UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

SOFTWARE DE GERENCIAMENTO DE LIBERAÇÃO E ATUALIZAÇÃO DE VERSÃO

EDUARDO SIEMANN

BLUMENAU
2012

2012/2-8

EDUARDO SIEMANN

SOFTWARE DE GERENCIAMENTO DE LIBERAÇÃO E ATUALIZAÇÃO DE VERSÃO

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

Prof. Jhony Alceu Pereira , Especialista - Orientador

BLUMENAU
2012

2012/2-8

SOFTWARE DE GERENCIAMENTO DE LIBERAÇÃO E ATUALIZAÇÃO DE VERSÃO

Por

EDUARDO SIEMANN

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

Presidente:	Prof. Jhony Alceu Pereira, Especialista – Orientador, FURB
Membro:	Prof. Everaldo Artur Grahl, Mestre – FURB
Membro:	Prof. Jacques Robert Heckmann, Mestre – FURB

Blumenau, 7 de dezembro de 2012.

Dedico este trabalho a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, que mesmo longe, sempre esteve presente.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador, Jhony Alceu Pereira, 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.

A Persistência é o Caminho do êxito.

Charles Chaplin

RESUMO

Este trabalho consiste no desenvolvimento de um sistema de controle de liberações. O sistema permite a geração dos pacotes de atualização e instalação, controle de testes utilizando formulários cadastrados pelos usuários, além do controle de atualizações e solicitações de licenças. Com o sistema pretende-se melhorar o fluxo de testes e liberações das versões, além de melhorar o processo de atualização dos clientes e a solicitação de novas licenças pelas revendas. O trabalho foi desenvolvido utilizando o Mentawai Framework, Java e PostgreSQL.

Palavras-chave: Gerenciamento de Liberação. Java. PostgreSQL. Mentawai *Framework*.

ABSTRACT

This work is the development of a control system releases. The system allows the generation of the update packages and installation, control tests using forms by registered users, beyond the control of updates and requests for licenses. With the system aims to improve the flow tests and releases of versions, in addition to improving the process of updating the clients and requesting new licenses for resale. The study was conducted using the Mentawai Framework, Java and PostgreSQL.

Key-words: Release Management. Java. PostgreSQL. Mentawai Framework.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

AFV – Automação da Força de Vendas

BSD – Biblioteca de Software Definitivo

DHD – Dispositivo de Hardware Definitivo

ERP – *Enterprise Resource Planning*

ITIL – *Information Technology Infrastructure Library*

PDA – *Personal Digital Assistant*

SQL – *Structured Query Language*

TI – Tecnologia da informação

SUMÁRIO

1 INTRODUÇÃO

Atualmente as empresas visam atender todos os seus clientes de uma forma única, porém utilizando softwares padrões. Utilizando esta forma de trabalho estas empresas sofrem com múltiplas versões, pois ao mesmo tempo em que as novas versões atendem as necessidades de novos clientes, criam uma grande variedade de versões do software atual devido às várias versões em uso pelos clientes.

O principal problema deste grande número de versões são as correções que já foram feitas e ainda não estão nos clientes, podendo gerar reclamações dos clientes devido à utilização de versões desatualizadas.

Segundo Carvalho, Silva e Torres (2003), a empresa deve possuir um plano de continuidade, garantindo manutenções e atualizações e priorizando a comunicação periódica de todas as pessoas envolvidas. Além de garantir introduções de alterações no plano de continuidade, permitindo a recuperação de um desastre caso as novas alterações não estejam de acordo com o negócio da mesma.

Sendo assim para manter o plano de continuidade a empresa deve possuir um sistema que reduza os ruídos na comunicação, reduza a realização de atividades e permita atividades em grupo para solucionar as tarefas, melhorando assim o fluxo de dados entre os envolvidos.

A empresa Progressiva Informática Ltda. na qual baseia se este trabalho, além de controlar liberações e novas versões, deve manter suas revendas atualizadas, mantendo assim uma versão única para todos os seus clientes e revendas, reduzindo os custos de instalação e treinamento.

Antes de utilizar o sistema desenvolvido neste trabalho, ao liberar uma versão a empresa comunicava a todos os seus clientes as novidades da nova versão. Se o mesmo possuísse interesse em atualizar comunica a empresa via e-mail para pedir a atualização de seu sistema. Após um período sem atualização a Progressiva Informática Ltda. entrava em contato com seus clientes perguntando se eles não desejam atualizar para a nova versão, podendo optar por permanecerem na mesma.

O produto deste trabalho controla o processo de liberação de uma versão pelo desenvolvimento e testes usando formulários até a liberação final e atualização automática nos clientes. Além do controle das versões ele possui o controle das solicitações de licenças feitas pelas revendas, que anteriormente eram por e-mail, as quais entravam em filtros de spam, atrasando o processo de licenciamento das revendas e da empresa.

Existiam vários sistemas de controles de testes que poderiam ser utilizados para o problema apresentado pela empresa, porém não seriam integrados aos seus sistemas e base de dados. Outro fator importante é que empresa está abrindo várias revendas no Brasil, gerando assim um grande fluxo de solicitações de licenças.

O trabalho otimiza os processos realizados pelos setores de desenvolvimento, suporte e comercial, reduzindo o tempo de liberações de novas versões do Apogeus Android, além de gerar informações integradas ao sistema atual da empresa, para cada nova versão lançada desde o início do desenvolvimento, liberação para testes, formulários de testes, liberação para clientes e revendas.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho foi desenvolver um sistema para realizar o controle de versões, testes e liberações do Sistema Apogeus Android da empresa Progressiva Informática Ltda. denominado Apogeus Update Manager.

Os objetivos específicos do trabalho proposto são:

a) permitir o setor de desenvolvimento criar arquivos para instalação, atualização e documentação da nova versão;
b) montar formulário de testes para cada versão do sistema;
c) após os testes realizados com sucesso, permitir ao setor comercial definir a data do lançamento da versão ou em caso de uma versão de correção ao setor de suporte liberar a versão.

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 gerenciamento de liberações, gerenciamento de testes, atualização de softwares

automatizados, Progressiva, Apogeus Android, Mentawai *Framework*, sistema atual e trabalhos correlatos.

O terceiro capítulo apresenta o desenvolvimento do sistema Apogeus Update Manager iniciando-se com o levantamento de informações, tendo na seqüência, operacionalidade, a funcionalidade de uma revenda, a funcionalidade de um desenvolvedor e a funcionalidade de um suporte.

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

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os assuntos gerenciamento de liberação, gerenciamento de testes, Progressiva Informática Ltda., Apogeus Android, sistema atual além de trabalhos correlatos.

2.1 GERENCIAMENTO DE LIBERAÇÃO

Atualmente tem-se a *Information Technology Infrastructure Library* (ITIL) que define várias regras a serem aplicadas na infra-estrutura da empresa, operação e manutenção de serviços de Tecnologia da Informação (TI), visando melhorar a qualidade no processo de desenvolvimento tendo foco o cliente.

O Gerenciamento de Liberação é responsável pelo armazenamento de todo o software com a Biblioteca de Software Definitivo (BSD) e hardware com Dispositivo de Hardware Definitivo (DHD) autorizado dentro da organização (DOROW, 2009).

Conforme o fluxo de desenvolvimento é acelerado, as empresas sofrem problemas freqüentes na liberação de versões a todos seus clientes ou usuários. Alguns exemplos são os softwares de antivírus, pois depois de realizarem testes, liberando assim uma versão nova, a empresa deve replicar os dados para todos os servidores disponíveis, além de criar pacotes de atualizações a todos os usuários atuais.

Com o Gerenciamento de Liberação tudo isto teria que ser previsto e testado antes de ser colocado em produção e ainda passar pela avaliação do comitê de mudanças (DOROW, 2009).

Os tipos e freqüências das liberações irão influenciar a política de riscos. Necessidades de liberações diárias podem requisitar planos de testes direcionados para o reuso ou testes automatizados. Necessidades de liberações com um prazo de folga permitem um maior controle de testes e validações (FREITAS, 2010, p. 243).

Neste trabalho as liberações são completas, ou seja, ao liberar uma versão todos os componentes do sistema serão atualizados, pois além da atualização completa existem a delta, que é composta por apenas itens de configuração os quais foram modificados desde a última liberação, a empacotada, na qual as liberações independentes são unificadas em um único

pacote de atualização e a de emergência na qual é realizado um ajuste crítico e liberado não respeitando o ciclo natural de liberação.

2.2 PROGRESSIVA INFORMÁTICA LTDA.

A Progressiva é uma empresa sediada em Blumenau (SC), especializada na área da computação móvel, desenvolvendo produtos para informatizar a força de vendas de distribuidores, atacadistas e indústrias em todo território nacional. Em 1995 desenvolveu um dos primeiros sistemas de automação a nível nacional, o WINNER´S para os equipamentos HP200LX. Durante o passar dos anos evoluiu sua carteira de clientes para o produto APOGEUS utilizando equipamentos com os Sistemas Operacionais PalmOS, Windows Mobile e Android.

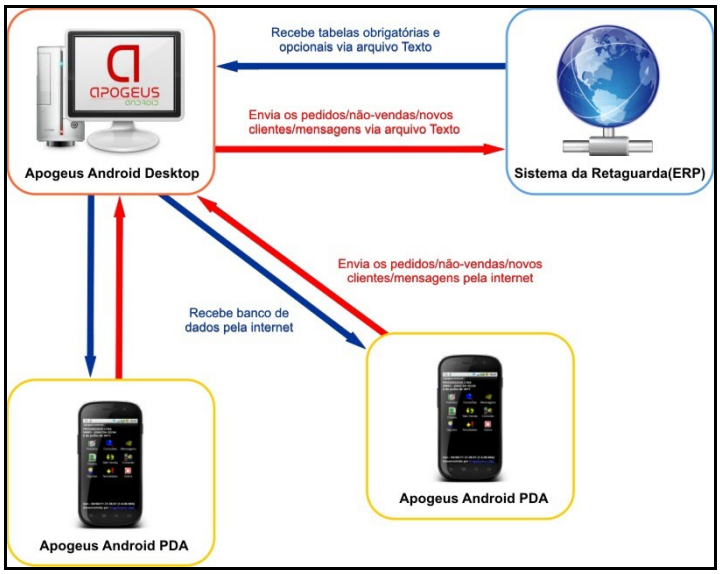
Em junho de 2011 iniciou a comercialização do novo produto para Automação da Força de Vendas (AFV), Apogeus Android, escrito em Java, disponibilizando recursos inovadores como catálogo de produtos com fotos e vídeos, pesquisa por voz de clientes e produtos, por código de barras 2D, integração do endereço dos clientes com Google Maps (auxiliando o vendedor a localizar o cliente), configurações pessoais de listas, ordenações dos dados exibidos nas listas. Atualmente conta com mais de 150 clientes espalhados em todo o território nacional, tornando-se uma das principais empresas desenvolvedoras de soluções para computação móvel (PROGRESSIVA, 2012a).

2.3 APOGEUS ANDROID

Apogeus Android é um sistema atualmente comercializado pela empresa Progressiva. Foi desenvolvido para a automação da força de vendas. O sistema permite emitir pedidos off-line, seguindo as regras de negócio da empresa.

Na Figura 1 é apresentado o fluxo dos dados, disponibilizado no site da Progressiva, sobre o funcionamento do Apogeus Android. Conforme a Progressiva o mesmo possui três partes principais, a retaguarda podendo ser um sistema *Enterprise Resource Planning* (ERP),

Apogeus Android Desktop (executado no servidor do cliente) e Apogeus Android *Personal Digital Assistant* (PDA), utilizado pelo vendedor. A retaguarda gera e importa os dados para o sistema Apogeus Desktop, que por sua vez disponibiliza e importa as informações de todos os PDA's dos vendedores.



Fonte: Progressiva (2012b).

Figura 1 – Fluxo de dados do Apogeus Android

2.4 MENTAWAI *FRAMEWORK*

O *framework* utilizado neste trabalho foi o Mentawai Framework, já utilizado pela empresa Progressiva Informática Ltda. deste 2007 por sua fácil utilização e depuração de erros. Além do conhecimento já adquirido durante os anos sobre o framework pelos desenvolvedores da empresa, que ao final deste trabalho irão incorporar ao sistema da intranet da Progressiva.

> Mentawai é um framework web full-stack, action-based, MVC e open-source em Java criado em Junho de 2005. Desde então ele tem sido fiel à sua filosofia que é oferecer uma solução simples de alto nível para o desenvolvimento de aplicações web sem XML e Annotations, usando uma configuração programática centralizada para preparar e ligar todas as partes de uma aplicação web. Ele suporta inteiramente o princípio KISS utilizando altos níveis de abstração para exterminar qualquer complexidade. Se você não for capaz de entender ou utilizar qualquer funcionalidade do framework a culpa é do framework e não sua (MENTAFRAMEWORK, 2012).

Conforme trecho acima, o Mentawai é baseado em *action-based*, sendo assim para todas as chamadas ao servidor que necessitam de algum processamento são criadas classes chamadas de a*ctions* que por sua vez executam um método da classe definida no *ApplicationManager*, conforme a Figura 2.

```
action("resultadoteste/resultadoteste", ActionResultadoTeste.class, "downloadInstall")
        .on(SUCCESS, new StreamConsequence("application/zip"))
        .on(ERROR, fwd("/error.jsp"));
```

Figura 2 – Código fonte do ApplicationManager definindo uma *action*

Na Figura 2 é definida a action para chamar o método downloadInstall da classe ActionResultadoTeste. Após definir no *ApplicationManager*, basta criar o método na classe especificada como a Figura 3.

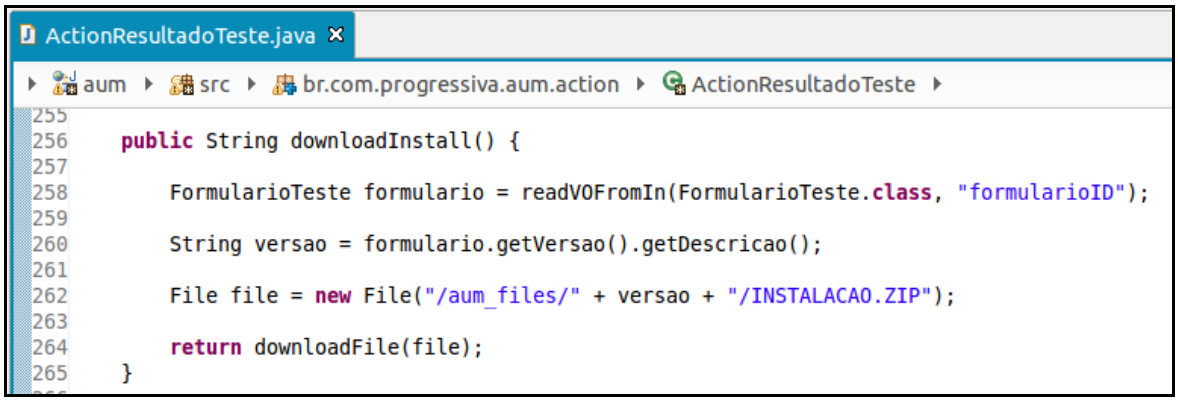


Figura 3 – Código fonte da classe ActionResultadoTeste

2.5 SISTEMA ATUAL

A empresa utiliza somente um *Workflow* desenvolvido por si mesma para controlar o desenvolvimento, testes e liberações. Esse utiliza o conceito de tarefas para cada solicitação, alteração e testes do sistema.

Ao planejar uma nova versão todas as correções, melhorias e novas funcionalidades são lançadas em tarefas separadas, as quais são passadas ao desenvolvimento que define um prazo a serem entregues.

Ao terminar todas as tarefas de desenvolvimento da versão atual, o desenvolvimento cria pacotes de atualização, além de passar as tarefas para o setor de suporte realizar testes de atualização com dados fictícios da empresa.

O setor de suporte utiliza um *checklist* para realizar os testes das funcionalidades padrões do sistema, porém elas não estão considerando as novas funcionalidades da versão a ser testada. Somente estão descritos os funcionamentos das novas funcionalidades nas tarefas passadas pelo *Workflow* da empresa. Sendo assim o controle realizado pelo *checklist* é feito por papeis e das novas funcionalidades é realizado no *Workflow* da empresa.

