

FACULDADES INTEGRADAS DE CARATINGA

FACULDADE DE CIÊNCIA DA COMPUTAÇÃO

**SISTEMA DE GERENCIAMENTO
DE CÓDIGO VERSIONADO
PARA SERVIDORES DE PRODUÇÃO**

REINALDO JOSÉ MOREIRA

CARATINGA
2012

REINALDO JOSÉ MOREIRA

**SISTEMA DE GERENCIAMENTO
DE CÓDIGO VERSIONADO
PARA SERVIDORES DE PRODUÇÃO**

Monografia apresentada à banca examinadora da Faculdade de Ciência da Computação das Faculdades Integradas de Caratinga, como requisito parcial para obtenção do título de bacharel em Ciência da Computação, sob orientação do professor Msc. Jacson Rodrigues Correia da Silva.

Caratinga

2012

REINALDO JOSÉ MOREIRA

**SISTEMA DE GERENCIAMENTO
DE CÓDIGO VERSIONADO
PARA SERVIDORES DE PRODUÇÃO**

Monografia submetida à Comissão examinadora designada pelo Curso de Graduação em Ciência da Computação como requisito para obtenção do grau de Bacharel.

Prof. Jacson Rodrigues Correia Silva
Faculdades Integradas de Caratinga

Prof. Glauber Luis da Silva Costa
Faculdades Integradas de Caratinga

Prof. Hebert Luiz Amaral Costa
Faculdades Integradas de Caratinga

Caratinga, 18/12/2012

AGRADECIMENTOS

Agradeço a Deus pela oportunidade a minha família que me deu a possibilidade de estudar e agradeço a todos os professores, amigos e colegas que me ajudaram na elaboração deste trabalho.

RESUMO

Os códigos fontes de aplicações desenvolvidas por uma empresa necessitam de um controle capaz de manter as versões de seus sistemas, chamados de produção, atualizados sem interferir na estrutura de segurança definida por uma hierarquia ou regra de negócio. Por esse motivo de segurança, devem existir níveis de autorização corretos quanto ao acesso ao servidor, evitando que nenhuma pessoa sem os devidos privilégios ou sem a documentação necessária acessem os servidores que alocam esses códigos. Com o objetivo de resolver o problema de atualização e documentação dos sistemas, foi desenvolvido nesse trabalho um sistema de atualizações de códigos fontes que proporciona o versionamento e a documentação das atualizações feitas nos sistemas em funcionamento no servidor. A metodologia baseia-se em uma série de rotinas (scripts e interface) que operam de maneira rápida, segura e com a menor necessidade de contato humano ao servidor ou ao sistema já em produção, para assim evitar problemas de privilégios e autorização.

O principal desafio do trabalho foi a necessidade de fornecer a possibilidade de recuar (rollback) ou um avançar (rollforward) determinadas versões de um aplicativo já em produção, garantindo que não ocorresse nenhum problema às modificações efetuadas, visto que o aplicativo já encontrava-se em uso pelos usuários finais. Os resultados apresentaram que a meta foi atingida e que o sistema desenvolvido garante uma atualização segura dos códigos, fornecendo aos programadores a possibilidade de desenvolver e atualizar os códigos sem acesso direto aos servidores e fornecendo à empresa a possibilidade de modificação dos aplicativos em produção. A utilização de tais sistemas de controle é fundamental para que uma empresa não tenha problemas de versionamento e atualização de código, além da garantia de acesso e perda de código por funcionários desatentos ou mal intencionados.

Palavras-chave: Subversion, BS7799, Ambiente segregado, Desenvolvimento compartilhado.

ABSTRACT

The application source code when developed by a company in need of a capable control of the maintaining versions with their systems, without interfering with the updated security structure defined by a hierarchy or business rule. Therefore security authorization levels must be adjust to control the access and to preventing access without privileges or without the necessary documentation of the codes. Aiming to solve the updating problem and system documentation this work was developed to control the source code update and providing versioning and documentation of actualizations made on servers systems running. The methodology is based on routines (scripts and interface) that operate faster, safer and without human contact on the server or on the running system when already in a production server, thus avoiding privileges problems and authorization.

The main challenge of this work was provide the possibility of rollback or rollforward certain versions of an application ensuring that does not occur any modifications made since the application already found itself in use by they users. The results showed that the system ensures a secure update of the code, giving developers the ability to develop and update the codes without direct access to the servers and providing the company the ability to modify applications in production . The use of such systems control is critical to any company that does not have versioning problems and code update, as well as ensuring access code and lost by careless or malicious employees.

Keywords: Subversion, BS7799, segregated Environment, Shared development

LISTA DE ILUSTRAÇÕES

Figura 1 - Túnel seguro e criptografado entre cliente e servidor.....	19
Figura 2 - SVN+SSH Utilizando túnel criptografado	19
Figura 3 - Exemplo de configuração de CRON.....	20
Figura 4 - Exemplo de tíquete de publicação	21
Figura 5 – Passos da execução do robô de publicação	23
Figura 6 - Gráfico comparativo I – Segunda parte da Entrevista	29
Figura 7 - Gráfico comparativo II – Primeira parte da Entrevista	30
Figura 8 - Exemplo de instalação do SVN	38
Figura 9 - Configuração do CRON.....	39
Figura 10 - Configuração do Servidor apache.....	40

LISTA DE TABELAS

Tabela 1 - Exemplo de utilização de políticas na definição de nome de repositórios.....	16
Tabela 2 - Scripts e suas etapas e funções	17
Tabela 3 – Participantes da pesquisa separados por cargo	27
Tabela 4 - Respostas versus Cargos	28
Tabela 5 - Resposta da Pesquisa por Pergunta	28
Tabela 6 - Sistema operacional.....	37
Tabela 7 - Pacotes e versões necessárias para o desenvolvimento do projeto.....	38
Tabela 8 - Módulos habilitados na configuração do Servidor WEB	41
Tabela 9 - Diferentes modos de acesso ao Subversion.....	41

LISTA DE ABREVIATURAS E SIGLAS

CSI “Computer Security Institute”

CERT “Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança”

NIST “National Institute of Standards and Technology”

SGSI “Sistemas de gestão de segurança da informação”

ABNT “Associação Brasileira de Normas Técnicas”

GCS “Gerenciamento de Configuração de Software”

SGBD “Sistema de Gestão de Base de Dados”

SUMÁRIO

1. INTRODUÇÃO	1
2. O VALOR DA INFORMAÇÃO E SUA PROTEÇÃO	3
2.1 A SEGURANÇA DO AMBIENTE COMPUTACIONAL.	4
2.1.1 Normas e a segregação de funções.	5
2.1.2 Segurança em ambientes com sistemas em produção.....	7
3. CONTROLE DE VERSÃO DE SOFTWARE	8
3.1 NECESSIDADE DE UM CONTROLADOR DE VERSÕES.....	9
3.1.1 Um controle de versão dentro do controle de versão	10
4. METODOLOGIA	13
4.1 O ROBÔ DE PUBLICAÇÃO	15
4.1.1 Arquitetura de scripts	17
4.1.2 Controlando o acesso usando Tunelamento seguro	18
4.1.3 Fases de execução do robô de publicação	20
4.1.3.1 Passos do robô de publicação	22
4.2 AVALIAÇÃO	25
5. RESULTADOS	27
6. CONCLUSÃO	31
7. TRABALHOS FUTUROS	32
REFERÊNCIAS	33
ANEXO I - Lista de pacotes	37
ANEXO II - Configurações do Sistema Operacional.....	38
ANEXO III - Configurações do apache	39
ANEXO IV - Configurações do Subversion.....	41
ANEXO V – SCRIPT de envio de tíquete - SVNDOWN.SH	42
ANEXO VI – SCRIPT de leitura de tíquete e atualização - SVNUP.SH	42
ANEXO VII – Script TCL/TK EXPECT de Criação de Projeto - attrPasswd_START.exp	44

ANEXO VIII – Script TCL/TK EXPECT de atualização de Projeto - attrPasswd_UPDATE.exp	45
ANEXO IX – Códigos da Interface de publicação.....	46
ANEXO X – Formulário de pesquisa de resultados.....	61

1. INTRODUÇÃO

A segurança da informação é um dos pontos cruciais para a sobrevivência das empresas e pessoas já que as informações são consideradas como um dos principais patrimônios das instituições. Sua aplicação não é somente sobre códigos-fonte de aplicativos e a não inserção de dados maliciosos e de falhas dentre os sistemas que essa empresa utiliza, mas também de prevenir a possibilidade de acesso às informações de maneira direta (BRASIL, 2007).

Toda alteração classificada como uma mudança no ambiente caracterizado como produção deve ser acessada exclusivamente por sistemas e programas previstos nas respectivas rotinas. O ambiente de produção é também definido como um ambiente mais seguro e mais robusto no qual os servidores e processos são acessados e utilizados pelos clientes finais da empresa.

As correções e evoluções de produtos, aplicativos e sistemas operacionais devem ser homologados em ambiente próprio, antes de serem implementados no ambiente de produção, devido, à possibilidade de ter indisponibilidade do ambiente (MIRANDOLA, MIRANDA e MARTINS, 2011). As alterações podem acarretar paradas e essas paradas, em prejuízo para a instituição ou empresa.

Este trabalho teve como objetivo apresentar o controle total dos códigos-fonte e gerenciar o acesso aos serviços e sistemas dos servidores. Foi utilizado um sistema de controle de versões SUBVERSION e a segregação de ambientes para garantir a atualização dos sistemas em funcionamento nos servidores de produção sem a necessidade de contato ou acesso direto dos desenvolvedores ao servidor de produção para atualizar os códigos dos sistemas. O desenvolvimento do trabalho é baseado em alguns itens da norma de segurança BS7799 (BS7799, 2012) que aconselham que o desenvolvedor evite fazer acessos diretos aos servidores de aplicação que estão em produção.

No capítulo 2 deste trabalho é apresentado o conceito básico da segurança da informação e os ambientes mínimos necessários para o desenvolvimento do trabalho. No capítulo 3, são abordados os conceitos e a necessidade do controle de versões e a possibilidade de desenvol-

vimento compartilhado através de ambientes segregados e a partir do o capítulo quatro é abordado o conceito do desenvolvimento do software/códigos que garantem a publicação dos códigos através do controle de versões, e nos capítulos seguintes é apresentado um modelo e exemplo de implementação e as configurações necessários para o uso prático e a possibilidade de trabalhos futuros.

2. O VALOR DA INFORMAÇÃO E SUA PROTEÇÃO.

Tenho uma segurança de que poderia me proteger contra as provocações, mas, de fato, há ações mais terríveis que não poderiam ser detidas por nenhuma segurança
(*Gary Kasparov*)

A informação é o principal patrimônio e o de maior valor de uma instituição e a segurança da informação é a proteção do conjunto de dados que tem como principal função e intenção de preservar o patrimônio para um indivíduo ou uma organização. Como citado pelo NIST (2004) a política de segurança é um documento importante ou o mais importante para o desenvolvimento de aplicações de sistema um de informação. Conforme Bishop (2004) os sistemas devem apoiar e orientar os procedimentos, normas e controles usados no projeto de arquitetura de segurança de TI onde os princípios básicos da segurança da informação podem ser caracterizados pela tríade chamada de Confiabilidade, Integridade e Disponibilidade (*Confidentiality, Integrity, Availability – CIA*).

Baseados na tríade CIA, a Confidencialidade pode ser definida como a propriedade que limita o acesso à informação tão somente às entidades legítimas, ou seja, aquelas autorizadas pelo proprietário da informação. A Integridade é a propriedade que garante que a informação manipulada mantenha todas as características originais estabelecidas pelo proprietário, incluindo controle de mudanças e ciclo de vida. A Disponibilidade é a propriedade que garante que a informação esteja sempre disponível para o uso legítimo por usuários autorizados pelo proprietário da informação.

Os principais conceitos de segurança da informação estão relacionados não apenas a segurança dos dados e da informação, mas também a dos sistemas e a infraestrutura que compõem todo o ambiente em si, onde a necessidade e controle de acesso são tão importantes quanto a proteção as falhas em aplicativos que podem fornecer acesso de usuários indesejados e tratamento incorreto dos dados e informações de uma instituição.

Na visão do administrador de um servidor/serviço, o acesso ao sistema operacional é considerado uma invasão de perímetro protegido independente da maneira como ou por quem foi feita. De qualquer maneira o acesso físico ou remoto ao servidor deve ser evitado ou restrito conforme mostrado por Ezingard (2006).

Nas seções seguintes são apresentados os conceitos de segurança no ambiente computacional com recomendações por normas e padrões de segurança além da criação de ambientes segregados com a finalidade de mostrar e classificar níveis de segurança e acesso aos ambientes.

2.1 A SEGURANÇA DO AMBIENTE COMPUTACIONAL.

A informação precisa ser protegida adequadamente, sendo especialmente importante quando os negócios são cada vez mais interconectados e possibilitando o aumento de falhas do ambiente de segurança devido à complexidade e números de interconexões. A segurança da informação é a proteção da informação de vários tipos de ameaças, garantindo assim a continuidade do negócio e minimizando seus riscos.

