à dificuldade de comparar manualmente os vários pacotes de classes de cada versão e falta de conhecimento técnico da estrutura da plataforma *Java J2EE*. Com isso, os servidores são sub-utilizados e freqüentemente o número de *builds* para serem testados supera os servidores de testes. Essa situação gera atraso na implantação dos projetos. Dentre os motivos para o sub- aproveitamento dos servidores de testes pode-se citar:

- a) falta de planejamento referente ao tempo de utilização do servidor de testes. Há casos em que uma única versão monopoliza um servidor de testes por vários dias ou até semanas;
- b) inexistência de um controle pontual de acompanhamento da situação atual e o histórico de reprovação dos *builds*.

Atualmente, para amenizar os problemas relacionados à liberação de *builds* nos servidores de testes, foi delegado a uma única pessoa a tarefa de controle de ocupação dos servidores e *deploy* das aplicações de testes. Com isso, evita-se erros até então comuns como paradas no servidor de aplicações devido a inobservância de dependências de versões. No entanto, sobrecarrega-se o analista de suporte, responsável por esta tarefa e cria-se uma grande dependência do mesmo.

Conforme diagrama de atividades representado na figura 6, a fábrica de software disponibiliza um *build* da aplicação para homologação ao analista de sistemas responsável pelo projeto. O mesmo solicita ao analista de suporte o *deploy* da aplicação em um dos servidores de testes. Este verifica a disponibilidade de servidores de acordo com as dependências da aplicação. Na planilha denominada “Servidores x Versão” é consultada a relação de versões base implantadas em cada servidor de testes. O controle de dependências é feito comparando-se manualmente os pacotes da nova versão liberados com os pacotes encontrados no diretório do servidor. Esta tarefa demanda um tempo considerável, uma vez que há um grande número de pacotes. Caso não houver disponibilidade de servidor o *build* deverá aguardar. Se houver, o analista de suporte faz o *deploy* da aplicação e atualiza uma planilha “Servidores x Versão”. O analista de sistemas executa os testes e quando aprovado solicita ao analista de suporte a liberação do *build* em um silo de testes. Finalmente, com a aprovação dos usuários do silo de testes é liberada a versão para todos os silos restantes. Caso haja alguma correção necessária, a fábrica de software é comunicada e deverá providenciar

uma nova versão que deve passar pelo mesmo fluxo de homologação.

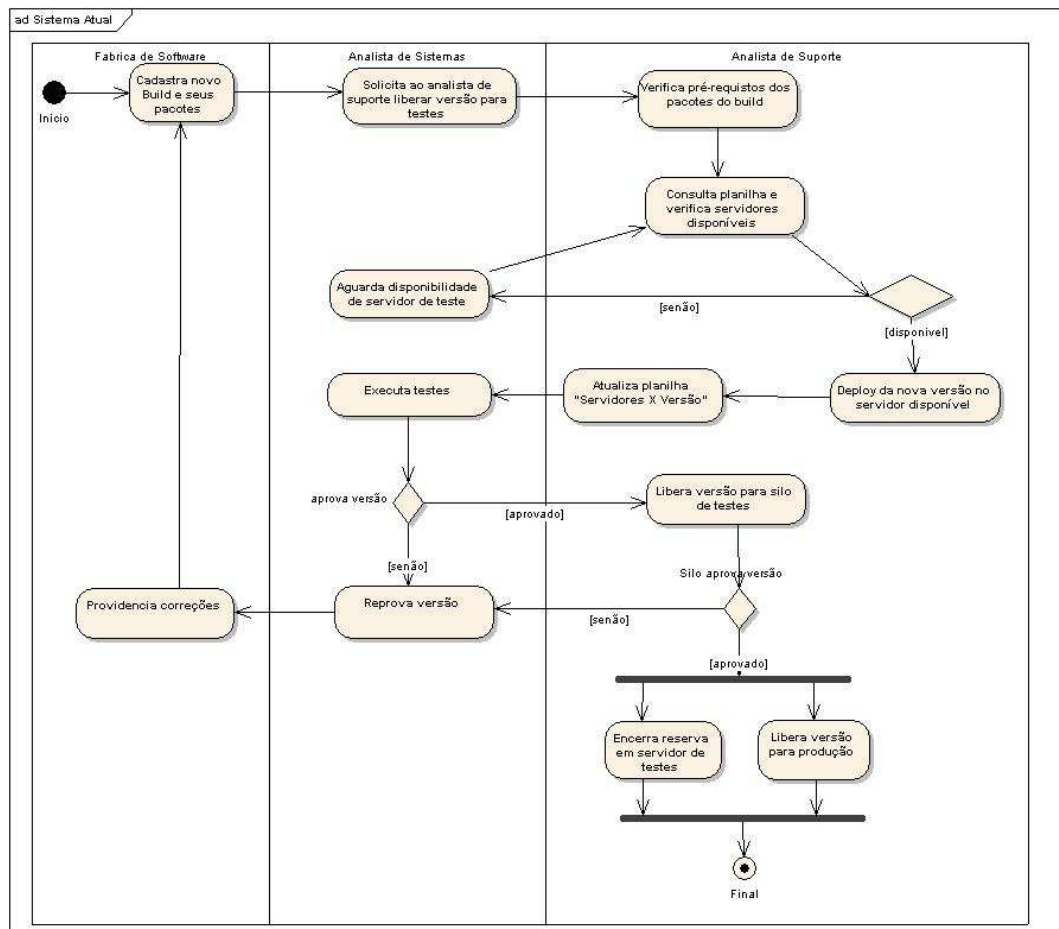


Figura 6 – Diagrama de Atividades para o Sistema Atual

3.2 SISTEMA DESENVOLVIDO

O sistema desenvolvido consiste em uma ferramenta auxiliar no processo de controle de homologação do Sistema Integrado de Gestão Silos. A intenção é retirar do analista de suporte a responsabilidade de verificação manual de versões nos servidores e *deploy* dos *builds* nos mesmos e transferi-la para o analista de sistemas. Para isso, a ferramenta deverá possibilitar a avaliação automática de dependência de pacotes de classes nos servidores de testes para um determinado *build*. Em seguida, o analista de sistemas deverá reservar um servidor por tempo limitado e só poderá estender este tempo se não houver *builds* em fila de espera. Ao finalizar os testes o *build* será aprovado ou reprovado.

O novo fluxo de processo para homologação de *builds* do sistema silos dispensa a função do analista de suporte, conforme ilustrado na figura 7. O fluxo se inicia quando a fábrica de software cadastra no sistema uma nova versão para homologação. Nesse momento serão informadas também as dependências dos pacotes da nova versão. Em seguida o sistema envia uma mensagem eletrônica para o analista de sistemas responsável pelo *build*. Quando melhor convier, o analista de sistemas verifica no sistema a disponibilidade de servidores de acordo com as dependências da versão. Então o mesmo deverá efetuar uma reserva de servidor. Caso não haja disponibilidade o sistema inclui o *build* em uma fila de espera e envia um e-mail para o analista tão logo disponibilize um servidor. Após efetuada a reserva o próprio analista deverá fazer o *deploy* da aplicação.

Desta forma, ganha-se agilidade no processo, pois o analista não depende mais de outro recurso para verificar disponibilidade de servidores, dependências de pacotes e *deploy* da aplicação. Ao finalizar os testes o processo de aprovação e reprovação de versões é o mesmo, exceto que ao invés de atualizar uma planilha o analista irá atualizar a situação do *build* na nova ferramenta.