Uma vez que a versão for liberada pelo suporte o desenvolvimento libera os pacotes de instalação e atualização para os repositórios da empresa.

A cada versão liberada são passadas a todos os clientes as novidades da versão. Caso o cliente desejar utilizar uma nova funcionalidade, serão passadas as informações de como ele deve proceder para a utilização. Além disso, setor de suporte testa a nova versão com os dados do cliente para avaliar se o mesmo pode atualizá-la.

Caso o cliente não entre em contado para reivindicar a atualização, o suporte da empresa entra em contato com o cliente para sugerir uma atualização, a fim de manter a maioria dos clientes na última versão do *software*.

Na atualização atual é utilizado um arquivo compactado, que é descompactado na pasta do sistema Apogeus Android. Ao descompactá-lo o suporte utiliza um arquivo script contido nos arquivos descompactados para atualizar os arquivos do sistema.

Na parte das revendas da Progressiva, o processo de solicitação de novas licenças é feito por e-mails, onde podem ocorrer problemas devido aos filtros de SPAMS, os quais filtram conteúdos com links para possíveis sites duvidosos, ocasionando a interrupção da comunicação entre revenda e a empresa.

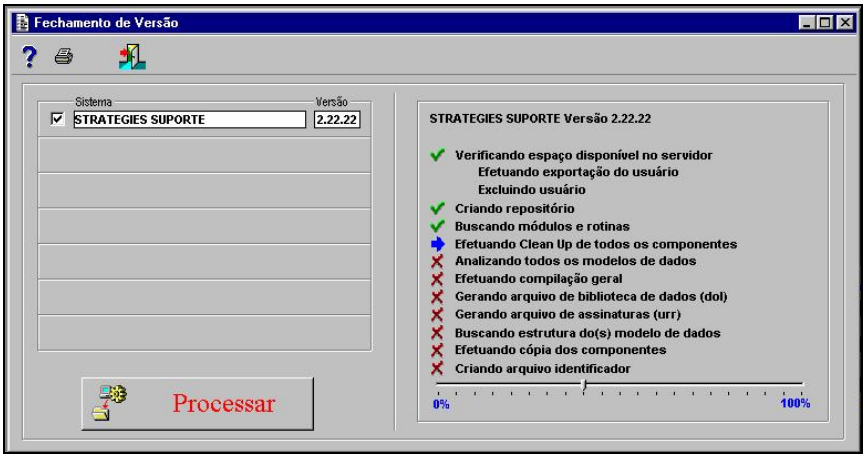
Após a análise dos processos atuais, foram verificados quais pontos poderiam ser melhorados na parte de liberação do desenvolvimento, no testes do suporte, e a criação do módulo de solicitações de licenças pelas revendas.

2.6 TRABALHOS CORRELATOS

Podem-se citar como correlatos as monografias realizadas pelos alunos Denis Cezar Metzner no qual fez um protótipo para atualização automatizada, Jeferson Roberto Samagaia com o controle de solicitações de alterações a serem feitas no sistema usando *Shell* UNIX e Airison Ambrosi com seu sistema de controle de atualizações usando Delphi. Todos foram feitos para conclusão do curso de Ciências da Computação na Universidade Regional de Blumenau.

Metzner (2001) propôs um protótipo para automatizar o fechamento e atualização das versões dos sistemas da empresa. No protótipo ele automatizou os principais processos da atualização. Porém, devido às limitações encontradas nas ferramentas utilizadas Uniface 7.0, algumas etapas deveriam ser executadas ainda manualmente pela pessoa responsável pela atualização. Além da linguagem Uniface ele utilizou banco de dados Oracle 8I.

Na Figura 4 é apresentada a tela desenvolvida por Metzner (2001), onde o usuário gera o pacote final para fechar a versão a ser liberada para os clientes. Reduzindo em muito o tempo de liberação de uma atualização, evitando em muito os trabalhos manuais.

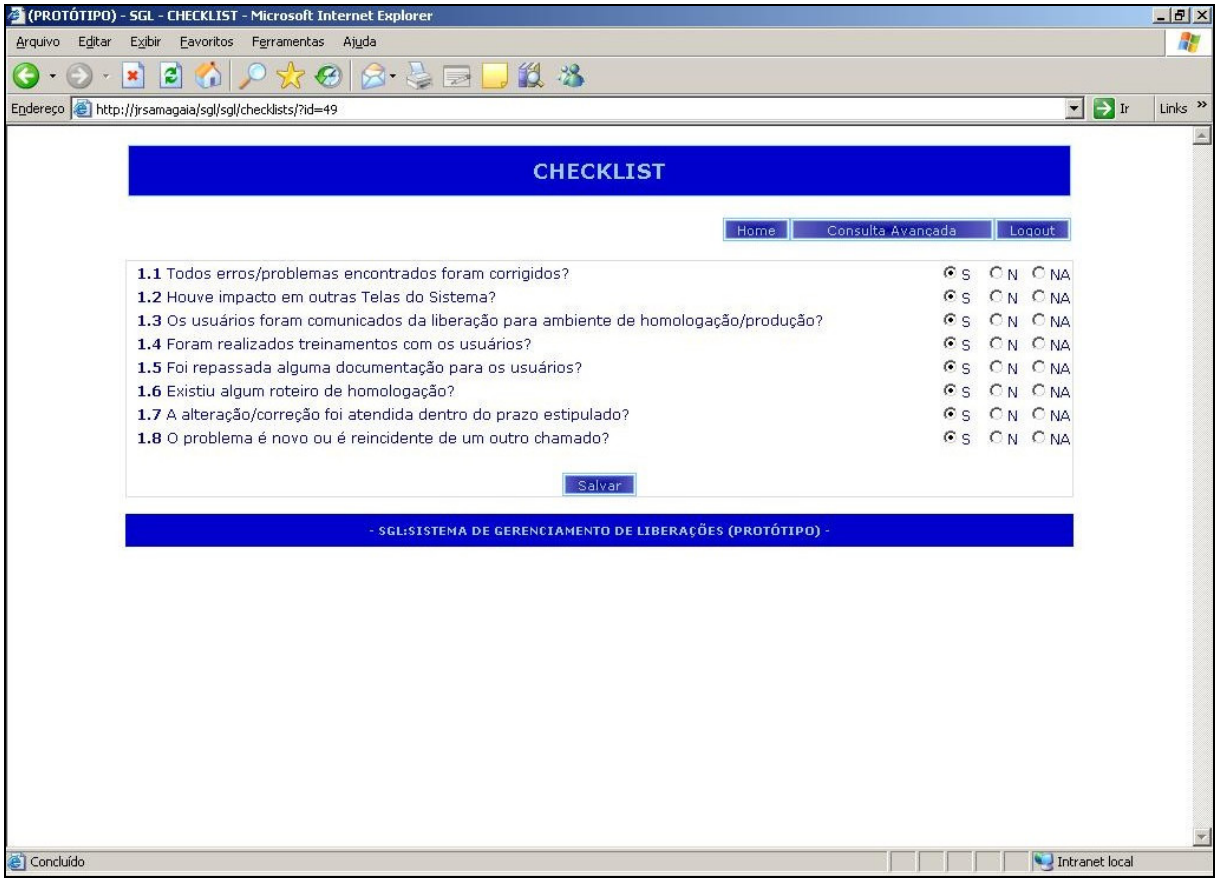


Fonte: Metzner (2001).

Figura 4 – Tela de fechamento da versão

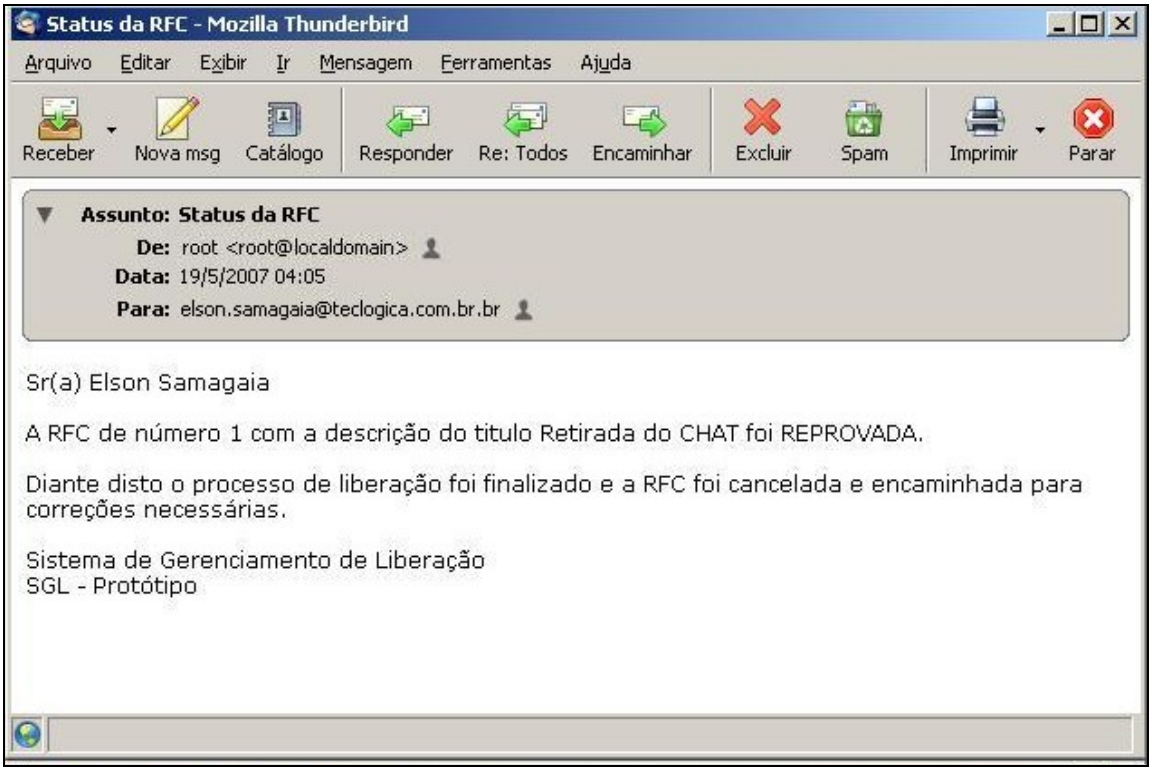
Samagaia (2007) desenvolveu um sistema para *Shell* UNIX para controlar o fluxo de liberações a serem homologados pelos clientes, melhorando o fluxo de atualização dos sistemas desenvolvidos pala empresa. Utilizou MySQL para armazenar os dados e PHP como linguagem principal de desenvolvimento do sistema web, além do Shell UNIX para executar comandos nativos no sistema.

O sistema de Samagaia (2007) permitia a criação de solicitações por um ambiente *Web*, conforme descrito anteriormente em PHP, o qual mostra quantas solicitações foram aprovadas, rejeitadas e seus motivos, retornando via e-mail ao solicitante um resumo do que ocorreu na sua solicitação, chamada no trabalho de *Request for Change* (RFC). A Figura 5 demonstra o *checklist* realizado no final de cada RFC, em seguida será enviado um e-mail para todos os envolvidos no RFC, sendo a Figura 6 se a RFC foi reprovada e Figura 7 se foi aprovada.



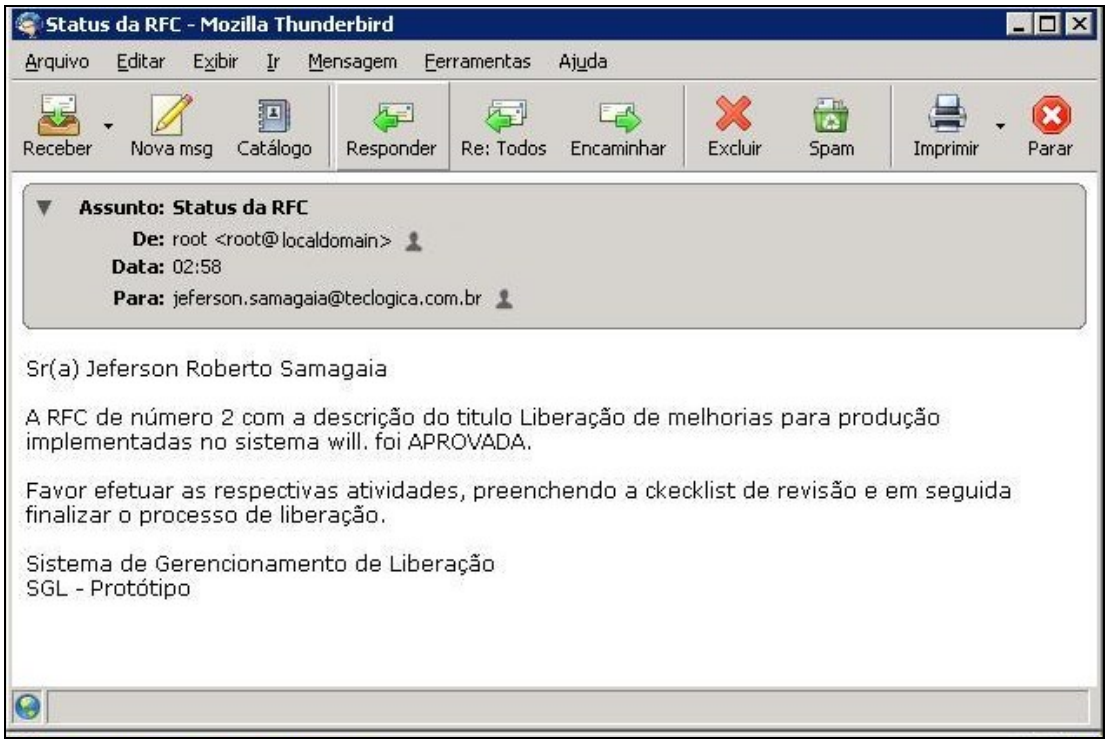
Fonte: Samagaia (2007).

Figura 5 – Tela de *checklist* da solicitação



Fonte: Samagaia (2007).

Figura 6 – E-mail recebido após RFC ser reprovada



Fonte: Samagaia (2007).

Figura 7 – E-mail recebido após a RFC ser aprovada

Ambrosi (2004) criou um protótipo de aplicativo de atualização automática de versões, que utiliza a Internet para atualizar, utilizando algoritmos *hashing* para garantir integridade dos arquivos transmitidos. Utilizou as tecnologias Delphi como linguagem, *Extensible Markup Language* (XML) para armazenar os dados do arquivo a ser controlado e *File Transfer Protocol* (FTP) para a transmissão dos arquivos. Na Figura 8 está demonstrada a tela de cadastros de um programa no qual o usuário informa todos os arquivos a serem enviados para uma atualização dos clientes.

Fonte: Samagaia (2007).

Figura 8 – Tela de cadastro para geração da atualização

3 DESENVOLVIMENTO DO SISTEMA

Neste capitulo estão descritos os dados técnicos sobre o sistema desenvolvido, bem como os requisitos funcionais, não funcionais, principais diagramas, fluxograma, diagrama de entidade relacionamento, técnicas e ferramentas utilizadas, operacionalidade e as funcionalidades de cada usuário.

3.1 LEVANTAMENTO DE INFORMAÇÕES

O levantamento de informações foi realizado com os envolvidos na empresa, envolvendo as etapas desde o desenvolvimento, passando pela liberação e atualização da versão. Tal levantamento, após coletados os dados, foi apresentado o sistema considerando os requisitos descritos neste capítulo.

Foram definidos 3 módulos distintos, o controle de liberação, o controle de atualização e o controle de solicitação de licenças. Onde o controle de liberação trataria a liberação das versões desde o inicio no desenvolvimento até a liberação da versão para os repositórios da empresa. O controle de atualização permite ao cliente solicitar atualizações das versões liberadas pelo controle de liberação de versão. E como último módulo foi definido o de solicitação de licenças, onde as revendas podem solicitar novas ou alterações de licenças.

O Quadro 1 apresenta os requisitos funcionais do módulo de controle de liberação e sua rastreabilidade, ou seja, a vinculação com o(s) caso(s) de uso associado(s).

