UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

GERENCIADOR DE AMBIENTES PARA TESTES MANUAIS

ALEXANDRE GIELOW

BLUMENAU
2014

2014/2-01

ALEXANDRE GIELOW

GERENCIADOR DE AMBIENTE PARA TESTES MANUAIS

Trabalho de Conclusão de Curso submetida à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso I do curso de Sistemas de Informação — Bacharelado.

Prof. Alexander Roberto Valdameri – Orientador

BLUMENAU
2014

2014/2-01

GERENCIADOR DE AMBIENTES PARA TESTES MANUAIS

Por

ALEXANDRE GIELOW

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. Marcel Hugo, Mestre – FURB
Membro:	Prof. Roberto Heinzle, Doutor – FURB

Blumenau, 9 de dezembro de 2014.

Dedico este trabalho á empresa de software, que foi essencial para motivação deste trabalho; aos meus amigos, que me ajudaram na elaboração; e principalmente minha família, que desde meus primeiros passos nesta universidade, me apoiou da melhor forma possível.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, meu pilar de conhecimento e exemplo.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador, professor Alexander Valdameri, por ter acreditado na conclusão deste trabalho.

Aos professores do Departamento de Sistemas e Computação da Universidade Regional de Blumenau por suas contribuições durante os semestres letivos.

Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível.

Charles Chaplin

RESUMO

Os testes manuais são de grande importância para a Qualidade de Software. A carência de materiais sobre o assunto e sistemas relacionados foram os motivadores para o desenvolvimento deste trabalho. O objetivo principal é melhorar o processo de manutenção de ambientes de testes manuais, através da criação de um software capaz de fazer o gerenciamento destes, automatizando vários dos processos que até então eram executados manualmente pelos profissionais da área de qualidade de uma empresa de software. Buscando maior performance no trabalho destes profissionais, busca-se garantir a redução de tempo em trabalhos manuais, aumentando então o tempo disponível para a principal tarefa do testador: planejar e testar. Tal objetivo foi alcançado através da confecção de uma ferramenta que vai fazer a gestão dos ambientes de testes, o banco de dados, o aplicativo e os serviços relacionados, tornando a gestão prática e compartilhável. A ferramenta foi construída na linguagem JAVA. A ferramenta funciona em ambiente de rede, permitindo compartilhamento dos ambientes de testes. Ao final da pesquisa e aplicação do trabalho, notou-se uma melhora na gerência de ambientes de testes, reduzindo perdas de informações, agilidade no compartilhamento de bases e principalmente, garantindo a integridade dos testes.

Palavras-chave: Testes manuais. Qualidade de software. Ambientes de testes manuais.

ABSTRACT

The manual tests are of great importance for Software Quality. The lack of material on the subject and related systems are the motivators for the development of this work. The main goal is to improve the maintenance process of manual testing environments, through the creation of a software capable of managing these, by automating many of the processes that were previously performed manually by professionals in the field of quality of a software company. Seeking higher job performance of these professionals, we seek to ensure the reduction of time in manual work, thus increasing the time available for the primary task of the tester, and test plan. This aim was achieved through the production of a tool that will make the management of testing environments, encompassing an environment with a whole database, application, and related services, and sharable making practice management. The tool was built in JAVA. The tool works in a network environment, allowing sharing of test environments. At the end of the research and application of the work, we noticed an improvement in the management of test environments, reducing loss of information, agility in sharing bases and foremost, ensuring integrity of the tests.

Keywords: Manual Testing. Software Quality. Manual Testing Environments.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

ABNT - Associação Brasileira de Normas Técnicas

CMD – *Prompt* de comando

CMMI - *Capability Maturity Model Integration*

CVS - *Concurrent Version System*

GCS - Gestão de Configuração de Software

ISO - *International Standardization Organization*

MER – Modelo de Entidade e Relacionamento

EA – *Enterprise Architect*

TCC – Trabalho de Conclusão de Curso

UC – *Use Case*

SUMÁRIO

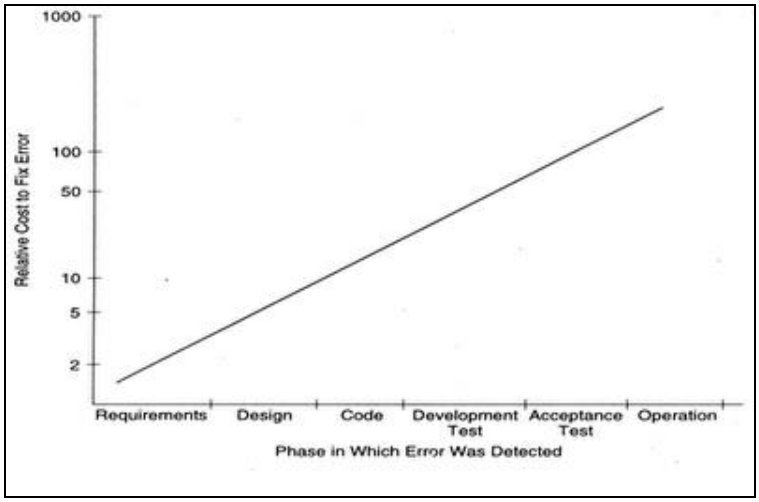
1 INTRODUÇÃO

A área de qualidade é um alicerce fundamental para que os produtos cheguem ao mercado sem defeitos. Para as empresas de desenvolvimento de software, a área de qualidade visa entregar ao cliente um sistema livre de *bugs* e dentro da expectativa. Para Bartié (2002), a qualidade de software é um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos.

Ambientes de testes bem definidos são fundamentais para garantir a qualidade de um software e consequentemente aumentar a competitividade. A redução dos custos de desenvolvimento e manutenção está diretamente relacionada a melhoria na qualidade dos produtos. Segundo Jones e Boehm (1986), 44% do esforço de um projeto são dedicados ao retrabalho, ou seja, refazer o trabalho já feito antes. Reduzir o retrabalho implica em realizar processos que identifiquem e corrijam defeitos o mais cedo possível de forma a evitar retrabalhar esses defeitos mais adiante no projeto, quando o custo será muito maior. Tais ações são de melhoria de qualidade. Segundo Rodrigues (2001), qualidade não é um fator de vantagem no mercado, mas é uma necessidade para a garantia da competitividade.

Conforme a Boehm e Basili (2001), quanto mais cedo erros forem identificados, menos custoso será para corrigi-los. A correção de erros durante a operação pode ser até 100 vezes mais custosa do que as correções feitas no desenvolvimento, conforme ilustra a Figura 1.

Figura 1 - Custos para correção de erros



Fonte: Anderson (2007).

Este trabalho apresenta a construção de uma ferramenta de automação do *backup* e restauração de ambientes de testes, aplicado ao processo de uma empresa desenvolvedora de software do sul do Brasil e de grande porte. Por atender grandes clientes, a empresa investe em qualidade, mas ainda possui demanda em melhorias no gerenciamento dos ambientes de testes. Portanto a gerência destes ambientes é fundamental para aumentar a qualidade de software, que por consequência, aumenta a satisfação dos clientes, que aumenta o valor agregado do sistema e finalmente, o lucro da empresa.

1.1 OBJETIVOS DO TRABALHO

O objetivo principal do trabalho é desenvolver um sistema para melhorar a disponibilidade dos ambientes de testes.

Os objetivos específicos do trabalho são:

a) manter ambientes de teste parametrizados para determinadas situações, como por exemplo, manter uma parametrização do sistema para geração de títulos de impostos na emissão de uma nota fiscal;
b) disponibilizar o ambiente de forma íntegra para outros usuários, sem depender de qualquer tipo de procedimento manual, como instalação, restauração de banco de dados, parametrização de sistema;
c) criar *backup* de versões do sistema para que seja possível "voltar no tempo" sem muito trabalho (criar outra instalação, encontrar a *build*, restaurar um *backup* no SQL ou no Oracle e restaurar o *middleware*);
d) facilitar o acesso ao *backup* de apenas banco de dados, ou aplicação, ou configurações de servidores para pessoas mais leigas na parte técnica;
e) facilitar os comandos de *middleware* (gerenciamento de domínio no Glassfish).

1.2 ESTRUTURA DO TRABALHO

No primeiro capítulo tem-se a introdução ao tema principal deste trabalho com a apresentação da justificativa e dos objetivos.

No segundo capítulo apresenta-se a fundamentação teórica pesquisada sobre Qualidade de Software, Gerência de Configuração, Ambientes de testes nas empresas, sistema atual e trabalhos correlatos.

O terceiro capítulo apresenta o desenvolvimento da ferramenta de gerência de ambientes para testes manuais, iniciando-se com o levantamento de informações, tendo na sequência, especificação com diagrama de casos de uso, diagrama de classes, modelo de entidade e relacionamento e diagrama de atividades, em seguida, implementação, resultados e discussão.

No quarto capítulo tem-se as conclusões deste trabalho bem como apresentam-se sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma visão sobre qualidade de software, gerência de configurações, ambientes de testes nas empresas e também sobre o sistema atual, além dos trabalhos correlatos.

2.1 QUALIDADE DE SOFTWARE

Qualidade de software provém do termo qualidade, que desde os primórdios da humanidade ficou em evidência. Segundo Juran e Gryna(1998), existem relatos históricos segundo os quais há mais de quatro mil anos os egípcios estabeleceram um padrão de medida de comprimento: o cúbito. Essa medida correspondia ao comprimento do braço do faraó reinante. Curiosamente, a troca de faraó significava que a medida deveria ser atualizada. Todas as construções deviam ser realizadas utilizando o cúbito como unidade de medida. Para isso eram empregados bastões cortados no comprimento certo e periodicamente – a cada lua cheia – o responsável por uma construção devia comparar o padrão que estava sendo utilizado com o padrão real. Se isso não fosse feito e houvesse um erro de medição, o responsável poderia ser punido com a morte.

Para que um software tenha entregas com qualidade, todas as áreas envolvidas com o produto devem estar alinhadas e prezar pela qualidade, desde o levantamento de requisitos até a documentação do sistema. Feigenbaum (1983) aborda a qualidade como uma estratégia que requer percepção de todos na empresa. Segundo ele, a qualidade se estende além dos defeitos no chão de fábrica; é um compromisso com a excelência; um modo de vida corporativa; um modo de gerenciamento.

A satisfação do cliente é primordial, para tanto, um dos maiores objetivos de abordagem de qualidade de software é a satisfação do cliente. Um software de qualidade não deve apenas não ter erros ou defeitos, deve atender os requisitos e ter as funcionalidades esperadas. Para Sanders (1994), um produto de software apresenta qualidade dependendo do grau de satisfação das necessidades dos clientes sob todos os aspectos do produto..

Qualidade também deve fazer parte da construção do software e na construção do software é fundamental manter o custo num nível aceitável. Uma das formas de manter o

custo é reduzir o tempo despendido em tarefas comuns para focar no que realmente é importante, testar o software e fazer análises de testes. Para Juran (1981), qualidade orientada pelo custo consiste na redução dos desperdícios e deficiências do processo de produção. Ele ainda afirma que, nesse sentido, alta qualidade implica no custo menor. Baseado nisto, aplicação de um sistema de gerenciamento de ambientes de testes se torna plausível, uma vez que o objetivo principal do sistema é reduzir o tempo com tarefas repetitivas e manuais, dispendendo mais tempo para focar na qualidade do software.

2.2 GERÊNCIA DE CONFIGURAÇÃO

Junto ao assunto de qualidade de software, tem-se o tema "Gestão de Configuração de Software (GCS)" que tratará da salva de informações, contendo histórico das versões de sistemas que serão tratados, informações que são tratadas como itens de configuração. Pressman (2005) afirma que a gerência de configuração de software é um conjunto de atividades projetadas para controlar as mudanças pela identificação dos produtos do trabalho que serão alterados, estabelecendo um relacionamento entre eles, definindo o mecanismo para o gerenciamento de diferentes versões destes produtos, controlando as mudanças impostas, e auditando e relatando as mudanças realizadas.

Para Molinari (2007) a história da GCS pode ser resumida da seguinte forma;

a) GCS baseada em arquivos (do início de 1970 até 1980):
   - voltada para controle e versionamento de arquivos pura e simplesmente;
   - volume menor de arquivos versionados;
   - suportava conceitos universais como *check-in* e *check-out;*
   - as ferramentas de GCS realizavam armazenamento das versões dos programas em arquivos separados;
b) GCS baseada em repositório (de início de 1980 até 1990);
   - as informações de cada arquivo versionado (metadados) passaram a ser armazenados em banco de dados;
   - o versionamento das versões dos programas continuava em arquivos separados
   - foco de organização para projetos;
   - ferramentas de GCS mais simplistas e intuitivas;

- introdução da gestão de mudanças;

c) GCS baseada em transparências (do início de 1990 até 2000);

- tanto os metadados como os arquivos de programas e documentos associados passariam a ser armazenados em repositórios centrais, segundo a linha de ferramentas de GCS;
- a preocupação com segurança transparente: quem acessa o quê, quem pode acessar o que, quando, o porquê acessou e o que foi acessado;
- a preocupação com rastreabilidade das informações;
- a visão de projetos ganhou força;
- a gestão de mudanças passa a ser usada como porta de entrada em processos GCS;
- a necessidade de integração entre as ferramentas de GCS e algumas soluções de software passa a ser um diferencial dos fornecedores;

d) GCS baseada em processos e produtividade (do início de 2000 até a atualidade);

- preocupação com produtividade;
- aplicação direta de conceitos de gerenciamento de projetos de forma direta, como gerenciamento por tarefas;
- preocupação com impacto de mudança;
- a linha de ferramentas de GCS torna-se fortemente visual e com cara de uma aplicação web, tendo produtos diferenciados por quem pratica e usa GCS (segmentação por tipo de usuário);
- a integração entre a ferramenta de GCS e seu ambiente de desenvolvimento passa a ser obrigatória.

A gerência de configuração se torna importante para este trabalho no momento que foi definido o controle de versionamento dos *backup*s que serão feitos pelo software, assim como a data que foi feito além de detalhes, compondo o conceito de metadados. Para Molinari (2007) metadados significa as informações relativas aos itens de configuração, tais como, a data de entrada em produção, a data de criação, quem criou o item de configuração e, principalmente, o relacionamento com os outros itens de configuração.

Cada item (aplicação, banco de dados, *middleware*) será tratado como um item de configuração, conforme o conceito de itens de configuração, que para Molinari (2007) significa que um item de configuração é o menor item de controle num processo de GCS. Ele pode ser qualquer coisa como um executável, uma aplicação corporativa, um documento ou um arquivo “.bat”.