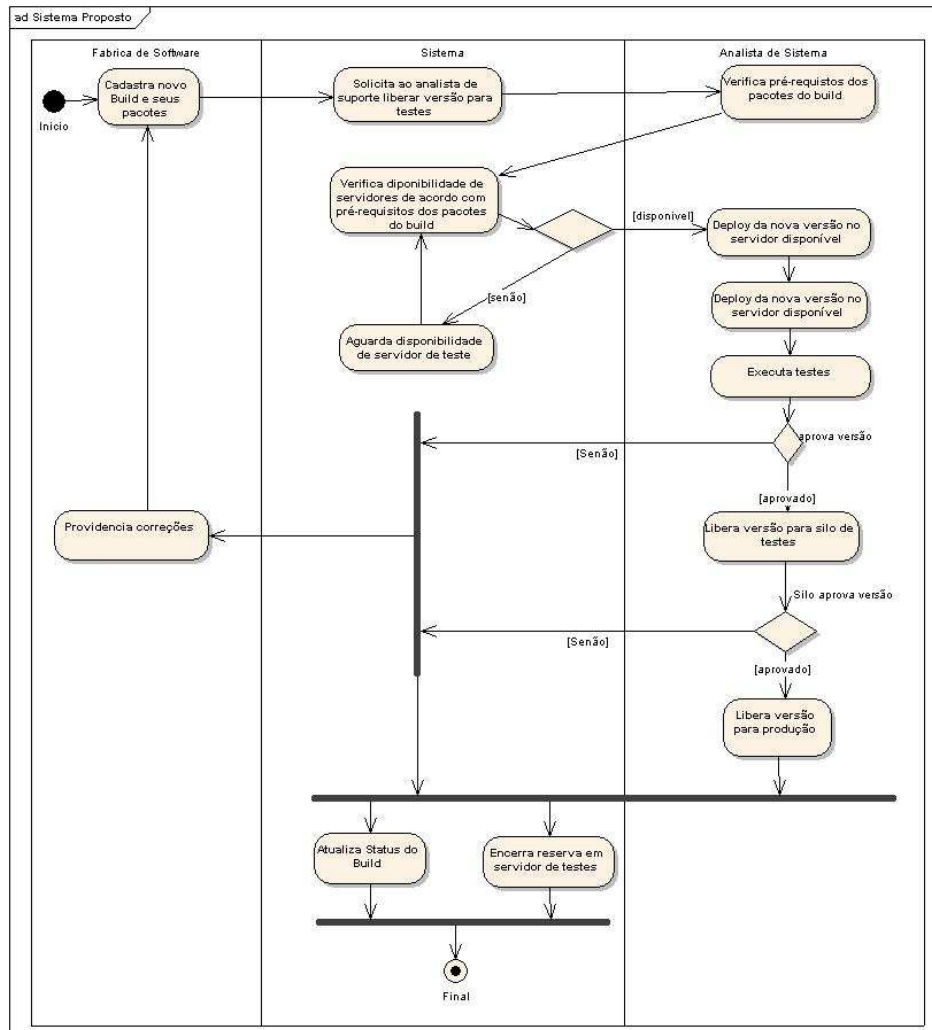


Figura 7 - Diagrama de Atividades para o Sistema Desenvolvido

3.3 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos funcionais especificam as principais características da camada de negócio do sistema. Estes requisitos descrevem os serviços que o sistema deve oferecer, como deve reagir a interação com o usuário e o comportamento do sistema em determinadas situações. Apenas os principais requisitos funcionais definidos na reunião realizada com os usuários foram contemplados neste trabalho, onde estes são:

- RF01: O sistema deverá permitir ao técnico da fábrica de software o cadastramento de *builds* para homologação, os pacotes que estão contidos no *build* e a versão pré-

requisito de cada pacote;

- b) RF02: O sistema deverá apresentar ao analista de sistemas os servidores disponíveis de acordo com o pré-requisito de cada pacote do *build* cadastrado em RF01;
- c) RF03: O sistema deverá possibilitar ao analista de sistemas a reserva de servidores para testes de homologação de *builds*;
- d) RF4: O sistema deverá possibilitar a consulta da situação de todos os *builds* em processo de homologação;
- e) RF5: O sistema deve registrar a aprovação/reprovação pelo analista de cada *build*;
- f) RF6: O sistema deverá permitir o cadastro de fábricas de software, de servidores de testes silos e analistas de sistemas.

3.4 REQUISITOS NÃO FUNCIONAIS

Requisitos de segurança, protocolo de comunicação e interface com o usuário são tratados como não-funcionais, mas apesar de não terem influência no negócio da aplicação, estão diretamente relacionados com a implementação do aplicativo. Assim sendo, são listados os requisitos não funcionais do sistema:

- a) RNF01: O sistema deverá ser construído em Java de acordo com a especificação J2EE. O mesmo deverá ser implantado no servidor Jboss;
- b) RNF02: O sistema deverá persistir dados no banco MS SQL Server 2000, padrão para aplicações em ambientes silos na Bunge Alimentos;
- c) RNF03: O sistema deverá verificar nos servidores de testes os pré-requisitos dos pacotes dos *builds* na tabela “Controle_Versao”, pertencente ao sistema de Controle de Versões.

3.5 ESPECIFICAÇÃO

A seguir serão abordados os casos de uso do sistema e diagrama de Entidade-Relacionamento.

3.5.1 Casos de Uso do projeto

Os casos de uso demonstram a interação do usuário com o sistema, destacando as ações que serão realizadas. Para a especificação do protótipo, foram definidos onze casos de uso:

- a) manter *build* para homologação;
- b) reservar servidor;
- c) adicionar *build* em fila de espera;
- d) manter reservas de servidor
- e) renovar reserva de servidor;
- f) aprovar *build*;
- g) reprovar *build*;
- h) cancelar reserva;
- i) consultar histórico de *builds*
- j) manter fábrica de software;
- k) manter analistas Bunge;
- l) manter servidores de testes para sistema silos
- m) manter situação de *build*

A seguir os casos de uso definidos são representados no diagrama de caso de uso, conforme as figuras 8, 9 e 10.

