CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

CollabDev: Gerenciador de Repositórios para Ambientes Colaborativos de Desenvolvimento

RODOLFO ADHENAWER CAMPAGNOLI MORAES

Marília, 2013

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

CollabDev: Gerenciador de Repositórios para Ambientes Colaborativos de Desenvolvimento

Monografia apresentada ao
Centro Universitário Eurípides de
Marília como parte dos requisitos
necessários para a obtenção do
grau de Bacharel em
Sistemas de informação
Orientador: Prof. Rodolfo Barros Chiaramonte

Marília, 2013

Campagnoli Moraes, Rodolfo

CollabDev: Gerenciador de Repositórios para Ambientes Colaborativos de Desenvolvimento / Rodolfo Adhenawer Campagnoli Moraes; orientador: Profº. MS. Rodolfo Barros Chiaramonte, SP: [s.n.], 2013.

62f.

Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) – Curso de Sistemas de Informação, Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, Marília, 2013.

1. Gerenciador Repositório 2. Controle Versão 3. Ambiente Colaborativo

CDD: 658.403801

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Rodolfo Adhenawer Campagnoli Moraes

CollabDev: Gerenciador de Repositórios para Ambientes Colaborativos de Desenvolvimento.

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Sistemas de Informação.

Nota: 09 (Nove)

Orientador: Rodolfo Barros Chiaramonte

1°. Examinador: Emerson Alberto Marconato

2°. Examinador: Adriano Bezerra

Marília, 02 de dezembro de 2013.

DEDICATÓRIA

Dedico esta monografia a minha família em especial aos meus pais Wagner Moraes e Selma Campagnoli Moraes que sempre me deram total apoio nos momentos difíceis da minha vida e me deram suporte e todas as condições necessárias para que eu pudesse chegar até aqui.

AGRADECIMENTOS

É difícil lembrar e agradecer a todas as pessoas que me acompanharam desde o início da graduação até aqui, então começo agradecendo primeiramente a minha família que é a base de tudo pelo incentivo e esforço para que eu superasse as dificuldades ao longo dos anos e concluísse a graduação.

Agradeço a meus amigos pelos anos de graduação, trabalhos, estudos em grupo, discussões e diferentes pontos de vista onde sempre contribuíram para o aumento do conhecimento coletivo e individual de cada um.

Agradeço também a todos meus grandes professores que me incentivaram desde o começo da graduação e transmitiram todo o conhecimento necessário para que eu pudesse ir vencendo os obstáculos ao longo do curso, em especial ao meu orientador Rodolfo Barros Chiaramonte.

Por fim agradeço a todas as pessoas que me ajudaram de alguma forma a superar os desafios ao longo dos anos de graduação, a todos vocês, meu muito obrigado.

"Nenhum homem realmente produtivo pensa como se estivesse escrevendo uma dissertação."
Albert Einstein

SUMARIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACL:	Access Control List
ADD:	Adicionar
CVS:	Concurrent Version System
DIFF:	Difference
GPL:	General Public License
HD:	Hard Drive
HTTP:	HyperText Transfer Protocol
HTTPS:	HyperText Transfer Protocol Secure
MIT:	Massachusetts Institute of Technology
MVC:	Model View Controller
PHP:	Hypertext Preprocessor
RM:	Remove
SCV:	System Control Version
SO:	Sistema Operacional
SSL:	Access Control List
SVN:	Subversion
TI:	Tecnologia da Informação

RESUMO

Um gerenciador de repositórios possibilita que se crie um ambiente de trabalho colaborativo onde todos os envolvidos em um determinado projeto trabalhem de forma simultânea. A criação e manutenção destes repositórios normalmente é feita manualmente por meio da execução de linhas de comando requerendo conhecimentos específicos. Propõe-se neste projeto o desenvolvimento de um sistema gerenciador de repositórios por meio de um ambiente web intuitivo e funcional que permita criar e manter um repositório com controle de grupos, usuários e suas respectivas permissões para ambientes colaborativos de desenvolvimento.

Palavras-chave: Subversion; SVN; GIT; Gerenciamento de Repositório; Controle de Versão; Ambientes Colaborativos.

ABSTRACT

A repository manager allows you to create a collaborative environment where everyone involved in a given project can work simultaneously. The creation and maintenance of these repositories is usually done manually by running command lines, requiring specific knowledge. This project aims to develop a repository manager by creating an intuitive and functional web environment allowing create and maintain a repository with control groups, users and their respective permissions to be used in collaborative environments of development.

Keywords—Subversion; SVN; GIT; Repository Management; Version Control; Collaborative Environments.

INTRODUÇÃO

O uso de um sistema de controle de versão visa fomentar a colaboração e o compartilhamento de dados no desenvolvimento de um projeto. Utilizando o controle de versão é possível prover um ambiente colaborativo de desenvolvimento para instituições de pesquisa e ensino, que estimule e acelere o crescimento de projetos em grupos com utilização de repositórios e controle de versionamento. "Em certo nível, a capacidade de várias pessoas modificarem e gerenciarem o mesmo conjunto de dados de seus próprios locais é o que fomenta a colaboração" (SUSSMAN et al, 2002).