Requisitos Funcionais do módulo de controle de liberação	**Caso de Uso**
RF01: O sistema deverá permitir ao desenvolvimento criar pacotes de instalação e atualização.	UC01.01
RF02: O sistema deverá permitir o cadastramento de formulário de testes pelo desenvolvimento para a versão liberada pelo desenvolvimento.	UC01.02
RF03: O sistema deverá permitir ao usuário do suporte seguir o formulário para realizar os testes, além de informar os resultados.	UC01.03
RF04: O sistema deverá permitir que ao término dos testes pelo usuário do	UC01.03

suporte, sejam enviadas tarefas para o setor de desenvolvimento com os problemas encontrados.	
RF05: O sistema deverá mudar o status da versão para “pré-liberada” ao terminar os testes com sucesso seguindo formulário de teste.	UC01.03
RF06: O sistema deverá permitir ao gerente comercial ou gerente de suporte liberar versões definidas como “pré-liberada”.	UC01.04

Quadro 1 – Requisitos funcionais do controle de liberação

O Quadro 2 apresenta os requisitos funcionais do controle de atualização e sua rastreabilidade, ou seja, vinculação com o(s) caso(s) de uso associado(s).

Requisitos Funcionais do módulo de controle de atualização	**Caso de Uso**
RF07: O sistema deverá permitir a consulta de uma listagem com todos os clientes e suas respectivas versões.	UC02.01
RF08: O sistema deverá permitir o cliente solicitar a atualização de versão. Enviando uma cópia dos dados para análise do suporte, para que assim possa liberar a atualização.	UC02.06
RF09: O sistema deverá permitir ao suporte listar todas as solicitações de atualização feitas pelos clientes.	UC02.03
RF10: O sistema deverá permitir ao suporte baixar os dados do cliente para realizar testes antes de liberar a atualização.	UC02.04
RF11: O sistema deverá permitir liberar a atualização automática ao cliente após a análise dos dados enviados na solicitação.	UC02.02
RF12: O sistema deverá realizar o *backup* dos dados ao iniciar a atualização.	UC02.02
RF13: O sistema deverá permitir o retorno à versão anterior, caso seja necessário. O mesmo poderá ser feito pelo suporte ou cliente.	UC02.05

Quadro 2 – Requisitos funcionais do controle de atualização

O Quadro 3 apresenta os requisitos funcionais do controle de solicitação de licenças e sua rastreabilidade, ou seja, vinculação com o(s) caso(s) de uso associado(s).

Requisitos Funcionais do módulo de controle de solicitação de licenças	Caso de Uso
RF14: O sistema deverá permitir às revendas a solicitação de novas licenças.	UC03.01
RF15: O sistema deverá permitir a solicitação de alteração em uma licença já existente para alterar o número de usuários.	UC03.02
RF16: O sistema deverá permitir à revenda consultar as solicitações de novas ou alterações de licenças.	UC03.03
RF17: O sistema deverá permitir a consulta das licenças disponíveis.	UC03.04

Quadro 3 – Requisitos funcionais do controle de solicitação de licenças

Além dos requisitos funcionais o sistema atende alguns requisitos não-funcionais descritos no Quadro 4.

Requisitos Não Funcionais
RNF01: O sistema deverá ser integrado ao sistema da atual Intranet da empresa.
RNF02: O sistema deverá utilizar a ferramenta pgAdmin para acessar o banco de dados.
RNF03: O sistema irá utilizar o banco de dados PostgreSQL 8.4.
RNF04: O ambiente de desenvolvimento que será utilizado deve ser o Eclipse.
RNF05: O sistema deverá utilizar mecanismos de *hashing* como o MD5 para garantir integridade dos arquivos transmitidos aos clientes.
RNF06: Os arquivos transmitidos devem utilizar o tipo de compactação ZIP.
RNF07: Deve ser utilizado o cadastro de usuários da base de dados atual da empresa.
RNF08: O servidor web utilizado deve ser Tomcat 5.5.

Quadro 4 – Requisitos não funcionais

3.2 ESPECIFICAÇÃO

Esta seção tem como objetivo demonstrar o problema utilizando diagramas, os quais estão relacionados diretamente aos requisitos funcionais. Os diagramas foram criados utilizando ferramenta Astah. Já o modelo de entidade relacionamento (MER) foi gerado pelo MySQL Workbench.

3.2.1 Diagrama de casos de uso

Esta subseção apresenta os diagramas de casos de uso dos módulos de controle de licença, controle de atualização e controle de solicitação de licenças, sendo que os detalhamentos dos principais casos de uso estão disponíveis no Apêndice A.

Na Figura 9 tem-se o com os casos de uso que mostram a criação de pacotes, cadastramento de formulário de testes, realização dos testes além da liberação da versão.

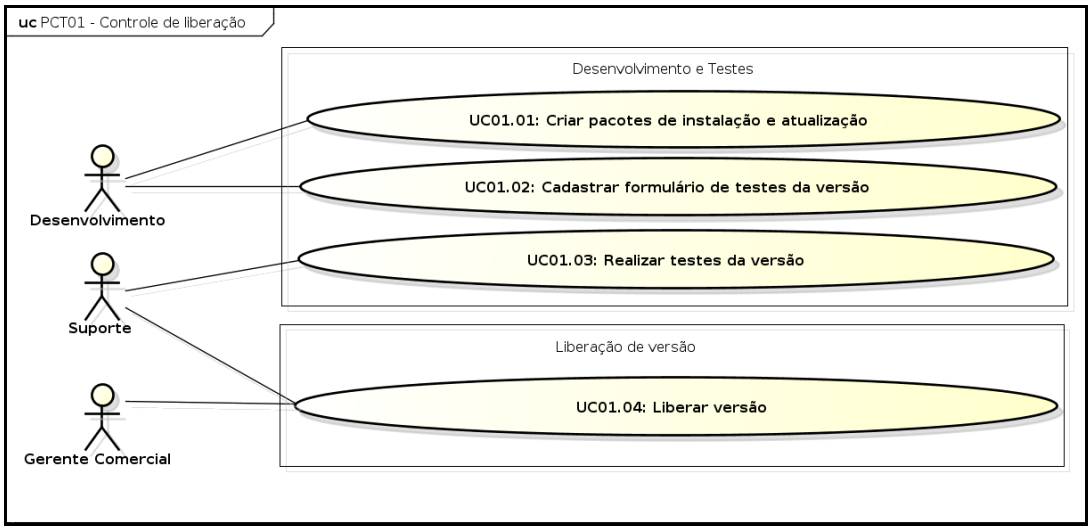


Figura 9 – Caso de uso do módulo de controle de liberação

Na Figura 10 tem-se como será o controle de atualização de versão do sistema proposto.

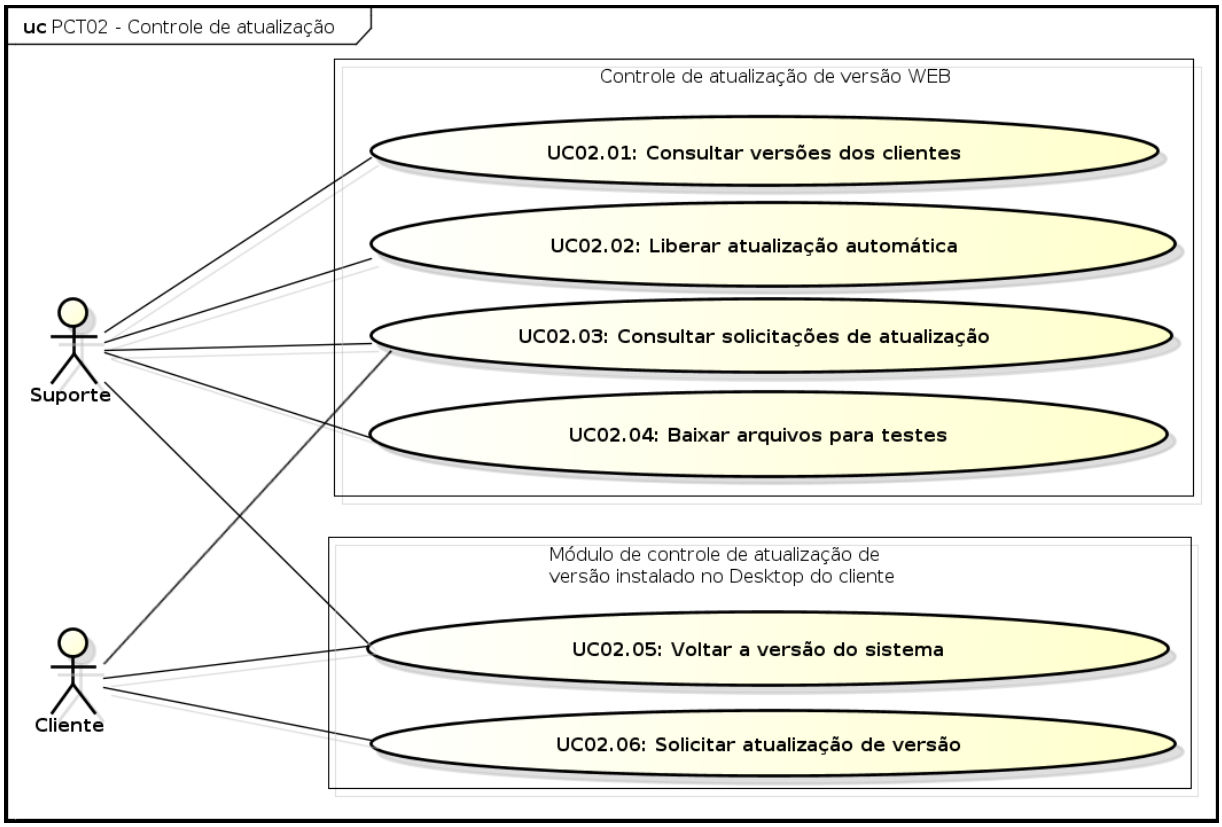


Figura 10 – Caso de uso do módulo de controle de atualização

Na Figura 11 estão descritos os casos de uso que controlam a solicitação de novas licenças e alteração de licenças existentes.

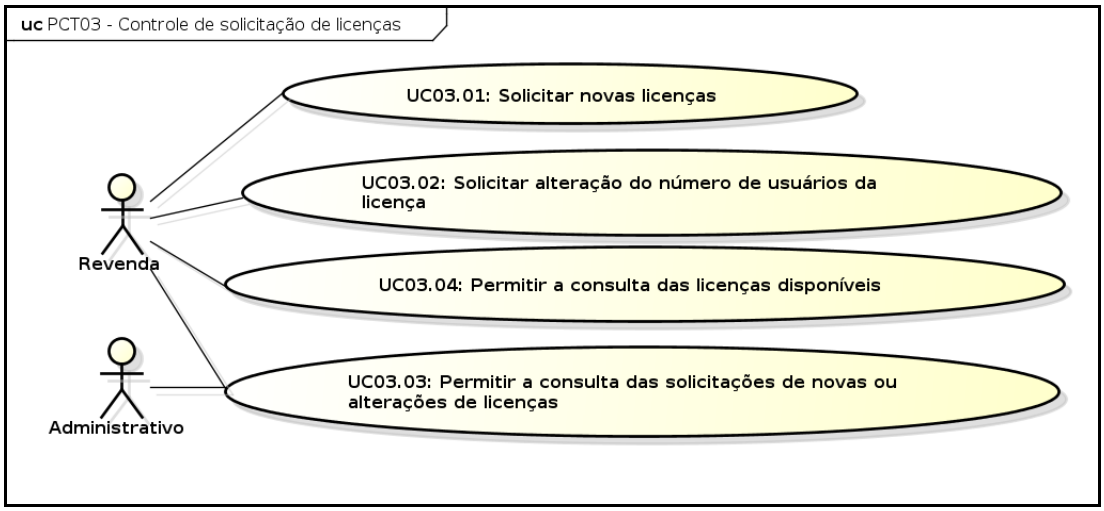


Figura 11 – Caso de uso do módulo de controle de solicitação de licenças

3.2.2 Fluxograma

O fluxograma contido na Figura 12 descreve o processo de liberação de uma nova versão desde o desenvolvimento, testes, correção de erros, liberação, solicitação de atualização e atualização automática do sistema.

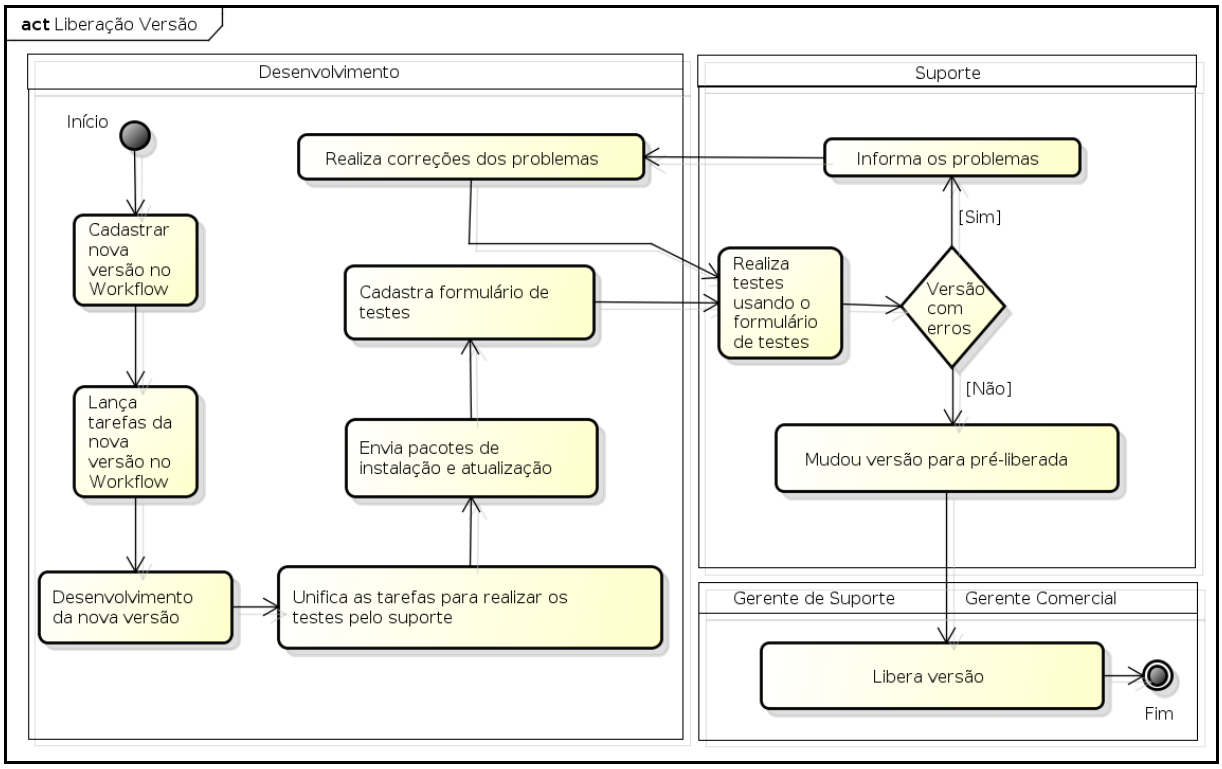


Figura 12 – Fluxograma de liberação de versão

O fluxograma inicia-se com o cadastramento da versão no sistema de *Workflow* da empresa. Em seguida o desenvolvimento lança as tarefas das novas funcionalidades. Após o término do desenvolvimento, as tarefas são unificadas em uma única tarefa para realizar os testes pelo departamento de suporte, além de enviar os pacotes de instalação e atualização da versão para o servidor. Usando a tarefa de testes, o desenvolvimento cria um formulário de testes da versão atual, permitindo assim o controle dos testes pelo Apogeus Update Manager.

Quando o suporte recebe a tarefa de teste da versão, inicia os testes baseados no formulário de testes cadastrado pelo desenvolvimento, permitindo assim um controle dos testes que estão sendo feitos. Mesmo que o usuário do suporte não terminá-lo o sistema permite a ele salvar os testes realizados até o momento.

Caso o suporte encontre algum problema na versão testada, o mesmo deve informar

onde foi encontrado o problema, retornando assim a versão para o desenvolvimento. Caso o problema não esteja relacionado a algum teste que o desenvolvimento determinou, o usuário do suporte informa o campo "Outros problemas" que será demonstrado no decorrer do trabalho. Uma vez que o suporte grava o teste com todas as perguntas respondidas e o mesmo possuir problemas, o sistema envia uma tarefa para o desenvolvimento com os problemas encontrados. Caso não seja encontrado algum problema, a versão muda o seu estado para pré-liberada.