A segurança da informação não pode ser considerada a mesma coisa que a segurança em Tecnologia da Informação (TI) ou segurança de ambiente computacional. Se o termo “segurança da informação” é usado da mesma forma que “segurança de TI”, isso significa que quase ninguém tem tomado decisões fundamentais e não técnicas de segurança que afetam os departamentos de TI. Um exemplo bem simples, a segurança do ambiente computacional se refere à proteção do Hardware e acessos ao sistema operacional e a segurança da informação é referente à proteção do software e sistemas WEB.

Entende-se que todo o ambiente computacional deve estar com um nível de segurança determinado como sistemas/ambientes de produção, podendo ser dentro ou fora da rede. Esses

sistemas devem ter o mais alto nível de proteção, pois esse será seu produto final dentro de um ambiente computacional, o trabalho visto pelos clientes. O objetivo desse trabalho foi apresentar a proteção que deve ser fornecida ao servidor de produção e a possibilidade e recomendação de um ambiente de homologação, sendo que para definir os níveis de segurança de cada ambiente é necessário conhecer algumas normas e a necessidade de segregação de funções.

2.1.1 Normas e a segregação de funções.

As recomendações de utilização e técnicas de segurança foram baseadas nos Sistemas de gestão de segurança da informação (SGSI), especificamente sob a norma *British Standard 7799* (BS7799, 2012). Sendo este um conjunto de boas práticas a serem aplicadas na infraestrutura, operação e manutenção de serviços e outros modelos de governança e gestão de TI. Criada em 1995, é utilizada como referência mundial em relação à segurança da informação assegurando que sejam adotados padrões rígidos de segurança como citado por Bon (2007).

A BS7799 torna-se cada vez mais importante para as organizações de todo o mundo, pois além de garantir que as informações internas sejam gerenciadas de forma mais segura, comprova aos clientes que a empresa dispõe de controles adequados para a proteção das informações. Ao ser traduzida e regularizada para os padrões brasileiros, a norma foi nomeada e aprovada como ABNT NBR ISO/IEC 27002:2005 Tecnologia da informação – Técnicas de segurança – Código de prática para a gestão da segurança da informação (ABNT, 2012).

A norma aborda de forma sucinta a segregação de ambientes, definindo-a e simplificando-a de maneira que todos os sistemas devem ser separados por estrutura física ou lógica dentro do mesmo ambiente.

“Convém que funções e áreas de responsabilidade sejam segregadas para reduzir as oportunidades de modificação ou uso indevido não autorizado ou não intencional dos ativos da organização” ABNT NBR ISO/IEC 27002:2005 (ABNT, 2012).

Como escrito por Coleman (2008), o conceito chave da segurança de TI é a separação de funções, também conhecida como *Separation of Duties* (SoD), que determina o controle interno e a separação das funções associadas a seus privilégios especificamente determinada pelo nível de segurança de cada ambiente.

Para as pequenas organizações, a segregação ou a utilização de normas de qualidade ou de controle podem ser consideradas difíceis de serem implementadas, devido à necessidade de ter vários ambientes e servidores onde algumas vezes, o desenvolvedor é também o gerente do projeto. De qualquer maneira, convém que o seu princípio seja aplicado sempre que possível, mesmo que a informação seja considerada de pouca ou até mesmo sem importância. Porque conforme Ezingard (2006), a segregação de ambientes facilita a garantia da integridade e a disponibilidade da informação.

Para que o item de confiabilidade das informações seja garantido, é necessário que ocorra a segregação de funções, que cada ambiente segregado (desenvolvimento, homologação e produção) seja idêntico e dessa maneira será evitado que um código funcione em um ambiente e não funcione no outro. Para facilitar o entendimento, é possível citar um exemplo de necessidade de segregação: vários erros poderiam ocorrer se um desenvolvedor atualizasse a sua página web a cada 10 minutos sem os mínimos testes ou verificações de segurança. É possível até mesmo que ocorra a total indisponibilidade da página web, comprometendo a confiabilidade do ambiente.

Para solucionar o problema enfrentado pelo desenvolvedor, é possível e mais prático se o mesmo tivesse um ambiente de desenvolvimento para trabalhar e um ambiente de homologação para fazer os testes. Quando aprovado pelo cliente, o gerente então poderia enviar para a produção os dados já testados. Dessa maneira o ambiente de produção não fica comprometido nem instável.

2.1.2 Segurança em ambientes com sistemas em produção

Conforme Pfleeger e Pfleeger (2006), a segurança em ambiente de produção é o foco principal do gestor do ambiente, pois os sistemas operacionais, serviços e aplicativos funcionando em seus servidores podem conter riscos em seu código. O recomendado e necessário é proteger e reconhecer das principais ameaças aos seus sistemas e trabalhar nesta informação de forma proativa. Não apenas dificultando ao máximo as atividades e acessos autorizados ou não autorizados mas também identificando, registrando e documentando quaisquer tentativas que violem a política de segurança adotada.

Conforme descrito pelo Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança do Brasil (CERT), cada organização deve definir o que é uma ameaça em relação ao seu ambiente, um incidente de segurança em computadores. Logo, as regras e políticas de defesa e segurança ao ambiente de produção dependem e variam em cada organização, pois dependerá muito de quanto vale a informação para a empresa que está protegendo seu ambiente de produção.

O controle do código no ambiente de produção garante que, no mínimo, todos os requisitos citados e todas as modificações possam ser verificadas, documentadas e comentadas, diferenciadas e avaliadas pelo gerente ou coordenador responsável do projeto. A atualização de certos serviços nem sempre são interessantes, pois podem introduzir mais vulnerabilidades e instabilidades ao ambiente do que a versão anterior.

Uma das maneiras de garantir a segurança do ambiente de produção é controlar o que entra e sai de dentro do servidor. Conforme Murta e Werner (2007), é necessário que alguns requisitos garantam o controle rápido de versões a fim de limitar os problemas de software e identificando os problemas antes de entrarem em produção, sendo um dos principais parâmetros de segurança a segregação dos ambientes de produção, homologação e desenvolvimento e também o controle de versões e o controle de acesso ao ambiente segregado.

3. CONTROLE DE VERSÃO DE SOFTWARE

Em seu livro, Collins-Sussman, Fitzpatrick e Pilato (2009) expõe que o software de controle de versões tem por finalidade resolver muitos problemas que ocorrem durante o desenvolvimento de software, já que sua causa costuma ser por falta de controle sobre os arquivos desenvolvidos durante o projeto. Dentre os principais problemas que convenceriam qualquer desenvolvedor a usar um controlador de versão é a perda ou o esquecimento de qualquer código. Até mesmo se a perda for maliciosa é possível descobrir quem fez e quando fez, simplesmente utilizando a documentação do próprio controlador de versão.

O sistema de controle de versão (ou versionamento) é um software com a finalidade de gerenciar diferentes versões de um documento qualquer. Esses sistemas são comumente utilizados no desenvolvimento de software com a finalidade de documentar o desenvolvimento do projeto e a linha de evolução, gerenciamento dos trabalhos compartilhados e até prazos de entregas de códigos das aplicações e documentações relacionadas à evolução para qualquer projeto. O sistema de controle de versão rastreia e controla todos os artefatos do projeto como código-fonte, arquivos de configuração, documentação, imagens e arquivos binários e desse modo consegue coordenar o trabalho paralelo de desenvolvedores. Através da disponibilidade de cada versão produzida de cada item do projeto, possibilita a existência de diferentes versões sendo trabalhadas ao mesmo tempo e estabelece uma política de sincronização de mudanças fornecendo um histórico completo do repositório (local onde estão armazenados os códigos fontes) e do projeto.

Como o repositório registra todas as alterações efetuadas, o sistema de controle de versão pode informar com facilidade quais as alterações que foram realizadas entre uma revisão e outra, quando isso ocorreu e quem fez as alterações. Essas informações permitem um controle maior sobre mudanças no projeto.

No desenvolvimento do projeto foi escolhido o Subversion (SVN) como controlador de versão porque é controlador de versão gratuito com código fonte aberto, mas poderia ser utili-

zado qualquer outro gerenciador de versões para o desenvolvimento do projeto como CVS, GIT, MERCURIAL, etc. A escolha do SVN foi devida a possibilidade de encontrar todas as funcionalidades e todos os passos de funcionamento e configuração em vários documentos e tutoriais encontrados com facilidade na Internet, além de livros gratuitos e pagos.

3.1 NECESSIDADE DE UM CONTROLADOR DE VERSÕES

Um das qualidades de um controlador de versões é a possibilidade de ter grupos de pessoas trabalhando simultaneamente no desenvolvimento de software mantendo um repositório central com as versões mais recentes dos arquivos e também resolvendo problemas de conflitos de desenvolvimento compartilhado.

Para o desenvolvedor ou a pessoa responsável pelo desenvolvimento ou infraestrutura, o sistema de versionamento é uma ferramenta que garante a documentação do trabalho de desenvolvimento e para as empresas, o sistema é uma alternativa eficiente e escalável. Seu uso intenso no suporte ao desenvolvimento e a sua adequação a grandes projetos e a equipes distribuídas geograficamente durante o desenvolvimento mantém todo o patrimônio intelectual na própria instituição.

Segundo Collins-Sussman, Fitzpatrick e Pilato (2009), o principal objetivo do controlador de versões é dar à equipe de desenvolvimento a possibilidade e a liberdade de errar, sendo muito mais fácil acertar quando você sabe que pode errar várias vezes para encontrar a melhor alternativa. Entretanto, é importante lembrar que a ferramenta de controle de versão não substitui o processo de Gerenciamento de Configuração de Software (GCS) que é o processo de identificar, organizar e controlar modificações ao software sendo construído. O GCS é uma considerável mudança cultural nas equipes de desenvolvimento e necessita de outros fatores, além do controlador de versão, para que o resultado seja bem sucedido. O trabalho de geren-

ciamento depende principalmente do analista, gestor do projeto e das regras de negócio serem bem definidas. Dessa maneira, facilita o desenvolvimento da aplicação para que o GCS seja bem implementado e otimizado principalmente no caso software com ambiente de desenvolvimento compartilhado.

O controlador de versões é uma ferramenta que auxilia e resolve a maioria dos conflitos do desenvolvimento compartilhado, reduzindo o número de problemas em um desenvolvimento compartilhado ou em grupo.

Os sistemas de versionamento são convenientes quando diversos desenvolvedores trabalham simultaneamente no mesmo projeto, resolvendo eventuais conflitos entre as alterações. Para que seja possível o trabalho em equipe, o sistema de controle de versão pode ter um ou mais sistemas de controle de usuários/acessos. Assim, é possível identificar cada usuário, que geralmente fica protegido por uma senha pessoal do sistema operacional ou do próprio ambiente de versionamento.

Em qualquer equipe de desenvolvimento, a confusão e o conflito no desenvolvimento de códigos é inevitável, pois mesmo em equipes bem definidas, duas pessoas podem desenvolver a mesma parte de código, bibliotecas, funções, ou classes.

3.1.1 Um controle de versão dentro do controle de versão

O SVN, como a maioria das ferramentas de controle de versão, permite a divisão do código-fonte de um projeto em uma hierarquia onde cada conjunto de código pode ser mantido no *Branch* (galho), na *Tag* (etiqueta) e no *Trunk* (tronco), sendo os mesmos entendidos respectivamente como Produção, Homologação e Desenvolvimento. Sua documentação reco-

menda a criação de uma estrutura inicial de diretórios com os mesmos nomes citados para ocorrer uma segregação de versões dentro do próprio repositório do projeto.

A diferença entre *Branch* e *Tag* é apenas conceitual. Nesse caso, poder-se-ia utilizar o *Branch* para controlar esses diferentes ambientes e fazer a integração *merge*, que é a fusão de dois setores ou partes de códigos que estão sendo desenvolvidos por uma pessoa ou por uma equipe em um mesmo arquivo, entre eles quando fosse necessário.

Para melhor entendimento, um exemplo de sua utilização seria: *Trunk* é onde se coloca toda sua estrutura da aplicação que é constantemente atualizada pelos desenvolvedores através do envio de dados *commit*. Quando houver a necessidade de criação de uma versão para produção, será necessário criar um *Branch* (na pasta *branches*) com a informação da revisão na qual se deseja criar. O mesmo procedimento irá ocorrer para homologação, onde será necessário criar um *Branch* para homologação no primeiro momento oportuno e a nova versão estiver disponível para homologação (a partir de uma revisão específica do seu repositório) será necessário fazer um *merge* entre uma revisão de código do *trunk* e o *branch* de homologação.