Um ambiente colaborativo consiste em um repositório que é uma central de armazenamento de dados que armazena informação de forma hierárquica, assim como um típico servidor de arquivos, com a diferença de que um repositório utilizando controle de versão registra cada alteração feita ao longo do desenvolvimento do projeto, então qualquer usuário que faça parte do grupo colaborativo pode se conectar ao repositório e ter acesso a última versão dos arquivos e todo o seu histórico de alterações, bem como adicionar novos arquivos e estruturas de diretórios e disponibilizar para todo o grupo de colaboração.

Durante o desenvolvimento de uma pesquisa ou projeto, os colaboradores produzem artefatos que podem ser compartilhados com os membros da equipe e dependendo da sequência de realização das atividades o artefato produzido por um colaborador pode servir como artefato de entrada para outro.

Um gerenciador de repositórios é responsável pela manipulação de comandos de execução de um sistema de controle de versão, onde o usuário que esteja fazendo uso do gerenciador não precise saber os comandos específicos para interagir com o repositório, as execuções destes comandos são feitas através de uma interface gráfica amigável ao usuário.

Sem um gerenciador de repositórios para prover um ambiente colaborativo de desenvolvimento são necessários alguns conhecimentos específicos, entre eles, administração de repositórios, servidores, gerenciamento de usuários e suas permissões, sendo necessário fazer todas as interações através de linha de comando no servidor. Com o software gerenciador de repositórios busca-se minimizar o custo deste trabalho aumentando a eficiência e diminuindo a complexidade em se gerenciar um repositório.

Existem diversos sistemas de controle de versão, entre eles encontram-se o SVN, GIT, Mercurial, CVS entre outros, todos têm em comum o uso de repositórios para armazenamento de informações referente aos documentos versionados, como histórico de alterações de

arquivos e ramificações do projeto, um sistema de controle de versão pode possuir vários repositórios com diferentes colaboradores com permissões específicas para cada um.

Com isso é proposta a criação de um software gerenciador de repositórios que irá buscar prover um meio de facilitar a administração de repositórios, bem como seus grupos colaborativos, para que o ambiente em questão não seja uma dificuldade ou uma barreira no desenvolvimento de uma pesquisa/projeto e que a preocupação e foco seja totalmente voltado para o que está sendo desenvolvido no ambiente.

Para o desenvolvimento do software gerenciador de repositórios proposto foi feito um estudo e uma análise das operações adotadas pelos softwares de controle de versão, como eles lidam com o repositório e como é feita sua administração no servidor.

Antes de iniciar o desenvolvimento do software foi gerada uma documentação de especificação de requisitos para nortear o andamento do projeto, nesta etapa foram incluídos os seguintes itens: (I) Documentar os requisitos, (II) Realizar a especificação de software e (III) Gerar diagramas, o software foi desenvolvido de acordo com os artefatos gerados nesta etapa.

O software desenvolvido foi nomeado como CollabDev, a linguagem utilizada para o seu desenvolvimento foi o PHP apoiado sobre o *framework open-source* CakePHP que implementa a estrutura de organização MVC, que facilitou na organização e manutenção do software, o banco de dados foi o MySQL.

No Capítulo 1 - "Trabalho Colaborativo" é apresentado a utilização de um controle de versão para criação de um ambiente colaborativo de desenvolvimento destacando os princípios e possibilidades de colaboração no meio tecnológico através do trabalho colaborativo, abordando o conceito geral sobre controle de versão bem como seus tipos de classificação de gerenciamento de repositórios.

Atualmente existem diversos gerenciadores de repositórios que proporcionam um ambiente de desenvolvimento colaborativo, no Capítulo 2 - “Outras Ferramentas de Gerenciamento de Repositórios” as ferramentas foram separadas em dois grupos, proprietárias e *open source* onde foi feito um levantamento de suas características e funcionalidades para critérios de comparações com a ferramenta desenvolvida.

O capítulo 3 - "Proposta" apresenta a ferramenta CollabDev, suas funcionalidades, como é feito o gerenciamento de repositórios e o seu processo de desenvolvimento que definiu sua arquitetura que facilita na integração de novos sistemas de controle de versão.

Com a ferramenta desenvolvida, no capítulo 4 - "Avaliações e resultados" foram feitos comparativos com as ferramentas proprietárias e open sources encontradas atualmente no mercado a fim de analisar vantagens e desvantagens em relação ao CollabDev.