As versões pré-liberadas ficam aguardando o usuário gerente comercial ou gerente de suporte para a liberação usando a tela de liberação final de versão.

3.2.3 Diagrama de Entidade Relacionamento

A Figura 13 demonstra as tabelas do Diagrama de Entidade Relacionamento, no diagrama estão as tabelas criadas para o funcionamento do Apogeus Update Manager em amarelo, já em azul as são utilizadas porém os dados são gerados por outros sistemas da empresa Progressiva. Segue uma breve descrição das entidades listadas na Figura 13:

a) `Um_SolicitacaoLicenca`: entidade responsável pelas solicitações de licenças realizadas pelas revendas;
b) `Um_QuantidadeSolicitacaoLicenca`: entidade que pertence a uma listagem de quantidade relacionadas a uma solicitação de licença;
c) `Um_FormularioTeste`: entidade responsável pelos formulários de testes cadastrados pelo desenvolvimento;
d) `Um_PerguntaFormularioTeste`: entidade que pertence a uma listagem de perguntas relacionadas a um formulário de testes;
e) `Um_ResultadoTeste`: entidade responsável pelo resultado das respostas fornecidas pelo formulário de testes;
f) `Um_RespostaResultadoTeste`: entidade que pertence a uma listagem de respostas relacionadas a um resultado de testes de um formulário de testes;
g) `Um_SolicitacaoAtualizacaoVersao`: entidade que permite a um cliente solicitar pela sua licença do desktop uma atualização do sistema.
h) `Um_SolicitacaoAtualizacao_Novidade`: entidade que relaciona a solicitação de atualização as novidades selecionadas.

O dicionário de dados está descrito no Apêndice B.

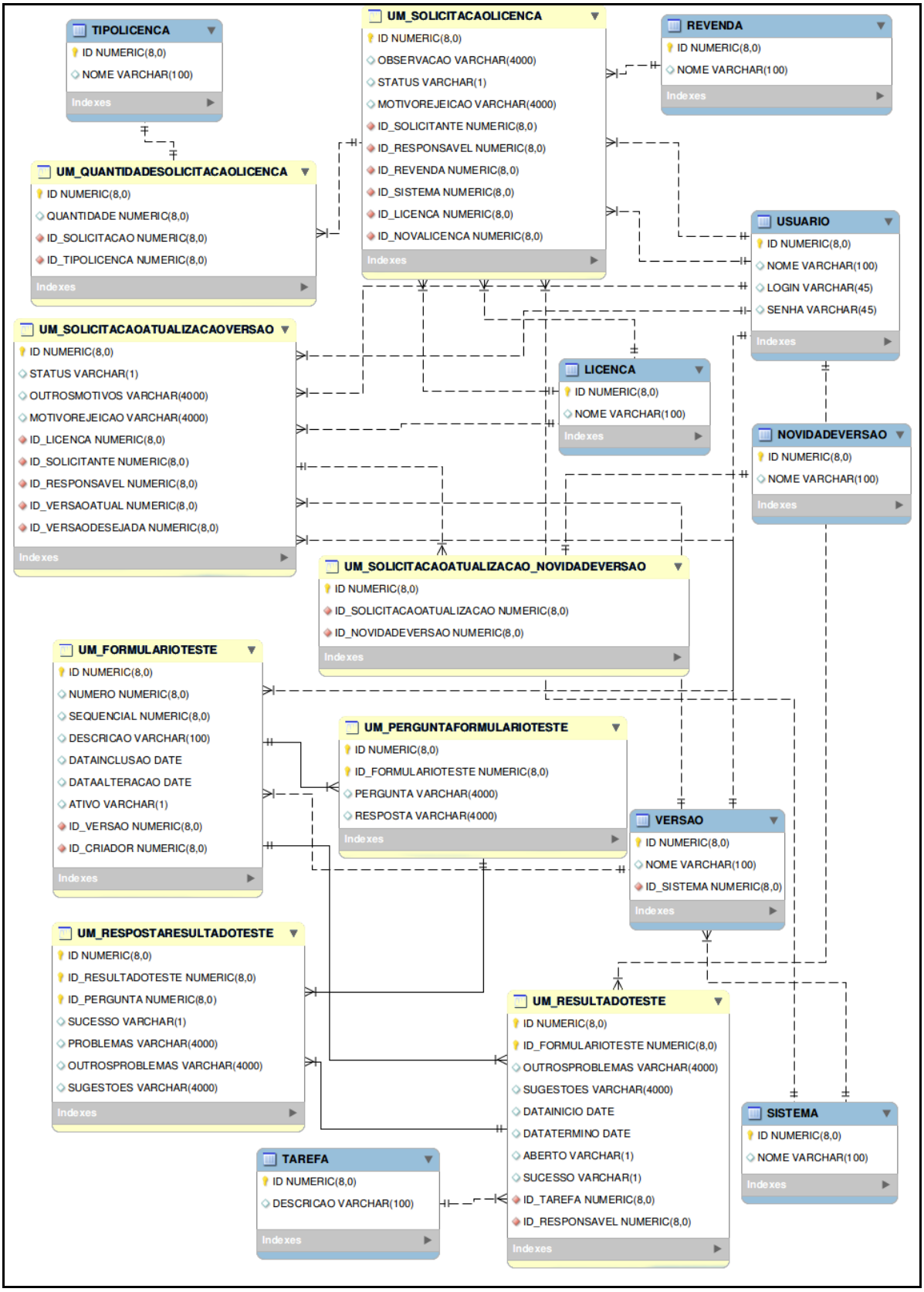


Figura 13 – Modelo de Entidade Relacionamento

3.3 IMPLEMENTAÇÃO

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

3.3.1 Técnicas e ferramentas utilizadas

O sistema foi desenvolvido sobre o sistema operacional Ubuntu 12.04, pois é o sistema utilizado pela empresa no setor de desenvolvimento, além da ferramenta Eclipse Indigo na qual foram desenvolvidas as partes de Java, SQL, JSP, JavaScript e HTML.

Como foi descrito nos requisitos não-funcionais, o banco de dados utilizado foi o PostgreSQL 8.4, já utilizado pela empresa. Para acesso ao banco de dados foi utilizado o pgAdmin III. Para acesso pelo sistema escrito em Java foi utilizado o *Driver Java Database Connectivity* fornecido pela comunidade do PostgreSQL.

No desenvolvimento em Java foi utilizado o *framework* brasileiro Mentawai *Framework*, devido à sua grande utilização pela empresa. Para rodar o sistema foi utilizado o Tomcat 5.5, pois todos os serviços da empresa estão rodando sobre várias instâncias de desta versão e ele foi especificado nos requisitos não-funcionais.

Na parte do servidor, chamada de Apogeus Update Manager, foram criados 2 novos projetos em Java usando *Swing*. O primeiro projeto é chamado de Apogeus Package Manager (APM), responsável pela criação dos pacotes pelo desenvolvimento para as versões. O segundo projeto é o Apogeus Update Manager Client (AUMC), responsável pela solicitação de atualização, *backup* da versão atual e atualizações automáticas.

Na Figura 14 demonstra a interação entre os módulos de atualização do Apogeus Update Manager (AUM) no servidor da Progressiva, com o módulo de atualização Apogeus Update Manager Cliente (AUMC), instalado no cliente.

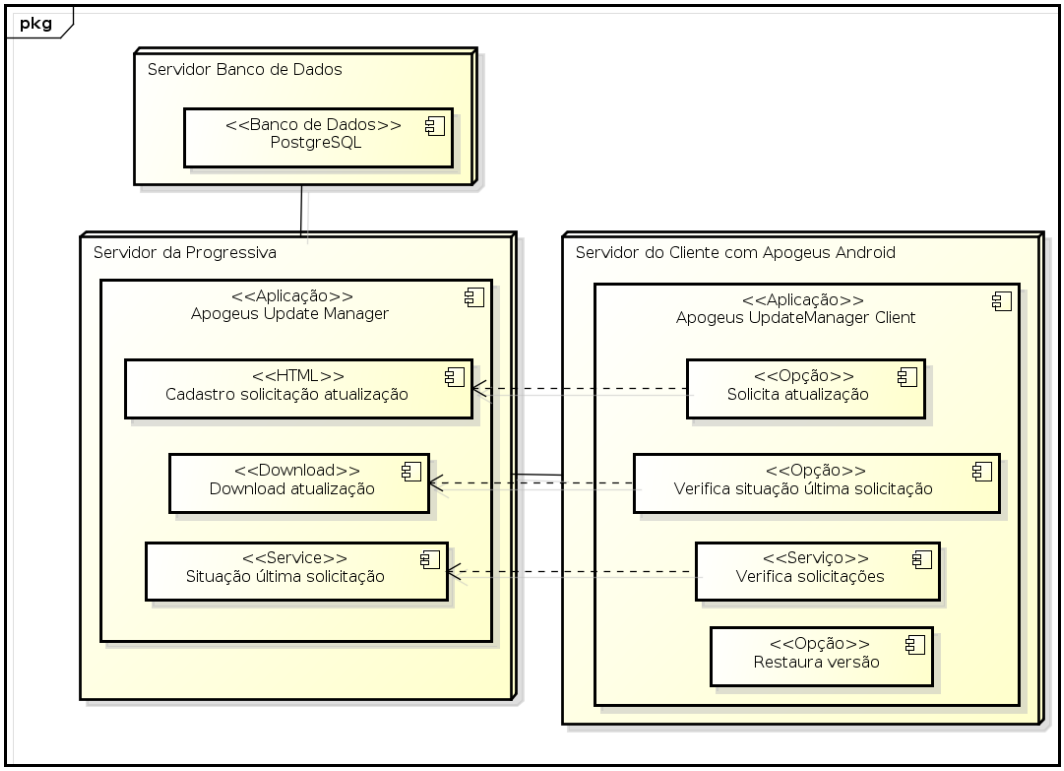


Figura 14 – Diagrama de implantação do módulo de atualização

3.3.2 Operacionalidade da implementação

A operacionalidade será separada em subseções, onde primeiramente será mostrada visão de um usuário de uma revenda, em seguida de um desenvolvedor, suporte e cliente.

3.3.2.1 Funcionalidades para revenda

Inicialmente o usuário loga no sistema conforme a Figura 15 com o seu usuário passado pela Progressiva a revenda.

Figura 15 – Tela de *login* da revenda

Após estar logado, o sistema apresenta a teça conforme mostra a Figura 16, onde o usuário terá acesso à três opções distintas: solicitar nova/alteração licença, consultar solicitações, licenças disponíveis, além do acesso à opção de solicitações de atualizações, pelo menu superior.

Figura 16 – Menu de opção da revenda

Caso o usuário da revenda opte por criar uma solicitação para nova licença ou alterar licença, será apresentada a tela da Figura 17. Nesta tela a revenda pode especificar na observação o motivo da nova licença ou outros dados, pois esta mesma observação irá servir pare preencher a observação da licença.

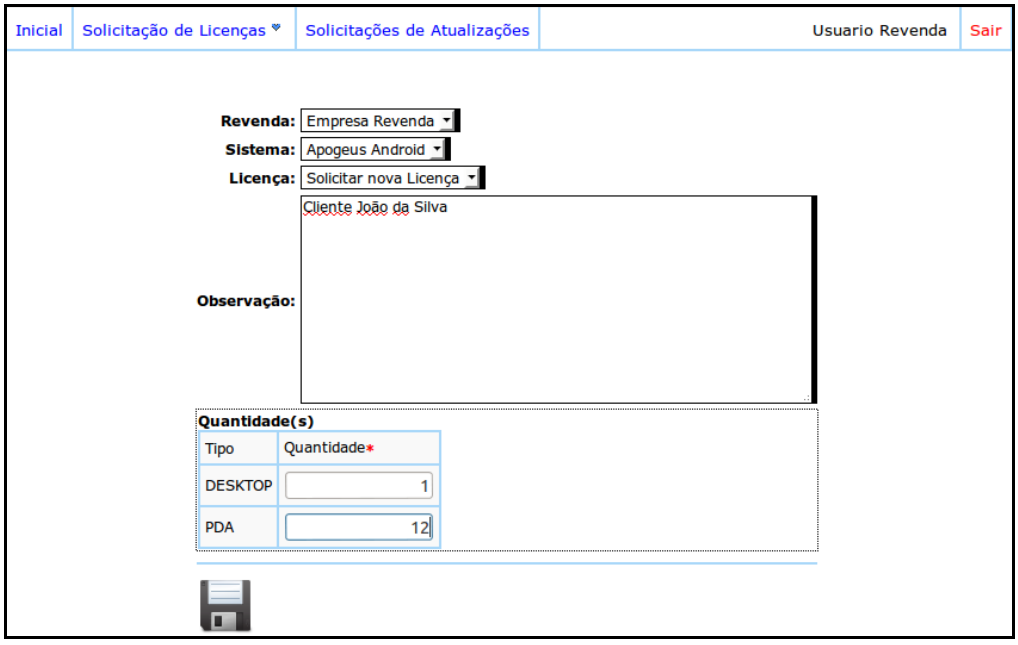


Figura 17 – Tela de solicitação de nova/alteração licença

Após preencher os dados da solicitação e clicar em salvar, o sistema apresenta a consulta de solicitações de licenças, conforme a Figura 18.

Figura 18 – Tela de consulta de solicitações de nova/alteração licença

Nas tabelas mostradas no sistema, possuem as opções de exportar para os tipos *Comma-separated values* (CSV), do programa Microsoft Excel ou *Extensible Markup Language* (XML), assim aos usuário pode manipular os dados em programas externos ao sistema.

A terceira opção do menu da revenda, chamada licenças disponíveis, redireciona o usuário para o gerenciamento de licenças da Progressiva, que não é feito por este sistema.

A quarta opção da revenda é visualizar suas solicitações de atualizações. Esta opção será descrita mais adiante nas funcionalidades de um cliente, pois a revenda pode ser um cliente se usar um servidor de testes na empresa com o Apogeus Android.

3.3.2.2 Funcionalidades para um cliente

O cliente irá acessar o Apogeus Update Manager somente para solicitar atualizações do seu sistema instalado no servidor. Para realizar isto o cliente usará o Apogeus Update Manager Client, o qual apresenta um ícone na barra de notificações e possui um menu quando clicado com o botão direito do mouse, conforme demonstrado na Figura 19.

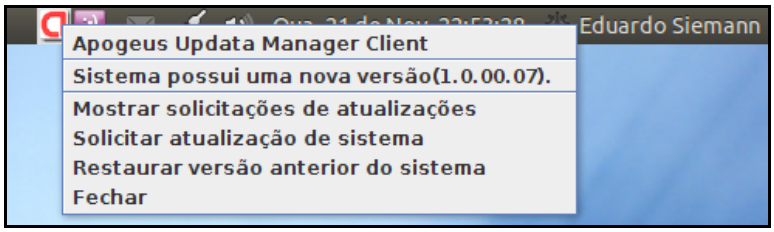


Figura 19 – Tela do Apogeus Update Manager Client mostrando suas opções

Nesta tela o sistema mostra na situação do sistema que existe uma atualização sua. Caso o cliente opte pela atualização, o sistema automaticamente compactará os arquivos textos correspondentes e irá enviá-los para o servidor, abrindo assim uma nova solicitação, conforme demonstrado na Figura 20.