Segundo Nagel (2005), o SVN pode ser definido como um servidor/repositório de arquivos comuns, portando simplesmente a funcionalidade de lembrar e documentar quando e onde um arquivo/diretório foi alterado da última vez. A forma mais utilizada no controlador de versões é manter uma versão local onde são feitas as alterações e depois submetidas ao repositório. A pasta de trabalho na máquina local dos integrantes da equipe onde são baixados todos os arquivos para consulta e edição, pois cada pessoa tem sua forma de organizar os arquivos no computador, porém, recomenda-se que seja definido um padrão de diretório para todos os integrantes da equipe, pois existem arquivos que referenciam outros dentro do mesmo projeto, e se a pasta de trabalho não for padronizada, essas referências ficam desorganizadas e com erros, ou seja, com referências denominadas quebradas.

Algumas pessoas quando trabalham em modo corporativo preferem trabalhar em um único servidor e manter as versões diretamente dentro dele, essas pessoas geralmente trabalham somente em modo *lock* de arquivos, ou somente uma pessoa altera e atualiza cada arquivo por vez e raramente usam o *merge* para trabalhar. Isso irá depender do nível de conhecimento da equipe.

Ao desenvolver um software, um planejamento define que funcionalidades o produto deve ter ao término de uma versão. Além disso, ele mostra o estado do projeto a quem está acompanhando-o e o que esperar das futuras versões. Esses novos perfis de gerenciamento de versões podem ser continuados em futuros projetos baseados no atual.

4. METODOLOGIA

Este trabalho teve como finalidade desenvolver uma série de *scripts* e uma interface de publicação amigável para apoiar no controle das versões de softwares que serão colocados em um ambiente de produção segregado baseado em recomendações contidas na norma BS7799. Esta norma foi usada como padrão por ter seus capítulos liberados gratuitamente para consulta, o que não acontece na antiga americana ISO/IEC 17799 (2005) e a atual ABNT NBR ISO/IEC 27002 (2005).

Esse software pode ser usado por gerentes, analistas e desenvolvedores como ferramenta de auxílio com publicações de novas versões de software em ambientes sem acesso remoto ou físico e tem como principal finalidade o controle de versões e publicações além de documentar a história do desenvolvimento dentro do ambiente segregado e fechado.

O uso de ambientes segregados tem como principais objetivos evitar os conflitos de interesse, atos falhos, erros, fraudes, controle de falhas, roubo de informação e controle de segurança. Os ambientes desse projeto foram segregados/separados da maneira com a seguinte ordem de segregação:

- Ambiente de desenvolvimento: todos os envolvidos no desenvolvimento de um projeto ou código terão acesso diretamente ao servidor sem qualquer restrição e o cliente não deverá ter acesso a esse ambiente.
- Ambiente de homologação: somente os analistas e gerentes do projeto terão acesso a essa parte do projeto, já que os clientes poderão acessar, opinar e confirmar sobre versões dos sistemas.
- Ambiente de produção: nesse ambiente somente os gerentes do projeto terão acesso para que todas as atualizações sejam confirmadas e documentadas já que será o ambiente no qual o cliente final irá acessar, evitando qualquer atualização desnecessária ou alguma atualização que deixara o sistema inoperante. Para ter o controle completo no

acesso aos ambientes, principalmente o ambiente de produção, podemos definir que qualquer acesso aos servidores deve ser evitado e somente quando for necessário o acesso deverá ser totalmente monitorado e documentado. Assim, é possível evitar qualquer tipo de fraude ou reconfiguração do ambiente.

Após a verificação de que era necessário efetuar as tarefas que garantissem a publicação nos diversos ambientes, garantissem um controle de versão seguro e que possibilitassem a modificação de aplicações em tempo de produção de maneira fácil por uma interface descomplicada, foram estudadas as possibilidades de desenvolver scripts de publicação e a interface de acesso, utilizando várias ferramentas e linguagens como Perl, Java e PHP, onde foi optado por utilizar a linguagem BASH para os *scripts* devido à facilidade de montar um sistema fácil de manter e com o código aberto e para a interface de publicação foi usado o CAKEPHP devido à possibilidade de desenvolvimento rápida que o framework oferece.

Foi possível criar os scripts chamados de Robô devido à possibilidade executar ações sem intervenção humana proporcionando a possibilidade de fazer atualizações quase automáticas do ambiente. Para fornecer a função de publicação ao usuário gerencial, foi desenvolvido uma interface de publicação baseada em banco dados que gera um arquivo/Tíquete informacional que é enviado para uma pasta específica que o robô de publicação fica periodicamente verificando. A interface tem a como função secundária facilitar a intervenção do responsável da publicação de aprovar as versões à frente da atual e até voltar as versões de acordo com o desenvolvimento e a necessidade do projeto (descrito a partir da seção 4.1).

Após a implementação, o sistema foi homologado em um ambiente muito heterogêneo, devido a grande quantidade de códigos em desenvolvimento que foi o ambiente de homologação e produção do MEC (Ministério da Educação e Cultura), a possibilidade de acesso e a necessidade de uma ferramenta do tipo no ambiente. O retorno da utilização da ferramenta pode ser descrita em três etapas.

Primeira Etapa – Convencimento da equipe de administração de redes, o que foi o mais fácil e bem aceito devido a facilidade da equipe em gerenciar e negar acessos para publicações de códigos nos servidores de produção.

Segunda Etapa – Convencimento dos gestores e gerentes de projetos, que após um breve treinamento, aceitaram com tranquilidade devido à documentação que a ferramenta proporciona do tempo de desenvolvimento e a possibilidade de *Rollback* do ambiente de maneira segura e rápida.

Terceira Etapa – Convencimento da equipe de desenvolvimento, essa pode ser considerada a parte mais difícil de ser executada, pois os desenvolvedores que não estiverem acostumados com o desenvolvimento versionado teriam uma mudança de cultura que o desenvolvimento compartilhado proporciona, além da necessidade de treinamento na ferramenta escolhida para o controle de versão.

Depois de completamente implementada, a ferramenta foi bem aceita por todas as equipes, porque deixava explicitado tanto no SVN quanto na interface de publicação toda a documentação do desenvolvimento quanto as autorizações necessárias para a publicação dos ambientes, principalmente determinando nomes e donos aos mal feitos em publicações errôneas.

4.1 O ROBÔ DE PUBLICAÇÃO

Esse trabalho foi desenvolvido utilizando como sistema operacional o Linux e como distribuição o UBUNTU (2012) que tem o foco na usabilidade e facilidade de manutenção e instalação. O robô de publicação pode ser definido como uma série de *scripts* baseados no interpretador de comandos BASH (2010), além de uma interface em PHP (2010) de publicação para gerenciar e autorizar as publicações das versões dos códigos servidores. A interface de publicação foi desenvolvida com base no framework CAKEPHP (2010), que foi escolhido devido à facilidade de desenvolvimento e também por proporcionar um ambiente agradável

para o administrador ou os gerentes dos sistemas que irão controlar o versionamento e utilizar o robô de publicação. Nas próximas seções serão utilizados como teste a criação dos repositórios do sistema exemplo chamado de “monografia” para apresentar a manipulação e o gerenciado utilizando o robô de publicação e sua automação.

Para o gerenciamento de LOGS (Arquivos ou base de dados que contem informações e ações tomadas dentro do sistema) e controle de acesso à interface foi utilizado um Sistema de Gerenciamento de Banco de Dados (SGBD) Mysql (2012).

O que foi apresentado nesse trabalho foi outra maneira de trabalhar com o SVN. Um exemplo simples foi utilizada no seu desenvolvimento, onde foi criado um repositório independente para cada ambiente do projeto (Desenvolvimento, Homologação e produção). Os nomes dos repositórios serão apresentados na Tabela 1.

Política utilizada de nomeação de repositórios		
Nome do projeto	Ambiente	Nome final
Solução		
monografia	Desenvolvimento	monografia_des
monografia	Homologação	monografia_hom
monografia	Produção	monografia_pro

Tabela 1 - Exemplo de utilização de políticas na definição de nome de repositórios

Depois de definida a política de nomes e a publicação, é possível mostrar em etapas todo o processo da criação, envolvendo o envio do Tíquete entre os servidores até a atualização feita pelo processo do robô de publicação.

4.1.1 Arquitetura de scripts

Os scripts podem ser classificados como uma sequência lógica de instruções ou operações para automatizar a execução de aplicações, transformando-se em poderosas ferramentas de trabalho, principalmente na execução diária e em trabalhos continuados. Podem também ser definidos como arquivos executáveis, capazes de possuir em seu conteúdo instruções definidas, conhecidas e claras, que são executadas por um interpretador.

O interpretador de comandos que foi utilizado nos *scripts* foi o BASH (denominando-os *bash-scripts*) que opera como um tradutor entre o sistema operacional e o usuário ao utilizar uma sequência de comandos de um trabalho rotineiro. Foi utilizada uma série de *bash-scripts* para facilitar e agilizar o desenvolvimento do trabalho. Os *bash-scripts* foram utilizados em diferentes etapas do sistema, da criação dos repositórios até a publicação final.

Na Tabela 2 é possível ver as etapas de execução e quais os scripts que executam a ação, além de como é feita a interação necessária para a troca de informação dentro do sistema.

Conhecendo o robô de publicação				
Nome do Script	Execução	Lingua-gem	Etapa	Função resumida
Interface de publicação	Humana	PHP	1	Criar Tiquete
svdown.sh	Automática	Bash	2	Enviar o Tiquete para o servidor Destino
svnup.sh	Automática	Bash	3	Interpretar o Tiquete
attrPasswd_START.exp ou attrPasswd_UPDATE.exp	Automática	Tcl TK	4	Interagir com as perguntas do Sistema Operacional
svnup.sh	Automática	Bash	5	Publicar os projetos baseados na informação do Tiquete

Tabela 2 - Scripts e suas etapas e funções

Alguns módulos de execução foram utilizados no *bash-script*. Foram adicionados os módulos attrPasswd_START.exp e o attrPasswd_UPDATE.exp que utilizam o comando EXPECT (Comando de interatividade) através de script TCL/TK(2012) que faz a autenticação de

um usuário padrão chamado REPORT. Esse é o usuário virtual responsável pela atualização dos sistemas, ele estará em todos os grupos de todos os sistemas ou repositórios que necessitarem utilizar o robô de publicação.

4.1.2 Controlando o acesso usando Tunelamento seguro

O Tunelamento é a criação de um túnel no qual as informações trafegam de maneira segura. Dentre as várias maneiras de se fazer esse tunelamento, foi utilizada a aplicação OPENSSH (2012) através do protocolo de *Secure Shell* (SSH).

Conforme escrito por Northcutt et. al. (2005), a conexão por tunelamento com o protocolo SSH é a maneira mais segura de trafegar as senhas e os dados, além de fornecer uma conexão por terminal diretamente no servidor onde ficarão os repositórios e os sistemas em produção. O SSH foi basicamente usado como um servidor de autenticação de usuários e também para criar um sistema de relação de confiança entre os servidores, fornecendo uma conexão segura quando trafegando os arquivos que estão sendo versionados no repositório de versões do SVN. Um exemplo de túnel seguro sendo criado através de uma rede baseada em conexão de cliente e servidor é apresentado na Figura 1 que representa como o tunelamento é feito entre o servidor de SSH e o cliente.

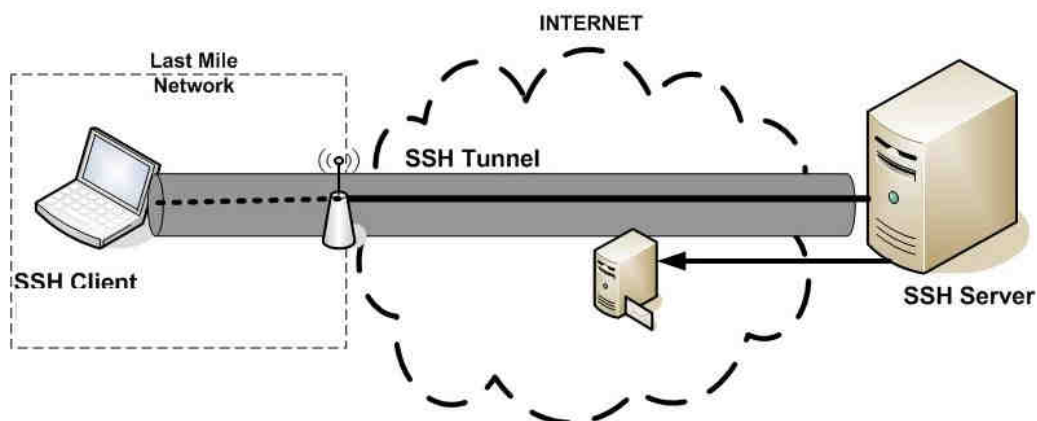


Figura 1 - Túnel seguro e criptografado entre cliente e servidor

Para garantir a comunicação segura tanto do tíquete quanto dos códigos trafegados, o SVN contém uma estrutura própria também baseada em túnel por SSH que garante o recebimento das informações de maneira criptografada e sem nenhuma intervenção complexa de criação, bastando adicionar um parâmetro na execução do comando do SVN, como mostrado na Figura 2.