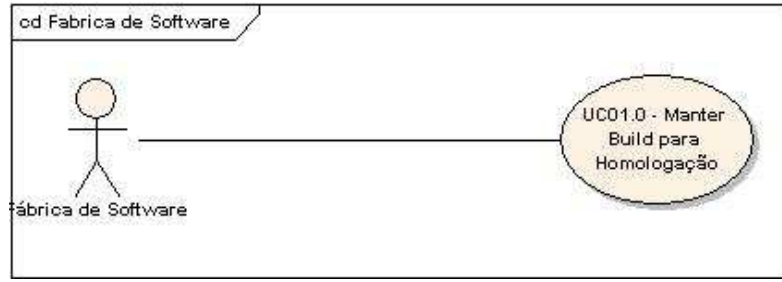


Figura 8 - Visão da fábrica de software

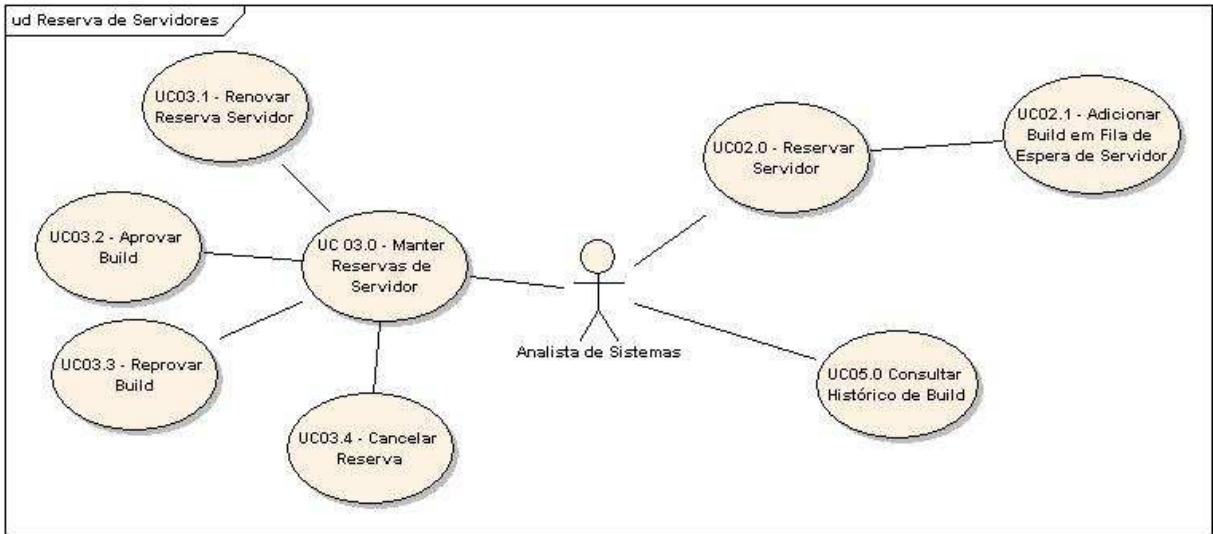


Figura 9 – Visão do analista de sistemas

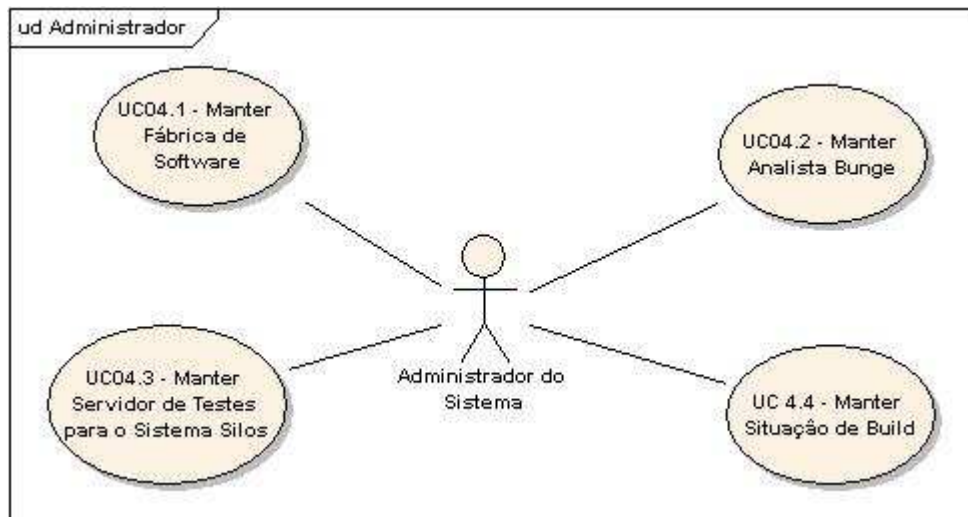


Figura 10 – Visão do administrador do sistema

3.5.2 Descrição dos casos de uso

A seguir serão detalhados os casos de uso especificados nas figuras 7, 8 e 9.

As descrições para os casos de uso podem ser vistas nos quadros de 1 à 5.

Caso de Uso	Manter <i>Build</i> para Homologação(UC01.0)
Sumário	Manter o cadastro(inclusão, remoção, alteração e consulta) dos dados sobre <i>builds</i> do sistema silos para serem homologados pela Bunge Alimentos.
Ator	Fábrica de Software
Pré Condições	Os pacotes de <i>builds</i> devem estar previamente cadastrados.
Fluxo principal	<ol style="list-style-type: none"> 1. O usuário requisita a manutenção do cadastro de <i>build</i>; 2. O sistema apresenta um formulário em branco com os dados do <i>build</i> (número, analista responsável Bunge, pacotes do <i>build</i>) e uma opção de consulta; 3. O usuário informa o número do <i>build</i>; 4. O sistema busca as informações referente ao número do <i>build</i> informado, habilita os campos para edição do registro e os botões de alterar e excluir; 5. O usuário efetua alterações no cadastro; 6. O sistema verifica a validade dos dados e confirma as alterações efetuadas.
Fluxo Alternativo(2): Consulta	<ul style="list-style-type: none"> • O usuário requisita consulta sobre lista de <i>builds</i> liberados; • O sistema apresenta uma lista de todos os <i>builds</i>, permitindo que o usuário selecione o <i>build</i> desejado; • O usuário seleciona o <i>build</i>; • O caso de uso continua a partir do passo 4.
Fluxo Alternativo(4): Inclusão	<ul style="list-style-type: none"> • O <i>build</i> informado no passo 4 não existe no cadastro e o sistema assume o modo de inclusão; • O usuário informa os detalhes do <i>build</i> (número, analista responsável Bunge, pacotes do <i>build</i> e versão pré-requisito do pacote); • O sistema verifica a validade dos dados e confirma a operação.
Fluxo Alternativo(4): Exclusão	<ul style="list-style-type: none"> • O usuário requisita a exclusão do registro; • O sistema verifica a exclusão do <i>build</i> e confirma a operação.
Fluxo de exceção	<ul style="list-style-type: none"> • Caso os dados sejam inválidos o sistema reporta o erro, solicita novos dados e repete a operação;
Pós-condições	Um <i>build</i> foi atualizado ou inserido ou removido do sistema.

Quadro 1 – Caso de uso “Manter *build* para Homologação”

Caso de Uso	Reservar Servidor (UC02.0)
Sumário	O analista de sistemas verifica a disponibilidade de servidores para o <i>build</i> selecionado e efetua a reserva do mesmo.
Ator	Analista de Sistemas
Pré Condições	Deve haver <i>builds</i> cadastrados para homologação pendentes de homologação.
Fluxo Principal: Reservar Servidor	<ol style="list-style-type: none"> 1. O usuário requisita a consulta de servidores para <i>build</i>; 2. O sistema exibe uma lista de valores com os <i>builds</i> pendentes de homologação; 3. O usuário seleciona o <i>build</i>; 4. O sistema verifica os pré-requisitos de cada pacote nos servidores de testes e exibe uma lista com detalhes referentes a disponibilidade de cada servidor e as seguintes opções: reservar servidor, adicionar <i>build</i> em fila de espera. 5. O usuário seleciona um servidor disponível para reserva e requisita reserva deste; 6. O sistema apresenta os dados da reserva para confirmação do usuário; 7. O usuário confirma a reserva; 8. O sistema registra a reserva de servidor emite uma mensagem de confirmação.
Fluxo Alternativo(5): adicionar servidor em lista de espera	<ul style="list-style-type: none"> • O usuário seleciona um servidor com a opção de “lista de espera” e requisita adição de <i>build</i> em lista de espera; • O sistema apresenta os dados da reserva para confirmação do usuário; • O usuário confirma a reserva; • O sistema registra a reserva de servidor emite uma mensagem de confirmação.
Fluxo de Exceção(4): ausência de servidor que satisfaça os pré-requisitos do <i>build</i>	<ul style="list-style-type: none"> • Se não houver servidor que contenha os pacotes que satisfaçam os pré-requisitos do <i>build</i> o sistema emite uma mensagem reportando o fato e o caso de uso termina.
Pós-condições	Um servidor foi reservado para testes de um <i>build</i> ou um <i>build</i> foi adicionado em lista de espera para um servidor.

Quadro 2 – Caso de uso “Reservar Servidor”

Caso de Uso	Manter Reservas de Servidor
Sumário	O analista de sistemas aprova/reprova o <i>build</i> e encerra reserva de servidor ou renova a reserva para continuar com o processo de homologação do <i>build</i> ou cancela a reserva de servidor.
Ator	Analista de Sistemas
Pré Condições	Deve haver reserva de servidores para homologação de <i>builds</i> pendentes de avaliação
Fluxo Principal: Aprovar <i>Build</i>	<ol style="list-style-type: none"> 1. O usuário requisita a consulta de <i>build</i> para homologação com reserva de servidores de testes; 2. O sistema exibe uma lista de <i>builds</i> pendentes de homologação que estejam alocados em servidores de testes e as operações que podem ser realizadas para cada <i>build</i>: a aprovação do

	<p><i>build</i>, reprovação do <i>build</i>, renovação de reserva de servidor ou cancelamento de reserva de servidor.</p> <ol style="list-style-type: none"> 3. O usuário requisita a aprovação do <i>build</i> selecionado; 4. O sistema apresenta os dados do <i>build</i> e da reserva de servidor para confirmação do usuário; 5. O usuário confirma aprovação do <i>build</i>; 6. O sistema atualiza a situação do <i>build</i> para “aprovado” e encerra a reserva de servidor.
Fluxo Alternativo(3): Reprovar <i>Build</i>	<ul style="list-style-type: none"> • O usuário requisita a reprovação do <i>build</i> selecionado; • O sistema apresenta os dados do <i>build</i> e da reserva de servidor para confirmação do usuário; • O usuário confirma reprovação do <i>build</i>; • O sistema atualiza a situação do <i>build</i> para “reprovado” e encerra a reserva de servidor.
Fluxo Alternativo(3): Renovar Reserva de Servidor	<ul style="list-style-type: none"> • O usuário requisita a renovação de reserva de servidor para continuação dos testes para homologação do <i>build</i>; • O sistema verifica se não há <i>build</i> em lista de espera para o servidor em questão; • Se não houver <i>build</i> em lista de espera para aquele servidor o sistema efetua a renovação de reserva de servidor; caso contrário, o sistema reporta o fato.
Fluxo Alternativo(3): Cancelar Reserva	<ul style="list-style-type: none"> • O usuário requisita o cancelamento de reserva de servidor para homologação de <i>build</i>; • O sistema apresenta o dados referentes à reserva de servidor e solicita confirmação do cancelamento; • O usuário confirma o cancelamento; • O sistema encerra a reserva de servidor para testes de homologação de <i>build</i>.
Fluxo de Exceção:	<ul style="list-style-type: none"> • Não há.
Pós-condições	Um <i>build</i> foi aprovado ou reprovado e a reserva relativo ao <i>build</i> foi encerrada ou a reserva de servidor foi renovada para continuação dos testes ou a reserva de servidor foi cancelada