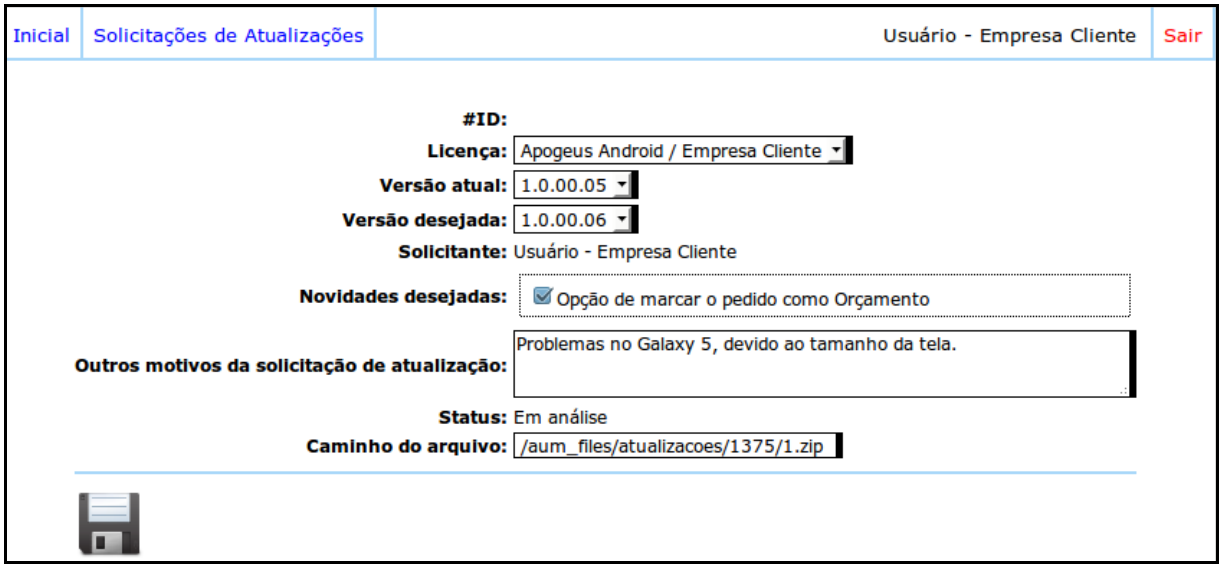


Figura 20 – Tela de solicitação de atualização

Na tela de solicitação de atualização o sistema apresenta as novidades que o cliente deseja utilizar, sendo elas lidas do sistema atual do *Workflow*, considerando as novidades da versão atual até a versão desejada permitindo assim o cliente selecionar quais funcionalidades terão que ser verificadas nos testes feitos pelo suporte.

Após salvar a solicitação de atualização o sistema apresenta a tela de consulta das solicitações de atualizações conforme a Figura 21.

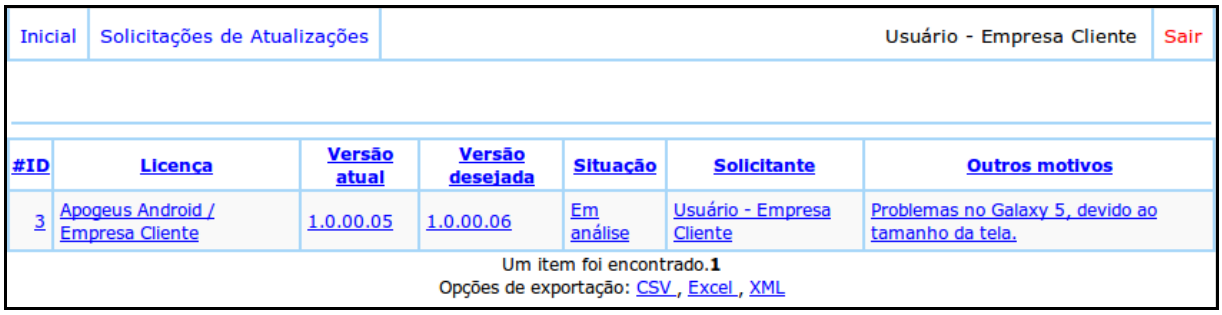


Figura 21 – Tela de consulta das solicitações de atualizações

Após o cadastramento o Apogeus Update Manager Client, que está executando no servidor do cliente, muda a sua situação para informar ao usuário que sua solicitação está em análise, conforme a Figura 22.

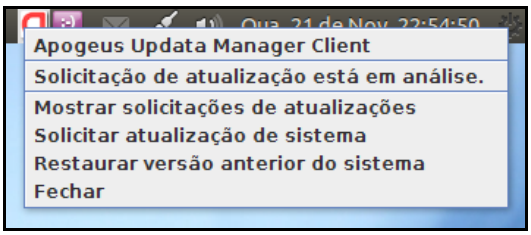


Figura 22 – Menu de opções do Apogeus Update Manager Client

Caso seja rejeitada a atualização o sistema irá mudar a situação da solicitação para rejeitada, como demonstrado na Figura 23. Se a solicitação for aprovada o sistema irá automaticamente avisar ao cliente que foi aprovada e pode ser atualizada. Caso o cliente deseja realizar a atualização, o sistema realiza o *backup* dos dados para que assim seja atualizado, conforme a Figura 24, além de mudar a situação novamente para atualizado, como demonstrado na Figura 25.

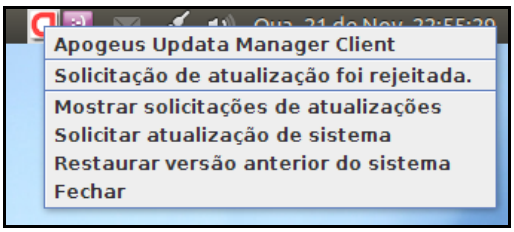


Figura 23 – Tela do Apogeus Update Manager Client com a solicitação rejeitada

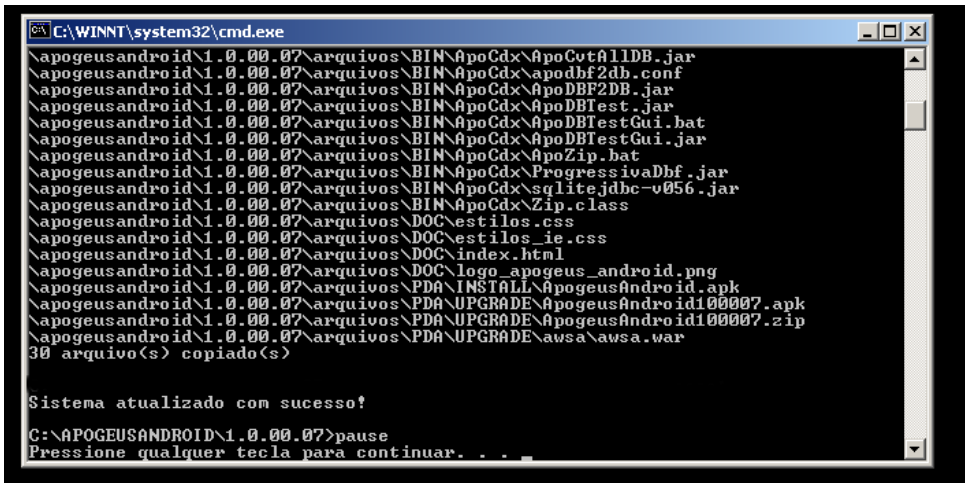


Figura 24 – Tela com o script de atualização executado

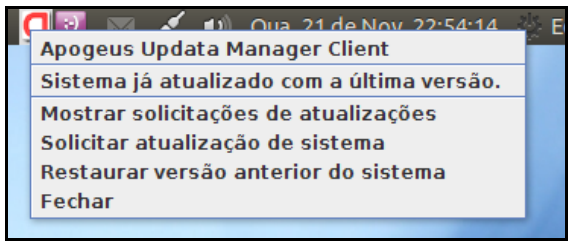


Figura 25 – Tela do Apogeus Update Manager Client com a solicitação aprovada

3.3.2.3 Funcionalidades para desenvolvedor

O desenvolvedor possui acesso ao Apogeus Package Manager, no qual preenche os dados para realizar a liberação da versão conforme a Figura 26.

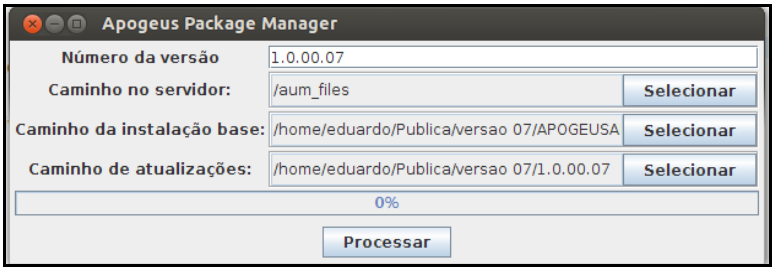


Figura 26 – Tela do Apogeus Package Manager utilizado para gerar os pacotes da versão

Após preencher os dados corretamente, o usuário do desenvolvimento pressiona o botão processar, o que gera um arquivo de instalação e o outro de atualização no caminho do servidor automaticamente.

O desenvolvedor possui acesso completo ao Apogeus Update Manager, no qual pode optar pelas opções conforme mostra a Figura 27.

Figura 27 – Menu do desenvolvedor

Ao selecionar a opção de criar novo formulário o sistema irá apresentar a tela da Figura 28, na qual o desenvolvedor colocará todas as perguntas e respostas que serão testadas no sistema, e gravará o formulário.

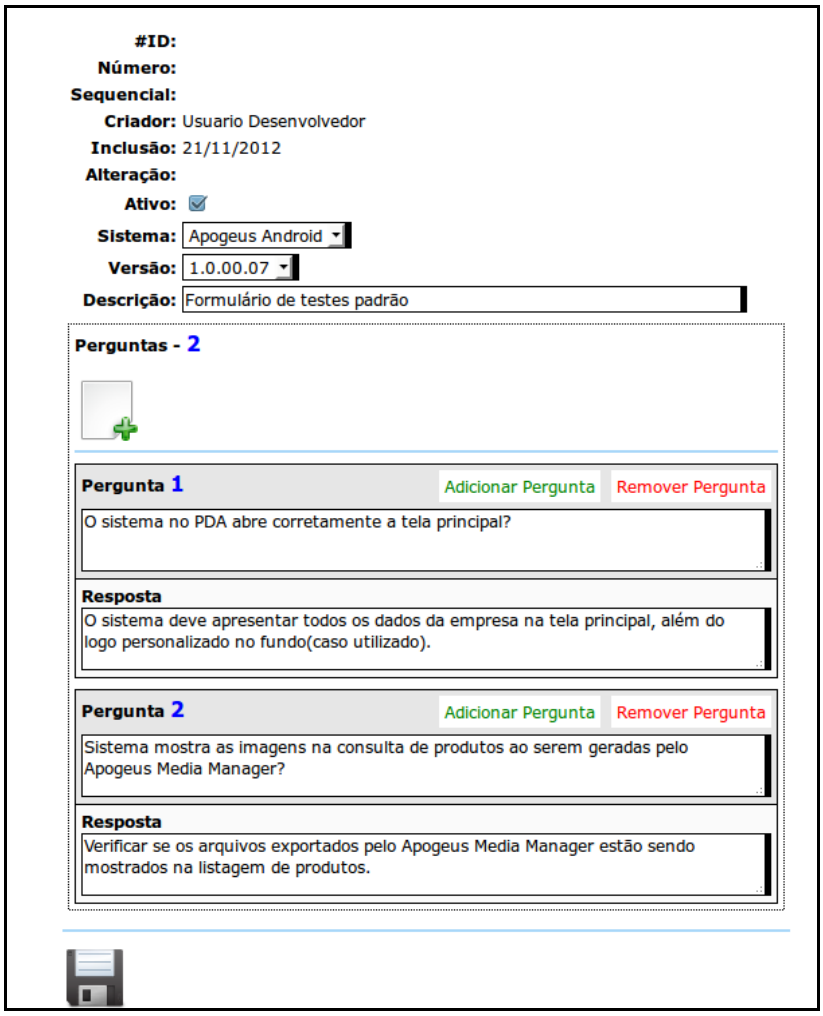


Figura 28 – Formulário de testes

Na tela o desenvolvedor irá preencher inicialmente o sistema desejado, e qual a versão do sistema que serão realizados os testes. Em seguida irá descrever o tipo de testes a realizar no campo descrição. Sendo que cada versão pode ter mais de um formulário de testes, pois em uma liberação emergencial pode-se criar um formulário reduzido de testes para que assim tenha uma liberação rápida e focada a correção do problema apresentado. Por fim as perguntas e respostas serão descritas a partir dos formulários padrões e novas funcionalidades que devem ser testadas pelo suporte.

Após gravar o formulário, o desenvolvedor é direcionado à consulta de formulários, conforme a Figura 29. O botão na parte superior com o desenho de um símbolo de mais permite ao usuário criar novos formulários, e para a alteração o usuário deve clicar sobre o item desejado.

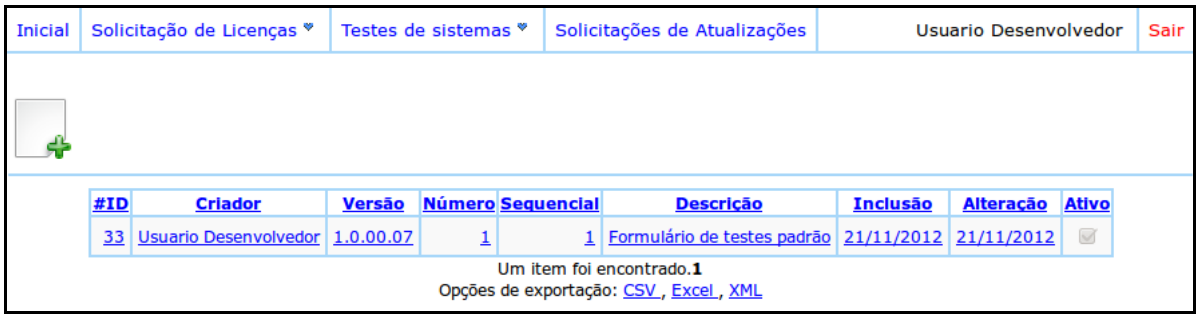


Figura 29 – Tela de pesquisa de formulário de testes

Na lista de formulários somente são mostrados os ativos. O campo ativo é somente utilizado como controle e filtro, pois a cada alteração do formulário é acrescentado um ao seqüencial e em é inativo o formulário anterior, mantendo assim um histórico das alterações dos formulários. O número do formulário também é seqüencial para cada versão, sendo assim o segundo formulário da mesma versão receberá o número dois, e se for o primeiro formulário da versão receberá o número um.

3.3.2.4 Funcionalidades para o suporte

O usuário do suporte possui todas as opções, iguais ao usuário do desenvolvimento conforme demonstrado na Figura 27, porém o mesmo realiza adicionalmente a análise das solicitações de atualizações, utilizando a consulta de solicitações, conforme a Figura 30.

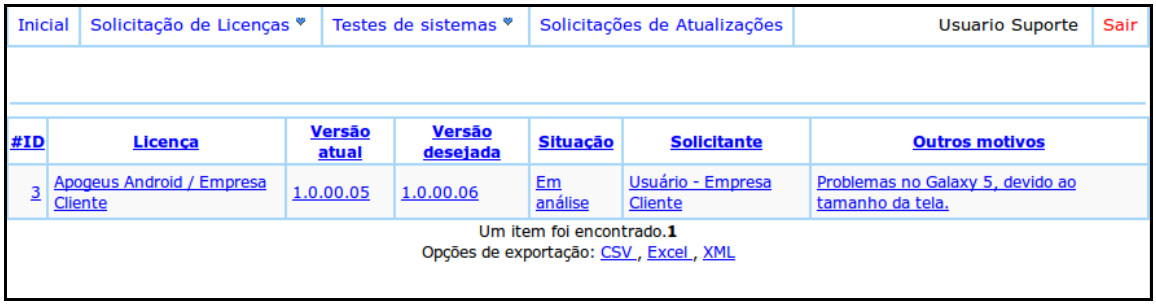


Figura 30 – Tela de pesquisa de solicitações de atualizações

Para analisar as solicitações o suporte clica sobre a solicitação desejada e verifica quais são as novidades ou outros motivos que o cliente possui para atualizar. Após baixar o arquivo compactado enviado pelo Apogeus Update Manager Client, o suporte pode rejeitar a solicitação informando o que não permite a atualização, como demonstrado na Figura 31. Caso os testes realizados permitam a atualização, ele muda a situação para liberada conforme a Figura 32.