A captura de tela mostra uma janela de terminal com o prompt 'reinaldo : bash'. O menu de aplicativos inclui 'Arquivo', 'Editar', 'Exibir', 'Histórico', 'Favoritos', 'Configurações' e 'Ajuda'. O comando executado é: `reinaldo@reinaldo-linux:~$ svn co svn+ssh://USUARIO@subversion.exemplo.com/svn/projeto projeto`. A barra de status na base da janela também indica 'reinaldo : bash'.

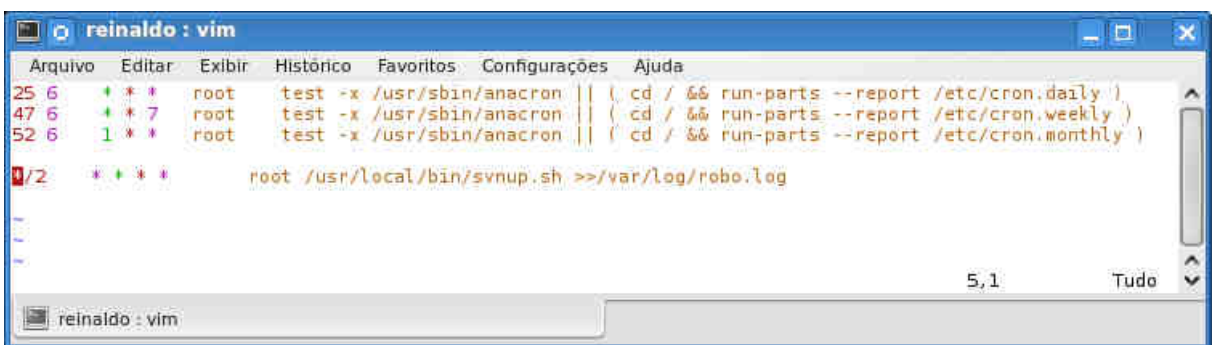
Figura 2 - SVN+SSH Utilizando túnel criptografado

Existem diversas maneiras de utilizar o SVN, mas a mais adequada devido à facilidade de implementação e à segurança no tráfego das informações, sendo então escolhida para o desenvolvimento deste trabalho foi a utilização da conexão por túnel usando SSH.

4.1.3 Fases de execução do robô de publicação

Nesse seção são apresentados quais os passos do processo de publicação e como foram divididas as etapas principais para a criação e envio do Tíquete ao recebimento e à publicação dos códigos. Essas etapas podem ser divididas em duas partes principais.

- A primeira parte é o envio, caracterizado como a criação e envio do tíquete para publicação. É criado um tíquete pela interface de publicação com as informações através das escolhas feitas na interface de publicação. As escolhas são pré-cadastradas no formato da política de nomes e a partir dessas informações o responsável autoriza a publicação pela interface. O tíquete é gerado e enviado a uma pasta do sistema operacional que é recolhido pelo robô de publicação. O robô de publicação é chamado para ser executado através do CRON (2012) que é um serviço agendador de tarefas do Linux e é enviado para o servidor de destino. Abaixo na Figura 3 segue um exemplo de configuração de CRON para executar o *script* do robô de publicação.



```
reinaldo : vim
Arquivo  Editar  Exibir  Histórico  Favoritos  Configurações  Ajuda
25 6     * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6     * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6     1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
0/2     * * * *   root    /usr/local/bin/svnup.sh >> /var/log/robo.log
5,1
Tudo
reinaldo : vim
```

Figura 3 - Exemplo de configuração de CRON

- A segunda parte é o recebimento, caracterizado como a interpretação do tíquete e publicação do sistema. O *script* de publicação interpreta qual projeto criar ou atualizar sendo um *RollForward* ou *RollBack* e através do *script* de interação, é executada a criação ou a atualização do ambiente através de comandos do SVN. O robô de publicação arquiva o tíquete com a finalidade de futuras revisões de código e LOGS. Para a interpretação das informações do tíquete são necessárias somente as informações de nome e tipo de servidor, do nome do projeto e da *release* (versão) a ser publicada para que o robô de publicação consiga executar a atualização necessária. Um exemplo é apresentado na Figura 4.

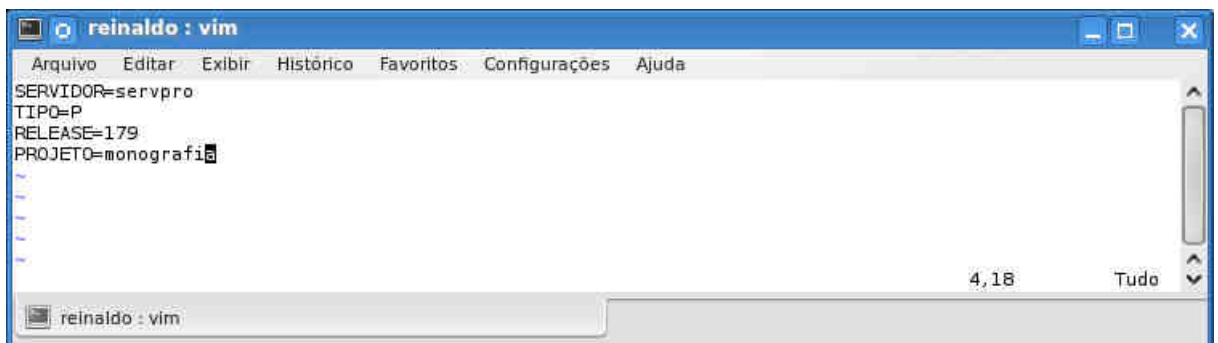


Figura 4 - Exemplo de tíquete de publicação

Para que as duas partes se comuniquem é necessário que ocorra uma relação de confiança entre as partes. Para que essa relação de confiança se estabeleça foi necessário criar uma chave de acesso e um ambiente criptografado através do túnel seguro, o tíquete é repassado de robô (envio) para o robô (recebimento) independente de onde os servidores se encontram na topologia de rede.

4.1.3.1 Passos do robô de publicação

Nessa subseção são detalhados os passos do robô de publicação e suas funções dentro do publicador. Podemos definir o robô como um tipo de software de interface à interface: da interface de publicação até a interface do sistema em um ambiente de produção. Na Figura 5 é mostrado todos os passos necessários para a publicação.

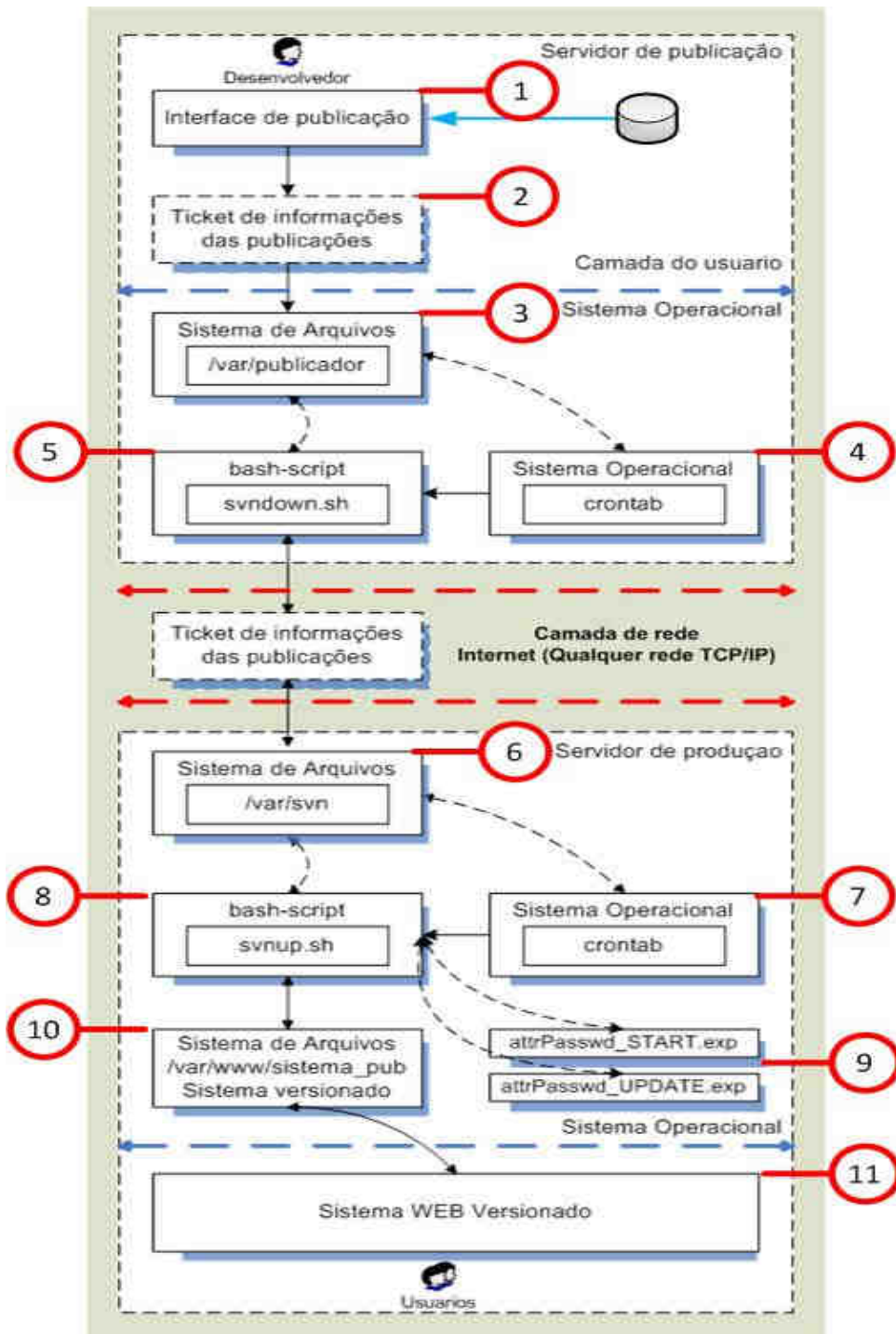


Figura 5 – Passos da execução do robô de publicação

- **Passo 1** - Acesso a interface de publicação: O desenvolvedor responsável pela publicação acessa a interface de publicação e escolhe os dados necessários. Como o sistema que será

atualizado e em qual ambiente é o principal que será a versão do software. Para facilitar a criação dos LOGS, o sistema guarda as informações dos tíquetes já publicados e os responsáveis pela publicação, além da hora da publicação e se foi feito algum *Rollback* de versão em um sistema de banco de dados;

- **Passo 2** – O tíquete é criado: É criado pela interface de publicação um arquivo texto com as informações fornecidas pelo publicador e o responsável da publicação;
- **Passo 3** – O tíquete vai para a pasta de publicação: O tíquete é salvo na pasta do sistema operacional “/var/publicador”, essa é a pasta onde o robô responsável pelo envio analisa se existe uma necessidade de publicação;
- **Passo 4** – O serviço do CRON executa a primeira parte do robô: A cada 2 minutos o serviço de agendamento do servidor Linux responsável pela publicação irá executar o *bash-script* “svndown.sh”, que é a parte responsável por interpretar e enviar o tíquete para o servidor no ambiente desejado;
- **Passo 5** – O robô de publicação envia o tíquete: O *script* “svndown.sh” interpreta as informações como servidor destino e as envia através de conexão segura por túnel SSH. Logo depois de enviado, o robô remove o arquivo da pasta, deixando-a vazia. No caso de vários tíquetes dentro da pasta, o robô executa cada um individualmente e não para de enviar até que a pasta esteja vazia novamente;
- **Passo 6** – O Tíquete chega à pasta de publicação: O tíquete é entregue pelo SSH na pasta “/var/svn”. Essa é a pasta onde o robô responsável pela publicação e execução do comando SVN procura os tíquetes para que sejam publicados e também é a pasta onde são salvos todos os tíquetes já publicados para futura análise, independente se foi feito um *RollForward* ou *RollBack*, os tíquetes nunca mais são aproveitados.
- **Passo 7** - O serviço do CRON executa a segunda parte do robô: A cada 2 minutos o serviço de agendamento do servidor Linux do servidor de produção irá executar o *bash-script* “svnup.sh” que é a parte responsável por interpretar as variáveis de publicação;

- **Passo 8** – O robô de publicação interpreta o tíquete: O *bash-script* “svnup.sh” executa os *scripts* que são responsáveis pela interação com o SVN e os módulos de autenticação verificam o usuário e sua permissão para que o sistema seja atualizado;
- **Passo 9** – Os *scripts* de interação verificam a publicação: Depois de interpretado, o comando de atualização é chamado pelo SVN e o *script* “attrPasswd_UPDATE.exp” verifica se o servidor e sua pasta onde estão localizados os códigos está pronta para receber a atualização. No caso de novos sistemas nunca publicados, o *script* “attrPasswd_START.exp” é chamado para criar toda a estrutura de pastas e para fazer a primeira publicação;
- **Passo 10** – Os *scripts* de interação fazem a publicação: Se a pasta, o sistema e as versões estiverem corretas e não corrompidas, o robô continua com a atualização e, para finalizar, ele troca as permissões das pastas e dos arquivos publicados naquela versão. Quando uma publicação está corrompida, o SVN cria uma série de arquivos que ele consegue interpretar para fazer novamente a publicação e corrigir os erros;
- **Passo 11** - O sistema é atualizado: O sistema foi atualizado e a nova versão é entregue para o usuário.