2.3 AMBIENTES DE TESTES NA EMPRESA

Ambientes de testes devem ser construídos de forma mais íntegra possível, respeitando uma série de fatores, tais como alta disponibilidade de dados, dados íntegros (sem alterações, de acordo com o ambiente dos clientes) além de ser fácil de ser encontrado. Segundo Bastos et al. (2007, p. 83) a criação, pela equipe de teste, de um ambiente isolado, organizado, representativo e mensurável, garante a descoberta de erros reais.

Para que os testes sejam garantidos e tenham desempenho, é recomendado o uso de ambientes independentes, ou seja, que apenas a equipe de qualidade tenha acesso. Segundo Mecenas e Oliveira (2005), os ganhos para o processo de qualidade dos projetos com a criação de um ambiente independente são:

a) ambiente controlado;
b) dados íntegros;
c) base de dados reduzida;
d) utilização de massa de dados construída, e não real;
e) facilidade no gerenciamento;
f) testes não tendenciosos;
g) processo proativo (trabalhar na prevenção de erros, e não na correção deles);
h) garantia da utilização das normas e dos padrões especificados;
i) teste de todos os módulos, e não apenas dos que sofreram alteração, garantindo que nada tenha sido alterado após a manutenção.

2.4 SISTEMA ATUAL

Atualmente a empresa não possui sistema para controlar ambientes de teste, o que torna o processo demorado, quando feito. A ideia principal é ter maior disponibilidade de ambientes para testes manuais.

Os ambientes que serão tratados pelo trabalho têm a seguinte estrutura: Programa, que é instalado no computador localmente, na plataforma Windows, com banco de dados, que pode ser Oracle ou Microsoft SQL Server e também uma camada de serviços que funciona sobre o Glassfish, servidor de aplicação *open source* concebido pela Sun Microsystems e

mantido pela Oracle após a aquisição da Sun. Este conjunto de aplicações forma um ambiente de testes.

Cada profissional de testes tem o seu próprio ambiente. Caso outra pessoa queira utilizar o ambiente deste, deve solicitar que o outro testador pare o que está fazendo, execute o comando para obter um *backup* no SQL Server e disponibilize esta mídia na rede. Para muitos testadores, este processo é complicado e exige capacidade técnica, o que muitas vezes impossibilita o processo. Testadores que usam o banco de dados Oracle, raramente tem conhecimento para efetuar *backup* do banco de dados, visto que na plataforma da concorrente, Microsoft, existe uma ferramenta com interface gráfica que facilita a utilização. Ainda assim, não é feito, pois depende de uma série de comandos e passos para produzir o resultado esperado.

Outra dificuldade está relacionada às versões compiladas durante uma semana, que podem ser diversas, impossibilitando que a versão instalada em um computador possa ser replicada para outro, além de seu ambiente no Glassfish, que para ser feita cópia, precisa de parametrizações complexas em determinados arquivos.

Desta forma, é necessário que o outro testador disponibilize seu computador para que os testes possam ser feitos, o que quebra a rotina de trabalho de um dos profissionais. Outro problema atual, quando algum teste tem quebra na base do testador, e este sente a necessidade de utilizar um ambiente de algum par para garantir que o problema não está no ambiente atual.

Outro agravante para a segurança nos testes feitos atualmente está relacionado ao isolamento dos ambientes de testes. Os testadores têm acesso restrito às bases que estão localizadas nos computadores dos atendentes de suporte, que nem sempre podem disponibilizar os dados de conexão ou então uma cópia da base de dados. Também pode haver problema nas bases dos testadores, caso algum desenvolvedor faça intervenção na base de dados do mesmo, comprometendo a massa de dados, parametrizações e cadastros. Segundo Fontes, Rios e et al. (2007), o ambiente de testes deve ser isolado, com processamento independente e características similares ao ambiente de desenvolvimento e produção e deve ser restrito à equipe de testes para garantir a integridade dos testes realizados.

Ainda é identificado que os programadores têm acesso á base dos testadores. Seria ideal que os desenvolvedores não tivessem acesso aos ambientes de testes. Eventualmente ao receber uma tarefa de volta devido a algum erro de programação, o desenvolvedor pode alterar a massa de dados para facilitar algum processo na base do testador. Segundo Mecenas

e Oliveira (2005, p. 83), o mais importante é oferecer a garantia de que não houve influência externa.

2.5 TRABALHOS CORRELATOS

A seguir são apresentados uma monografia, um TCC relacionado ao tema do trabalho e um software, que atende parcialmente algumas funcionalidades do trabalho.

A monografia de Cunha (2007) destaca informações sobre ambientes de testes, tais como seus itens a equipe, a documentação, o hardware, o software, a rede, o ambiente físico, os suprimentos e interfaces, a preparação de ambientes de testes e quais os três tipos de ambientes, separados por objetivos dos testes. De acordo com Cunha (2007) tem-se:

a) ambiente de desenvolvimento: compreende aos testes unitários e de integração;
b) ambientes de testes: composto pelo teste de integração, sistema, regressão, aceitação e funcionais;
c) ambiente de produção: composto pelo teste de estresse, performance e de aceitação além de abordar o tema “testes automatizados”.

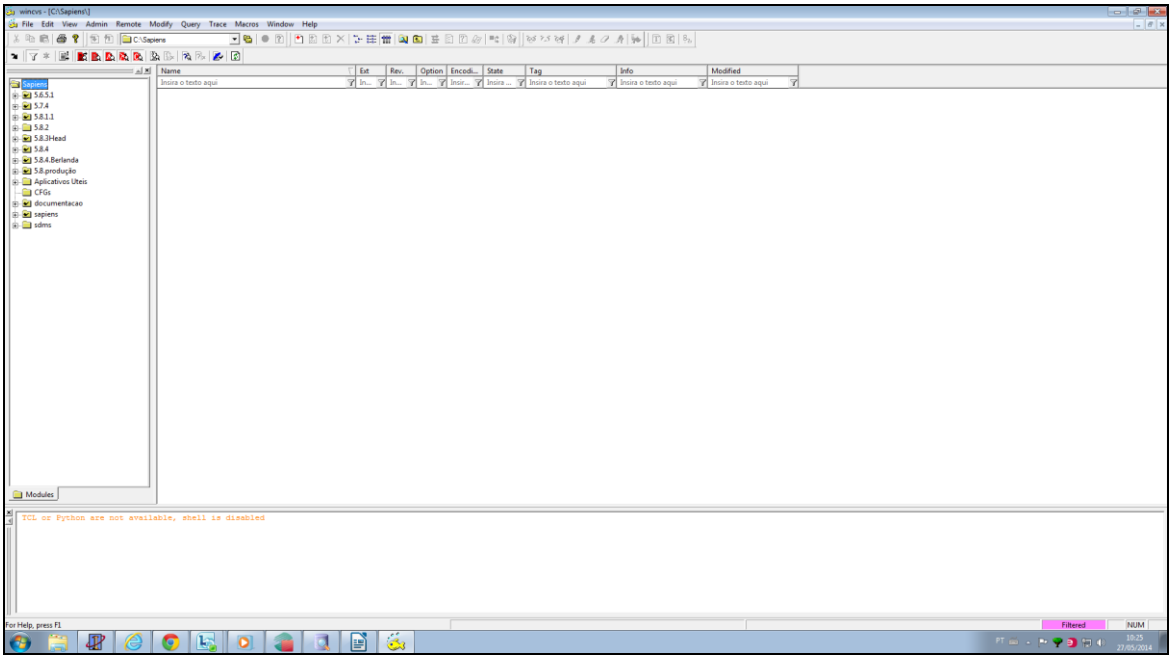
Saldanha (2009) em seu TCC destacou a importância da qualidade de software e a importância dos testes nas empresas de desenvolvimento de software, falando sobre certificação de qualidade, modelo de qualidade de software, normas ABNT, ISO, CMMI, processo de teste de software entre outros. No entanto, não aborda o tema ambiente de testes, foco desta proposta.

O software WinCVS (Figura 2), derivação gráfica do CVS, criado em 1984 por Dick Grune. Ele permite salvar arquivos em diretórios locais ou remotos (cliente-servidor), salvando a versão gravada e quem manipulou os arquivos em seus *logs*. É muito utilizado em projetos de desenvolvimento de sistemas, pelas equipes de desenvolvimento, já que permite até comparar arquivos (CVSGUI, 2007).

O software conta com alguns pontos fracos e funcionalidades que não atenderiam todas as melhorias propostas neste projeto como um todo, apesar de algumas funcionalidades serem semelhantes a desenvolvida pelo software deste trabalho, tal como a guarda de arquivos num repositório padronizado. Não se trata do que fazer, mas de como fazer, melhor e de forma fácil.

O WinCVS conta basicamente com a funcionalidade de salvar e versionar arquivos, enquanto o software que foi desenvolvido tem como princípio facilitar a vida do testador/desenvolvedor, agrupando o código fonte da aplicação, o banco de dados, e domínio do Glassfish. Além disto, o WinCVS não seria capaz de parar o banco de dados para efetuar a cópia do arquivo, e também parar o serviço de do Glassfish para efetuar a cópia das informações no domínio.

Figura 2 - CVS e a estrutura de versionamento de um dos produtos



O WinCVS trabalha com compressão *delta* dos dados salvos em repositório, tendo alto custo de processamento para armazenamento de arquivos grandes (por exemplo, no caso de bancos de dados com tamanhos superiores a 400mb), a aplicação proposta tem a opção de escolha se deseja ou não compactar o conteúdo, tornando o *backup* menor, porém mais lento para restauração (caso for usado menos vezes, porém que ocupe menos espaço) ou então sem compactação, para ser usado mais vezes, com a restauração mais rápida, a custo de mais espaço para armazenamento.

3 DESENVOLVIMENTO DA FERRAMENTA

Neste capítulo são abordadas as etapas de desenvolvimento do projeto. Primeiramente são descritos os requisitos do trabalho. Em seguida são apresentadas a especificação nas seções de diagrama de casos de uso, diagrama de classes, diagrama de entidade e relacionamento e diagrama de atividades, também a seção de implementação que contém técnicas e ferramentas utilizadas e operacionalidade da implementação destacando e explicando as principais funções do aplicativo.

3.1 LEVANTAMENTO DE INFORMAÇÕES

A ferramenta tem o propósito de facilitar o trabalho de testadores, reduzindo o tempo de testes de cada tarefa. Com o testador não se preocupando com ambiente de teste, terá mais tempo para aplicar as atividades de teste, melhorando a abrangência e profundidade dos testes.

A primeira funcionalidade prevista, motivadora do projeto, é permitir armazenar o ambiente de testes atual, para poder utilizar, criar, apagar e alterar registros sem que haja qualquer problema no sistema, já que uma imagem dele poderia ser restaurada em minutos, momentos após a bateria de testes, dependendo do tamanho da massa de dados.

No cenário atual, caso haja a necessidade de criar um ambiente idêntico ao do par de testes em sua máquina, era necessário fazer uma nova instalação do sistema (com uma série de parametrizações, restaurar um banco de dados, efetuar uma instalação do Glassfish e configuração de domínio e então atualizar com a mídia do software desejado). Com o sistema proposto, desde que haja um ambiente pronto (outro par tenha disponibilizado) bastará escolher o ambiente e clicar no botão “restaurar”. Aguardando, todo o processo acima será feito automaticamente, evidenciando o ganho de tempo, já que outras tarefas podem ser feitas durante o processo de restauração, já que não haverá nenhuma interrupção por parte do software.

A disponibilização de ambientes, uma vez que estejam parametrizados no gerenciador de ambiente para testes manuais (apenas na instalação do software ou quando um novo ambiente é instalado), bastará escolher o ambiente, parâmetro de *backup* e clicar em “*backup*”, e em minutos a mídia estará disponível para todos que usarem o software, via rede.

As mídias ficarão em um servidor comum a todos que utilizarem o software, ainda permitindo que o diretório de destino/origem seja alterado quando necessário.

O software trará uma lista de ambientes disponíveis, destacando o sistema, a versão, a *build*, nome do banco de dados, nome do ambiente *middleware*, usuário gerador da mídia, data que foi feito e observações, caso necessário. Toda essa informação será gerada automaticamente no momento da geração do ambiente, com exceção da observação, que é opcional.

No momento de efetuar o *backup* ou restauração, foi solicitada pela equipe de desenvolvimento de produto da empresa, a possibilidade de execução de *scripts* (*shell CMD* e *SQL*), para possíveis customizações no processo em termos de alterações nas massas de dados, exclusão de diretórios e arquivos, envio de e-mails entre outros.

O Quadro 1 apresenta os requisitos funcionais previstos para o sistema e sua rastreabilidade, ou seja, vinculação com o(s) caso(s) de uso associado(s).

Quadro 1 - Requisitos Funcionais X Caso de Uso

Requisitos Funcionais	**Caso de Uso**
RF01: O sistema deverá possibilitar o registro dos ambientes de testes.	UC01
RF02: O sistema deverá permitir o usuário Administrador excluir e alterar ambientes de teste.	UC02
RF03: O sistema deverá permitir ao usuário efetuar *backup* total ou parcial dos seus ambientes de testes.	UC03
RF04: O sistema deverá ter uma lista de ambientes disponíveis para efetuar restauração, de acordo com o produto selecionado.	UC03
RF05: O sistema deverá compactar a mídia de ambiente de testes e domínio do Glassfish.	UC03
RF06: O sistema deverá permitir o usuário escolher compactar ou não o *Backup* do banco de dados.	UC03
RF07: O sistema deverá permitir observações em cada *backup* a ser feito.	UC03
RF08: O sistema deverá armazenar o gerador do *backup* e a data.	UC03
RF09: O sistema deverá abrir uma janela de confirmação antes de iniciar qualquer procedimento.	UC03 e UC04
RF10: O sistema deverá permitir a importação e execução de *scripts* (via arquivo de texto) após a restauração (*CMD*, comandos Java e *SQL*).	UC03 e UC04
RF11: O sistema deverá permitir especificar o sistema a ser restaurado.	UC03 e UC04

RF12: O sistema deverá configurar automaticamente as parametrizações do Glassfish.	UC05
RF13: O sistema deverá permitir a configuração de acesso a bancos de dados para efetuar *backup*.	UC06 e UC07
RF14: O sistema deverá permitir alterar o local de armazenamento dos *backups*, de acordo com o produto a ser feito *backup*.	UC08
RF15: O sistema deverá permitir alterar e excluir serviços do Windows relacionados aos sistemas.	UC09
RF16: O sistema deverá ter uma tela de *login* para acesso ao sistema e respectivo *logout.*	UC10
RF17: O sistema deverá permitir alteração do local de armazenamento das mídias de instalação (origem e destino).	UC12 e UC13
RF18: O sistema deverá permitir alterar o local que ficam armazenadas as mídias de restauração, de acordo com o produto a ser restaurado.	UC13

O Quadro 2 lista os requisitos não funcionais previstos para o sistema.