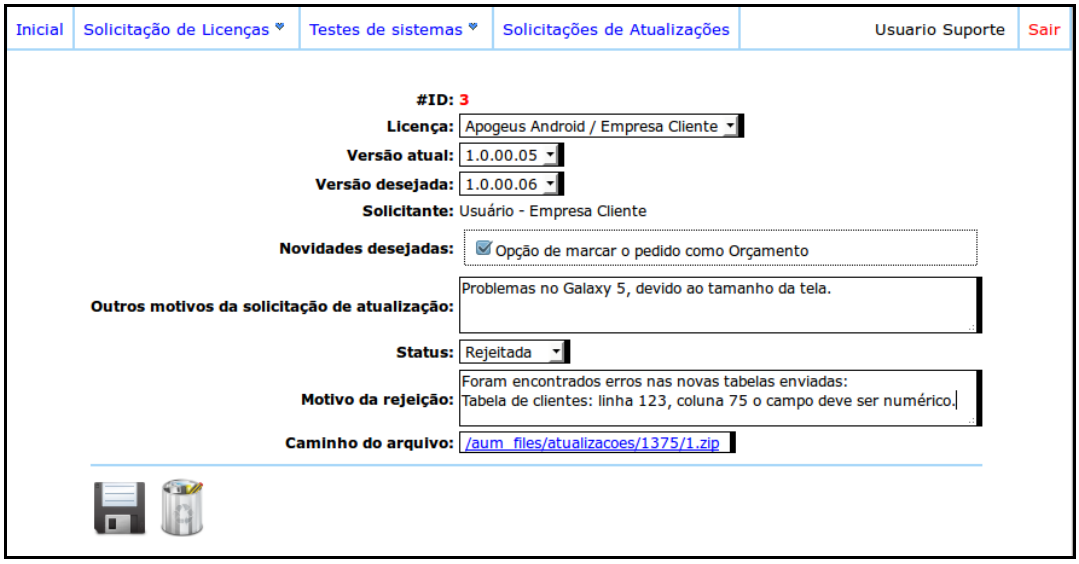


Figura 31 – Tela de solicitação de atualização rejeitada

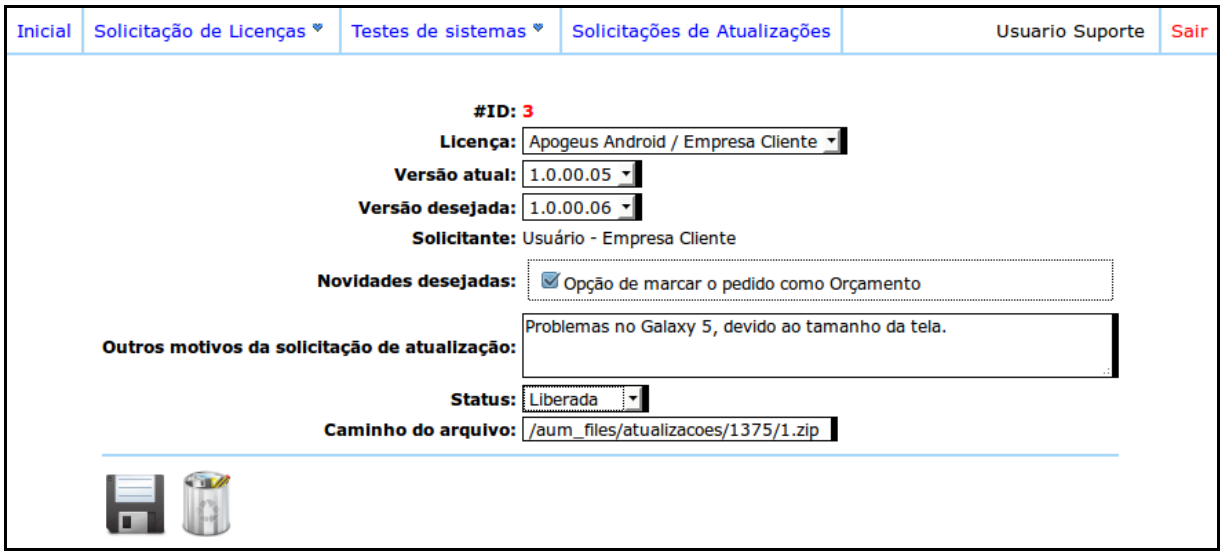


Figura 32 – Tela de solicitação de atualização liberada

Além da permissão de aprovar atualizações nos clientes, o suporte possui permissão para realizar testes das versões utilizando os formulários criados pelo desenvolvimento. Quando o desenvolvimento passa as tarefas pelo *Workflow* da empresa a pessoa encarregada pelos testes inicia os testes acessando a opção de iniciar novo teste, do menu de teste de sistemas, que por sua vez mostra a tela de testes, conforme demonstrado na Figura 33.

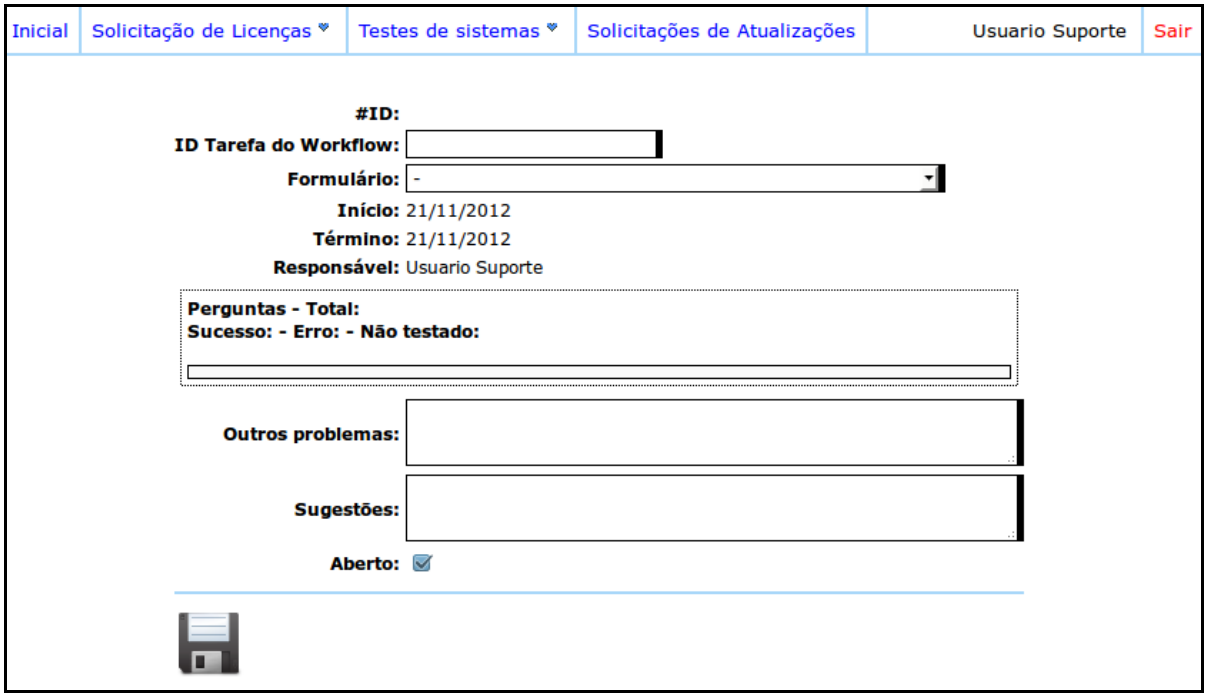


Figura 33 – Tela de resultado de testes da versão sem formulário selecionado

Após selecionar o formulário o sistema apresenta as perguntas cadastradas pelo desenvolvimento conforme a Figura 34.

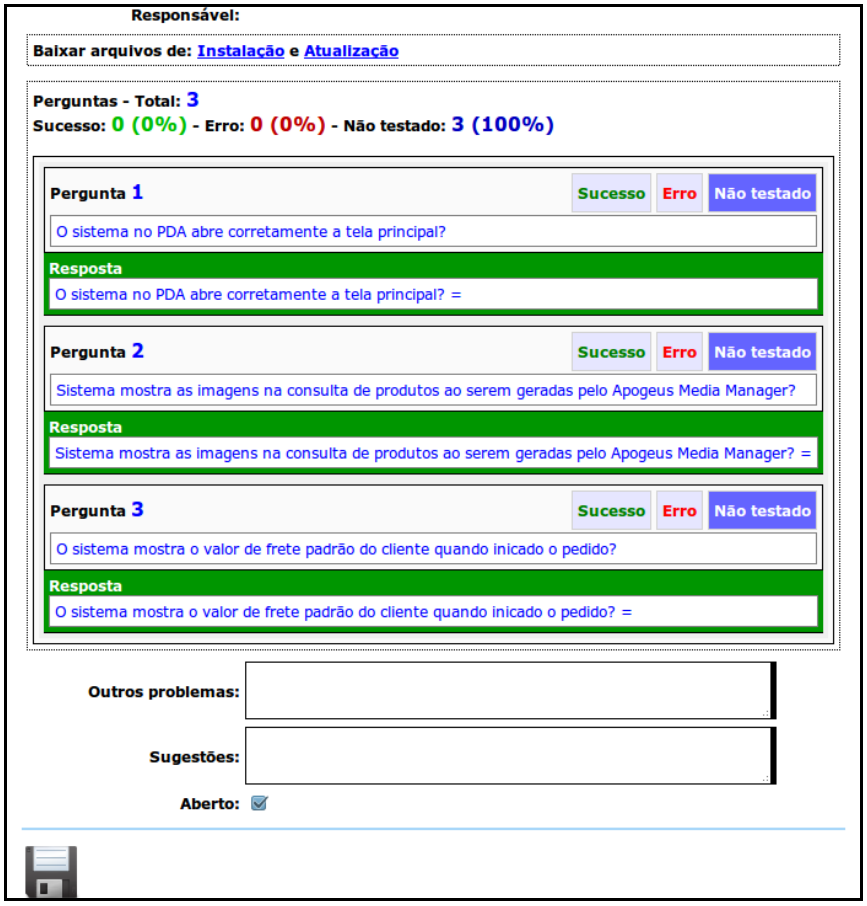


Figura 34 – Tela de resultado de testes da versão com sucesso

Após preencher as respostas o sistema apresenta quantas perguntas ainda faltam ser respondidas. Caso todas forem respondidas, o sistema permite ao usuário desmarcar o campo de `aberto`, pois assim o sistema assume que todas as perguntas foram testadas. Na Figura 35, pode-se ver um teste realizado, porém não fechado.

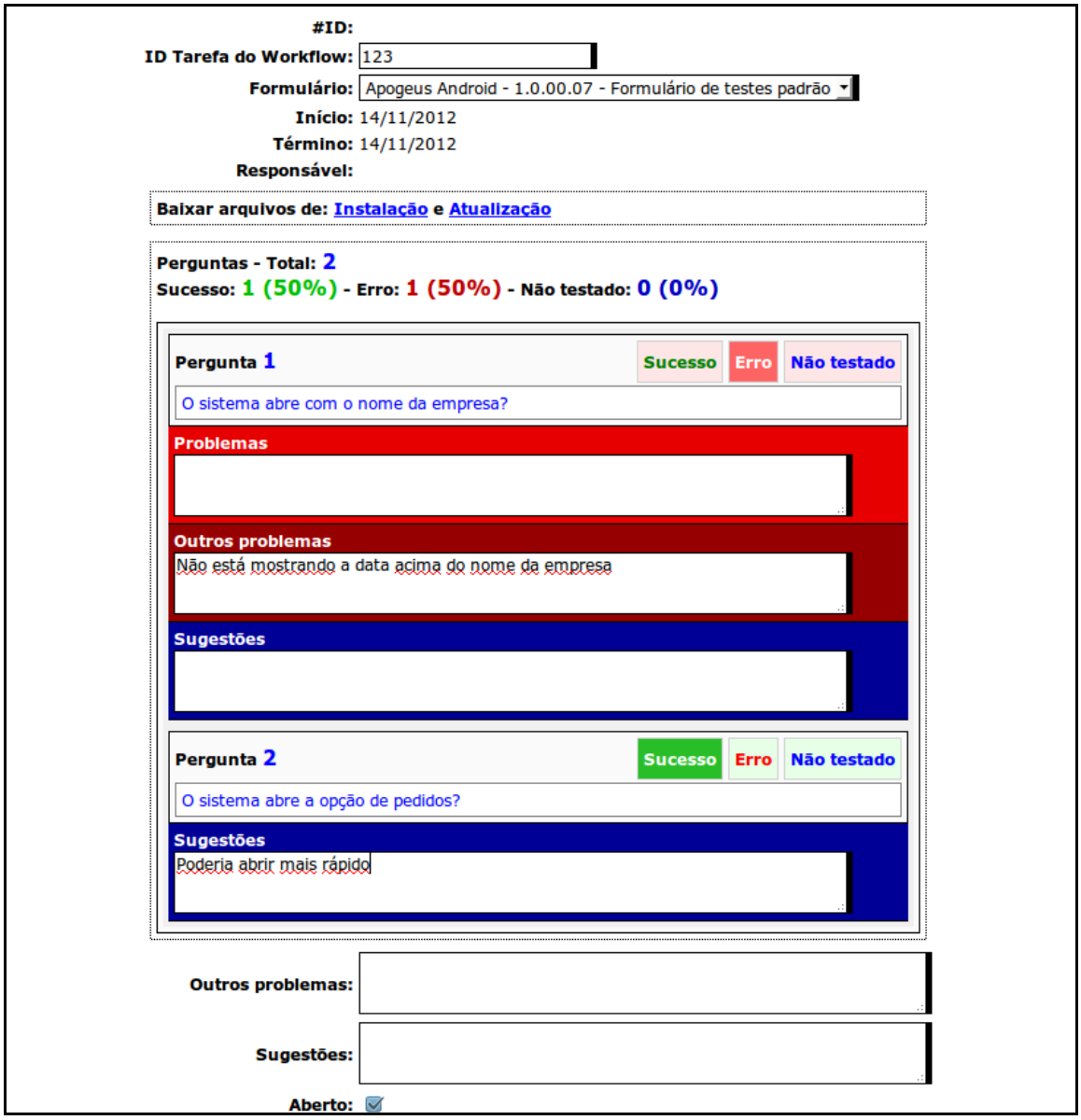


Figura 35 – Tela na qual são informados os resultados de um teste de versão

3.3.2.5 Linhas de códigos

Na Figura 36 pode-se ver a rotina de solicitação de atualização quando selecionada a opção de solicitar uma nova solicitação. No código fonte pode-se ver o recebimento do arquivo pelo parâmetro chamado `arquivo`, após escrever os dados escritos para a pasta `/aum_files/atualização`. O sistema retorna o local onde foi gravado arquivo para que assim o módulo de atualização preencha todos os dados da solicitação.

```
public String enviarArquivo() {
    FileItem arquivo = (FileItem) input.getValue("arquivo");

    try {
        HttpServletResponse res = ((SessionContext) session).getResponse();

        String fileKey = "/aum_files/atualizacao/" + new Date().getTime() + ".zip";

        res.setHeader("file_key", fileKey);

        File f = new File(fileKey);

        if (!f.getParentFile().exists())
            f.getParentFile().mkdirs();

        arquivo.write(f);
        arquivo.delete();

    } catch (Exception e) {
        e.printStackTrace();
        return ERROR;
    }

    return SUCCESS;
}
```

Figura 36 – Código fonte de envio de arquivos da solicitação de atualização

Na Figura 37 está a rotina que permite o *download* do arquivo enviado pelo cliente quando solicitada a atualização do sistema. Esta opção é acessada pelo suporte quando é feita a análise da solicitação.

```
public String baixarArquivo() {
    String pathname = readStringFromIn("arquivo");

    // evita baixar outros arquivos
    if (!pathname.startsWith("/aum_files/atualizacao/"))
        return ERROR;

    File file = new File(pathname);
    return downloadFile(file);
}
```

Figura 37 – Código fonte do *download* do arquivo da solicitação de atualização

Na Figura 38 está a rotina que escreve os valores das solicitações de versão para a tela de cadastro, os quais os mesmos serão preenchidos pelo *framework*.

```
@Override
public void fromVOToOutput(SolicitacaoAtualizacaoVersao idata) throws Exception {

    writeLongToOut("id", idata.getId());

    writeVOToOut("versaoAtualID", idata.getVersaoAtual());
    writeVOToOut("versaoDesejadaID", idata.getVersaoDesejada());
    writeVOToOut("solicitanteID", idata.getSolicitante());
    writeStringToOut("solicitanteStr", idata.getSolicitante());

    writeVOToOut("licencaID", idata.getLicenca());

    writeVOToOut("responsavelID", idata.getResponsavel());
    writeStringToOut("caminhoArquivo", idata.getCaminhoArquivo());
    writeStringToOut("outrosMotivos", idata.getOutrosMotivos());
    writeEnumToOut("statusENUM", idata.getStatus());
    writeStringToOut("statusStr", idata.getStatus() == null ? null : idata.getStatus().toString());

}
```