4.2 AVALIAÇÃO

Para a avaliação do sistema implementado, foi aplicado um questionário para os funcionários do Ministério da Educação MEC na DFTI Tecnologia da Informação. O intuito deste questionário foi conhecer as opiniões sobre os aspectos gerais do sistema, tais como facilidade de uso, funcionalidades, interface, compreensão do conteúdo e satisfação em usá-lo. O questi-

onário foi feito através de visitas aos departamentos e células de desenvolvimento do Ministério da Educação, além de pesquisa interna na DFTI sobre as regras de segurança e acesso.

5. RESULTADOS

Para analisar o ganho que este trabalho representa para a para o ambiente de uma empresa, foram realizados alguns testes de migração para o ambiente versionado com funcionários do Ministério da Educação e da DFTI Tecnologia da informação, Os resultados mostram o que há de aproveitamento e usabilidade no robô de publicação e também onde deve ser investido mais pela empresa para a implantação completa do robô em ambientes segregados.

Os testes foram feitos com 30 pessoas de diferentes cargos para saber onde o controlador de versões e o publicador tiveram melhor desempenho, a lista de cargos e a quantidade de pessoas por cargo da pesquisa segue na Tabela 3. A grande maioria dos entrevistados é de desenvolvedores porque são as pessoas mais afetadas na existência de um ambiente segregado e além do mais, o objetivo desse trabalho foi dar uma ferramenta para evitar o contato físico e remoto do desenvolvedor no ambiente de produção.

Cargo	Quantidade de pessoas	Empresa
Analista de Segurança	1	DFTI
Gerente de Redes	1	MEC
Analista de produção	5	MEC
Gerente de projetos	6	MEC
Desenvolvedores	15	MEC / DFTI
Analistas de testes	2	MEC
TOTAL	30	

Tabela 3 – Participantes da pesquisa separados por cargo

Na Tabela 4 é possível ver os números das respostas por cargo. Também é possível ver nos números que os desenvolvedores são as pessoas que mais tem um problemas pelo ambiente versionado e pela criação de documentações e também é possível notar que as pessoas

mais satisfeitas com o desenvolvimento do trabalho são as pessoas que gerenciam projetos ou gerencia a parte física ou infraestrutura computacional (P1 refere-se a pergunta de número 1).

Cargo	Quant.	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Analista de Segurança	1	1	0	1	1	1	0	1	1	1	1
Gerente de Redes	1	1	1	1	1	1	0	1	1	0	1
Analista de produção	5	2	3	4	5	5	0	5	5	5	5
Gerente de projetos	6	1	5	5	6	6	1	6	6	5	6
Desenvolvedores	15	5	12	9	14	15	7	15	15	15	15
Analistas de testes	2	0	0	1	2	2	1	1	0	2	2
TOTAL - SIM	30	10	21	21	29	30	9	29	28	28	30
TOTAL - NÃO		20	9	9	1	0	21	1	2	2	0

Tabela 4 - Respostas versus Cargos

Na Tabela 5 mostra em porcentagem das respostas referente às perguntas feitas para os entrevistados.

	1 - Você já utiliza o Subversion ou algum controle de versões?	2 - Você costuma usar algum sistema de documentação na criação de códigos?	3 - Achou fácil usar o Subversion?	4 - Você entende porque os códigos devem ser versionados?	5 - A interface de publicação é fácil de usar?
SIM	33%	70%	70%	97%	100%
NÃO	67%	30%	30%	3%	0%

	6 - Você prefere ter acesso ao servidor de produção do que deixar para o robô publicar?	7 - Você entende porque os ambientes devem ser segregados?	8 - Deu muito trabalho para fazer a migração de um sistema não versionado para um sistema versionado?	9 - Todas as publicações foram bem sucedidas?	10 - Depois dos testes, você usaria o robô normalmente no dia a dia?
SIM	30%	97%	93%	93%	100%
NÃO	70%	3%	7%	7%	0%

Tabela 5 - Resposta da Pesquisa por Pergunta

Através das respostas da Tabela 5 foi possível notar uma grande aceitação (100%) no robô de publicação (Perguntas 5 e 10), mostra também a falta de conhecimento ou interesse das pessoas (67%) quando o assunto é relacionado a ferramentas de controle de versão (Perguntas 1 e 2). Mesmo após a grande maioria (Perguntas 3 e 4) aceitar o uso e as facilidades do SVN muitos acharam difícil (93%) de fazer a migração para um ambiente totalmente versionado e mesmo aceitando completamente a usabilidade do publicador e entendendo a necessidade de criar um ambiente segregado algumas pessoas (30%) ainda preferem usar a atualização manual (Pergunta 6) do que usar o robô de publicação mesmo quando 93% das pessoas perguntadas conseguirem fazer atualizações bem sucedidas.

Na Figura 6 é mostrado no gráfico de maneira mais intuitiva a porcentagem da segunda parte das respostas e dos questionamentos da pesquisa. Essa parte da pesquisa refere-se basicamente a usabilidade e confiança do robô de publicação de acordo com a opinião dos entrevistados por isso será mostrada na ordem inversa no gráfico, aonde a última questão vem primeiro.

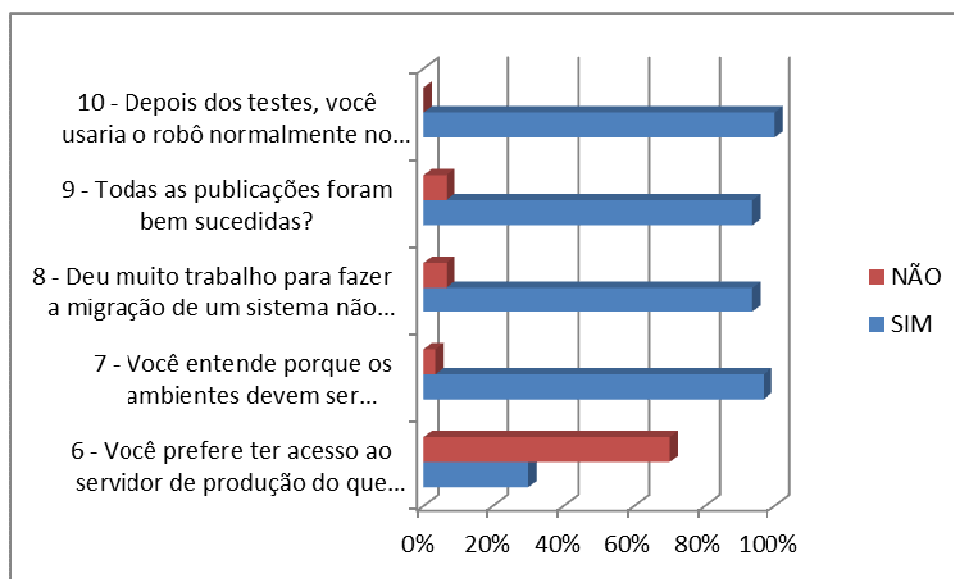


Figura 6 - Gráfico comparativo I – Segunda parte da Entrevista

Já na Figura 7 é possível mostrar a porcentagem da primeira parte das respostas da pesquisa. Essa parte da pesquisa refere-se basicamente ao conhecimento das pessoas em relação a ferramentas de versionamento e documentação de desenvolvimento.

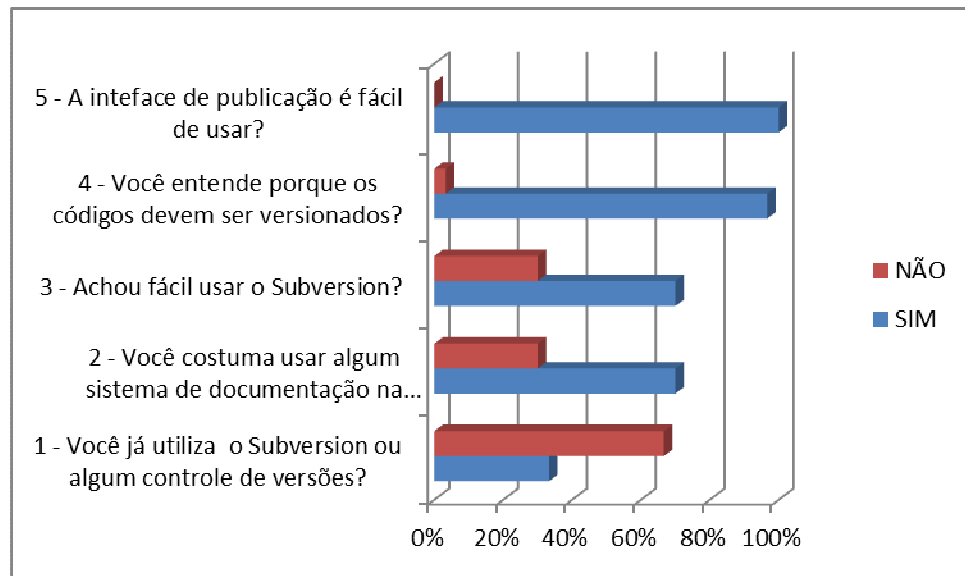


Figura 7 - Gráfico comparativo II – Primeira parte da Entrevista

6. CONCLUSÃO

O ganho no projeto foi a melhorar no desempenho, praticidade e a documentação dos códigos já que sempre são itens mais desejados por todos os gerentes de projetos. Mesmo que essas variáveis não agradem muito os desenvolvedores o controle de código e de acesso favorecem muito, não só a segurança do ambiente de produção, mas também na garantia da confiança e da integridade dos sistemas.

O presente trabalho apresentou como é possível atualizar sistemas de produção sem que ocorra acesso direto ao servidor e ainda garantindo a documentação e o controle de versões quase que obrigatório dos códigos. No trabalho, foi citado como e o porquê das normas e regras de publicação e segregação de ambientes, além de mostrado também os requisitos para a instalação, como a instalação pode ser feita e as recomendações de uso do robô de publicação.

Para avaliar o publicador foi aplicado um questionário, contendo 10 questões de avaliação e uma questão para comentar sobre o robô de publicação para trabalhos futuros, para os profissionais testados nos quais 30 pessoas de diferentes cargos, mas envolvidos no processo responderam a pesquisa. Nesta avaliação o robô de publicação teve uma aceitação de 100% dos profissionais que apesar de reclamarem que ele irá adicionar mais alguns passos ao desenvolvimento de códigos, definiram que utilizariam o robô de publicação como complemento ao ambiente de trabalho a fim de garantir uma melhor segurança devido ao ambiente segregado e sem acesso direto.

Concluindo, o robô de publicação foi muito bem aceito pelos profissionais entrevistados, alcançando o objetivo de ser utilizado como apoio ao trabalho em ambientes segregados.

7. TRABALHOS FUTUROS

Com base nos testes realizados com o robô de publicação usando SVN e a dicas recebidas durante a entrevista para verificação dos resultados, muitos trabalhos futuros podem ser desenvolvidos para melhorar a interface de comunicação com o desenvolvedor além do desempenho do software de publicação, como:

- Interface de integração com sistema operacional Windows;
- Migração dos scripts do robô para trabalharem em ambiente Windows com ASP e IIS;
- Melhorar a relação de confiança entre os servidores através de troca de certificado SSL;
- Criação de um serviço usando conexões por sockets sem a necessidade de criação de tíquetes de publicação;
- Criar um sistema para checar a integridade de arquivos binários através de MD5SUM para arquivos binários publicados.

REFERÊNCIAS

ALBING, Carl, VOSSSEN, JP, NEWHAM, Cameron. *Bash Cookbook: Solutions and Examples for bash Users*. Estados Unidos da América: Apress, 2007.

ABNT, Associação Brasileira de Normas Técnicas. *NBR ISO/IEC 27002: Tecnologia da informação – Técnicas de segurança – Código de prática para a gestão da segurança da informação*. Disponível em: <<http://www.abnt.gov.br>>. Acesso em: Novembro 2012.

BASH. Bourne Again SHell - *Bash Reference Manual*. Disponível em <http://www.gnu.org/software/bash/manual/bashref.html#What-is-Bash_003f>. Acesso em: Setembro 2012.

BERLIN, Daniel, ROONEY, Garrett. *Practical Subversion, Expert's Voice in Open Source*). Nova Iorque: Apress, 2 ed., 2006.

BISHOP, Matt. *Introduction to Computer Security*. Estados Unidos da América: Addison-Wesley Professional, 2004.

BON, Jan van, et al. *Foundations of IT Service Management Based on ITIL V3*. Estados Unidos da América: itSMF International, 3 ed., setembro 2007.

BRASIL, Tribunal de Contas da União. *Boas práticas em segurança da informação* - Tribunal de Contas da União. 2. ed. Brasília: TCU, Secretaria de Fiscalização de Tecnologia da Informação, 2007.