Quadro 3 – Caso de uso “Atualizar situação de *build*”

Caso de Uso	Manter Fábricas de Software
Sumário	Manter o cadastro(inclusão, remoção, alteração e consulta) das fábricas de software que prestam serviços para Bunge Alimentos.
Ator	Administrador do sistema
Pré Condições	Não há.
Fluxo Principal	<ol style="list-style-type: none"> 1. O usuário requisita a manutenção do cadastro de fábrica de software; 2. O sistema apresenta um formulário em branco com os dados pertinentes (código, nome, analista responsável) e uma opção de consulta; 3. O usuário informa o código da fábrica de software; 4. O sistema busca as informações referente à fábrica de software informada, habilita os campos para edição do registro e os

	<p>botões de alterar e excluir;</p> <p>5. O usuário efetua alterações no cadastro;</p> <p>6. O sistema verifica a validade dos dados e confirma a operação.</p>
Fluxo Alternativo(2): Consulta	<ul style="list-style-type: none"> • O usuário requisita consulta sobre lista de fábricas de software; • O sistema apresenta uma lista de todas as fábricas de software, permitindo que o usuário selecione a fábrica desejada; • O usuário seleciona a fábrica de software; • O caso de uso continua a partir do passo 4.
Fluxo Alternativo(4): Inclusão	<ul style="list-style-type: none"> • A fábrica de software informada no passo 4 não existe no cadastro e o sistema assume o modo de inclusão; • O usuário informa os detalhes da fábrica de software (código, nome, analista responsável); • O sistema verifica a validade dos dados e confirma a operação.
Fluxo Alternativo(4): Exclusão	<ul style="list-style-type: none"> • O usuário requisita a exclusão do registro; • O sistema verifica a exclusão e confirma a operação.
Fluxo de Exceção	<ul style="list-style-type: none"> • Caso os dados sejam inválidos o sistema reporta o erro, solicita novos dados e repete a operação;
Pós-condições	Uma fábrica de software foi atualizada ou inserida ou removida do sistema.

Quadro 3 – Caso de uso “Manter fábrica de software”

Caso de Uso	Manter Analistas Bunge
Sumário	Manter o cadastro (inclusão, remoção, alteração e consulta) de analistas de sistemas envolvidos nos projetos do sistema silos.
Ator	Administrador do sistema
Pré Condições	Não há.
Fluxo Principal: Atualização	<ol style="list-style-type: none"> 1. O usuário requisita a manutenção do cadastro de analistas Bunge; 2. O sistema apresenta um formulário em branco com os dados pertinentes (código e nome) e uma opção de consulta; 3. O usuário informa o código do analista Bunge; 4. O sistema busca as informações referente ao analista Bunge informado, habilita os campos para edição do registro e os botões de alterar e excluir; 5. O usuário efetua alterações no cadastro; 6. O sistema verifica a validade dos dados e atualiza o cadastro.
Fluxo Alternativo(2): Consulta	<ul style="list-style-type: none"> • O usuário requisita consulta sobre lista de analistas Bunge; • O sistema apresenta uma lista de todos os analistas Bunge, permitindo que o usuário selecione o analista desejado; • O usuário seleciona analista Bunge; • O caso de uso continua a partir do passo 4.
Fluxo Alternativo(4): Inclusão	<ul style="list-style-type: none"> • O código de analista Bunge informado no passo 4 não existe no cadastro e o sistema assume o modo de inclusão; • O usuário informa os detalhes do analista Bunge (código e nome); • O sistema verifica a validade dos dados e confirma a inclusão

	do registro.
Fluxo Alternativo(4): Exclusão	<ul style="list-style-type: none"> • O usuário requisita a exclusão do registro; • O sistema verifica a exclusão do registro e confirma a operação.
Fluxo de exceção	<ul style="list-style-type: none"> • Caso os dados sejam inválidos o sistema reporta o erro, solicita novos dados e repete a operação;
Pós-condições	Um analista Bunge foi atualizado ou inserido ou removido do sistema.

Quadro 4 – Caso de uso “Manter analistas Bunge”

Caso de Uso	Manter Servidores de Testes Silos
Sumário	Manter o cadastro (inclusão, remoção, alteração e consulta) de servidores para testes de novas versões do sistema silos.
Ator	Administrador do sistema
Pré Condições	Não há.
Fluxo Principal: Atualização	<ol style="list-style-type: none"> 1. O usuário requisita a manutenção do cadastro de servidor de teste; 2. O sistema apresenta um formulário em branco com os dados pertinentes (código e nome) e uma opção de consulta; 3. O usuário informa o código do servidor de teste; 4. O sistema busca as informações referente ao código de servidor de teste informado, habilita os campos para edição do registro e os botões de alterar e excluir; 5. O usuário efetua alterações no cadastro; 6. O sistema verifica a validade dos dados e confirma a atualização do registro.
Fluxo Alternativo(2): Consulta	<ul style="list-style-type: none"> • O usuário requisita consulta sobre lista de servidor de teste; • O sistema apresenta uma lista de todos os servidores de teste, permitindo que o usuário selecione o servidor desejado; • O usuário seleciona o servidor de teste; • O caso de uso continua a partir do passo 4.
Fluxo Alternativo(4): Inclusão	<ul style="list-style-type: none"> • O código de servidor de teste informado no passo 4 não existe no cadastro e o sistema assume o modo de inclusão; • O usuário informa os detalhes do servidor de teste (código e nome); • O sistema verifica a validade dos dados e o sistema confirma a inclusão do registro.
Fluxo Alternativo(4): Exclusão	<ul style="list-style-type: none"> • O usuário requisita a exclusão do registro; • Se o servidor de teste pode ser excluído, o sistema realiza a exclusão; caso contrário, o sistema reporta o fato.
Fluxo de Exceção	<ul style="list-style-type: none"> • Caso os dados sejam inválidos o sistema reporta o erro, solicita novos dados e repete a operação;
Pós-condições	Um servidor de teste foi atualizado ou inserido ou removido do sistema.

Quadro 5 – Caso de uso “Manter servidores de testes silos”

3.5.3 Modelo de Dados

Segundo Petrovic(2001), um Modelo de Dados é descrição do banco de dados, onde, a partir da especificação de requisitos, são representadas quais as entidades (descritas por seus atributos) serão armazenadas no banco de dados e os relacionamentos existentes entre elas. Representa um conjunto de requerimentos de informações de negócio.

A abordagem que se dispensa ao assunto normalmente atende a duas perspectivas: Modelagem Lógica e Modelagem Física. A primeira é usada como representação de alto nível e considera o ponto de vista do usuário criador do dado. Neste nível é definido como as entidades serão armazenadas na estrutura do Banco de Dados. Na modelagem física são definidos detalhes de implementação dos dados, descrevendo a estrutura de armazenamento e os métodos utilizados para acessar os dados efetivamente. Estes fatores estão, diretamente, relacionados ao um Sistema gerenciador de Banco de Dados – SGBD (PETROVIC, 2001). A figura 11 ilustra o modelo de dados físico do sistema criado na ferramenta Enterprise Manager que acompanha a *suite* banco de dados Microsoft SQL Server.