Figura 38 – Código fonte da inclusão ou edição das solicitações de versões

Na Figura 39 está a parte do código fonte da página de formulários de testes, o sistema percorre a lista de perguntas cada pergunta já cadastrada cria os componentes necessários.

```
<script type="text/javascript">

<mtw:isNull test="perguntas_list" negate="true">
<mtw:list value="perguntas_list">
<mtw:loop var="pergunta">
novaPergunta('<prg:outJs value="pergunta.pergunta"/>', '<prg:outJs value="pergunta.resposta"/>', true);
</mtw:loop>
</mtw:list>
</mtw:isNull>

</script>
```

Figura 39 – Código fonte da inclusão ou edição das solicitações de versões

Na Figura 40 está a parte do código fonte que cria a classe que controla as conexões do Apogeus Update Manager Cliente (AUMC) com o servidor Apogeus Update Manager (AUM).

```
private AumcHttpMethodController createHttpController(HttpConfig config) {
    MultiThreadedHttpConnectionManager tipoConn = new MultiThreadedHttpConnectionManager();
    HttpConnectionManagerParams p1 = new HttpConnectionManagerParams();
    p1.setIntParameter(HttpConnectionManagerParams.MAX_HOST_CONNECTIONS, 10);
    tipoConn.setParams(p1);

    HttpConnectionManagerParams p2 = new HttpConnectionManagerParams();
    p2.setIntParameter(HttpConnectionManagerParams.MAX_TOTAL_CONNECTIONS, 100);
    tipoConn.setParams(p2);

    HttpClient httpClient = new HttpClient(tipoConn);

    AumcHttpMethodController controller = new AumcHttpMethodController(config, httpClient);

    return controller;
}
```

Figura 40 – Código fonte da inclusão ou edição das solicitações de versões

3.4 RESULTADOS E DISCUSSÃO

O principal objetivo deste trabalho era melhorar o controle de solicitações de novas licenças, criação de pacotes das versões, controle dos testes da versão e permitir a atualização automatizada quando liberada pelo suporte. O objetivo foi alcançado nos testes realizados no ambiente de testes da empresa, utilizando uma cópia da base de dados oficial do *Workflow* e em seguida foi apresentado o para todos os envolvidos da empresa, os mesmos realizaram testes durante alguns dias verificando a necessidade da criação de alguma nova opção. Falou-se que essas opções seriam integradas ao sistema interno da empresa, sendo assim a localização das opções iriam mudar, mas as funcionalidades seriam as mesmas. Porém o sistema somente será utilizado oficialmente na empresa a partir de meados de março de 2013.

O real motivo para tal é que todos os clientes com Apogeus Android devem possuir uma versão compatível com o Apogeus Update Manager, a qual deverá sair no final de janeiro de 2013.

No desenvolvimento do trabalho foram utilizados *frameworks* já utilizados pela empresa, o que facilitou o desenvolvimento do sistema. Além disso acesso à estrutura física de testes da empresa reduziu o tempo em criar ambientes de testes fora dela, pois assim não seriam necessárias as configurações dos ambientes de desenvolvimento e testes como banco de dados e servidor *Web*.

Com o passar do tempo o sistema como um todo foi adaptando-se às necessidades dos usuários. Um exemplo disso foi a criação do projeto Apogeus Package Manager (APM) desenvolvido para rodar localmente na máquina do usuário do desenvolvimento, cuja idéia inicial era de enviar os arquivos compactados diretamente à página do Apogeus Update Manager. Após a utilização da primeira versão, foi definido que seria melhor o usuário do desenvolvimento somente definir a versão e automaticamente enviar a uma pasta no servidor, sem a utilização de uma interface Web.

O trabalho correlato de Metzner (2001) era responsável pela criação de pacotes de versões finais para atualização do sistema de forma automatizada. O Apogeus Update Manager trabalha de forma similar, pois ele possui um servidor onde os arquivos serão armazenados até que um programa cliente, que neste trabalho foi denominado Apogeus Update Manager Client (AUMC), solicite-os para efetuar uma atualização.

Ainda comparando trabalhos correlatos, Samagaia (2007) desenvolveu um controle de

solicitações, que funciona de forma similar ao resultado dos formulários de testes, pois permite que nos testes sejam solicitadas alterações ou até correções por meio de tarefas ao setor de desenvolvimento pelo setor do suporte.

Quando analisado o trabalho correlato de Ambrosi (2004), o mesmo desenvolveu um sistema que permite a criação de atualizações de programas enviando-os para um servidor *File Transfer Protocol* (FTP), no qual é verificado se a versão do cliente já está atualizada usando *Extensible Markup Language* (XML). O principal diferencial no trabalho desenvolvido para Progressiva é a utilização de um servidor *Web* (HTTP) e a integração com o Apogeus Android para que os dados sejam enviados para que o suporte analise se o cliente pode atualizar o sistema.

Após a liberação para o uso na base de testes foi definido que seriam necessárias alterações na estrutura do Apogeus Android. Já foi solicitada uma reestruturação nos pacotes de atualização para que essa seja executada de forma automatizada quando o Apogeus Update Manager for colocado em funcionamento.

4 CONCLUSÕES

O trabalho teve como principal objetivo desenvolver um sistema de controle de versões, testes e liberações do sistema Apogeus Android da empresa Progressiva, além de permitir solicitar novas licenças. Por reduzir o trabalho na criação de pacotes, facilitou os testes de versão, eliminando os formulários de testes impressos, além da eliminação do uso de e-mail na solicitação de novas ou alterações de licenças.

Analisando os objetivos do trabalho, todos foram cumpridos, mesmo que tenham sido realizadas alterações no projeto durante o desenvolvimento devido à sugestões dos envolvidos. Assim, criou-se um sistema que irá atender por completo às necessidades da empresa em relação ao que foi tratado no trabalho.

A redução do tempo do desenvolvimento na criação dos pacotes foi devido ao módulo do Apogeus Package Manager (APM), o qual permite em poucos cliques liberar todos os arquivos de forma fácil. Após a criação o desenvolvimento utiliza os formulários para criar os *checklists* baseados nas funcionalidades da versão liberada pelo APM, que novamente reduz o tempo de modificação dos arquivos do *checklist* devido a sua tela de fácil inclusão das perguntas e respostas.

Após a todos os testes e correções com sucesso, a versão fica com a situação pré-liberada onde é mostrada ao gerente de suporte e gerente comercial para que um ou outro libere a versão facilmente para todos os clientes em apenas um clique.

Como dito anteriormente, este projeto somente será utilizado em meados de março de 2013, devido a alguns ajustes que serão realizados no Apogeus Android, para que permita o script de atualização automática. Estas alterações somente estarão disponíveis na versão 1.0.0.00.08 do Apogeus Android prevista para janeiro de 2013. Além desta alteração, o projeto Apogeus Update Manager será incorporado ao sistema *Intranet* Progressiva, pois assim uma vez logado no atual sistema de intranet, o usuário poderá navegar nas outras opções já existentes da Progressiva.

Conclui-se com este trabalho que, mesmo sendo desenvolvido como um sistema externo ao sistema de intranet da Progressiva, o software deste trabalho pode funcionar independentemente dos outros serviços, mesmo utilizando dados gerados por outros aplicativos, tais como: controle de licença, sistema de e-mail marketing, *Workflow*, dentre outros serviços da intranet que não são citados aqui.

4.1 EXTENSÕES

Para continuar o sistema seria necessária a criação dos módulos que não são disponibilizados pela Progressiva. Pode-se citar alguns módulos que não estão presentes neste trabalho que seriam necessários, como cadastro de usuários, clientes, revendas, sistema e versões, além do desenvolvimento do módulo de controle de liberação de versões, permitindo assim a empresa controlar a criação, testes e liberação da versão.

Além do módulo de testes de versão, é necessário o controle de licenças de dispositivos, o qual teria os mesmos cadastros do controle de testes, além dos cadastros de licenças e dispositivos.

REFERÊNCIAS

AMBROSI, Airison. **Protótipo de software para atualização automática de versão de arquivos**. 2004.43 f, il. Trabalho de Conclusão de Curso - Universidade Regional de Blumenau, Curso de Ciência da Computação, Blumenau, 2004. Disponível em: <http://www.bc.furb.br/docs/MO/2004/305285_1_1.pdf>. Acesso em: 31 out. 2012.

CARVALHO, Hugo; SILVA, Pedro Tavares; TORRES, Catarina Botelho. **Segurança dos Sistemas de Informação - Gestão Estratégica da Segurança Empresarial**. Centro Atlântico, Lda, 2003.

DOROW, Everson. **ITIL: Gerenciamento de liberação**. [S.l.], 2009. Disponível em: <http://www.profissionaisti.com.br/2009/07/itil-gerenciamento-de-liberacao/>. Acesso em: 31 out. 2012

FREITAS, Marcos André dos Santos. **Fundamentos do gerenciamento de serviços de TI:** preparatório para a certificação ITIL V3 Foundation. Rio de Janeiro : Brasport, c2010. xvi, 351 p, il., grafs.

MAGALHÃES, Ivan Luizio; PINHEIRO, Walfrido Brito. **Gerenciamento de serviços de TI na prática:** uma abordagem com base na ITIL, inclui ISO/IEC 20.000 e IT Flex. São Paulo : Novatec, 2007. 667 p, il.

MENTAFRAMEWORK. **Introdução,** [S.l.], 2012. Disponível em: <http://www.mentaframework.org/mtw/Page/Intro/mentawai-fullstack-actionbased-mvc-opensource-java-web-framework> Acesso em: 3 nov. 2012

METZNER, Denis Cezar. **Um protótipo de software assistente de atualização de versão de software**. 2001. xi, 53p, il. Trabalho de Conclusão de Curso - (Graduação em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2001.

PROGRESSIVA. **Empresa,** Blumenau, 2012a. Disponível em: <http://sites.google.com/site/progressivaltda1/Empresa/>. Acesso em: 31 out. 2012

_____ **Funcionamento Apogeus Android,** Blumenau, 2012b**.** Disponível em: <http://sites.google.com/site/progressivaltda1/produtos/Apogeus-Android/funcionamento>. Acesso em: 13 out. 2012

SAMAGAIA, Jeferson Roberto**. Sistema de gerenciamento de controle de liberação de versões de sistemas Web baseado na recomendação Itil utilizando Shell Unix.** 2007.93 f, il. Trabalho de Conclusão de Curso - (Graduação em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau. Blumenau, 2007. Disponível em: <http://www.bc.furb.br/docs/MO/2008/329175_1_1.pdf>. Acesso em: 16 out. 2012.

APÊNDICE A – Descrição dos Casos de Uso

Este Apêndice apresenta a descrição dos principais casos de uso descritos na seção de especificação deste trabalho. No Quadro 5 estão descritos todos os casos de uso do sistema.

UC01.01 Criar pacotes de instalação e atualização

Permite ao usuário do desenvolvimento selecionar duas pastas, onde primeira deve ser a instalação original do Apogeus Android, já a segunda deve ser a pasta onde estão os arquivos da atualização. O sistema deverá criar dois arquivos ZIP, no qual um será a instalação atualizada e outro somente irá conter arquivos de atualização.

Constraints

Pré-condição . O usuário do desenvolvimento deve estar cadastrado no sistema.

Pré-condição . A versão deve estar cadastrada no sistema *Workflow* da empresa.

Pós-condição .Dois arquivos ZIP serão gerados e enviados para o servidor para serem realizados os testes.

Cenários

Consulta versões {Principal}.

1. Usuário do desenvolvimento seleciona a versão na qual serão enviados os arquivos.
2. Sistema apresenta uma tela para informar pasta de instalação e outra de atualização.
3. Usuário do desenvolvimento seleciona as respectivas pastas e clica no botão de enviar dados.
4. Sistema cria um pacote de instalação e outro de atualização e envia ao servidor para a respectiva versão.

UC01.02 Cadastrar formulário de testes da versão

Permite ao usuário do desenvolvimento criar formulários de testes para o setor de suporte. O mesmo deve permitir a criação de formulários com perguntas do tipo texto, verdadeiro/falso, múltiplas alternativas e escala de 0 a 10. Ao cadastrar cada pergunta o desenvolvimento pode ou não cadastrar a resposta, permitindo assim ao sistema analisar se os testes foram feitos com sucesso se o mesmo possuir uma resposta.

Constraints

Pré-condição . O usuário do desenvolvimento deve estar cadastrado no sistema.

Pré-condição . O usuário deve estar logado no sistema.

Pré-condição . A versão deve estar cadastrada no sistema *Workflow* da empresa.

Pré-condição . A versão deve possuir arquivos enviados pelo desenvolvimento.

Pós-condição . Um formulário de testes é cadastrado no sistema.

Cenários

Consulta versões {Principal}.

1. Usuário do desenvolvimento seleciona a versão na qual será cadastrado o formulário de testes.
2. Sistema apresenta uma tela para cadastrar as perguntas, permitindo selecionar o tipo da mesma.
3. Usuário do desenvolvimento preenche a pergunta, em seguida seleciona o tipo da resposta, podendo preencher a resposta correta.
4. Sistema permite ao usuário cadastrar a próxima pergunta ou finalizar o cadastro do formulário.

UC01.03 Realizar testes da versão

Permite ao usuário do suporte selecionar a versão do sistema para iniciar os testes, além de permitir baixar arquivos de instalação e atualização do sistema.

Constraints

Pré-condição . O usuário do suporte deve estar cadastrado no sistema.

Pré-condição . O usuário deve estar logado no sistema.

Pré-condição . O formulário de teste deve estar cadastrado no sistema.

Pós-condição . Uma versão foi aprovada e alterada para o estado de pré-liberada.

Pós-condição . Uma versão foi rejeitada pelas respostas e voltou para o desenvolvimento.

Pós-condição . Uma versão foi rejeitada devido a um erro que não estava no formulário.

Pós-condição . Uma tarefa no Workflow foi lançada ao desenvolvimento ao encontrar erros.

Cenários

Consulta versões {Principal}.

1. Usuário do suporte solicita o formulário de testes de uma versão específica.
2. Sistema apresenta tela para registro dos testes.
3. Usuário do suporte preenche as respostas.

4. Sistema analisa as respostas e apresenta se o sistema está ou não aprovado.
5. Usuário descreve outros erros encontrados nos testes e grava as informações.
6. Sistema mostra mensagem ao usuário de que todas as repostas estavam corretas.
7. Usuário confirma a mudança do estado da versão para pré-liberada.
8. Sistema muda o estado da versão para pré-liberada.

Cancelar os testes {Alternativo}.

No passo 3, caso o usuário opte por cancelar os testes na versão selecionada

3.1 Usuário cancela os testes.

3.2 Sistema apresenta tela para selecionar outra versão.

Respostas erradas {Alternativo}.

No passo 6, caso o sistema encontre respostas erradas

6.1 Sistema alerta ao usuário que existem respostas erradas.

6.2 Sistema pergunta se o usuário quer refazer os testes.

6.3 Usuário aceita refazer os testes.

6.4 Sistema registra os dados dos testes.

6.5 Sistema limpa os campos da tela e retorna ao passo 2.

Respostas erradas e encaminhadas ao desenvolvimento {Alternativo}.

No passo 6.3, caso o usuário opte por não refazer os testes

6.3.1 Sistema envia uma tarefa do Workflow para o desenvolvimento com os resultados dos testes.

Versão sem formulário de testes {Exceção}.

No passo 2, caso a versão não possua um formulário de testes, apresenta mensagem "Versão não possui formulário de testes, informe ao desenvolvimento."

UC01.04 Liberar versão

Permite ao gerente comercial ou usuário do suporte a liberação final da versão, no qual serão enviados os arquivos testados para os repositórios da empresa

.

Constraints

Pré-condição . O gerente comercial ou usuário do suporte deve estar cadastrado no sistema.

Pré-condição . O usuário deve estar logado no sistema.

Pré-condição . A versão a ser liberada deve possuir um formulário de deste realizado com sucesso.

Pós-condição . Uma versão é liberada para os repositórios da empresa.