BS7799. *British Standard 7799*. Disponível em: <<http://www.itgovernance.co.uk/bs7799.aspx>>. Acesso em: Outubro de 2012.

CAETANO, Cristiano. *CVS: Controle de Versões e Desenvolvimento Colaborativo de Software*. Rio de Janeiro: Novatec, 2004.

CAKEPHP. *CAKEPHP Manual*. Disponível em: <<http://book.cakephp.org/1.2/view/3/The-Manual>>. Acessado em: Setembro de 2012.

CERT. Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança. Disponível em: <www.cert.br>. Acessado em: Novembro 2012

COLEMAN, Kevin. *Separation of duties and IT Security*. Disponível em: <<http://www.csoonline.com/article/446017/separation-of-duties-and-it-security>>. Acessado em: Outubro de 2012.

COLLINS-SUSSMAN, Bem, FITZPATRICK, Brian W., PILATO, Michael. *Version Control with Subversion*, for Subversion 1.4. Stanford: 2008. Disponível em <<http://svnbook.red-bean.com/en/1.4/svn-book.html>>. Acessado em: Agosto de 2009.

COLLINS-SUSSMAN, Bem, FITZPATRICK, Brian W., PILATO, Michael. *Version Control with Subversion – The Official Guide And Reference Manual*. Stanford, California, Estados Unidos da América: Soho Books, 2009.

CRON. *Intro to CRON*. Disponível em: <http://www.unixgeeks.org/security/newbie/unix/cron-1.html>. Acessado em: Novembro de 2012.

DAWEL, George. *A segurança da informação nas empresas, ampliando Horizontes além da Tecnologia*. São Paulo: Editora Ciência Moderna, 2007.

EZINGEARD, Jean-Noel. *Information Security Standards: Adoption Drivers*. Disponível em: <<http://www.springerlink.com/index/83444q4r6q221605.pdf>>. Acessado em: Outubro de 2012.

ISO/IEC 17799, ISO/IEC 27002. *The Information Security Standard*. Disponível em: <<http://www.17799.com>, <http://www.standardsdirect.org/iso17799.htm/>>. Acesso em: Outubro de 2012.

LEBLANC, Dee-Ann. *Linux System Administration Black Book: The Definitive Guide to Deploying and Configuring the Leading Open Source Operating System*. Califórnia, Estados Unidos da América: CoriolisOpen Press, 2009.

MIRANDOLA, Ederaldo Antônio. MIRANDA, Ivan Antônio Junior. MARTINS, Jeferson Adriano. Controle de Mudanças em Ambiente de Produção. Disponível em: <http://www.lyfreitas.com/ant/artigos_mba/controle-de-mudancas.pdf>. Acessado em: Setembro de 2012.

MURPHY, David J. *Managing Software Development with Trac and Subversion: simple project management for software development*. Estados Unidos da América: Packt Publishing, 2007.

MURTA, Leonardo Gresta Paulino. WERNER, Cláudia Maria Lima. Gerência de Configuração no Desenvolvimento Baseado em Componentes. PESCCOPPE – Universidade Federal do Rio de Janeiro, 2007. Disponível em http://www.ic.uff.br/~leomurta/papers/murta_2007.pdf>. Acessado em: Outubro de 2012.

MYSQL. *MySQL Reference Manual*. Disponível em <<http://dev.mysql.com/doc/>>. Acessado em: Setembro de 2012.

NAGEL, Willian. *Subversion Version Control: Using the Subversion Version Control System in Development Projects*. Estados Unidos da America: Prentice Hall PTR, 2005.

NIST, National Institute of Standards and Technology. *Engineering Principles for Information Technology Security (A Baseline for Achieving Security)*. Disponível em: <<http://csrc.nist.gov/publications/nistpubs/800-27A/SP800-27-RevA.pdf>>. Acessado em: Outubro de 2012.

NORTHCUTT, Stephen. ZELTSER, Lenny. WINTERS, Scott. KENT, Karen. RITCHEY, Ronald W. *Inside Network Perimeter Security*. 2 ed., Estados Unidos da America: Sams Publishing, 2005.

PFLEEGER, Charles P., PFLEEGER, Shari Lawrence. *Security in Computing*. 4 ed., Estados Unidos da América: Prentice Hall PTR, 2006.

PHP. *Manual do PHP*. Disponível em: <http://www.php.net/manual/pt_BR/index.php>. Acessado em: Setembro de 2012.

STANFIELD, Vicki, SMITH, Roderick W. *Linux System Administration: Craig Hunt Linux Library*. 2 ed., Estados Unidos da América, Califórnia: Sybex, 2008.

OPENSSSH. *Web Manual Pages*. Disponível em: <<http://www.openssh.org/manual.html>>. Acessado em: Setembro de 2012.

TCL/TK. *TCL Developer Exchange*. Disponível em: <<http://www.tcl.tk/>>. Acessado em: Novembro de 2012

UBUNTU. *Server*. Disponível em: <<http://www.ubuntu.com/business/server>>. Acessado em: Novembro de 2012.

ANEXO I - Lista de pacotes

Para o desenvolvimento do projeto foram utilizados os softwares e as versões abaixo:

O sistema operacional	
GNU/Linux – Kernel 2.6.X	Ubuntu 8.04 LTS Server/Intepriid

Tabela 6 - Sistema operacional

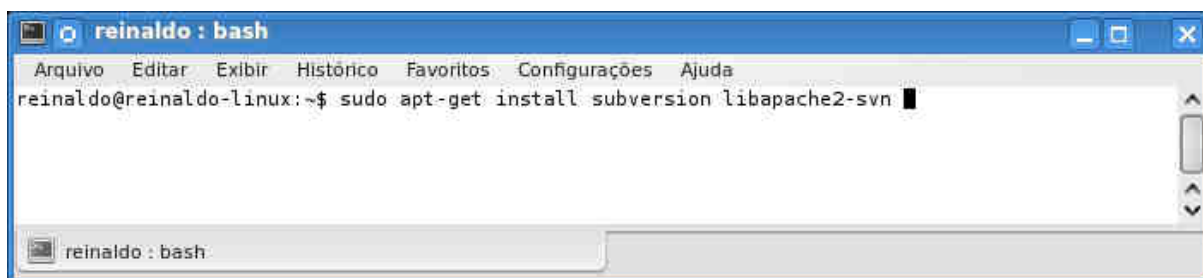
Softwares necessários		
Software	Versão	Descrição
apache2-mpm-prefork apache2-utils apache2.2-common	2.2.x	Servidor Web é um serviço que processa solicitações HTTP (Hyper-Text Transfer Protocol), o protocolo padrão da Web. Quando você usa um navegador de internet para acessar um site, este faz as solicitações devidas ao servidor Web do site através de HTTP.
php5 libapache2-mod-php5 php5-cli php5-common php5-curl php5-gd php5-mysql	5.2.x	Série de pacotes da linguagem e módulos do PHP que é uma linguagem de programação de computadores interpretada, livre e muito utilizada para gerar conteúdo dinâmico na World Wide Web.
subversion subversion-tools	1.4.x	Atualmente principal software cliente/servidor de controle de versões.

libapache2-svn		
mysql-client-5.0	5.0.x	Pacotes de banco de dados ou sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada).
mysql-common		
mysql-server-5.0		

Tabela 7 - Pacotes e versões necessárias para o desenvolvimento do projeto

ANEXO II - Configurações do Sistema Operacional

Todas as configurações serão feitas nos aplicativos e nada será feita no sistema operacional nem será fixado em uma determinada distribuição e visando a garantia da liberdade de escolha de distribuição das milhares existentes no sistema Linux.



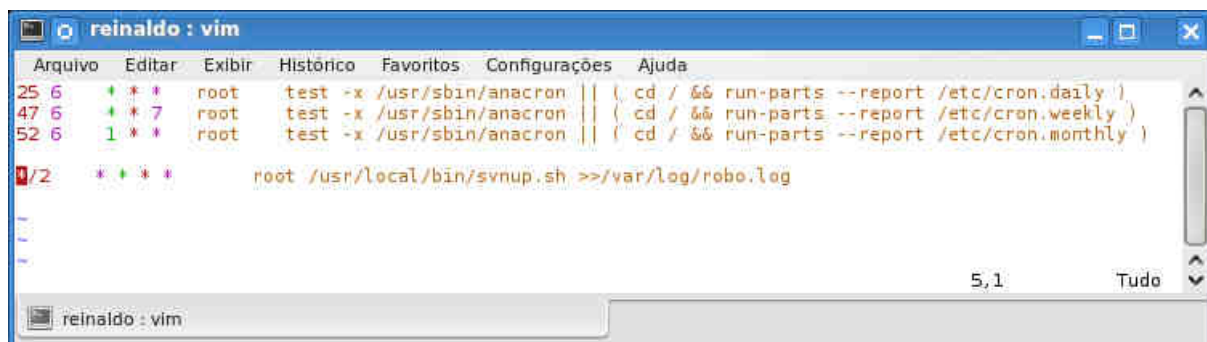
```

reinaldo : bash
Arquivo  Editar  Exibir  Histórico  Favoritos  Configurações  Ajuda
reinaldo@reinaldo-linux:~$ sudo apt-get install subversion libapache2-svn

```

Figura 8 - Exemplo de instalação do SVN

A principal configuração a ser feita no sistema operacional é a instalação de um comando no CRONTAB (Sistema de agendamento do Linux) do sistema operacional e tomando o cuidado do pacote do CRON estar instalado.



```
reinaldo : vim
Arquivo  Editar  Exibir  Histórico  Favoritos  Configurações  Ajuda
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
*/2    * * * *  root    /usr/local/bin/svnup.sh >>/var/log/robo.log
5,1     Tudo
```

Figura 9 - Configuração do CRON

Para a atualização do sistema o script de verificação (svnup.sh) será executado a cada 2 minutos, logo cada analista poderá definir a variação de horários dependente de cada sistema.

ANEXO III - Configurações do apache

O Servidor APACHE (2011) é um servidor WEB que provê serviços e é compatível com o protocolo HTTP e suas funcionalidades são mantidas através de da estrutura de módulos. Mas nada impede que outros servidores WEB possam ser instalados e testados para funcionar a única necessidade é a necessidade executar o Framework em PHP.

A única configuração necessária é a que habilita os websites do servidor apache como mostrada na imagem abaixo, essa configuração define qual pasta (diretório) e qual o domínio o servidor apache deve responder quando a requisição chegar do navegador pela internet.

Alguns módulos deverão ser habilitados no servidor apache. Esses módulos garantem o funcionamento e fazem toda a conexão e a troca de informações para que o sistema funcione corretamente.

```

reinaldo : vim
Arquivo  Editar  Exibir  Histórico  Favoritos  Configurações  Ajuda
NameVirtualHost
<VirtualHost *>
    ServerAdmin webmaster@localhost.com
    DocumentRoot /var/www
    ServerName subversion.localhost.com
    ErrorLog /var/log/apache2/subversion_error.log
    CustomLog /var/log/apache2/subversion.log combined
</VirtualHost>
1,17  Tudo
reinaldo : vim
  
```

Figura 10 - Configuração do Servidor apache

Módulos necessários no servidor apache	
php5.conf php5.load	Arquivo de configuração do módulo do PHP e arquivo de carregamento do módulo do PHP, usado para definir o endereço do módulo que será carregado.
rewrite.load	Módulo Rewrite para que .htaccess possa ser utilizado.
autoindex.load autoindex.conf	Arquivo de configuração do módulo do AutoIndex que evita que o usuário do apache possa listar os arquivos que estão em determinado diretório.
dav_svn.conf dav_svn.load	Módulo de autenticação do SVN para ser utilizado no servidor apache

Tabela 8 - Módulos habilitados na configuração do Servidor WEB

ANEXO IV - Configurações do Subversion

Nenhum item em especial é necessário ser configurado no aplicativo do Subversion, a configuração será a padrão da instalação, já que nenhum item em especial será tratado no Subversion e sim nos demais softwares que acompanham ou trabalham com ele.

A integração do Subversion com demais aplicações é feita através de uma camada de acesso que permite a implementação em diversos mecanismos de acesso.

Tipo	Método de Acesso
file://	Acesso diretamente no sistema de arquivos do servidor, acesso local
http://	Acesso utilizando um servidor WEB com módulos de acesso do Subversion habilitados
https://	Mesmo acesso Web, mas criptografia SSL
svn://	Acesso via protocolo svnserve e usando o próprio servidor do Subversion
svn+ssh://	Usando o mesmo servidor svnserve mas com a possibilidade de tunelar as informações por SSH

Tabela 9 - Diferentes modos de acesso ao Subversion

A arquitetura básica de funcionamento do SVN pode variar de acordo com aplicações que de acesso do usuário ou de acesso ao repositório, diversos softwares que podem ser usados como clientes de publicação podendo variar de acordo a opção de cada usuário.