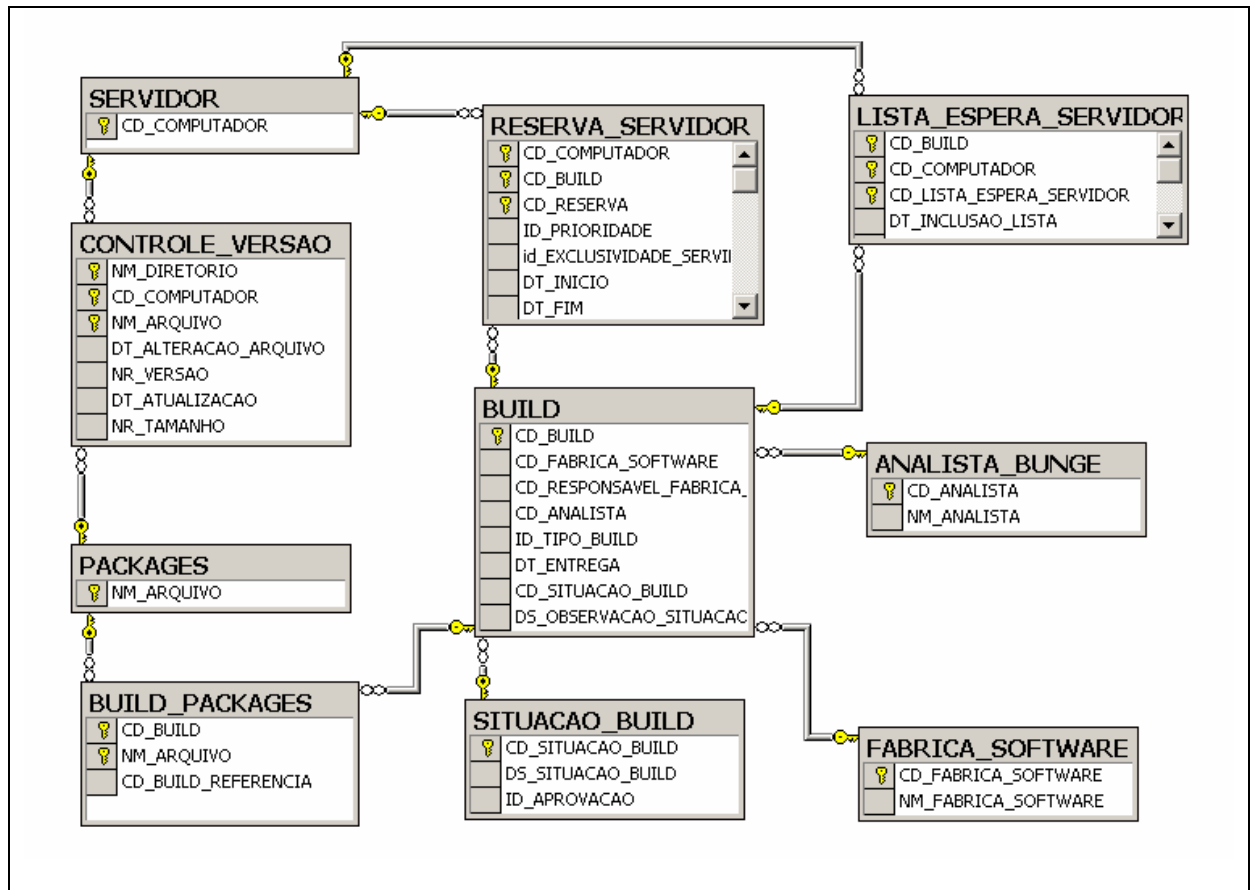


Figura 11 – Modelo Físico de dados

As tabelas build, build_package, fabrica_software, analista_bunge, reserva_servidor e servidor constituem o aplicativo Gerenciador de Homologações. As tabelas controle-versao e packages pertence ao sistema Controle de Versão da Bunge Alimentos e é apenas consultada pelo sistema desenvolvido.

3.6 IMPLEMENTAÇÃO

Os assuntos seguintes descrevem as ferramentas e técnicas utilizadas para o desenvolvimento do trabalho e a operacionalidade do sistema.

3.6.1 Técnicas e ferramentas utilizadas

Para a fase de desenvolvimento e implementação do sistema utilizaram-se ferramentas e técnicas que auxiliaram no decorrer do trabalho, mantendo as informações organizadas e consistentes, e agilizando o processo de desenvolvimento. A seguir são descritas as funcionalidades utilizadas de cada ferramenta e técnicas de desenvolvimento.

3.6.1.1 Genexus

Segundo Artech (2003), Genexus é uma ferramenta de desenvolvimento acelerado (RAD), multi-plataforma que permite a geração e manutenção automáticas de aplicações de missão crítica. Genexus incorpora uma tecnologia única no mundo que permite ao desenvolvedor capturar os requerimentos dos usuários de forma independente da plataforma de execução, e daí gerar o 100% da aplicação do cliente. Com isso, a ferramenta permite o desenvolvimento de aplicativos, seu gerenciamento e manutenção, com aumentos de produtividade reportados pelos clientes de até 500%.

Genexus trabalha com um novo paradigma – focar na visão de cada usuário do futuro sistema e, a partir daí, gerar tanto a base de dados como os programas de aplicação. Foi a investigação original da Artech a que permitiu descobrir e usar no Genexus o fato que dado um conjunto de visões de dados, existe uma só base relacional mínima capaz de suportar todas as necessidades (ARTECH, 2003).

Os principais benefícios para o cliente, de acordo com Artech (2003) são:

- a) reduzir o tempo ao mercado (*time-to-market*) de seus aplicativos;
- b) reduzir o custo de desenvolvimento, manutenção e suporte de seus aplicativos;
- c) focalizar-se no seu negócio, não na tecnologia;

- d) desenvolver os sistemas em forma incremental, sem investir hoje em eventuais requisitos do futuro;
- e) ter-se a liberdade de escolha de sistema operacional e banco de dados.

Genexus gera código de suas aplicações em diversas linguagens de programação como Cobol, Visual Basic, Java, C e C# e vários bancos de dados, entre eles Oracle, Sql Server e MySql. A interface com o usuário é feita em formulários *Windows (Win Forms)* ou páginas Html Dinâmicas (*Web Forms*).

Em ambiente *web*, Genexus é capaz ainda de gerar aplicações com grande interatividade, com utilização de Javascript e Ajax. Ajax é a sigla para "*Asynchronous Javascript and XML*". É uma técnica onde utiliza-se Javascript e XML para transformar suas aplicações, de modo que não precise recarregar a tela cada vez que o usuário submeter um evento.

O sistema desenvolvido, utilizou Genexus para geração de código Java na arquitetura J2EE com banco de dados MS SQL Server e interface *Web*.

3.6.1.2 Arquitetura J2EE

Bond (2003) descreve a especificação J2EE como um padrão dinâmico para a produção de aplicativos corporativos seguros, escaláveis e altamente disponíveis. A arquitetura J2EE é um conjunto de especificações e práticas que possibilitam soluções para o desenvolvimento, instalação e gerenciamento de aplicações multi-camadas baseadas em servidores.

Conforme Sun Microsystem (2002), a arquitetura J2EE é composta dos seguintes elementos:

- a) modelo de aplicações J2EE: modelo padrão de programação para o desenvolvimento de aplicações multi-camadas;
- b) especificada como um conjunto de API's e políticas requeridas. Pode ser implementada por diversos fornecedores;
- c) bateria de testes de compatibilidade J2EE: bateria de testes de compatibilidade que verificam se uma determinada plataforma J2EE é compatível com a especificação da plataforma J2EE;
- d) implementação de referência J2EE: uma implementação de referência que demonstra as capacidades da arquitetura J2EE e que provê uma definição operacional da plataforma J2EE.

3.6.2 Operacionalidade da implementação

Para demonstrar o funcionamento do aplicativo desenvolvido, são apresentadas a seguir algumas funções passo a passo da utilização do sistema simulando um caso real de utilização.

Ao iniciar a aplicação uma tela inicial com menu é apresentada. De acordo com os requisitos definidos com os usuários, solicitou-se não implantar qualquer controle de acessos no sistema. O motivo é que qualquer sistema no ambiente da Bunge Alimentos o controle de acesso deve ser obrigatoriamente centralizado no sistema corporativo. Para isso seria necessário envolver outros recursos da empresa e, portanto, optou-se por não implantá-lo. Conforme figura 12, o usuário pode escolher opções à partir de dois módulos. O primeiro se refere novos *builds*, ou seja *builds* que precisam passar por um processo de homologação

antes de ser implantado em produção. As opções para este módulo são:

- a) disponibilizar *build* para homologação ;
- b) reservar servidor;
- c) reservas de Servidor - Avaliação de *build*.
- d) histórico de *builds* – Situação

Para o módulo de cadastro as opções são:

- a) fábrica de software;
- b) servidores de testes;
- c) analistas homologadores.



Figura 12 – Tela inicial do sistema

Ao clicar no link “Disponibilizar *build* para homologação” o sistema abre uma tela para manter *Build*, ilustrada na figura 13. É nesta tela que os analistas de sistemas das fábricas de software da Bunge deverão cadastrar novos *builds* que deverão ser homologados por

analistas da Bunge. Eventualmente outros usuários podem utilizar esta tela para fazer consultas por situação de *build*, por exemplo.