Quadro 2 - Requisitos não funcionais

Requisitos Não Funcionais
RNF01: O sistema deverá ser compatível com as plataformas *SQL* Server e Oracle.
RNF02: O sistema deverá rodar na plataforma Windows.
RNF03: O sistema deverá ser desenvolvido em Java.
RNF04: O sistema deverá rodar em rede TCP/IP.

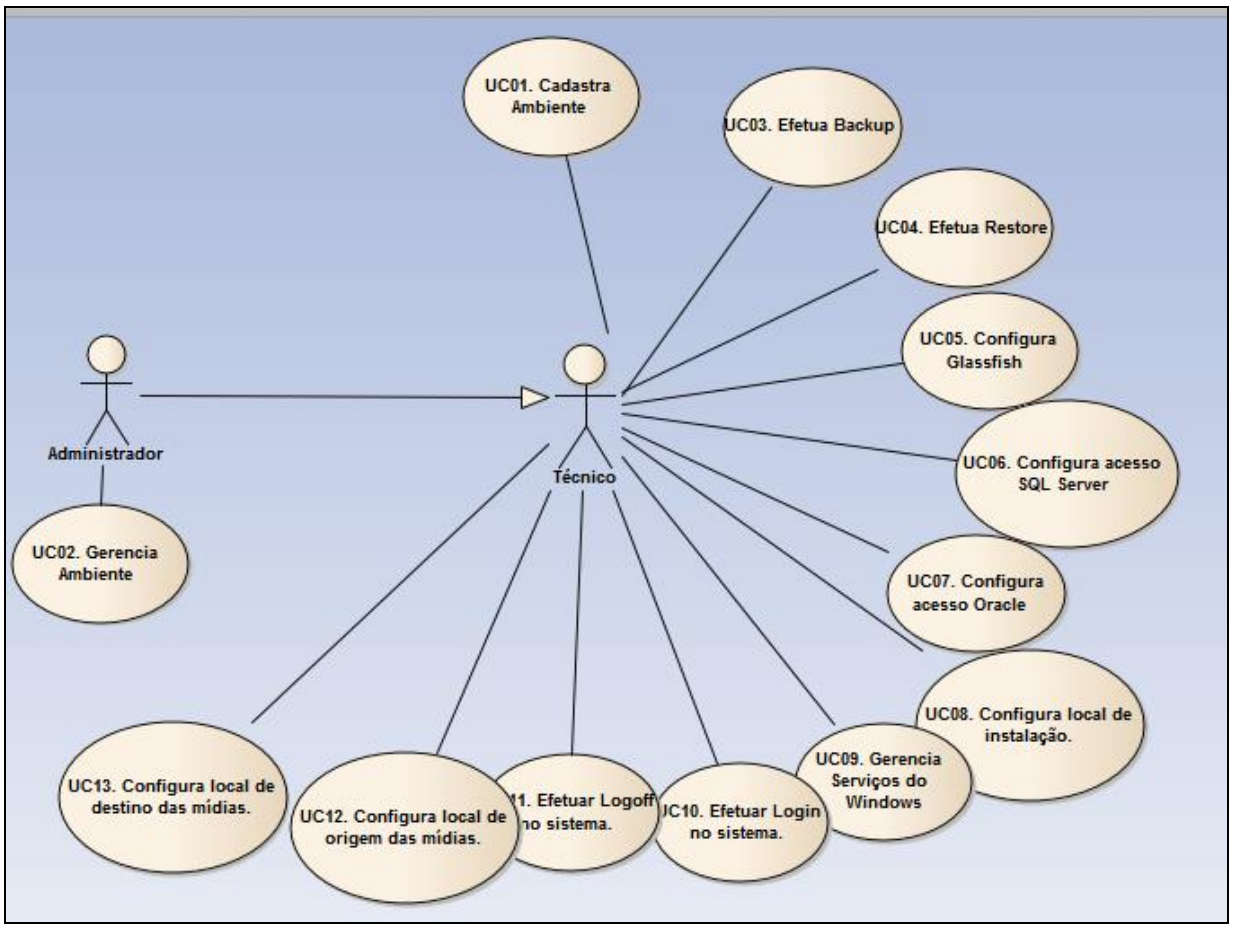
3.2 ESPECIFICAÇÃO

Esta seção apresenta os diagramas de caso de uso e os modelos de entidade relacionamento (MER) do módulo desenvolvido. Para gerar os diagramas foi utilizada a ferramenta Enterprise Architect (EA) e para o MER utilizou-se a ferramenta SQL Server Management Studio.

3.2.1 Diagrama de casos de uso

Na Figura 3, localiza-se o diagrama de caso de uso, contendo 13 UCs e dois atores, que são o técnico, que usará o sistema para criar os ambientes de testes e efetuar restaurações, e o administrador, que terá todos acessos do técnico mas poderá excluir *backup*s que estarão no repositório na rede.

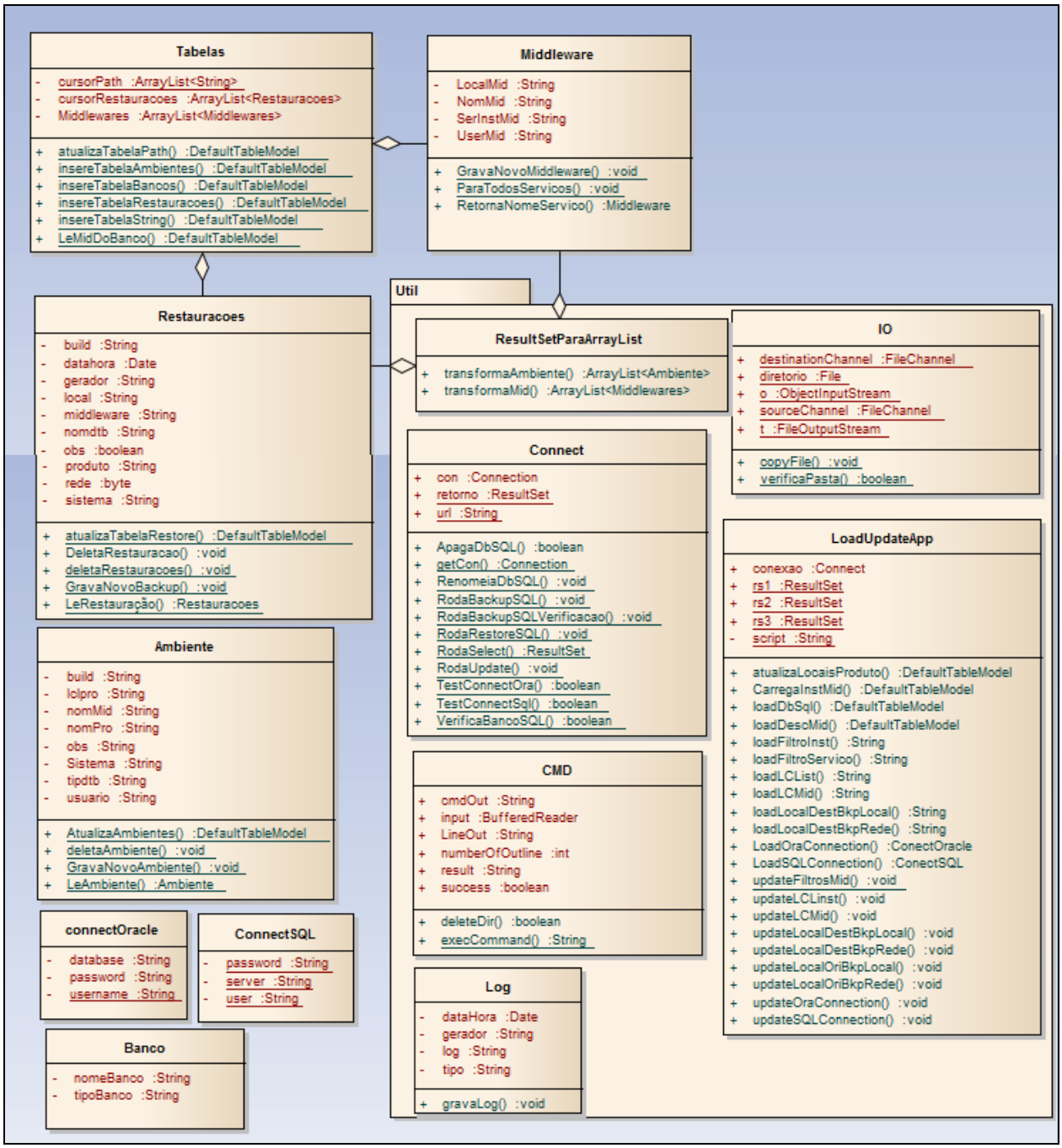
Figura 3 - Diagrama de Caso de Uso



3.2.2 Diagrama de classes

As classes da aplicação possuem dependências, como por exemplo, métodos que usam objetos provenientes de outras classes, conforme a Figura 4.

Figura 4 - Diagrama de Classes

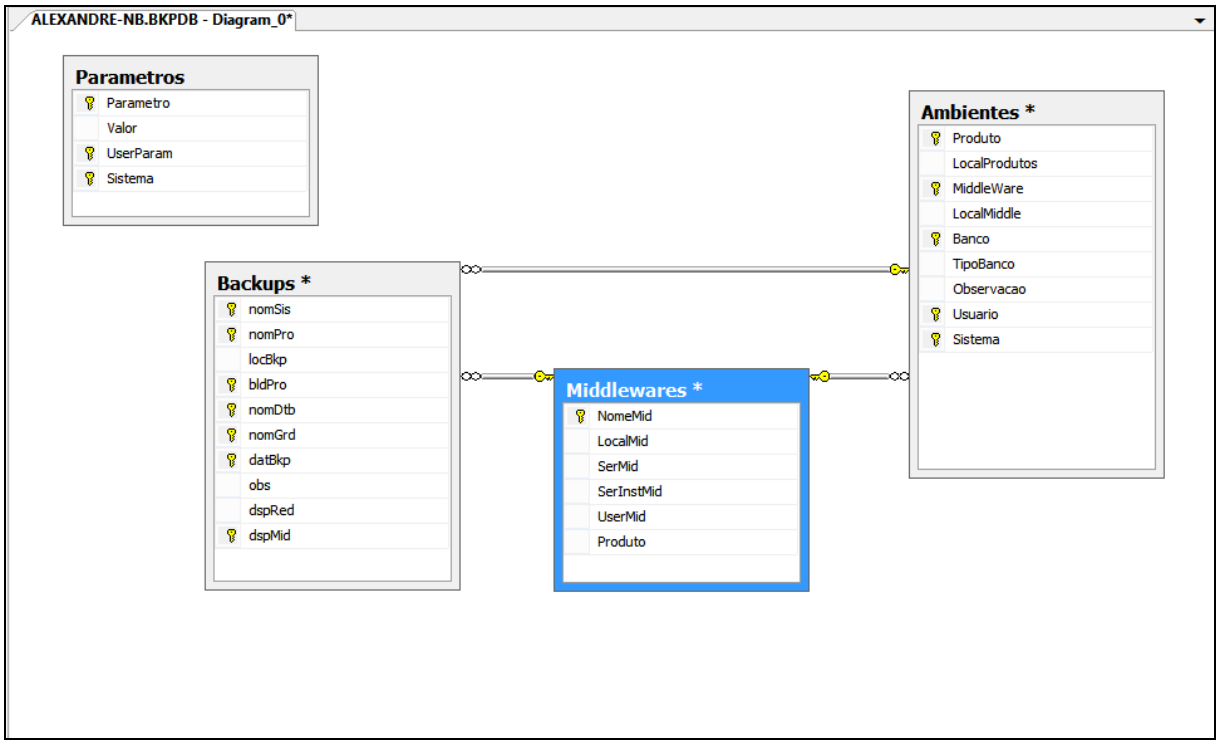


3.2.3 Modelo de entidade e relacionamento

Para desenvolvimento da aplicação foi criado um modelo de entidade e relacionamento (MER). No MER (Figura 5), destaca-se a simplicidade do modelo de dados, onde pode-se observar que existem apenas duas chaves estrangeiras, para controle de integridade de dados, que são os relacionamentos entre a tabela de ambientes *versus*

middlewares e *backup*s *versus middlewares*, não seguindo portanto, as formas normais de modelagem de dados.

Figura 5 - MER



A seguir uma breve descrição das tabelas do diagrama:

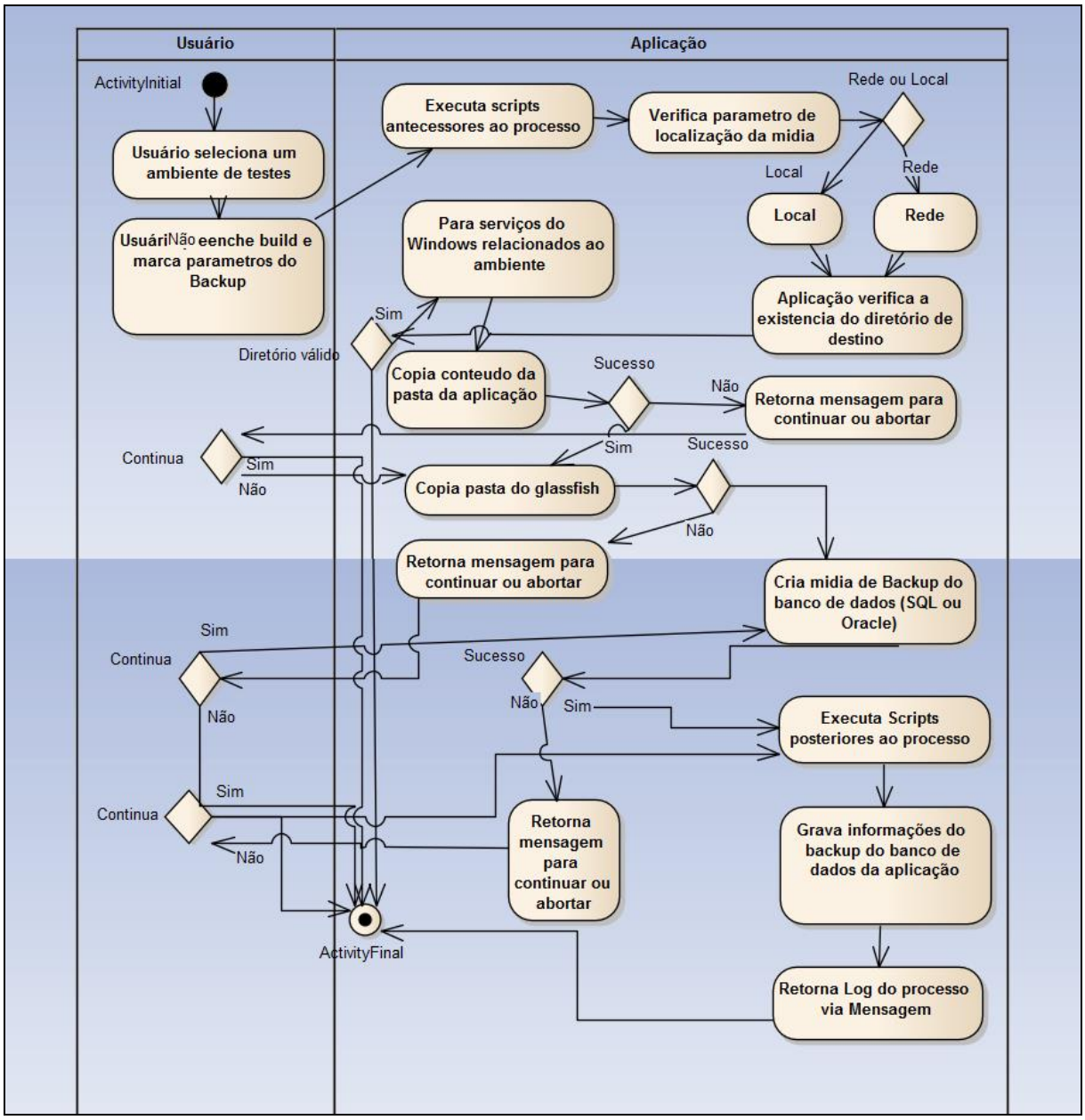
a) parâmetros: representa cadastro dos parâmetros para gravar valores padrões por usuários;
b) ambientes: representa as informações dos ambientes de testes disponíveis para ser efetuado *backup;*
c) *backup*s: representa as informações dos *backup*s efetuados pelos testadores;
d) *middlewares*: representa os ambientes de *middleware* vinculados ao Sistema.