ANEXO V – SCRIPT de envio de tíquete - SVNDOWN.SH

```
#!/bin/bash
# Script para envio de arquivo/Tíquete para os servidores que publicarão os
# códigos
# Desenvolvido por Reinaldo Moreira - reinaldo.j.moreira@gmail.com
DIR=/var/publicador
DIR_OK="$DIR/OK/"
if [ ! -d $DIR_OK ] ; then
    mkdir -p $DIR_OK
fi
for arquivo in `ls $DIR | grep -v OK`
do
    srv=`cat $DIR/$arquivo | grep SERVIDOR | cut -d= -f2`
    scp $DIR/$arquivo root@$srv:/var/svn/
    mv $DIR/$arquivo $DIR_OK$arquivo.ok
done
```

ANEXO VI – SCRIPT de leitura de tíquete e

atualização - SVNUP.SH

```
#!/bin/bash
# Script para atualização dos sistemas utilizando o subversion
# Desenvolvido por Reinaldo Moreira - reinaldo.j.moreira@gmail.com
DIR="/var/svn"
DIR_OK="$DIR/OK/"
USER="report"
PASS="123"
# Expect para passar a senha diretamente para o usuario report publicar
COMMAND_START="attrPasswd_START.exp" COMMAND_UPDATE="attrPasswd_UPDATE.exp"
# Expect para update
APACHE_REPO="/var/www/"
EMAIL="dgr@mec.gov.br"
LOG="/var/log/robo.log"
MENSAGEM="ERRO DE PUBLICACAO NO SERVIDOR"
if [ ! -d $DIR_OK ] ; then
    mkdir -p $DIR_OK
fi
for arquivo in `ls $DIR | grep -v OK` ; do
    if [ "$arquivo" == " " ];then
        exit 0
    fi
    # filtra as informações do Tíquete
    RELEASE=`cat $DIR/$arquivo | grep RELEASE | grep RELEASE | cut -d= -f2`
    PROJETO=`cat $DIR/$arquivo | grep PROJETO | cut -d= -f2`
    TIPO=`cat $DIR/$arquivo | grep TIPO | cut -d= -f2`
    if [ "$TIPO" == " " ];then
        exit 0
    fi
    # Seleciona a arvore de acordo com o ambiente
    case "$TIPO" in
        D)
            ARVORE=des
            ;;
        H)
            ARVORE=hom
            ;;
        P)
            ARVORE=pro
            ;;
    esac
    # Cria pasta do sistema
    if [ ! -d $APACHE_REPO/$PROJETO ] ; then
        $COMMAND_START $USER $PASS $PROJETO $APACHE_REPO $ARVORE
    fi
    # Entra na pasta do projeto e executa um update na release desejada
    cd $APACHE_REPO/$PROJETO
    # Linha do expect --- spawn "svn up -r$release"
    $COMMAND_UPDATE $PASS $RELEASE
    # Move arquivo para não ser publicado mais nada
    mv $DIR/$arquivo $DIR_OK/$arquivo.ok
    # Muda a permissao da pasta
    if [ -d /$APACHE_REPO/$PROJETO ];then
```

```

        chown -R www-data.www-data /$APACHE_REPO/$PROJETO
        chmod -R 755 /$APACHE_REPO/$PROJETO
    fi
    # Verifica erro de publicacao no log e envia email avisando
    TESTE="`tail -nl $LOG | grep Failed >/dev/null; echo $?`"
    if [ "$TESTE" = "0" ];then
        tail -nl $LOG | grep Failed | mail -s "$MENSAGEM `hostname` $PROJE-
TO" $EMAIL
        echo "$MENSAGEM - AVISADO NO $EMAIL" >> /var/log/robo.log
    fi
    cd - >/dev/null
done

```

ANEXO VII – Script TCL/TK EXPECT de Criação de Projeto - attrPas-swd_START.exp

```

#!/usr/bin/expect
# Script de interação com o subversion quando usando Tunel SSH
# Script de criação de projeto
# Desenvolvido por Reinaldo Moreira - reinaldo.j.moreira@gmail.com
set user [lindex $argv 0]
set passNew [lindex $argv 1]
set project [lindex $argv 2]
set address [lindex $argv 3]
set tree [lindex $argv 4]
spawn svn co svn://$user@subversion.mec.gov.br/svn/$project\_ $tree
$address/$project
expect "are you sure you want to continue connecting (yes/no)? $"
send "yes\r"
expect "Password: $"
send "$passNew\r"
expect "Password: $"
send "$passNew\r"
expect "Password: $"
send "$passNew\r"
Interact

```

ANEXO VIII – Script TCL/TK EXPECT de atualização de Projeto - attrPas-swd_UPDATE.exp

```
#!/usr/bin/expect
# Script de interação com o subversion quando usando Tunel SSH
# Script de Atualização de projeto
# Desenvolvido por Reinaldo Moreira - reinaldo.j.moreira@gmail.com
set passNew [lindex $argv 0]
set release [lindex $argv 1]
spawn svn up -r$release
expect "Password: $"
send "$passNew\r"
expect "Password: $"
send "$passNew\r"
expect "Password: $"
send "$passNew\r"
interact
```


ANEXO IX – Códigos da Interface de publicação.

acao.php

```
<?php
class Acao extends AppModel {

    var $name = 'Acao';
    var $validate = array(
        'no_acao' => VALID_NOT_EMPTY,
    );

    //The Associations below have been created with all possible keys,
    those that are not needed can be removed
    var $hasOne = array(
        'AcaoPai' =>
            array('className' => 'Acao',
                'foreignKey' => 'acao_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'dependent' => ''
            ),
    );

    /**
     * Retorna array com acoes filha do grupo passado
     *
     * Caso o grupo passado seja 0, retorna array com todas as acoes sem
    pai
     *
     * @param int $intGroup
     * @return array
     */
    public function selectGroup( $intGroup = 0 )
    {
        if( $intGroup == 0 )
        {
            $strSql = "SELECT *
            FROM acoes as Acao
            WHERE id = acao_id";
            return( $this->query( $strSql ) );
        }
        else
    }
}
```

```

        {
            $this->recursive = 0;
            return( $this->findAllByAcaoId( $intGroup ) );
        }
    }

    /**
     * Faz arvore de ações baseado em herança
     *
     * @param array $arrAcoes
     */

    function makeTree( $arrScrumble , & $arrChildren = null , $intPai = 0 )
    {
        $arrChildren['children'] = self::getChildren( $arrScrumble , $intPai );
        //debug( $arrChildren );
        if( count( $arrChildren['children'] ) )
        {
            foreach ( $arrChildren['children'] as & $arrValor )
            {
                // $arrValor['children'] = $arrChildren;
                self::makeTree( $arrScrumble , $arrValor , $arrValor['id'] );
            }
        }
    }

    /**
     * Retorna filhos de uma ação
     *
     * @param array $arrScrumble    Array com todas as ações
     * @param integer $intFather    Id do pai
     * @return array
     */
    function getChildren( $arrScrumble , $intFather = 0 )
    {
        $arrFather = array();
        foreach ( $arrScrumble as $arrValor )
        {
            if( $arrValor['Acao']['acao_id'] == $intFather )
            {
                $arrFather[] = $arrValor['Acao'];
            }
        }
        return $arrFather;
    }

    /**
     * Define ação como pai de um grupo
     *
     * @param array $data
     */
    public function setAsGroup( $data )
    {
        $arrAcao['Acao'] = array(
            'id' => $data['Acao']['acao_id'] ,
            'is_grupo_acao' => "1"
        );
        $this->save( $arrAcao );
    }
}

```

```

    public function teste()
    {
    }
}
?>

```

Acao_perfil.php

```

<?php
class AcaoPerfil extends AppModel {

    var $name = 'AcaoPerfil';

    var $useTable = "acoes_perfil";

    //The Associations below have been created with all possible keys,
    those that are not needed can be removed
    var $belongsTo = array(
        'Perfil' =>
            array('className' => 'Perfil',
                'foreignKey' => 'perfil_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'counterCache' => ''
            ),

        'Acao' =>
            array('className' => 'Acao',
                'foreignKey' => 'acao_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'counterCache' => ''
            ),
    );
    function saveAcao( $arrAcao , $perfil_id )
    {
        if( intval( $perfil_id ) )
        {
            $strSql = "DELETE FROM acoes_perfil WHERE perfil_id = "
.$perfil_id;
            $this->execute( $strSql );
        }
        foreach ( $arrAcao as $intAcao )
        {
            $strSql = "INSERT INTO acoes_perfil (acao_id, perfil_id) valu-
es(
                " . (int) $intAcao . " ,
                " . (int) $perfil_id . "
            )";
            $this->execute( $strSql );
        }
        return true;
    }
}
?>

```

Acao_projeto_usuario.php

```
<?php

class AcaoProjetoUsuario extends AppModel
{
    var $useTable = 'acao_projeto_usuarios';

    var $belongsTo = array(
        'Projeto' => array(
            'className' => 'Projeto',
            'foreignKey' => 'projeto_id'
        ),

        'Acao' => array(
            'className' => 'Acao',
            'foreignKey' => 'acao_id'
        ),

        'Usuario' => array(
            'className' => 'Usuario',
            'foreignKey' => 'usuario_id'
        )
    );

    /**
     * Salva ações do usuario
     *
     * @param array $arrAcoes
     * @param int $usuario_id
     * @param int $projeto_id
     */
    function salvarAcoes( $arrAcoes , $usuario_id , $projeto_id )
    {
        $strSql = "delete from acao_projeto_usuarios where usuario_id =
" . intval( $usuario_id ) . " and projeto_id = " . intval( $projeto_id );
        $this->execute( $strSql );

        if( ! is_array( $arrAcoes ) ) return true;
        foreach ( $arrAcoes as $intAcao )
        {
            $strSql = "INSERT INTO acao_projeto_usuarios ( acao_id ,
usuario_id , projeto_id ) values(
                " . (int) $intAcao . " ,
                " . (int) $usuario_id . " ,
                " . (int) $projeto_id . " )";
            $this->execute( $strSql );
        }
        return true;
    }
}
?>
```

Acao_usuario.php

```
<?php
class AcaoUsuario extends AppModel {

    var $name = 'AcaoUsuario';
    var $useTable = 'acao_usuarios';
    var $validate = array(
        'acao_id' => VALID_NOT_EMPTY,
        'usuario_id' => VALID_NOT_EMPTY,
    );

    //The Associations below have been created with all possible keys,
    those that are not needed can be removed
    var $belongsTo = array(
        'Acao' =>
            array('className' => 'Acao',
                'foreignKey' => 'acao_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'counterCache' => ''
            ),

        'Usuario' =>
            array('className' => 'Usuario',
                'foreignKey' => 'usuario_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'counterCache' => ''
            ),

    );

    public function saveActions( $arrUsuario , $arrAcao )
    {
        $this->execute( "delete from acao_usuarios where usuario_id = " .
intval( $arrUsuario['Usuario']['id'] ) );
        if( is_array( $arrAcao ) )
        {
            foreach ( $arrAcao as $intAcao )
            {
                $data['AcaoUsuario']['acao_id'] = $intAcao;
                $data['AcaoUsuario']['usuario_id'] = $arrUsua-
rio['Usuario']['id'];
                $strSql = "insert into acao_usuarios ( acao_id , usuario_id
) values( " . intval( $intAcao ) . " , " . intval( $arrUsua-
rio['Usuario']['id'] ) . " );";
                $this->execute( $strSql );
            }
        }
    }
}
?>
```

Error.php

```
<?php
class Error extends AppModel
{
    var $useTable = false;
}
?>
```

Linguagem.php

```
<?php
class Linguagem extends AppModel {
    var $name = 'Linguagem';
    var $validate = array(
        'ds_linguagem' => VALID_NOT_EMPTY,
    );
}
?>
```

Perfil.php

```
<?php
class Perfil extends AppModel
{
    var $name = "Perfil";
    var $useTable = 'perfis';
    var $validate = array(
        'ds_perfil' => VALID_NOT_EMPTY
    );

    var $hasMany = array(
        "AcaoPerfil" => array(
            'className' => "AcaoPerfil" ,
            'foreignKey' => 'perfil_id'
        )
    );
}
?>
```

Principal.php

```
<?php
```

```

class Principal extends AppModel
{
    var $useTable = false;
}

```

```
?>
```

Projeto.php

```

<?php
class Projeto extends AppModel {

    var $name = 'Projeto';
    var $validate = array(
        'no_projeto' => VALID_NOT_EMPTY,
        'sg_projeto' => VALID_NOT_EMPTY
    );

    //The Associations below have been created with all possible keys,
    those that are not needed can be removed
    var $belongsTo = array(
        'Linguagem' =>
            array('className' => 'Linguagem',
                'foreignKey' => 'linguagem_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'counterCache' => ''
            ),

    );

    var $hasMany = array(
        'ProjetoUsuario' =>
            array( 'className' => 'ProjetoUsuario' ,
                'foreignKey' => 'projeto_id'
            ),
        'ProjetoServidor' =>
            array( 'className' =>'ProjetoServidor' ,
                'foreignKey' => 'projeto_id'
            )
    );

    /**
     * Cria arquivo para a criacao do repositorio automatico no SVN
     *
     * Ao adicionar um projeto, eh criado esse arquivo para que o servidor
    leia
     * e automaticamente ja crie os repositorios de dev, hom, e prod do
    projeto
     *
     * @param string $sigla
     * @return boolean
     */
    function createFile( $sigla )
    {
        #$fileName = DIR_ARQUIVOS . "project/newproject-" . $sigla .
        ".txt";
    }
}