The screenshot shows the BUNGE system interface. At the top left is a vertical image of wheat. The BUNGE logo is at the top center. Below the logo are navigation buttons: |<, <, >, >|, and Selecionar. A red message states: "Dados com a chave especificada não foram encontrados." Below this is a form with the following fields:

- BUILD: 01_40_01_06
- Fabrica Software: Tecnologica
- Responsavel: Pedro Pacheco
- Analista: Andre Spengler
- Tipo Build: Solicitacao de Servico
- DT_ENTREGA: 07/06/06 08:50

Below the form is a table with two columns: Package Liberada and PRE_REQUISITO.

Package Liberada	PRE_REQUISITO
sassCa	
SassCadastrrosClient.jar	
SassCadastrrosEJB.jar	
SassCadastrrosServer.jar	

At the bottom of the interface are buttons: Aplicar Mud, Verificar, Fechar, Apagar Toda, and Ajuda.

Figura 13 – Manter cadastro de *build*

O link “Reservar Servidor” leva o usuário a tela que pode ser considerada como a principal funcionalidade do sistema. É nessa tela que o analista de sistemas poderá selecionar um *build* pendente de homologação e listar os servidores disponíveis de acordo com os pré-requisitos de cada pacote do *build* selecionado. Caso haja servidor disponível e que atenda os pré-requisitos do *build* verificado pelo sistema, o analista poderá clicar no link para efetuar a reserva de servidor. Caso haja servidor disponível mas o servidor não possui a versão do pacote correta, o sistema informa o número da versão desatualizada para o analista saber o motivo da impossibilidade de efetuar a reserva. Se o servidor já estiver ocupado, o analista poderá incluir o *build* em fila de espera de servidor e assim garantir preferência no momento do encerramento da reserva. A figura 14 ilustra todas as situações mencionadas.

BUNGE

Disponibilidade de Servidores baseados nos pre-requisitos do build 01_85_51_03

Exclusividade no Servidor?

Selecione o Servidor Disponível abaixo e clique na Opção desejada:

Computador	Status	Build Pkg	Lista de Espera
C1090	DISPONIVEL	01_62_51_01	
C2014	RESERVA P/ BUILD 01_85_51_02	01_85_51_01	ADICIONAR
C2016	DISPONIVEL	01_14_003	
C2017	DISPONIVEL	Reservar	

Figura 14 – Reserva de Servidor

A opção “Reservas de Servidor – Avaliação de *build*” abre uma tela na qual serão listados todos os *builds* pendentes de avaliação do analista que possuem uma reserva de servidor, conforme mostra a figura 14. O usuário poderá aprovar ou reprovar o *build* caso tenha finalizado os testes ou renovar a reserva de servidor se não houver *build* em fila de espera para aquele servidor. Há também a opção de cancelamento de reserva, caso o analista desista de efetuar o teste do *build* naquele dia.

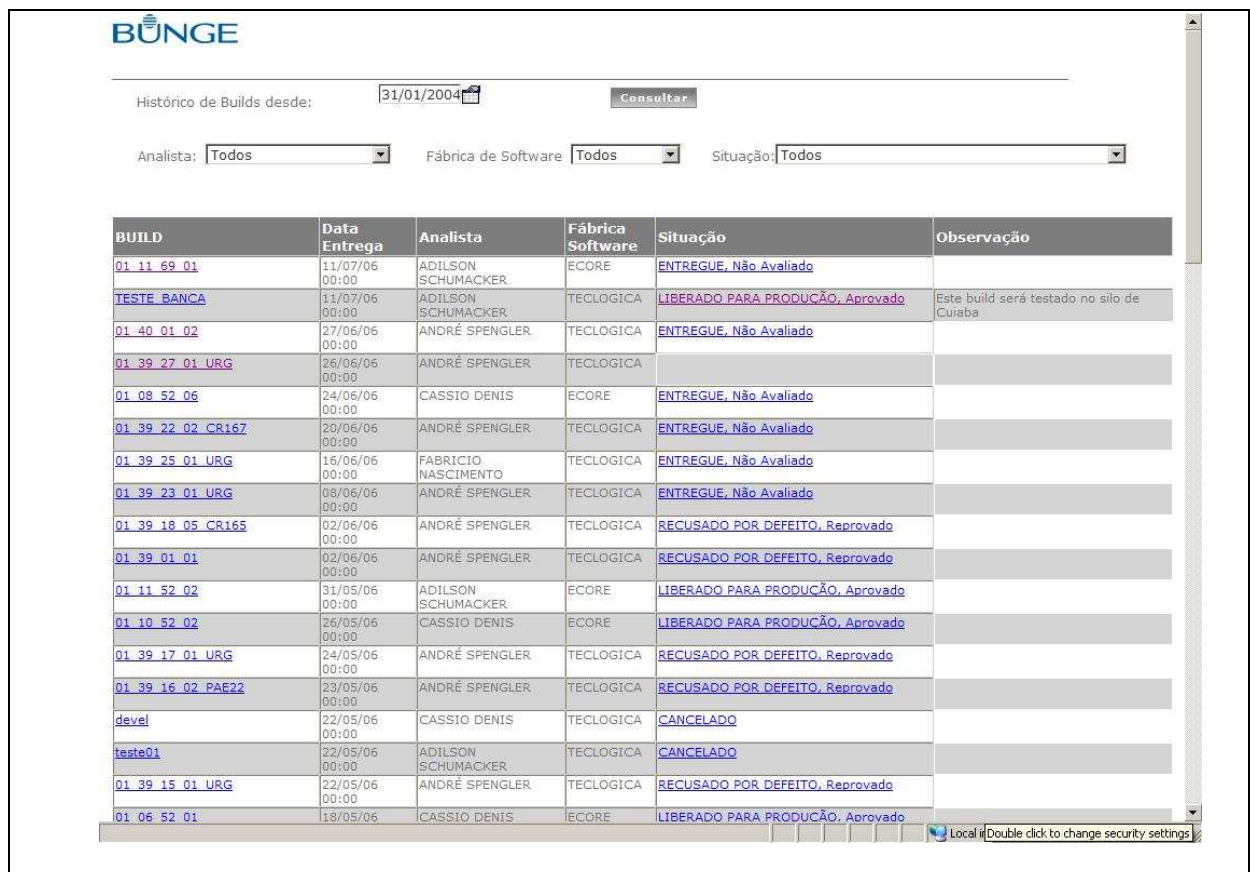


Lista de Builds pendentes de homologação com reservas de servidor efetuadas

BUILD	Computador	Data Inicio	Data Fim	Nome Analista	Aprovar	Reprovar	Prolongar Reserva	Cancelar Reserva
01_40_01_05	C2014	06/06/06	07/06/06	Andre Spengler	Aprovar	Reprovar	Renovar Reserva	Cancelar
01_85_51_02	C2014	07/06/06	08/06/06	Andre Spengler	Aprovar	Reprovar	Renovar Reserva	Cancelar
01_37_01_01	C2017	24/05/06	25/05/06	Andre Spengler	Aprovar	Reprovar	Renovar Reserva	Cancelar

Figura 15 – Lista de builds pendentes de avaliação e com reservas de servidor

A opção “Histórico de Builds – Situação” disponibiliza ao usuário a consulta de acompanhamento de homologação de builds. Existe a possibilidade de filtrar a busca de *builds* por data, analista responsável, fábrica de software e situação do *build*.