3.2.4 Diagrama de atividades

Nesta sub seção serão apresentados dois diagramas de atividades (Figuras 6 e 7) para representar os principais processos do sistema, *Backup* e restauração do sistema, destacando principalmente os processos internos do aplicativo.

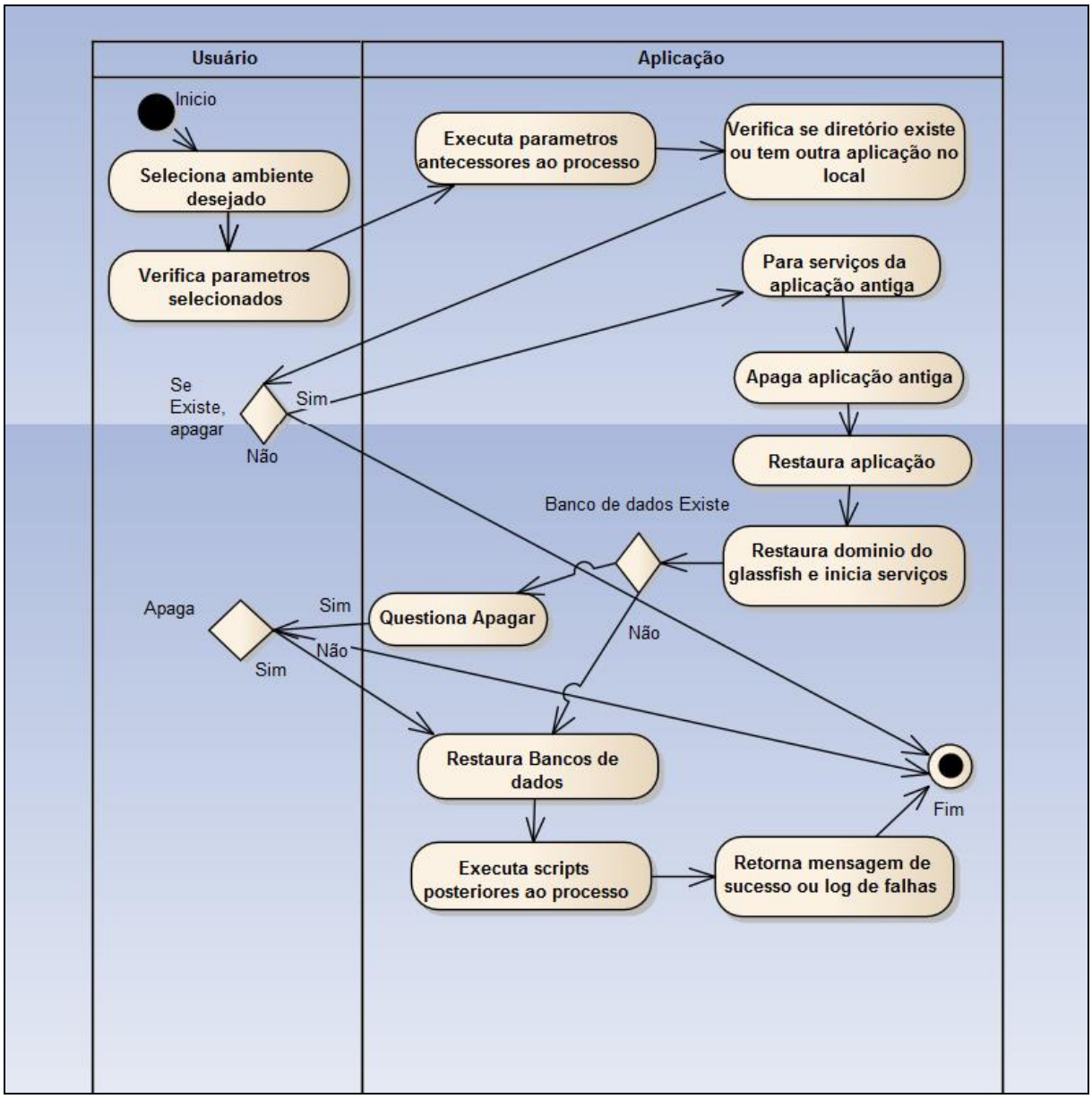
Na Figura 6, apresenta-se o diagrama de atividades, do processo de backup, contendo duas raias, a de usuário, com suas ações, e da aplicação, com seus processos, que dependem da parametrização do *backup*.

Figura 6 - Diagrama de Atividades - *Backup*



Na Figura 7, apresenta-se o diagrama de atividades, do processo de restauração, contendo duas raias, a de usuário, com suas ações, e da aplicação, com seus processos, que dependem da parametrização da restauração.

Figura 7 - Diagrama de Atividades - Restauração



3.3 IMPLEMENTAÇÃO

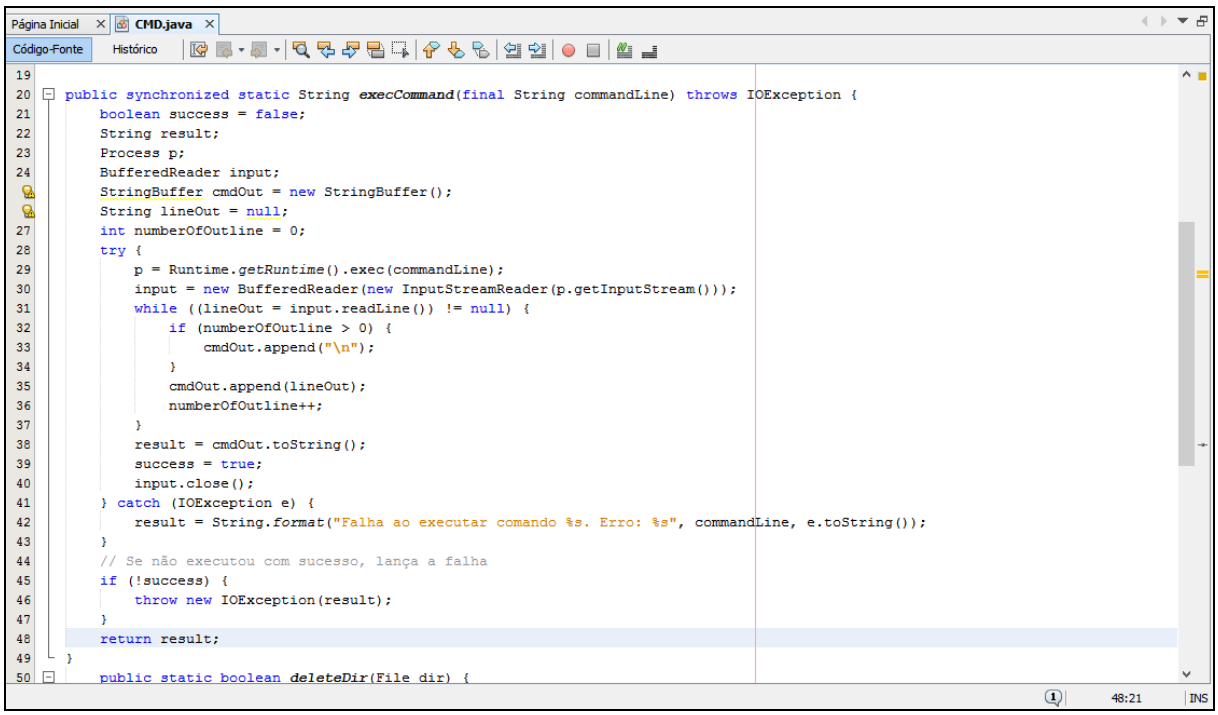
A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento do sistema foram utilizadas tecnologias de desenvolvimento disponibilizadas pela empresa Oracle. O desenvolvimento foi feito através do Java através de seu *Software Development Kit* (SDK) na versão 1.8.0 juntamente com a *Integrated Development Environment* (IDE) NetBeans, em sua versão 8.0.1.

A seguir são destacados alguns dos métodos da aplicação e suas respectivas funcionalidades, entrada e saída de dados. Na classe CMD, o método "ExecCommand" (Figura 8) é responsável por efetuar a comunicação do aplicativo com a plataforma de linha de comandos nativa do Windows. O método tem uma variável do tipo *String*, que repassa o comando a ser executado através do *prompt* de comando do Windows, de forma que o usuário não veja e nem tenha interação com a interface do *prompt*. O método retorna uma *String* com o resultado do comando executado, tais como confirmações de resultado ou até mesmo uma série de dados, para alimentar tabelas, por exemplo.

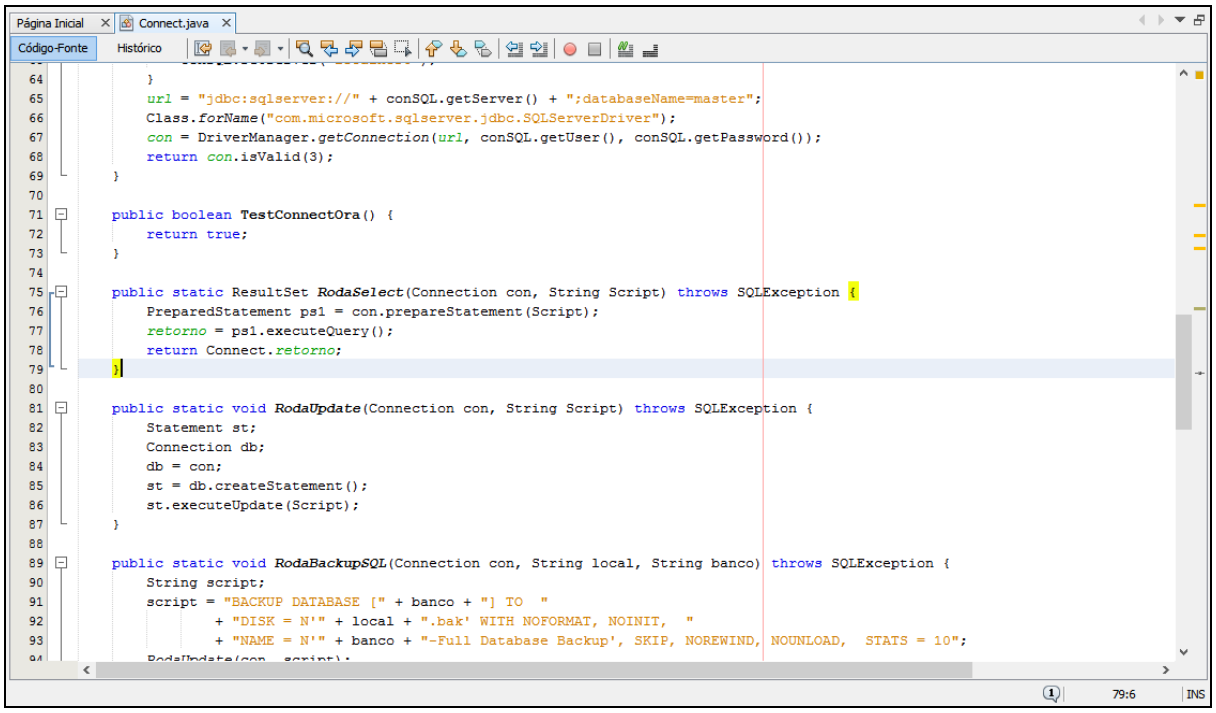
Figura 8 - Classe CMD – Método "ExecCommand"



Na classe *Connect*, tem-se o método "RodaSelect" (Figura 9) que é responsável por rodar comandos no banco de dados SQL Server da aplicação ou então no banco de dados do ambiente de testes. Como entrada, necessita de uma *string*, com a linha de comando desejada

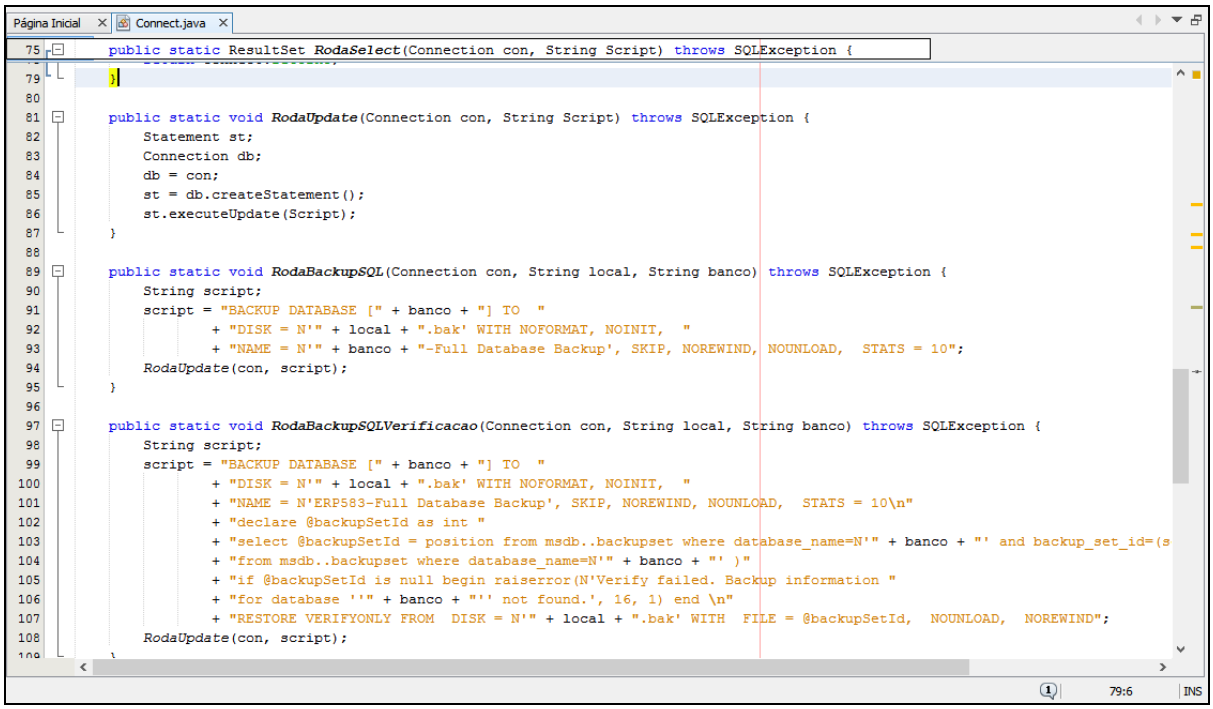
e um objeto do tipo *connection* que contém os dados de conexão de onde será rodado o comando. O método retorna um conjunto de dados no formato “ResultSet”, um formato padronizado da biblioteca “java.sql.ResultSet” de retorno de dados em formato semelhante ao de uma tabela do banco de dados SQL Server.