```

```

//$res = `subversion -p $sigla ----`;
//$res1 = `chown -R nobody.$sigla /svn/$sigla`;

//debug( $res );
//debug( $res1, 1);

#$content = "PROJETO=$sigla";
#if( @file_put_contents( $fileName , $content ) )
#{
#   return true;
#}
#else
#{
#   exit("erro ao escrever arquivo" );
#}
}

}
?>

```

Projeto_servidor.php

```

<?php
class ProjetoServidor extends AppModel {

    var $name = 'ProjetoServidor';
    var $useTable = 'projeto_servidores';
    var $validate = array(
        'projeto_id' => VALID_NOT_EMPTY,
        'servidor_id' => VALID_NOT_EMPTY,
    );

    //The Associations below have been created with all possible keys,
    those that are not needed can be removed
    var $belongsTo = array(
        'Projeto' =>
            array('className' => 'Projeto',
                'foreignKey' => 'projeto_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'counterCache' => ''
            ),

        'Servidor' =>
            array('className' => 'Servidor',
                'foreignKey' => 'servidor_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'counterCache' => ''
            ),

        'Servidorfuncao' =>
            array('className' => 'Servidorfuncao',
                'foreignKey' => 'servidorfuncoes_id',
                'conditions' => '',
                'fields' => ''
            )
    );
}

```



```

        'order' => '',
        'counterCache' => ''
    ),
);

public function saveServidores( $arrServidores , $arrProjeto )
{
    $strSql = "delete from projeto_servidores where projeto_id = " .
intval( $arrProjeto['Projeto']['id'] );
    $this->execute( $strSql );

    if( ! is_array( $arrServidores ) ) return true;

    foreach ( $arrServidores as $intId => $arrServidor )
    {
        $strSql = "insert into projeto_servidores (projeto_id , servi-
dor_id , servidorfuncoes_id ) values(
            " . (int) $arrProjeto['Projeto']['id'] . " ,
            " . (int) $intId . " ,
            " . (int) $arrServidor['servidorfuncoes_id'] . ")";
        $this->execute( $strSql );
    }
    return true;
}
}
?>

```

Projeto_usuario.php

```

<?php
class ProjetoUsuario extends AppModel {

    var $name = 'ProjetoUsuario';
    var $validate = array(
        'projeto_id' => VALID_NOT_EMPTY,
        'usuario_id' => VALID_NOT_EMPTY,
    );

    //The Associations below have been created with all possible keys,
    those that are not needed can be removed
    var $belongsTo = array(
        'Projeto' =>
            array('className' => 'Projeto',
                'foreignKey' => 'projeto_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'counterCache' => ''
            ),

        'Usuario' =>
            array('className' => 'Usuario',
                'foreignKey' => 'usuario_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'counterCache' => ''
            )
    );
}

```

```

    ),
);

public function saveUsuarios( $arrUsuarios , $arrProjeto )
{
    $strSql = "delete from projeto_usuarios where projeto_id = " . intval( $arrProjeto['Projeto']['id'] );
    $this->execute( $strSql );

    if( ! is_array( $arrUsuarios ) ) return true ;
    foreach ( $arrUsuarios as $intId => $arrUsuario )
    {
        $strSql = "insert into projeto_usuarios (projeto_id , usuario_id ) values(
            " . (int) $arrProjeto['Projeto']['id'] . " ,
            " . (int) $intId . " )";
        $this->execute( $strSql );
    }
    return true;
}
}
?>

```

Publicação.php

```

<?php
class Publicacao extends AppModel {

    var $name = 'Publicacao';

    //The Associations below have been created with all possible keys,
    those that are not needed can be removed
    var $belongsTo = array(
        'Servidor' =>
            array('className' => 'Servidor',
                'foreignKey' => 'servidor_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'counterCache' => ''
            ),

        'Usuario' =>
            array('className' => 'Usuario',
                'foreignKey' => 'usuario_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'counterCache' => ''
            ),

        'Projeto' =>
            array('className' => 'Projeto',
                'foreignKey' => 'projeto_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',

```

```

        'counterCache' => ''
    ),
    'Servidorfuncao' =>
        array( 'className' => 'Servidorfuncao' ,
              'foreignKey' => 'servidorfuncoes_id'
            )
    );

function salvarPublicacoes( $arrData )
{
    if( ! self::valida( $arrData ) ) return false;

    $projeto_id = $arrData['Projeto']['id'];

    if( ! is_array( $arrData['Publicacao'] ) ) return false;

    foreach ( $arrData['Publicacao'] as $intProjetoServidor => $arrPu-
blicacao )
    {
        if( ! ( $arrPublicacao['publicar'] ) ) continue;

        $projetoServidor = new ProjetoServidor();

        #busca informacoes do servidor
        $arrProjetoServidor = $projetoServidor->read( null , $intProje-
toServidor );

        #busca informacoes do projeto
        $this->Projeto->recursive = -1;
        $arrProjeto = $this->Projeto->read( null , $projeto_id );
        if( self::createFile(
            $arrProjeto['Projeto']['sg_projeto'] ,
            $arrProjetoServidor['Servidor']['no_servidor'] ,
            $arrProjetoServidor['Servidorfuncao']['ds_servidor_funcao']
        ,
            $arrPublicacao['nu_revisao']
        )
        )
        {
            $sql = "INSERT INTO publicacoes ( SERVIDOR_ID , USUARIO_ID
, PROJETO_ID , NU_REVISAO , DT_PUBLICACAO , SERVIDORFUNCOES_ID ) VALUES(
            " . (int) $arrProjetoServidor['Servidor']['id'] . " ,
            " . (int) $this->Session->getId() . " ,
            " . (int) $arrProjeto['Projeto']['id'] . " ,
            " . (int) $arrPublicacao['nu_revisao'] . " ,
            " . date("Y-m-d H:i:s") . " ,
            " . (int) $arrProjetoServidor['Servidorfuncao']['id'] .
" )";

            $this->execute( $sql );
        }
    }
    return true;
}

private function valida( $arrData )
{
    if( ! is_array( $arrData['Publicacao'] ) ) return false;

    foreach ( $arrData['Publicacao'] as $intProjetoServidor => $arrPu-
blicacao )
    {

```

```

        if( ! ( $arrPublicacao['publicar'] ) ) continue;

        if( ! $arrPublicacao['nu_revisao'] ) return false;
    }

    return true;
}

function createFile( $sigla , $servidor , $servidorfuncao , $release )
{
    $fileName = DIR_ARQUIVOS . $sigla . '-' . $servidor . '-' . substr(
    $servidorfuncao , 0 , 1 ) . '-' . $release . '-' . date("YmdHis") . ".txt";

    $content = "SERVIDOR=$servidor" . PHP_EOL .
        "TIPO=" . substr( $servidorfuncao , 0 , 1 ) . PHP_EOL .
        "RELEASE=$release" . PHP_EOL .
        "PROJETO=$sigla" . PHP_EOL;
    if( @file_put_contents( $fileName , $content ) )
    {
        return true;
    }
    else
    {
        exit("erro ao escrever arquivo" );
    }
}
}
?>

```

Servidor.php

```

<?php
class Servidor extends AppModel {

    var $name = 'Servidor';
    var $validate = array(
        'no_servidor' => VALID_NOT_EMPTY,
        'tiposervidor_id' => VALID_NOT_EMPTY,
    );

    //The Associations below have been created with all possible keys,
    those that are not needed can be removed
    var $belongsTo = array(
        'Tiposervidor' =>
            array('className' => 'Tiposervidor',
                'foreignKey' => 'tiposervidor_id',
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'counterCache' => ''
            ),
    );

}

/**
 * Prepara lista para preencher combo box
 *
 * @param array $arrServidores

```

```

    */
    public function doList( $arrServidores )
    {
        $arrList = array();
        foreach ( $arrServidores as $arrServidor )
        {
            $arrIndex = array(
                'id' => $arrServidor['Servidor']['id'] ,
                'nome' => $arrServidor['Servidor']['no_servidor'] ,
                'tipo' => $arrServidor['Tiposervidor']['ds_tipo_servidor']
            );

            $strIndex = str_replace( ' ' , "" , json_encode( $arrIndex ) );

            $arrList[ $strIndex ] = $arrServidor['Servidor']['no_servidor']
            . " ( " . $arrServidor['Tiposervidor']['ds_tipo_servidor'] . " )";
        }
        return ( $arrList );
    }
}
?>

```

Servidorfuncao.php

```

<?php
class Servidorfuncao extends AppModel {

    var $name = 'Servidorfuncao';
    var $validate = array(
        'ds_servidor_funcao' => VALID_NOT_EMPTY,
    );

}
?>

```

Tipo_servidor.php

```

<?php
class TipoServidor extends AppModel {

    var $name = 'TipoServidor';
    var $validate = array(
        'ds_tipo_servidor' => VALID_NOT_EMPTY,
    );

}
?>

```

Usuario.php

```

<?php
class Usuario extends AppModel {
    var $name = 'Usuario';
}

```

```

var $validate = array(
    'no_usuario' => VALID_NOT_EMPTY,
    'no_login' => VALID_NOT_EMPTY,
    'ds_senha' => VALID_NOT_EMPTY,
    'perfil_id' => VALID_NOT_EMPTY
);

var $belongsTo = array(
    'Perfil' => array(
        'className' => 'Perfil',
        'foreignKey' => 'perfil_id' ,
    )
);

function beforeValidate()
{
    if( array_key_exists( 'ds_senha' , $this->data['Usuario'] ) )
    {
        if( empty( $this->data['Usuario']['ds_senha_repeticao'] ) )
        || ! Validation::isEqual( $this->data['Usuario']['ds_senha' ] , $this-
>data['Usuario']['ds_senha_repeticao'] ) )
        {
            $this->validationErrors[ 'ds_senha_repeticao' ] = 1;
        }
        else
        {
            $this->data['Usuario']['ds_senha' ] = md5( $this-
>data['Usuario']['ds_senha' ] );
        }
    }

    if( array_key_exists( 'ds_senha_antiga' , $this-
>data['Usuario'] ) )
    {
        if( strcmp( md5( $this->data['Usuario']['ds_senha_antiga']
) , $this->Session->getDsSenha() ) != 0 )
        {
            $this->validationErrors[ 'ds_senha_antiga' ] = 1;
        }
    }

    return true;
}

/**
 * Verifica se as senhas passadas na inserção de usuário são iguais
 *
 * @param string $strSenha1
 * @param string $strSenha2
 * @return boolean
 */
function checkSenha( $strSenha1 , $strSenha2 )
{
    if( strcasecmp( $strSenha1 , $strSenha2 ) == 0 )
    {
        return true;
    }
    else
    {
        return false;
    }
}

```

```

}

/**
 * Prepara lista para preencher combo box
 *
 * @param array $arrUsuarios
 */
public function doList( $arrUsuarios )
{
    $arrList = array();
    foreach ( $arrUsuarios as $arrUsuario )
    {
        $arrIndex = array(
            'id' => $arrUsuario['Usuario']['id'] ,
            'nome' => htmlentities( $arrUsua-
rio['Usuario']['no_usuario'] ) ,
            'perfil' => htmlentities( $arrUsua-
rio['Perfil']['ds_perfil'] ) ,
            'perfil_id' => $arrUsuario['Usuario']['perfil_id']
        );
        $strIndex = str_replace( "'", '"', json_encode( $arrIndex
) );
        $arrList[ $strIndex ] = $arrUsua-
rio['Usuario']['no_usuario'] . " (" . $arrUsuario['Perfil']['ds_perfil'] .
" )";
    }
    return ( $arrList );
}
}
?>

```

ANEXO X – Formulário de pesquisa de resultados

PERGUNTA → RESPOSTA ↓	1 - Você já utiliza o Subversion ou algum controle de versões?	2 - Você costuma usar algum sistema de documentação na criação de códigos?	3 - Achou fácil usar o Subversion?	4 - Você entende porque os códigos devem ser versionados?	5 - A interface de publicação é fácil de usar?
SIM					
NÃO					

PERGUNTA → RESPOSTA ↓	6 - Você prefere ter acesso ao servidor de produção do que deixar para o robô publicar?	7 - Você entende porque os ambientes devem ser segregados?	8 - Deu muito trabalho para fazer a migração de um sistema não versionado para um sistema versionado?	9 - Todas as publicações foram bem sucedidas?	10 - Depois dos testes, você usaria o robô normalmente no dia a dia?
SIM					
NÃO					