Histórico de Builds desde: 31/01/2004 [Consultar](#)

Analista: Todos Fábrica de Software: Todos Situação: Todos

BUILD	Data Entrega	Analista	Fábrica Software	Situação	Observação
01_11_69_01	11/07/06 00:00	ADILSON SCHUMACKER	ECORE	ENTREGUE, Não Avaliado	
TESTE BANCA	11/07/06 00:00	ADILSON SCHUMACKER	TECLOGICA	LIBERADO PARA PRODUÇÃO, Aprovado	Este build será testado no silo de Curitiba
01_40_01_02	27/06/06 00:00	ANDRÉ SPENGLER	TECLOGICA	ENTREGUE, Não Avaliado	
01_39_27_01_URG	26/06/06 00:00	ANDRÉ SPENGLER	TECLOGICA		
01_08_52_06	24/06/06 00:00	CASSIO DENIS	ECORE	ENTREGUE, Não Avaliado	
01_39_22_02_CR167	20/06/06 00:00	ANDRÉ SPENGLER	TECLOGICA	ENTREGUE, Não Avaliado	
01_39_25_01_URG	16/06/06 00:00	FABRICIO NASCIMENTO	TECLOGICA	ENTREGUE, Não Avaliado	
01_39_23_01_URG	08/06/06 00:00	ANDRÉ SPENGLER	TECLOGICA	ENTREGUE, Não Avaliado	
01_39_18_05_CR165	02/06/06 00:00	ANDRÉ SPENGLER	TECLOGICA	RECUSADO POR DEFEITO, Reprovado	
01_39_01_01	02/06/06 00:00	ANDRÉ SPENGLER	TECLOGICA	RECUSADO POR DEFEITO, Reprovado	
01_11_52_02	31/05/06 00:00	ADILSON SCHUMACKER	ECORE	LIBERADO PARA PRODUÇÃO, Aprovado	
01_10_52_02	26/05/06 00:00	CASSIO DENIS	ECORE	LIBERADO PARA PRODUÇÃO, Aprovado	
01_39_17_01_URG	24/05/06 00:00	ANDRÉ SPENGLER	TECLOGICA	RECUSADO POR DEFEITO, Reprovado	
01_39_16_02_PAE22	23/05/06 00:00	ANDRÉ SPENGLER	TECLOGICA	RECUSADO POR DEFEITO, Reprovado	
devel	22/05/06 00:00	CASSIO DENIS	TECLOGICA	CANCELADO	
teste01	22/05/06 00:00	ADILSON SCHUMACKER	TECLOGICA	CANCELADO	
01_39_15_01_URG	22/05/06 00:00	ANDRÉ SPENGLER	TECLOGICA	RECUSADO POR DEFEITO, Reprovado	
01_06_52_01	18/05/06	CASSIO DENIS	ECORE	LIBERADO PARA PRODUÇÃO, Aprovado	

Figura 16 – Lista de *builds* pendentes de avaliação e com reservas de servidor

Os links de “Fábrica de software”, “Servidores de testes” e “Analistas homologadores” são utilizados para manutenção de cadastros conforme especificados na descrição dos casos de uso UC04.1, UC04.2 e UC04.3. Devido essas telas não apresentarem nenhuma funcionalidade vital do sistema as mesmas não foram relacionadas.

3.7 RESULTADOS E DISCUSSÃO

A etapa de testes e validação deste trabalho foi efetuada em conjunto com principais usuários do sistema. Foram avaliados aspectos de desempenho, navegabilidade além dos requisitos funcionais e não funcionais definidos no projeto. Para os requisitos não funcionais foi efetuado o *deploy* do sistema no servidor de aplicações Jboss e posterior acesso ao sistema. Neste momento, foi verificado que a ferramenta Genexus cumpre o propósito de geração de código Java para plataforma J2EE, acessando banco de dados SQL Server e empacotamento da aplicação compatível com o servidor de aplicações JBoss. Não houve necessidade de codificação de programas externos pois o Genexus foi capaz de gerar a aplicação completa. Ainda referente aos requisitos não funcionais, não houve problemas para leitura da tabela “Controle_Versao” proveniente do sistema Controle de Versão. Os aspectos de desempenho e navegabilidade foram também aprovados pelos usuários.

Para os requisitos não-funcionais foram inicialmente feitos os cadastros básicos seguidos de cadastros de *builds* homologados na semana anterior e simulação de reservas de servidores para esses *builds*. Com esse testes, foi possível validar os requisitos funcionais e não-funcionais propostos no trabalho com sucesso, não sendo necessário ajustes de maior impacto na aplicação. Em seguida a aplicação foi disponibilizada em produção para os utilização dos usuários que mostraram-se satisfeitos, uma vez que a aplicação atingiu seus objetivos planejados.

O Quadro 6 relaciona os trabalhos correlatos mencionados na fundamentação teórica com a aplicação desenvolvida, evidenciando aspectos e características em cada uma delas.

Características	Este Projeto	Heusi (2005)	Schumacker (2005)
aplicado à necessidade específica da Bunge Alimentos	X	X	X
Ferramenta de apoio a homologação de versões do sistema silos	X		
verificar versões em servidores	X	X	
Verificação de produtividade			X
interface web	X	X	
Implementação com Java para J2EE	X	X	

Quadro 6 – Características de cada trabalho

4 CONCLUSÕES

À partir do entendimento do fluxo atual de homologação foi possível o aperfeiçoamento deste fluxo através do sistema de Gerenciador de Homologação de Versões.

Os resultados obtidos nos testes com os usuários da ferramenta desenvolvida neste trabalho foram satisfatórios, dado o incremento de produtividade no fluxo de homologação de *builds* reportado pelos próprios usuários.

Além da agilidade que o aplicativo proporcionou ao automatizar o processo de verificação de pré-requisitos de pacotes do *build*, o aplicativo trouxe outros benefícios importantes como a maximização da utilização dos servidores, uma vez que no fluxo anterior era evitado implantar vários *builds* por servidor.

Outro benefício constatado, foi que os analistas de sistemas da Bunge Alimentos passaram a realizar os testes de homologação mais rapidamente. Com a aplicação do conceito de reservas de servidor, os analistas passaram a se programar para realizar os testes no período da reserva de servidor.

A ferramenta Genexus atendeu ao propósito deste trabalho de gerar totalmente o código Java para sistemas *web* na plataforma J2EE e empacotamento da aplicação compatível com o servidor de aplicação Jboss.

Este trabalho contribuiu ainda para aprimorar o domínio da ferramenta Genexus adquirido em função da necessidade de pesquisa e esforço no decorrer do desenvolvimento da aplicação.

4.1 EXTENSÕES

Nesta seção são apresentadas sugestões de extensões e modificações para este trabalho, que estão descritas a seguir:

- a) incluir um workflow de aprovação via e-mail em todo o fluxo de homologação do *build*;
- b) adicionar relatórios com indicadores referentes à estatísticas de aprovação, reprovação tempo de homologação, etc;
- d) adicionar a funcionalidade de liberação de *build* no momento da reserva de servidor;
- e) integrar a aplicação Gerenciador de Homologação de *Builds* com o sistema Gerenciador de Fontes da Bunge Alimentos.

REFERÊNCIAS BIBLIOGRÁFICAS

- ARTECH DO BRASIL. **Genexus**. São Paulo, 2002. Disponível em: <<http://www.genexus.com/br/portal/hgxpp001.aspx?10,2,26,O,P,0,MNU;E;7;1;MNU;,>>. Acesso em: 19 mai 2006.
- BEZZERA, Eduardo. **Princípios de análise e projetos de sistemas com UML**. Rio de Janeiro: Campus, 2002.
- BOND, Martin. **Aprenda J2EE: com EJB, JSP, Servlets, JNDI, JDBC e XML**. São Paulo: Pearson Education, 2003., 962p.
- BUNGE ALIMENTOS S/A. **Quem somos**. Gaspar, 2003. Disponível em: <<http://www.bunge.com.br/quemsomos/historico.asp>>. Acesso em: 31 mar 2006.
- HEUSI, Silvio Alves de Miranda. **Controle de versão do sistema silos da Bunge Alimentos**. 2005. 59f. Trabalho de Conclusão de curso (Graduação em Sistemas de Informação) – Fundação Universidade Regional de Blumenau, Blumenau.
- MARTINS, Vitor Hugo. **Estudos complementares**. Osasco, 2006. Disponível em: <<http://vhmartins.blogspot.com/>>. Acesso em: 13 abr 2006.
- OLIVEIRA, A. A. A. C. P. **Gerência de configuração de software: evolução de software sob controle**. Instituto Nacional de Tecnologia e Informação. Disponível em <<http://www.iem.efei.br>> Acessado em 15 nov 2005.
- PETKOVIC, Dusan. **SQL Server 2000 – Guia Prático**. São Paulo: Makron Books, 2001.
- PRESSMAN, Roger S. **Engenharia de software**. Rio de Janeiro: Makron Books, 1995.
- SCHUMACKER, ADILSON VALDECIR. **Sistemas de informação aplicado a gestão de outsourcing na TI da Bunge Alimentos**. 2005. 50f. Trabalho de Conclusão de curso (Graduação em Sistemas de Informação) – Fundação Universidade Regional de Blumenau, Blumenau.
- SILVA, C. E. S.; ALMEIDA, D. P; BARRETO, D.; TURRIONI, J. B. **Uso da gestão de configuração de software pelas organizações em busca de certificação**. São Paulo, 2005. Disponível em <<http://www.iem.efei.br>> Acesso em 14 mar. 2006.
- SUN MICROSYSTEM. **The J2EE Tutorial**. San Francisco, 2002. Disponível em: <<http://java.sun.com/j2ee/tutorial/>>. Acesso em: 23 mar 2006.
- VILAS BOAS, André Luis de Castro. **Gestão de Configuração para Teste de Software**. 2003. 147f. Dissertação (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2003.