Figura 9 - Classe *Connect* – Método “RodaSelect”



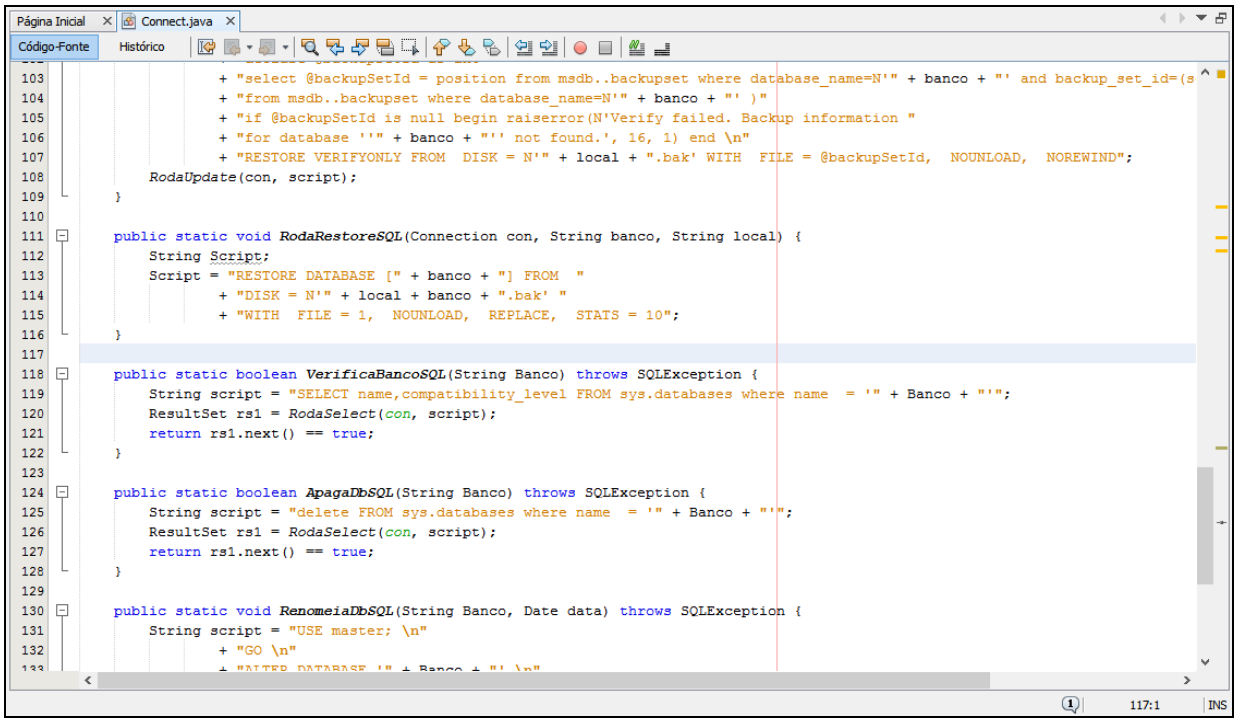
Ainda na classe *Connect*, o método “RodaBackupSQL” (Figura 10) é responsável por efetuar o *backup* (gerar a mídia, no formato .BAK) do banco de dados SQL Server. O método recebe como parâmetro uma conexão, no formato *connection*, uma *string*, para definir o local de destino da mídia e outra *string*, que contém o nome do banco de dados a ser feito *backup*. Este método não tem nenhum retorno.

Figura 10 - Classe *Connect* – Método “RodaBackupSQL”



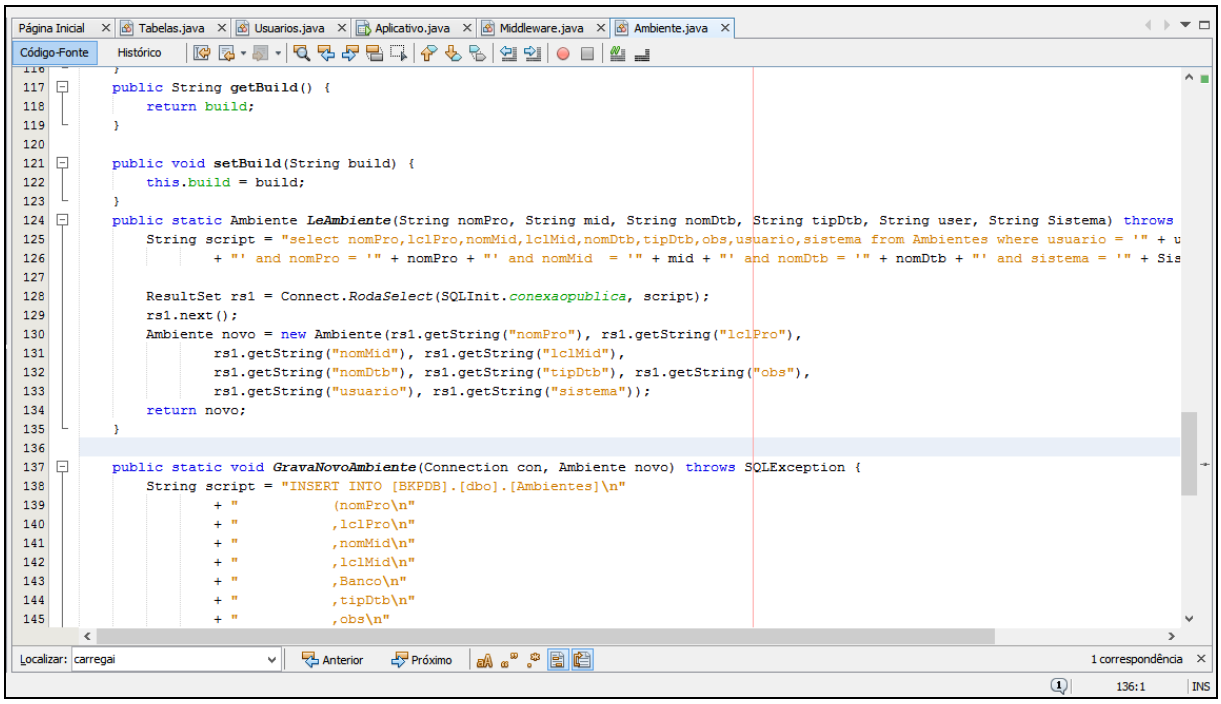
O Método “RodaRestoreSQL” (Figura 11) é responsável por executar o comando de restauração do banco de dados SQL Server, executando o comando necessário para transformar a midia (arquivo .bak) num banco de dados anexado no SQL Server. O método recebe como parâmetro um objeto do tipo *connection*, uma String com o nome do banco de dados e o local que a mídia se encontra. Este método não possui nenhum retorno.

Figura 11 - Classe *Connect* – Método “RodaRestoreSQL”



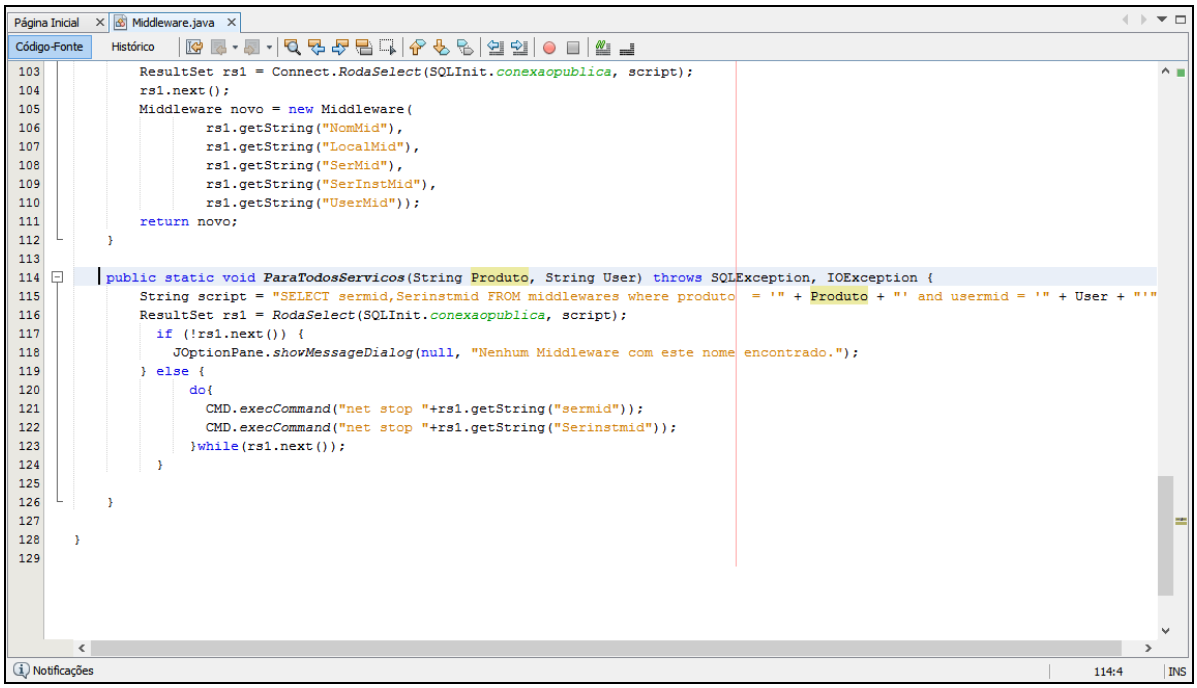
Na classe Ambiente, o método “LeAmbientes” (Figura 12) é responsável por retornar do banco de dados um ambiente de teste disponível e transformar num objeto do tipo Ambiente, para que o sistema trabalhe com as informações contidas no objeto, tais como, nome do ambiente, nome do banco de dados, localização da instalação, sistema, etc. O método recebe como parâmetros várias *strings* que são utilizadas para encontrar o ambiente correto no banco de dados da aplicação. Como retorno o método fornece um objeto do tipo Ambiente.

Figura 12 - Classe ambiente – Método “LeAmbientes”



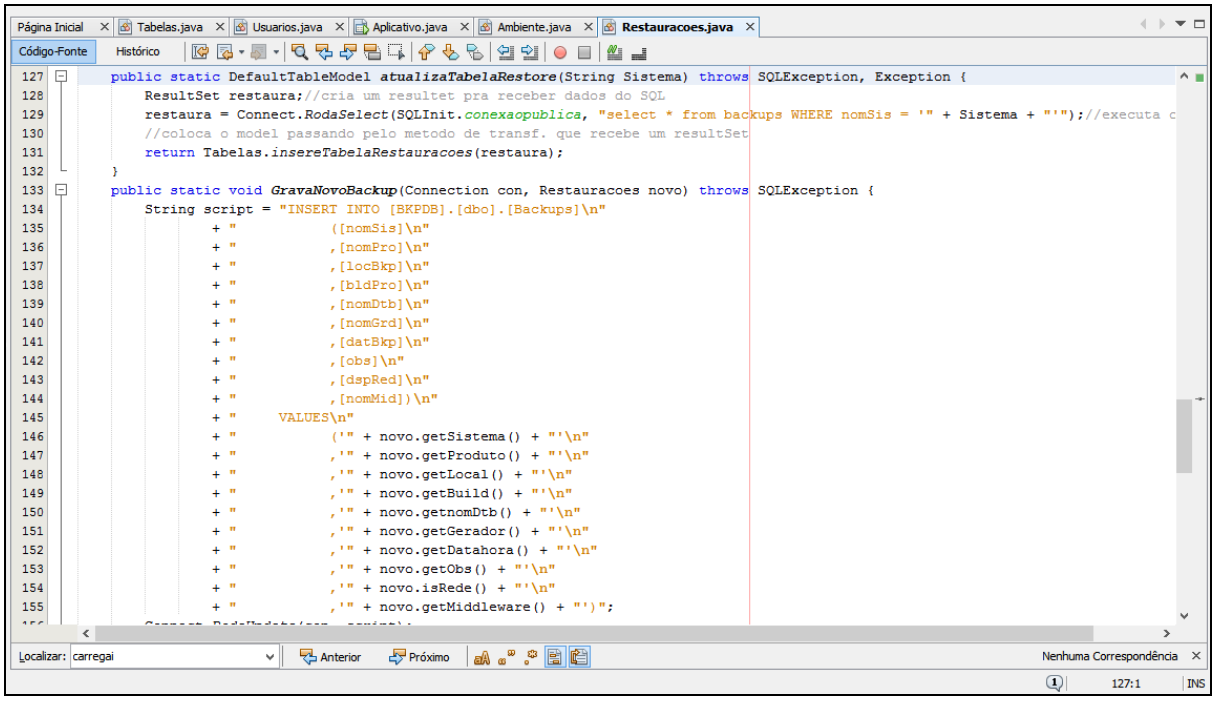
Na classe *Middleware* O método “ParaTodosServicos” (Figura 13) tem a funcionalidade de parar os serviços da aplicação que será feito o *backup*. O método recebe como parâmetro duas *strings*, nome do Produto e usuário. Então, baseado nestas informações busca o nome dos serviços de *middleware* e instalação, para que um comando seja executado para parar os serviços. O método não possui nenhum retorno.