Cenários

Consulta versões {Principal}.

1. Usuário solicita a tela de liberação de uma versão específica.
2. Sistema apresenta tela para confirmação da liberação.
3. Usuário confirma a liberação.
4. Sistema envia todos os arquivos para os repositórios da empresa.

UC02.01 Consultar versões dos clientes

Permite ao usuário do suporte consultar as versões dos clientes. O mesmo deve permitir o filtro por nome, C.N.P.J. ou revenda.

Constraints

Pré-condição . O usuário do suporte deve estar cadastrado no sistema.

Pré-condição . O usuário deve estar logado no sistema.

Pós-condição . Mostra uma lista das versões dos clientes respeitando os filtros.

Cenários

Consulta versões {Principal}.

1. Usuário do suporte solicita a tela de versões dos clientes.
2. Sistema apresenta tela com a listagem dos clientes e suas respectivas versões, permitindo o filtro por nome, C.N.P.J. ou revenda.
3. Usuário preenche o filtro desejado e clica em filtrar.
4. Sistema apresenta os dados filtrados.

UC02.02 Liberar atualização automática para o cliente

Permite ao usuário do suporte liberar a atualização automática ao cliente, sendo que para o mesmo o cliente deve solicitar a atualização ao suporte pelo módulo instalado no desktop.

Constraints

Pré-condição . O usuário do suporte deve estar cadastrado no sistema.

Pré-condição . O usuário deve estar logado no sistema.

Pré-condição . O cliente deve solicitar uma atualização.

Pré-condição . A solicitação de atualização deve ter um teste realizado com sucesso.

Pós-condição . Uma nova versão é instalada no cliente.

Cenários

Consulta versões {Principal}.

1. Usuário do suporte solicita a tela para liberação da atualização para o cliente.
2. Sistema mostra tela de liberação da atualização.
3. Usuário confirma a liberação da atualização.
4. Sistema marca a solicitação de atualização como liberada.
6. Sistema mostra ao usuário do suporte que está baixando o arquivo no cliente.
7. Módulo de atualização instalado no desktop baixa atualização.
8. Sistema mostra ao usuário do suporte que está realizando backup.
9. Módulo de atualização realiza o backup das informações.
10. Sistema mostra ao usuário do suporte que está executando o script.
11. Módulo de atualização executa o script de atualização.
12. Módulo de atualização marca o status da solicitação como realizada.
13. Módulo de atualização avisa ao cliente que a atualização ocorreu com sucesso.
14. Sistema mostra ao usuário do suporte que foi atualizado com sucesso.

UC02.03 Consultar solicitações de atualização

Permite ao usuário do suporte consultar todas as solicitações de atualização feitas pelos clientes. Deve possuir os filtros de nome, C.N.P.J., revenda e data de solicitação. O mesmo deve listar primeiramente as solicitações não analisadas por data de solicitação crescente.

Constraints

Pré-condição . O usuário do suporte deve estar cadastrado no sistema.

Pré-condição . O usuário deve estar logado no sistema.

Pós-condição . O usuário pode analisar a solicitação do cliente.

Cenários

Consulta versões {Principal}.

1. Usuário do suporte solicita a tela para consultar as solicitações de atualizações dos clientes.
2. Sistema apresenta tela com a listagem das solicitações de atualizações dos clientes, permitindo o filtro por nome, C.N.P.J. ou revenda.
3. Usuário preenche o filtro desejado e clica em filtrar.
4. Sistema apresenta os dados filtrados.

UC02.04 Baixar arquivos para testes

Permite ao usuário do suporte baixar os arquivos utilizando a consulta de solicitações de atualização, após baixar os arquivos deve permitir ao usuário liberar a atualização ao cliente ou cancelar a atualização informando o motivo e assim finalizando a solicitação.

Constraints

Pré-condição . O usuário do suporte deve estar cadastrado no sistema.

Pré-condição . O usuário deve estar logado no sistema.

Pré-condição . O usuário deve usar a consulta de solicitações de atualização.

Pós-condição . O usuário analisa a solicitação do cliente.

Cenários

Consulta versões {Principal}.

1. Usuário do suporte solicita a tela para baixar os arquivos da solicitação de atualização de um cliente.
2. Sistema apresenta tela com os arquivos a serem baixados para testes.
3. Usuário baixa um arquivo com todos os dados do cliente.
4. Sistema muda o estado da solicitação para testando e fecha a tela.

UC02.05 Voltar a versão do sistema

Permite ao usuário do suporte ou ao cliente restaurar ao estado do sistema instalado no desktop para antes da atualização. O mesmo deve solicitar três confirmações aos usuários, no qual a primeira irá informar o que irá ocorrer, a segunda os possíveis problemas ao retornar a versão, e a terceira deve forçar ao usuário ler um termo de aceitação para o retorno da versão.

Constraints

Pré-condição . O usuário deve confirmar a restauração da versão.

Pós-condição . A versão do sistema do cliente é restaurada.

Cenários

Consulta versões {Principal}.

1. Usuário do suporte ou cliente solicita a restauração da versão instalada no desktop.
2. Sistema de atualização do desktop apresenta uma pergunta informando o que irá ocorrer.
3. Usuário confirma a pergunta.
4. Sistema de atualização apresenta uma segunda pergunta mostrando os possíveis problemas ao voltar uma versão.
5. Usuário confirma a segunda pergunta.
6. Sistema de atualização apresenta uma terceira pergunta mostra uma pergunta forçando a aceitação dos termos.
7. Usuário lê os termos, marca como aceita e confirma a pergunta.
8. Sistema de atualização retorna o backup realizado na atualização.

UC02.06 Solicitar atualização de versão

Permite ao cliente solicitar a atualização do seu sistema, enviando uma solicitação de atualização ao servidor com seus arquivos para análise do setor de suporte.

Constraints

Pré-condição . O usuário do cliente deve estar cadastrado no sistema.

Pré-condição . O usuário deve estar logado no sistema.

Pós-condição . O usuário envia uma solicitação de atualização a empresa.

Cenários

Consulta versões {Principal}.

1. Usuário do cliente solicita a tela para enviar uma solicitação de atualização.
2. Módulo de atualização compacta os arquivos e envia-os para o servidor, em seguida, apresenta no navegador a tela de preenchimento de solicitação de atualização
3. Usuário seleciona as novidades desejadas, descreve se existe outro motivo e confirma atualização.
4. Sistema grava os dados e apresenta a consulta de solicitações, além de mudar a situação no cliente que sua solicitação está em análise.

UC03.01 Solicitar novas licenças

Permite ao usuário da revenda solicitar uma licença informando o sistema, número de usuários de cada tipo de dispositivo e uma observação para a licença. Depois de enviados os dados ao servidor serão feitos os processos atuais da empresa analisando a solicitação, e em seguida será enviado um e-mail para revenda com a chave da licença.

Constraints

Pré-condição . O usuário da revenda deve estar cadastrado no sistema.

Pré-condição . O usuário deve estar logado no sistema.

Pós-condição . O usuário envia uma solicitação de nova licença a empresa.

Cenários

Consulta versões {Principal}.

1. Usuário da revenda solicita a tela para enviar uma solicitação de uma nova licença.
2. Sistema apresenta a tela para preencher os dados para a solicitação da uma nova licença.
3. Usuário preenche os dados e confirma a solicitação.
4. Sistema grava os dados da solicitação.

UC03.02 Solicitar alteração do número de usuários da licença

Permite ao usuário da revenda digitar a chave da licença, em seguida informar quais serão as novas quantidades de usuários para cada tipo de licença.

Constraints

Pré-condição . O usuário da revenda deve estar cadastrado no sistema.

Pré-condição . O usuário deve estar logado no sistema.

Pós-condição . O usuário envia uma solicitação de alteração de uma licença existente.

Cenários

Consulta versões {Principal}.

1. Usuário da revenda solicita a tela para enviar uma solicitação de uma alteração de licença.
2. Sistema apresenta a tela para preencher os dados para a solicitação da uma nova licença.
3. Usuário preenche os dados e confirma a solicitação.
4. Sistema grava os dados da solicitação.

UC03.03 Permitir a consulta das solicitações de novas ou alterações de licenças

Permite ao usuário da revenda ou administrativo consultar a situação das solicitações de novas ou alterações licenças.

Constraints

Pré-condição . O usuário da revenda ou administrativo deve estar cadastrado no sistema.

Pré-condição . O usuário deve estar logado no sistema.

Pós-condição . O consulta o estado de sua solicitação.

Cenários

Consulta versões {Principal}.

1. Usuário da revenda ou administrativo solicita a tela de consulta de solicitações de novas licenças ou alterações de licenças.
2. Sistema apresenta todas as solicitações da revenda, caso seja do administrativo serão mostradas todas as solicitações de todas as revendas.
3. Usuário seleciona uma solicitação.
4. Sistema mostra os dados da solicitação.
5. Usuário do administrativo pode mudar o estado da solicitação para finalizado, caso seja a revenda estará como somente leitura.
6. Sistema volta para listagem de solicitações.

UC03.04 O sistema deverá permitir a consulta das licenças disponíveis

Permite ao usuário da revenda consultar uma lista suas licenças já liberadas.

Constraints

Pré-condição . O usuário da revenda deve estar cadastrado no sistema.

Pré-condição . O usuário deve estar logado no sistema.

Pós-condição . O usuário consulta as suas licenças disponíveis.

Cenários

Consulta versões {Principal}.

1. Usuário da revenda solicita a tela de consulta de licenças.
2. Sistema apresenta uma tela com todas as licenças disponíveis.
3. Usuário seleciona uma licença.
4. Sistema mostra os dados da licença.

Quadro 5 – Descrição dos casos de uso

APÊNDICE B – Dicionário de dados

Este Apêndice descreve as entidades mostradas na subseção 3.2.3. O nome das tabelas foram padronizadas de acordo com a empresa, sendo as duas primeiras letras definindo qual módulo elas pertencem, assim todas as tabelas possuem as iniciais de *Update Manager* (UM).

Outro padrão adotado pela empresa foi escrever `id_` na frente de cada chave estrangeira, além da chave primária sempre ser um numérico de oito posições denominado `id`.

Os Quadros de 6 até 13 descrevem todas as entidades criadas para este trabalho, não estão descritas entidades já existentes nos sistemas da Progressiva como versões e sistemas.

Entidade: Um_SolicitacaoLicenca		
Descrição: Solicitação de alteração ou nova licença		
Atributo	**Tipo**	**Descrição**
Id	Numeric (8,0)	Identificador.
Id_Solicitante	Numeric (8,0)	Usuário solicitante (Tabela externa).
Id_Responsavel	Numeric (8,0)	Usuário responsável (Tabela externa).
Id_Revenda	Numeric (8,0)	Revenda (Tabela externa).
Id_Sistema	Numeric (8,0)	Sistema (Tabela externa).
Id_Licenca	Numeric (8,0)	Licença a ser alterada, caso solicitação de nova não deve ser preenchido.
Observacao	Varchar (4000)	Observação da solicitação.
Status	Varchar (1)	Situação da solicitação.
Id_NovaLicenca	Numeric (8,0)	Nova licença gerada.
MotivoRejeicao	Varchar (4000)	Motivo da rejeição.

Quadro 6 – Entidade de solicitação de licença

Entidade: Um_QuantidadeSolicitacaoLicenca		
Descrição: Quantidades da solicitação de alteração ou nova licença		
Atributo	**Tipo**	**Descrição**
Id	Numeric (8,0)	Identificador.
Id_Solicitacao	Numeric (8,0)	Solicitação de licença.

Id_TipoLicenca	Numeric (8,0)	Tipo de licença (Tabela externa).
Quantidade	Numeric (8,0)	Quantidade solicitada.

Quadro 7 – Entidade da quantidade de solicitação de licença

Entidade: Um_FormularioTeste		
Descrição: Formulário de testes		
Atributo	**Tipo**	**Descrição**
Id	Numeric (8,0)	Identificador.
Id_Criador	Numeric (8,0)	Usuário criador (Tabela externa).
Id_Versao	Numeric (8,0)	Versão (Tabela externa).
Numero	Numeric (8,0)	Número do formulário é seqüencial por versão, ou seja, inicia do um para cada versão.
Sequencial	Numeric (8,0)	Número seqüencial para cada número de formulário, gerando um novo formulário a cada alteração.
Descricao	Varchar (100)	Descrição do formulário.
DataInclusao	Date	Data de inclusão.
DataAlteracao	Date	Data da última alteração.
Ativo	Varchar (1)	Pode ser zero para inativo ou um para ativo, sendo que será inativo o registro anterior a cada alteração.

Quadro 8 – Entidade do formulário de testes

Entidade: Um_PerguntaFormularioTeste		
Descrição: Pergunta do formulário de testes		
Atributo	**Tipo**	**Descrição**
Id	Numeric (8,0)	Identificador.
Id_FormularioTeste	Numeric (8,0)	Formulário de teste.
Pergunta	Varchar (4000)	Pergunta que deve ser respondida.
Resposta	Varchar (4000)	Resposta que o usuário deve encontrar ao testar.

Quadro 9 – Entidade da pergunta do formulário de teste

Entidade: Um_ResultadoTeste		
Descrição: Resultado do formulário de teste		
Atributo	**Tipo**	**Descrição**
Id	Numeric (8,0)	Identificador.
Id_FormularioTeste	Numeric (8,0)	Formulário de teste.
Id_Responsavel	Numeric (8,0)	Usuário responsável (Tabela externa).
Id_Tar	Numeric (8,0)	Tarefa (Tabela externa).
OutrosProblemas	Varchar (4000)	Outros problemas encontrados nos testes
Sugestoes	Varchar (4000)	Sugestões a serem consideradas no sistema.
DataInicio	Date	Data de início dos testes.
DataTermino	Date	Data de fechamento dos testes.
Aberto	Varchar (1)	Pode ser zero para fechado ou um para aberto.
Sucesso	Varchar (1)	Pode ser zero para testes com problemas ou um para testes realizados com sucesso.

Quadro 10 – Entidade do resultado do formulário de testes

Entidade: Um_RespostaResultadoTeste		
Descrição: Respostas dos resultados do formulário de testes		
Atributo	**Tipo**	**Descrição**
Id	Numeric (8,0)	Identificador.
Id_ResultadoTeste	Numeric (8,0)	Resultado de um formulário de teste.
Id_Pergunta	Numeric (8,0)	Pergunta do formulário de teste.
Sucesso	Varchar (1)	Pode ser zero para testes com problemas ou um para testes realizados com sucesso.
Problemas	Varchar (4000)	Problemas encontrados nos testes da pergunta.
OutrosProblemas	Varchar (4000)	Outros problemas encontrados nos testes desta pergunta, porém não possuem relação à mesma.
Sugestoes	Varchar (4000)	Sugestões a serem consideradas nesta pergunta.

Quadro 11 – Entidade da resposta do resultado do formulário de testes

Entidade: Um_SolicitacaoAtualizacaoVersao		
Descrição: Solicitação de atualização de versão do sistema		
Atributo	**Tipo**	**Descrição**
Id	Numeric (8,0)	Identificador.
Id_Licenca	Numeric (8,0)	Licença (Tabela externa).
Id_Solicitante	Numeric (8,0)	Usuário solicitante (Tabela externa).
Id_Responsavel	Numeric (8,0)	Usuário responsável (Tabela externa).
Id_VersaoAtual	Numeric (8,0)	Versão atual do cliente (Tabela externa).
Id_VersaoDesejada	Numeric (8,0)	Versão desejada (Tabela externa).
Status	Varchar (1)	Situação da solicitação de atualização.
OutrosMotivos	Varchar (4000)	Outros motivos para solicitar a atualização.
MotivoRejeicao	Varchar (4000)	Motivo da rejeição da atualização.

Quadro 12 – Entidade da solicitação de atualização de versão

Entidade: Um_SolicitacaoAtualizacao_Novidade		
Descrição: Novidades selecionadas na solicitação de atualização de versão		
Atributo	**Tipo**	**Descrição**
Id	Numeric (8,0)	Identificador.
Id_SolicitacaoAtualizacao	Numeric (8,0)	Solicitação atualização de versão.
Id_Novidade	Numeric (8,0)	Novidade da versão (Tabela externa).

Quadro 13 – Entidade das novidades da solicitação de atualização de versão