APÊNDICE A – Trecho de Código Gerado pela ferramenta Genexus

```

/*
    File: hreservarservidor_impl
    Description: Reservar Servidor
    Author: GeneXus Java Generator version 9_0_0-136
    Generated on: July 31, 2006 21:29:1.0
    Program type: Callable routine
    Main DBMS: sqlserver
*/
import com.genexus.*;
import com.genexus.db.*;
import com.genexus.distributed.*;
import com.genexus.webpanels.*;
import java.sql.*;

public final class hreservarservidor_impl extends GXWebPanel
{
    public hreservarservidor_impl( com.genexus.internet.HttpContext context )
    {
        super(context);
    }

    public hreservarservidor_impl( int remoteHandle )
    {
        super( remoteHandle , new ModelContext( hreservarservidor_impl.class ));
    }

    public hreservarservidor_impl( int remoteHandle ,
                                   ModelContext context )
    {
        super( remoteHandle , context);
    }

    protected void createObjects( )
    {
        dynavCd_buildselecao = new HTMLChoice();
        chkavExclusividade = UIFactory.getCheckbox(this);
    }

    public void webExecute( )
    {
        pa0B2( );
        if ( ( gxajaxcallmode == 0 ) )
        {
        }
        if ( ( gxajaxcallmode == 0 ) && ( GxWebError == 0 ) )
        {

```

```

        ws0B2();
        if ( ( gxajaxcallmode == 0 ) )
        {
            we0B2();
        }
    }
cleanup();
}

public void renderHtmlHeaders()
{
    wbTemp = httpContext.setContentType( "text/html" );
    wbTemp = httpContext.setHeader( "pragma", "no-cache" );
    httpContext.setStream();
    httpContext.writeTextNL( "<html>" );
    httpContext.writeTextNL( "<head>" );
    idxLst = 1 ;
    while ( ( idxLst <= radFormheadermetaname.getItemCount() ) )
    {
        httpContext.writeText( "<meta name=\"" +GXutil.rtrim(
radFormheadermetaname.getItemValue((short)(idxLst))+"\" content=\"" );
        httpContext.writeValue( GXutil.rtrim(
radFormheadermetaname.getItemText((short)(idxLst)))) );
        httpContext.writeTextNL( "\"/>" );
        idxLst = (int)(idxLst+1);
    }
    httpContext.writeTextNL( "<meta http-equiv=\"Content-Type\" content=\"text/html;
charset=UTF-8\"/>" );
    idxLst = 1 ;
    while ( ( idxLst <= radFormheadermetahttpequiv.getItemCount() ) )
    {
        httpContext.writeText( "<meta http-equiv=\"" +GXutil.rtrim(
radFormheadermetahttpequiv.getItemValue((short)(idxLst))+"\" content=\"" );
        httpContext.writeValue( GXutil.rtrim(
radFormheadermetahttpequiv.getItemText((short)(idxLst)))) );
        httpContext.writeTextNL( "\"/>" );
        idxLst = (int)(idxLst+1);
    }
}

public void renderHtmlOpenForm()
{
    httpContext.writeText( "<title>" );
    httpContext.writeText( "Reservar Servidor" );
    httpContext.writeTextNL( "</title>" );
    if ( ( GXutil.len( sDynURL ) > 0 ) )
    {
        httpContext.writeText( "<BASE href=\"" +sDynURL+"\" />" );
    }
    define_styles();
}

```

```

    httpContext.writeTextNL( "<script language=\"JavaScript\"
src=\""+httpContext.convertURL( "gxtypes.js")+\"></script>" );
    httpContext.writeTextNL( "<script language=\"JavaScript\"
src=\""+httpContext.convertURL( "gxballoon.js")+\"></script>" );
    httpContext.writeTextNL( "<script language=\"JavaScript\"
src=\""+httpContext.convertURL( "gxfwddcl.js")+\"></script>" );
    httpContext.writeTextNL( "<script language=\"JavaScript\"
src=\""+httpContext.convertURL( "gxfrmutil.js")+\"></script>" );
    httpContext.writeTextNL( "<script language=\"JavaScript\"
src=\""+httpContext.convertURL( "gxcallrpc.js")+\"></script>" );
    httpContext.writeTextNL( "<script language=\"JavaScript\"
src=\""+httpContext.convertURL( "gxautosuggest.js")+\"></script>" );
    httpContext.writeText( "" );
    httpContext.writeTextNL( "</head>" );
    FormProcess = (true ? " onload=\"window.document.forms[0].reset();"+\""+
onkeyup=\"form_onkeyup(event)\\" onkeydown=\"form_onkeypress(event,false);\" : "" );
    httpContext.writeText( "<body>" );
    if ( ! ( ((GXutil.strcmp("", GXutil.rtrim( ""))==0) ) ) )
    {
        httpContext.writeText( " background=\""+httpContext.convertURL( "")+\"" );
    }
    httpContext.writeText( " "+title=\"Garante a exclusividade no servidor\""+
"+class=\"Form\""+\" "+ bgcolor=\""+WebUtils.getHTMLColor(
(int)(0xFFFFFFFF))+\""+FormProcess+">" );
    httpContext.skipLines( 1 );
    httpContext.writeTextNL( "<form id=\"MAINFORM\" onsubmit=\"try{return
GXValidForm()}catch(e){return true;}\" name=\"MAINFORM\" method=\"POST\"
ACTION=\""+formatLink("hreservarservidor") +\""+\">" );
    WebStandardFunctions.gx_hidden_field( httpContext, "_EventName", "" );
    WebStandardFunctions.gx_hidden_field( httpContext, "_EventGridId", "" );
    WebStandardFunctions.gx_hidden_field( httpContext, "_EventRowId", "" );
}

public void renderHtmlCloseForm0B2()
{
    /* Send hidden variables. */
    /* Send saved values. */
    WebStandardFunctions.gx_hidden_field( httpContext, "nRC_Grid1", GXutil.ltrim(
localUtil.ntoc( nRC_Grid1, (byte)(4), (byte)(0), ",", ""));
    WebStandardFunctions.gx_hidden_field( httpContext, "sCallerURL", GXutil.rtrim(
httpContext.sCallerURL));
    httpContext.writeTextNL( "</form>" );
    include_jscripts( );
    httpContext.writeTextNL( "</body>" );
    httpContext.writeTextNL( "</html>" );
}

public void wb0B0()
{
    if ( ( wbLoad == false ) )

```

```

    {
        renderHtmlHeaders( );
        renderHtmlOpenForm( );
        WebStandardFunctions.gx_msg_list( httpContext, "",
httpContext.GX_msglist.getDisplaymode(), "", "");
        httpContext.writeText( "<P>" );
        wb_table1_3_0B2( true );
    }
    else
    {
        wb_table1_3_0B2( false );
    }
    return ;
}

public void wb_table1_3_0B2e( boolean wbgen )
{
    if ( ( wbgen == true ) )
    {
        httpContext.writeTextNL( "</P>" );
        httpContext.writeTextNL( "<P>&nbsp;</P>" );
        httpContext.writeText( "<P>&nbsp;</P>" );
    }
    wbLoad = true ;
}

public void start0B2( )
{
    wbLoad = false ;
    wbEnd = 0 ;
    wbTemp = (byte)(0) ;
    radFormheadermetaname.addItem("Generator", "GeneXus Java", (byte)(0));
    radFormheadermetaname.addItem("Version", "9_0_0-136", (byte)(0));
    radFormheadermetaname.addItem("Description", "Reservar Servidor", (byte)(0));
    httpContext.wjLoc = "" ;
    httpContext.nUserReturn = (byte)(0) ;
    httpContext.wbHandled = (byte)(0) ;
}

```