Figura 13 - Classe *Middleware* – Método “ParaTodosServiços”



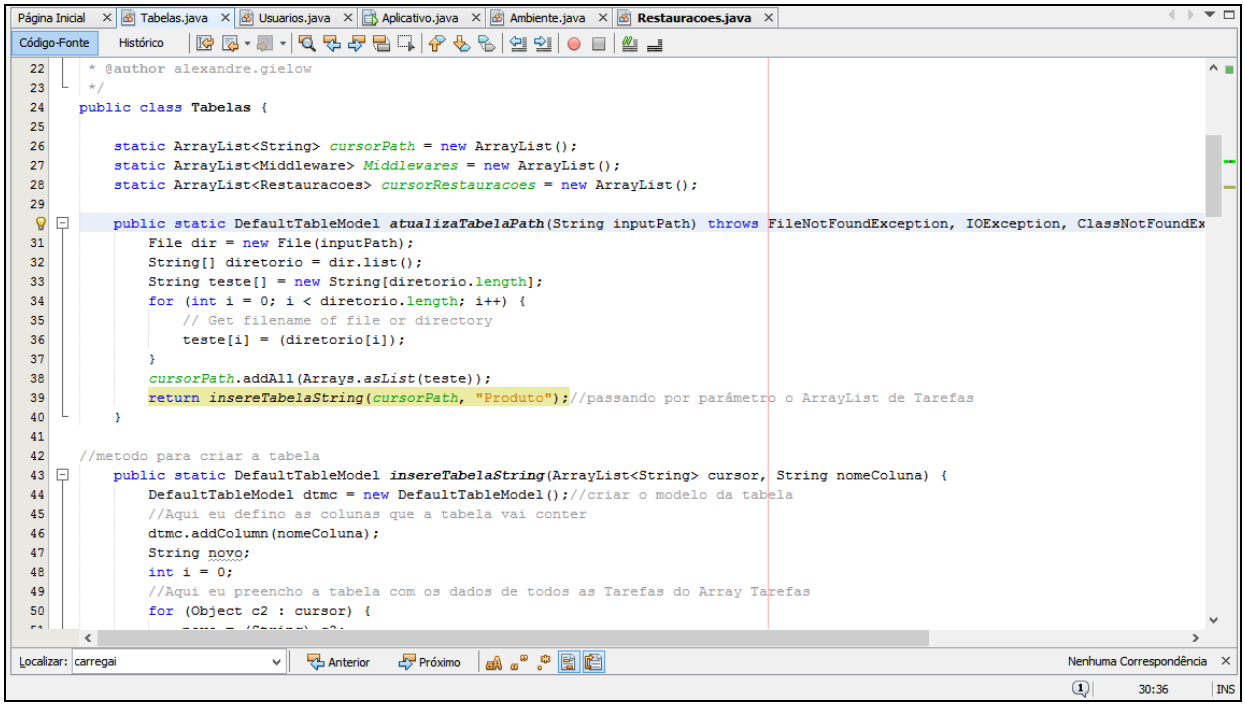
Na classe Restaurações, o método **“**GravaNovoBackup” (Figura 14) recebe como parâmetro um objeto do tipo *connection* e um objeto do tipo Restaurações, que contém todas as informações para gravar no banco de dados um novo *backup*. O método não possui retorno.

Figura 14 - Classe Restauracoes – Método “GravaNovoBackup”



Na classe Tabelas o Método “LeMidDoBanco” (Figura 15), é um exemplo de como funcionam os demais métodos da classe. Ele recebe o nome do usuário e o nome do produto, em seguida, efetua um *select* no banco de dados da aplicação, transforma o conjunto de resultados *ResultSet* num *arraylist* de objetos do tipo *middleware*, em seguida, cria uma *defaultTableModel*, contendo os dados requisitados. Por fim, o método retorna a *defaultTableModel*, para ser exibida ao usuário na camada de aplicação.

Figura 15 - Classe Tabelas – Método “LeMidDoBanco”

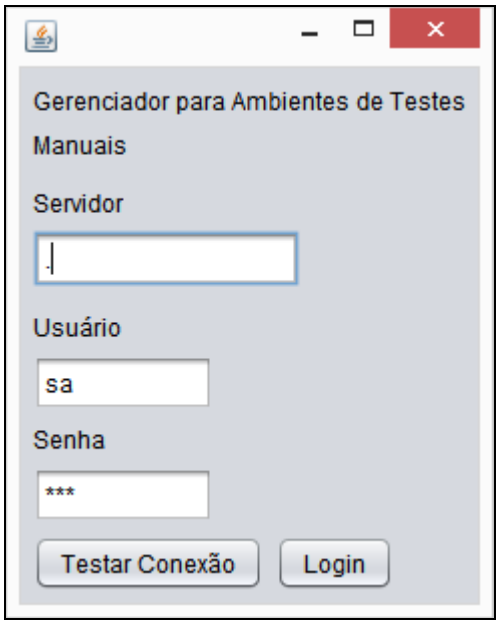


3.3.2 Operacionalidade da implementação

Para demonstrar o uso da aplicação, serão simulados cenários que usam todo o conjunto de funcionalidades do sistema. A tela de *login* (Figura 16) tem por objetivo receber servidor, usuário e senha para conectar a aplicação com o banco de dados onde estão gravados os parâmetros de usuário e informações sobre *backup*s. Usuários não cadastrados não terão acesso à ferramenta. Usuários que sejam cadastrados, porém não tenham parâmetros (primeiro acesso, por exemplo), serão automaticamente parametrizados através de parâmetros

padronizados, para que a ferramenta funcione.

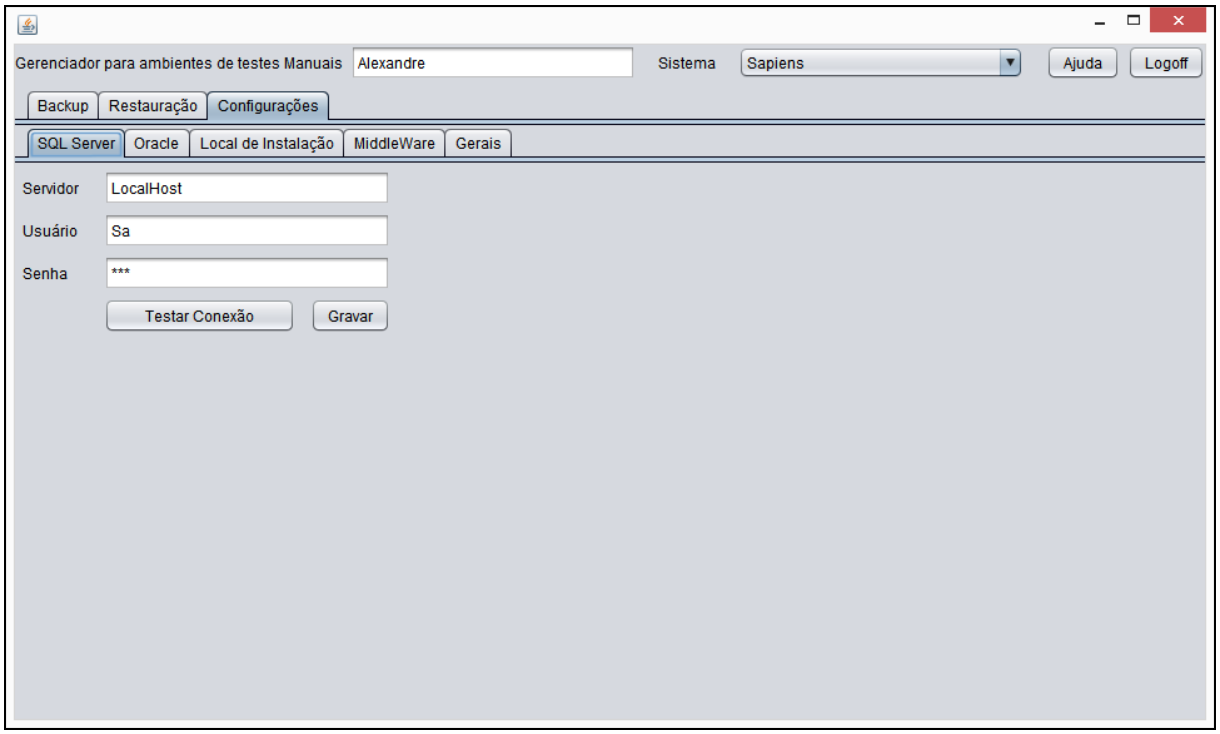
Figura 16 - Tela de *Login*



Após o *login* na aplicação, deve-se parametrizar o sistema de acordo com o usuário e suas necessidades. As telas de configurações serão sub agrupadas para um melhor entendimento, pois além de configurações, têm funcionalidades de controle de serviços e criação de novos ambientes.

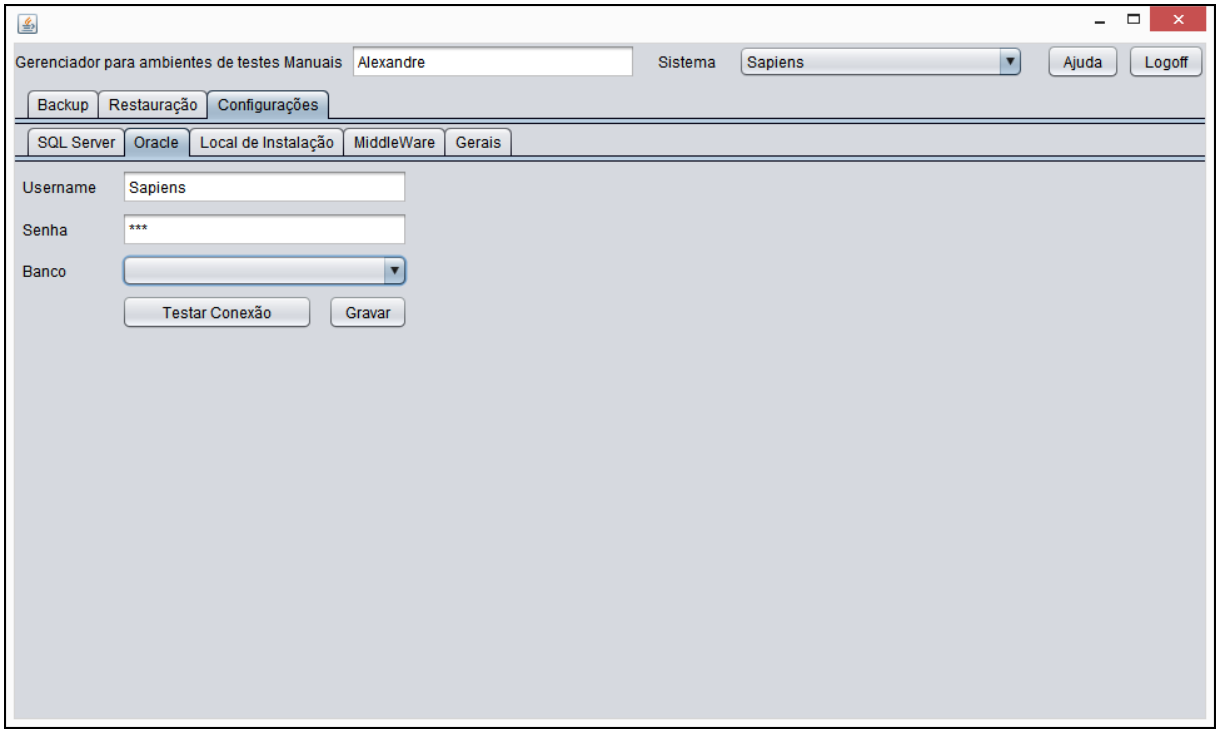
Inicialmente, na aba de configurações de conexão do SQL Server (Figura 17) tem por objetivo parametrizar a conexão de origem dos bancos de dados de aplicativos, para o SQL Server, preenchendo os campos servidor, usuário e senha.

Figura 17 - Aba de Configurações – SQL Server



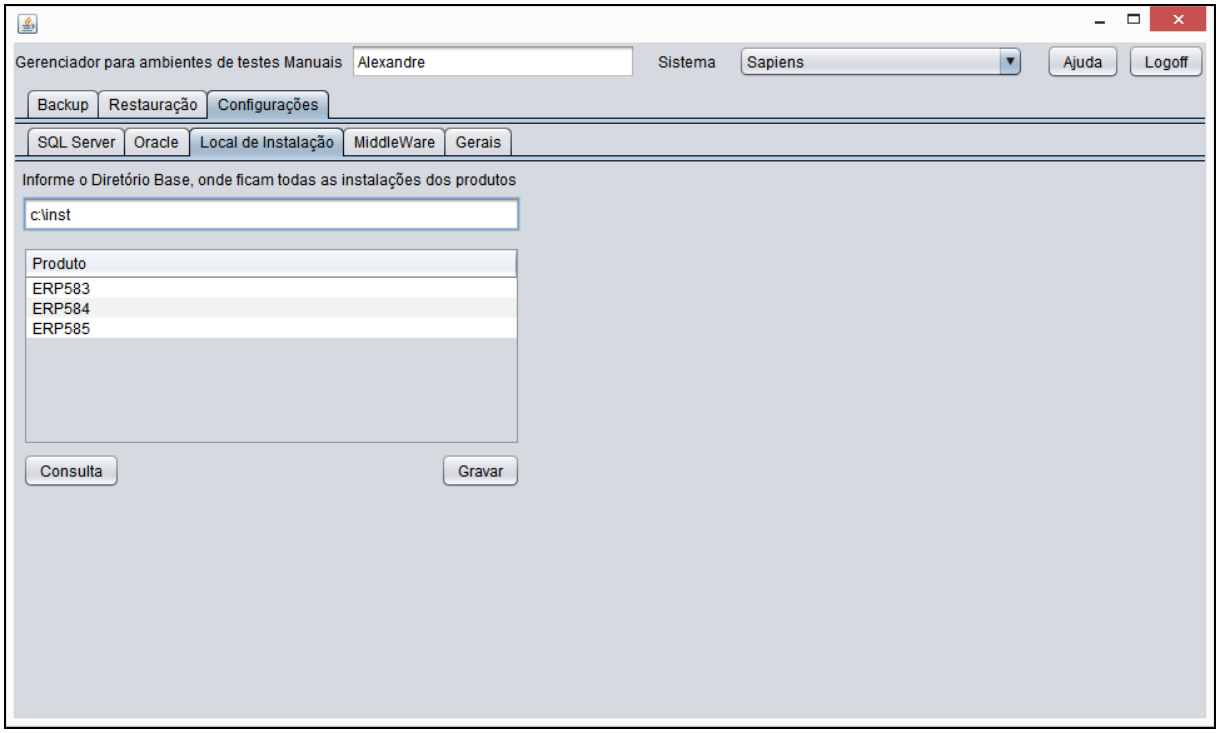
Na segunda aba, de configurações de conexão do Oracle (Figura 18), tem por objetivo parametrizar a conexão de origem dos bancos de dados de aplicativos, para o Oracle, preenchendo os campos usuário, senha e Banco.

Figura 18 - Aba de Configurações – Oracle



Na terceira aba, local de instalação (Figura 19), fica registrado o local onde os produtos estão instalados, isto é, diretório de onde será originado o *backup*, de acordo com o sistema selecionado no *comboBox* Sistema, no topo da tela.

Figura 19 - Aba de Configurações – Local de Instalação

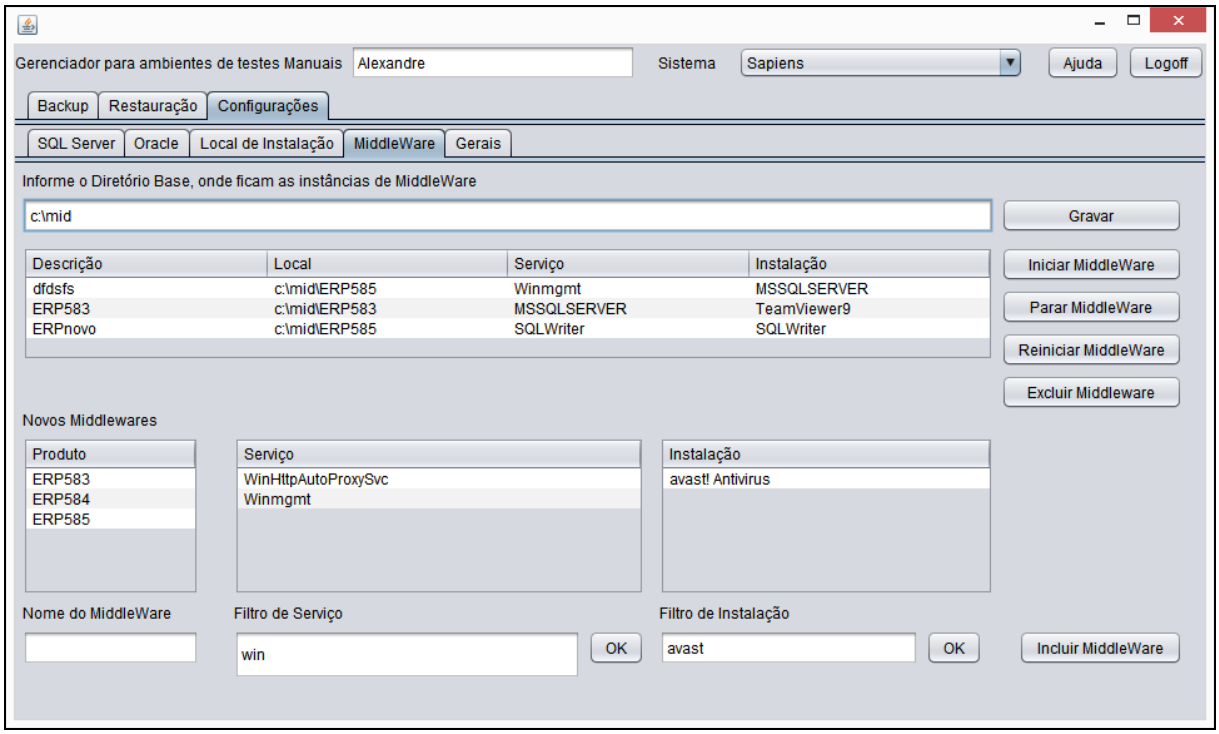


A aba de *Middleware* (Figura 20) contém o cadastro de ambientes *middleware*, logo abaixo do meio da tela, lendo a pasta de origem (onde fica localizado o Glassfish), o serviço de *middleware* da aplicação (que deve ser filtrado no campo abaixo da tabela) e também o serviço de instalação (que deve ser filtrado no campo abaixo da tabela). Após a seleção destas configurações, é criado um ambiente de *middleware*, e passa a aparecer na tabela localizada do meio para cima da tela.

A tela consta os seguintes botões:

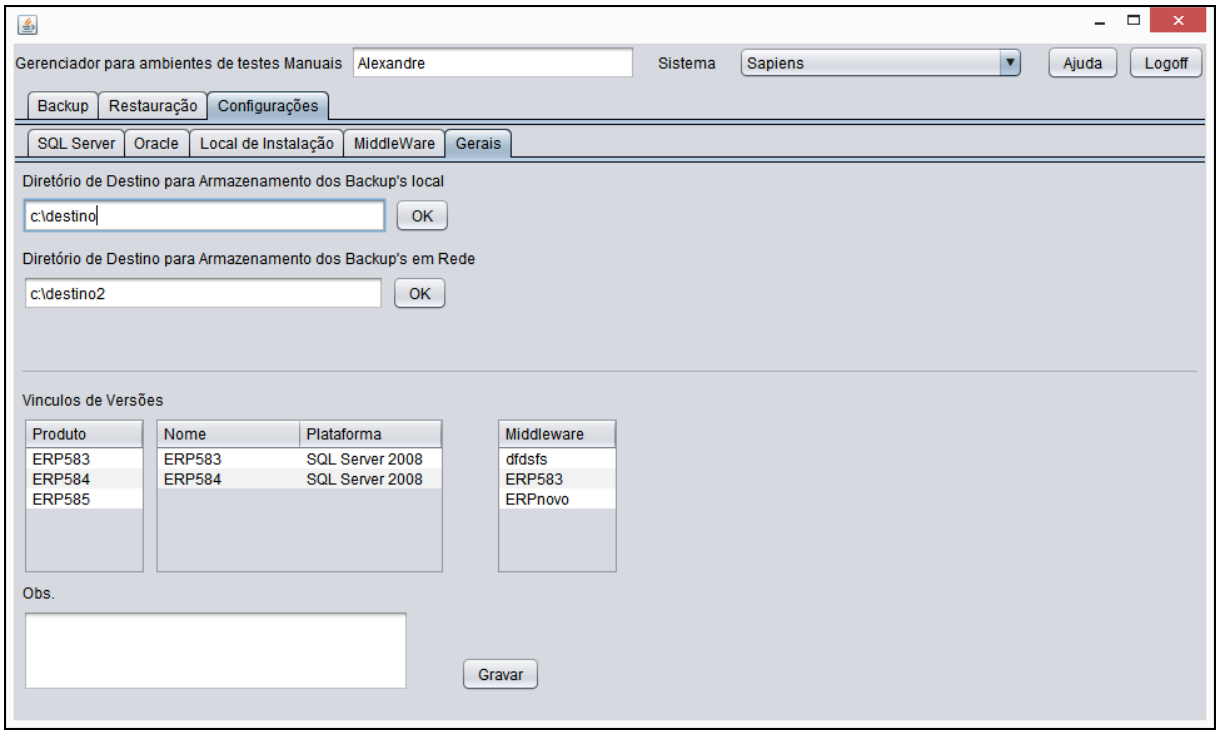
a) gravar: grava o diretório onde estão localizados os Glassfish;
b) iniciar *middleware*: para o(s)ambiente de *middleware* atualmente rodando e inicia o que estiver selecionado na tabela;
c) para *middleware*: para o(s)ambiente de *middleware* selecionado na tabela;
d) reiniciar *middleware*: para e inicia novamente os serviços *middleware*;
e) excluir *middleware*: exclui do registro o ambiente *middleware*, com a opção de remover os serviços e domínio do Glassfish.

Figura 20 - Aba de Configurações – *Middleware*



Na aba de configurações gerais (Figura 21), são vinculados: produto, banco de dados e ambiente *middleware*, formando então um ambiente de teste, que terá destino a aba "*backup*". Além do vinculo de versões, existem os dois campos de destino das das midias geradas pelo aplicativo, um deles para destino local (no próprio disco do computador do usuário) e outro para o local de destino em rede (para salvar num sevidor remoto por exemplo). Este é o ponto final das configurações. A partir deste momento o sistema já está em condições de funcionamento.

Figura 21 - Aba de Configurações – Gerais



Agora será feito um *backup* de um ambiente de testes, previamente configurado, utilizando de todas as opções dispostas pelo sistema, buscando a maior combinação possível de parametrizações. A tela de ambientes de Testes (Figura 22) guarda as informações dos ambientes de testes atualmente parametrizados no sistema, que são configurados previamente na tela de configurações.Seu uso baseia em escolher o sistema desejado pelo *Combobox* ”Sistema”, um ambiente de testes clicando em uma das linhas da tabela, informar qual a *build* do ambiente e escolher os parametros de configurações do *backup*:

a) aplicativo: parâmetro que define se o aplicativo do ambiente de testes será incluído na mídia de *backup*;
b) *middleware*: parâmetro que define se o *middleware* do ambiente de testes será incluído na mídia de *backup*;
c) banco de dados: parâmetro que define se deve ser feito uma cópia do banco de dados para ser incluso na mídia de *backup*;
d) grava na rede: parâmetro que define se o usuário deseja disponibilizar o *backup* no diretório de rede;
e) validar *backup*: parâmetro que define se é necessário fazer uma verificação da mídia de *backup*, verificando consistencia;
f) executa *script* SQL antes:permite executar um *script* SQL antes do *backup*, deve

ser um arquivo no formato “.txt”;

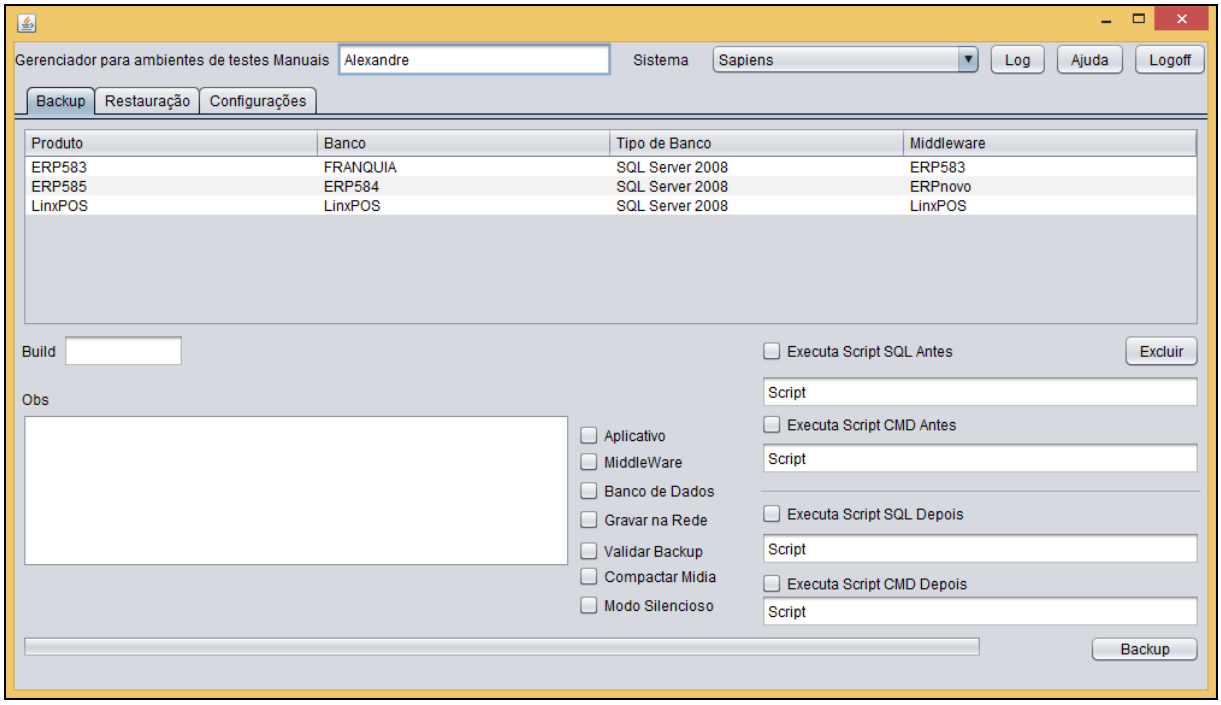
g) executa *script* CMD antes:permite executar um *script* CMD antes do *backup* , deve ser um arquivo no formato “.txt”;

h) executa *script* SQL depois:permite executar um *script* SQL depois do b*ackup* , deve ser um arquivo no formato “.txt”;

i) executa *script* CMD depois:permite executar um *script* CMD depois do b*ackup*, deve ser um arquivo no formato “.txt”.

Após escolhidas as configurações, e clicar no botão “*backup*” (Figura 22) o processo é iniciado, criando a mídia para ser posteriormente restaurada, executando passo a passo conforme parâmetros acima citados.

Figura 22 - Aba de *Backup*



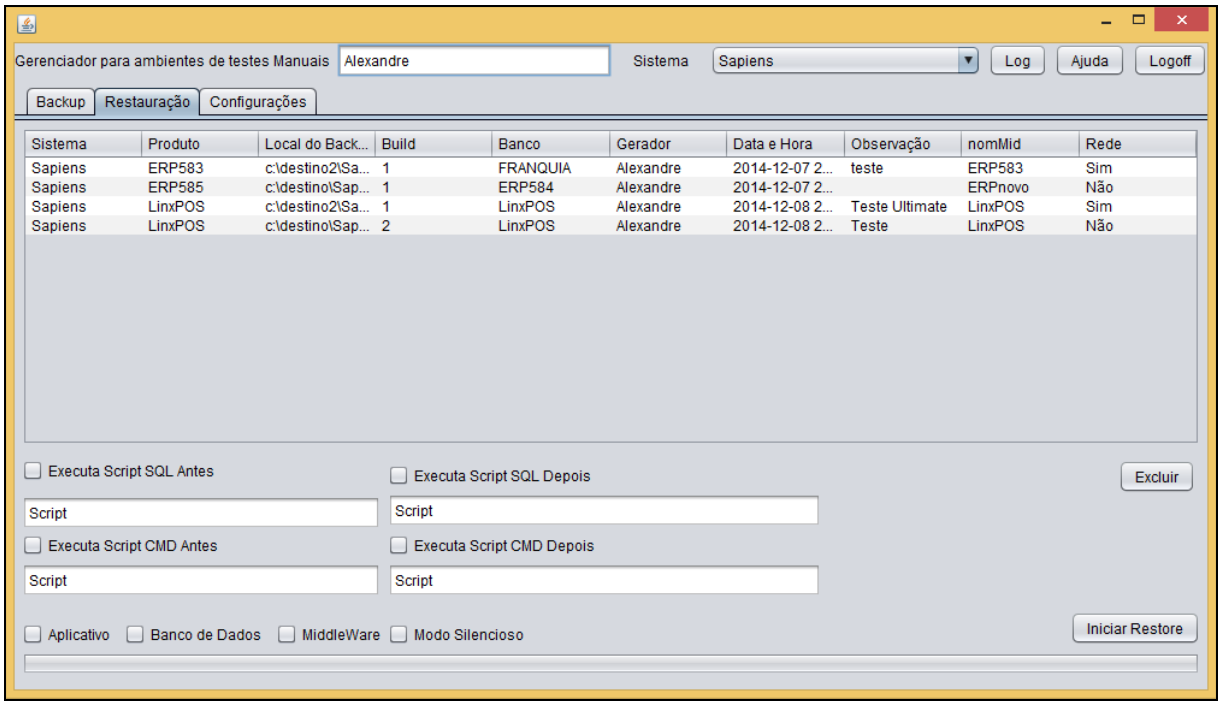
Na aba de Ambientes para Restaurações (Figura 23) pode-se observar a funcionalidade de exibir os ambientes de testes disponiveis para ser feita restauração, carrega os ambientes gravados em rede e também os que estão localizados no próprio disco local, para que o testador escolha o ambiente desejado, de acordo com as informações contidas em cada ambiente, tais como, Sistema, Produto, local do *backup*, *build*, banco, gerador, data, observação, nome do *middleware* e se está disponivel na rede. Será feito uma Restauração de um ambiente de testes, listado na tabela, utilizando de todas as opções dispostas pelo sistema, buscando a maior combinação possível de parametrizações.

Assim como a tela de *backup*, existem parâmetros para serem selecionados, resultando em diferentes tipos de restaurações:

a) aplicativo: parâmetro que define se o aplicativo do ambiente de testes deve ser restaurado;
b) *middleware*: parâmetro que define se o *middleware* do ambiente de testes deve ser restaurado;
c) banco de dados: Parametro que define se deve ser feito restauração do banco de dados do ambiente de teste;
d) executa *Script SQL* antes:permite executar um *script SQL* antes da restauração;
e) executa *Script CMD* antes:permite executar um *script CMD* antes da restauração;
f) executa *Script SQL* depois:permite executar um *script SQL* depois da restauração;
g) executa *Script CMD* depois:permite executar um *script CMD* depois da restauração.

Após escolhidas as configurações, e clicar no botão “Iniciar Restauração” (Figura 27) o processo é iniciado, sobrescrevendo o ambiente atual (caso ele já exista) ou criando um novo ambiente

Figura 23 - Aba de Restauração



3.4 RESULTADOS E DISCUSSÃO

O presente trabalho apresentou o desenvolvimento de uma plataforma para gerência de ambientes de testes manuais. Os requisitos para criação da plataforma foram elaborados com base nas necessidades de testadores, programadores e gerentes da empresa desenvolvedora de software.

O sistema foi testado e validado com base nas operações previstas nos casos de uso onde atendeu aos requisitos de sistema. A aplicação roda locamente, portanto, não são necessários testes de estresse para avaliar o desempenho. Além dos testes feitos, foram implementados *logs* nos procedimentos executados nos principais processos da aplicação, disponivel na aba de *logs* do aplicativo

Em relação aos trabalhos correlatos, o Quadro 3 apresenta as principais semelhanças e diferenças entre o aplicativo WinCVS e o aplicativo construído.

Quadro 3 - Comparativo Software Correlato

Plataforma	Aplicação	WinCvs
Versionamento	Sim	Sim
Armazenamento em Rede	Sim	Sim
Engloba o ambiente de testes como um todo	Sim	Não
Pré parametrizavel	Sim	Não
Criptografa Dados	Não	Sim

O artigo de Cunha (2007) trouxe um embasamento teórico de grande contrubuição para este trabalho, destacando ponto a ponto informações sobre ambientes de testes, que aumentaram a abrangência dos assuntos para implementação da ferramenta construida.

Saldanha (2009) em seu TCC, destacou o assunto Qualidade de Software, pilar deste trabalho, e suas principais vertentes, certificações, modelos e normas, que também contribuiram para o desenvolvimento da monografia e aplicação.

A empresa de desenvolvimento de software recebeu uma apresentação do sistema em dezembro de 2014 e agendou implantação em caráter de teste para janeiro de 2015.

4 CONCLUSÕES

O ambiente desenvolvido permite que técnicos de qualidade façam uma real administração de seus ambientes de testes, podendo compartilhar os mesmos entre outros testadores de forma simples, esquecendo os principais problemas operacionais relacionados a cópia e replicação de dados, além de não precisarem mais se preocupar com massa de dados.

Observou-se que para o lançamento do ambiente como um produto comercial (ficou agendado para janeiro de 2015 o início dos testes na empresa de desenvolvimento de software) que ajustes ainda são necessários, principalmente no que tange a maiores opções de cadastros, para que usuários cadastrem sistemas e alguns parâmetros do sistema.

As tecnologias utilizadas mostraram-se apropriadas para a criação do ambiente, permitindo facilmente conexão com plataformas de dados, reaproveitamento de código e facilidade na criação das telas.

A pesquisa e a implementação permitiram a oportunidade de estudar a fundo tecnologias e aprimorar o conhecimento na linguagem de programação JAVA, SQL Server e principalmente na plataforma Oracle.

4.1 EXTENSÕES

O sistema foi planejado e feito para atender a demanda de uma empresa de software, mas foi construído e tem potencial para atender muitos ramos da área de desenvolvimento de software, tornando sua utilização vantajosa para muitos outros clientes. Desta forma, com mais horas de trabalho sobre a aplicação, será possível torná-la mais abrangente, aumentando a quantidade de parametrizações, sem necessidade de alteração na estrutura da aplicação.

Também seria ideal a implementação de uma classe para compactar arquivos e criptografia dos mesmos, para que se possa ter redução de espaço em disco no armazenamento das mídias de *backup*, além de oferecer também a criptografia dos dados, caso necessário.

Outra necessidade detectada foi uma melhor gestão dos *logs* da aplicação, para que possam ser aplicados filtros, facilitando encontrar informações específicas, por exemplo.

REFERÊNCIAS

ANDERSON, K. Kenneth. **Software Process, Part 1**. Colorado, 2007. Disponível em: <http://www.cs.colorado.edu/~kena/classes/5828/s07/lectures/04/index.html>. Acesso em: 29 maio 2014.

BARTIÉ, Alexandre. **Garantia da qualidade de software**. Rio de Janeiro: Campus, 2002.

BASTOS, Anderson et al. **Base de conhecimento em teste de software**. São Paulo: Martins Fontes, 2007.

BOEHM, Barry; BASILI, Victor R. **"Software Defect Reduction Top 10 List.".** IEEE Computer, Jan 2001.

CUNHA, Simone. **Engenharia de software:** uma abordagem à fase de testes. 2007. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação), Belo Horizonte, UFMG.

CVSGUI. **CVCvsGui**. [S.l.], 2007. Disponível em: <http://cvsgui.sourceforge.net/>. Acesso em: 16 dez. 2014.

FEIGENBAUM, Armand V. **Total quality control, 3ª ed**. Nova York, McGraw-Hill, 1983.

FONTES, Martins; RIOS, Emerson et al. **Base de Conhecimento em Teste de Software**. São Paulo: Martins, 2007.

JONES, Capers; BOEHM, Barry. **Programming Productivity**. Mcgraw-Hill College, 1986.

JURAN, Josef. **Juran on quality improvement workbook**. Nova York: Juran Enterprises, 1981.

JURAN, Josef. GRYNA, Frank. **Juran's Quality Control Handbook Hardcover**. Mcgraw-Hill College, 1998.

MECENAS, Ivan; OLIVEIRA, Vivianne. **Qualidade em software.** Alta Books, 2005.

MOLINARI, Leonardo. **Gerência de Configuração** - Técnicas e Práticas no Desenvolvimento do Software. Florianópolis: Visual Books, 2007.

PRESSMAN, Roger; MAXIM, Bruce. **Software engineering:** a practitioner's approach. Ney York: McGraw-Hill, 2005.

RODRIGUES, Marcos Roberto. **Qualidade de Software**. Belo Horizonte, 2001. (Texto não publicado).

SALDANHA, Sara Alves. **Qualidade de software e a importância dos testes nas empresas de desenvolvimento de software**. São Paulo, 2009. Disponível em: <http://fateczl.edu.br/TCC/2009-2/tcc-58.pdf>. Acesso em: 8 mar. 2014.

SANDERS, Joc. **Software Quality**: A Framework for Success in Software Development and Support. Boston: ACM Press, 1994.

APÊNDICE A – Descrição dos Casos de Uso

Neste Apêndice apresenta-se a descrição dos casos de uso. No Quadro 4 estão os casos de uso e suas respectivas descrições.

Quadro 4 - Casos de Uso

UC01 Cadastra Ambiente Permite ao Técnico cadastrar e vincular ambientes de testes, compostos por mídia de instalação, Banco de dados (SQL Server e Oracle) e domínios Glassfish.
UC02 Gerencia Ambiente Permite ao Administrador excluir, alterar e vincular ambientes de testes, compostos por mídia de instalação, Banco de dados (SQL Server e Oracle) e domínios Glassfish.
UC03 Efetua *Backup* Permite ao Técnico incluir *backup* (de bancos de dados, domínios Glassfish e mídias de instalação) separadamente ou de forma vinculada, gravando a mídia em um local (remoto ou local), rodando os *scripts* vinculados.
UC04 Efetua *Restore* Permite ao Técnico restaurar um *backup*, disponível na lista de *Backups* do servidor da aplicação, de forma completa ou então de partes da mídia previamente gerada por outro usuário (BD, domínio ou mídia) em sua máquina, rodando os *scripts* vinculados.
UC05 Configura Glassfish Permite ao Técnico reconfigurar automaticamente o domínio do servidor Glassfish com os parâmetros necessários para rodar os aplicativos do software a ser restaurado.
UC06 Configura acesso SQL Server Permite ao Técnico configurar o acesso ao banco de dados SQL Server para que permita o sistema executar *scripts* e criar a mídia de *backup*.
UC07 Configura acesso Oracle Permite ao Técnico configurar o acesso ao banco de dados Oracle para que permita o sistema executar *scripts* e criar a mídia de *backup*.
UC08 Configura local de instalação. Permite ao Técnico configurar o local que estão as instalações de sistema no computador local, para que o sistema possa interagir com ele.
UC09 Gerencia Serviços do Windows Permite ao Técnico Gerenciar os serviços dependentes para funcionamento dos softwares (*Middleware*, serviço de instalação e Serviço do Glassfish).
UC10 Efetuar *Login* no sistema. Permite ao Técnico efetuar *login* para ter acesso ao sistema.
UC10 Efetuar *Logof* no sistema. Permite ao Técnico efetuar *Logoff* para encerrar o sistema.
UC11 Configura local de origem das mídias. Permite ao Técnico configurar o local que estão às mídias de b*ackup*
UC12 Configura local de destino das mídias. Permite ao Técnico configurar o local de destino das mídias de b*ackup*
UC13 Configura local de destino das mídias. Permite ao Técnico configurar o local de destino das mídias de *backup*

APÊNDICE B – Descrição do Dicionário de Dados

Este Apêndice apresenta a descrição das tabelas do banco de dados apresentadas na seção de especificação deste trabalho. Os tipos de dados utilizados nos atributos são:

a) *integer: Para variáveis numéricas inteiras;*
b) *string: Para variáveis de texto;*
c) *date: Para variáveis do tipo Data;*
d) *boolean: Para varáveis com valor verdadeiro ou falso.*

No Quadro 5 estão disponíveis as informações de atributo, tipo e descrição da tabela de ambientes

Quadro 5 - Tabela de Ambientes

Entidade: Ambientes		
Descrição: destinada a gravar os ambientes de testes (instalações)		
Atributo	Tipo	Descrição
NOMPRO(PK)	*Varchar(20)*	Nome do produto
LCLPRO	*Varchar(150)*	Local do produto
NOMMID(PK)	*Varchar(20)*	Nome do *middleware*
LCLMID	*Varchar(150)*	Local onde está instalado o *middleware*
NOMDTB(PK)	*Varchar(20)*	Nome do banco de dados
TIPDTB	*Varchar(20)*	Tipo do banco de dados
OBS	*Varchar(150)*	Observações
USUARIO(PK)	*Varchar(30)*	Usuário dono do ambiente
SISTEMA(PK)	*Varchar(20)*	Sistema que está vinculado.

No Quadro 6 estão disponíveis as informações de atributo, tipo e descrição da tabela de *backups.*

Quadro 6 - Tabela de *Backups*

Entidade: Backups		
Descrição: destinada a gravar os ambientes para restaurações		
Atributo	Tipo	Descrição
NOMSIS(PK)	*Varchar(20)*	Nome do sistema
NOMPRO(PK)	*Varchar(20)*	Nome do produto
LOCBKP	*Varchar(150)*	Local onde está a mídia do *Backup*
BLDPRO(PK)	*Varchar(10)*	Build (versão) do *Backup*
NOMDTB(PK)	*Varchar(20)*	Nome do banco
NOMGRD	*Varchar(30)*	Nome do gerador do *Backup*
DATBKP	*Datetime*	Data
OBS	*Varchar(150)*	Observação
DSPRED	*Bit*	Indicador se é público ou privado
NOMMID(PK)	*Varchar(20)*	Nome do *middleware*

No Quadro 7 estão disponíveis as informações de atributo, tipo e descrição da tabela de *log.*

Quadro 7 - Tabela de *Log*

Entidade: Log		
Descrição: Tabela para gravar os logs do sistema		
Atributo	Tipo	Descrição
TIPO	*Varchar(20)*	Tipo do log (criação de ambiente, *Backup*, restauração...)
LOG	*Varchar(500)*	Descrição do Log
GERADOR	*Varchar(30)*	Usuário que gerou o log.
DATAHORA	*Datetime*	Data e hora do registro.

No Quadro 8 estão disponíveis as informações de atributo, tipo e descrição da tabela de *middleware.*

Quadro 8 - Tabela de *Middleware*

Entidade: *Middleware*		
Descrição: Destinada a gravar informações do ambiente *middleware*		
Atributo	Tipo	Descrição
NOMMID(PK)	*Varchar(20)*	Descrição do *middleware*
LOCALMID	*Varchar(150)*	Local onde o *middleware* está instalado
SERMID	*Varchar(50)*	Serviço de *middleware*
SERINSTMID	*Varchar(50)*	Serviço de instalação do *middleware*
USERMID(PK)	*Varchar(30)*	Usuário proprietário do *middleware*
PRODUTO	*Varchar(20)*	Produto que está vinculado

No Quadro 9 estão disponíveis as informações de atributo, tipo e descrição da tabela de parâmetros.

Quadro 9 - Tabela de Parâmetros

Entidade: Parâmetros		
Descrição: Destinada a gravar os parâmetros de sistema por usuário		
Atributo	Tipo	Descrição
PARAMETRO(PK)	*Varchar(50)*	Descrição do parâmetro
VALOR	*Varchar(50)*	Valor do parâmetro
USERPARAM(PK)	*Varchar(30)*	Usuário do parâmetro
SISTEMA(PK)	*Varchar(20)*	Sistema vinculado