1. TRABALHO COLABORATIVO

Trabalho colaborativo é a distribuição de tarefas que podem ocorrer em quaisquer tipos de grupos e independente da especialidade, ocorre com cientistas de organizações de pesquisa, em equipes de empresas de desenvolvimento de software e em pesquisas e desenvolvimento de projetos do meio acadêmico. Seja qual for a área de atividade em que ocorra a colaboração, ela nunca é um relacionamento independente e sim subordinado à meta que se tem traçado para alcançar seus objetivos.

Pode-se dizer que as colaborações são exemplos de gerência por objetivo em que o foco sobre a meta é tão intenso que dispensa qualquer preocupação com formalidades gerenciais. Os grupos colaborativos se formam dentro de cada setor de atividade e tendem a responder questões específicas de suas áreas. Assim, o trabalho colaborativo terá mais chance de sucesso quando seus colaboradores, cada um em sua especialidade, expressarem suas ideias ligadas a uma meta comum, desenvolvendo esforços para realizar um propósito compartilhado.

Segundo Parrila (Parrilla, 2004), "grupos colaborativos são aqueles em que todos os componentes compartilham as decisões tomadas e são responsáveis pela qualidade do que é produzido em conjunto, conforme suas possibilidades e interesses".

Nas colaborações de sucesso não se adota qualquer método de gerência tradicional, ou técnicas de administração científica. Em geral, espera-se que o indivíduo, a quem foi designada uma responsabilidade específica, produza resultados dentro do cronograma e em um determinado nível de qualidade.

A colaboração é possibilitada pelos recursos computacionais que podem ser partilhados em tempo real, dando acesso a diferentes sistemas simbólicos como o texto, o som e a imagem, email e de fóruns de discussão, entre outros. "Estes recursos têm sido utilizados como meio de comunicação e de trabalho colaborativo por profissionais e estudiosos cujas áreas de atividade abrangem vários domínios, entre eles, a medicina, a arte, a ciência, e a educação" (Schooler, E., 1996). Já Mclagan e Nel (Mclagan e Nel, 2000) dizem que "a colaboração é necessária para a melhoria da produtividade e para resolver problemas empresariais específicos".

Nos dias de hoje as tecnologias de trabalho colaborativo permitem que se criem ou gerenciem equipes de projeto independentemente de sua localização geográfica, essas tecnologias fomentam de certo modo o trabalho colaborativo que segundo Jamil (Jamil, 2001), "o trabalho colaborativo é a forma de trabalho, apoiada em um ambiente

tecnologicamente adequado, que possibilita que todos os profissionais trabalhem de forma simultânea no projeto ou atividade em que estão envolvidos".

Uma destas tecnologias que possibilita o trabalho colaborativo é o controle de versão, "o sistema de controle de versões consiste, basicamente, em um local para armazenamento de artefatos gerados durante o desenvolvimento de sistemas de software" (MASON, M, 2006). Provê a colaboração de vários desenvolvedores em paralelo sobre o mesmo projeto e até o mesmo arquivo, sem que um sobrescreva o trabalho do outro, o controle de versão é primordial para que se fomente um ambiente de trabalho colaborativo, nele é possível manter diferentes linhas de evolução de um projeto com diferentes colaboradores.

1.1. CONTROLE DE VERSÃO

Um controle de versão proporciona o gerenciamento de mudanças de diretórios e arquivos de um determinado projeto, voltado mais especificamente para versionar código fonte de software, porém pode-se versionar quaisquer tipos de arquivos, tais como imagem, vídeos, textos e planilhas, possibilita voltar a versão específica de um arquivo sempre que necessário.

É composto por duas partes principais: o repositório e a área de trabalho. O repositório armazena todos os arquivos e histórico de mudanças de um projeto, assim como o autor e a data de cada alteração feita em cada arquivo. Os colaboradores de um determinado projeto que utilizam controle de versão não trabalham diretamente nos arquivos do repositório, todas as alterações são feitas em uma área de trabalho que contém uma cópia dos arquivos do projeto que é monitorada para identificar as mudanças realizadas.

A sincronização entre o repositório e a área de trabalho é feita pelo colaborador, que pode submeter ou requisitar uma atualização dos arquivos do projeto conforme mostra a Figura 1.

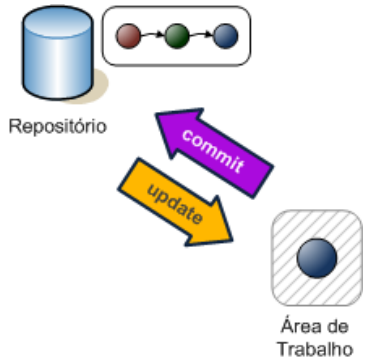

Figura 1 - Comunicação entre repositório e área de trabalho.

Os sistemas de controle de versão podem ser classificados em dois modelos de gerenciamento de repositório: centralizado e distribuído.

No modelo centralizado existe apenas um repositório central e várias cópias de trabalho como exemplificado na Figura 2. As operações de commit e update são responsáveis por consolidar as alterações e atualizar a cópia local que acontecem entre cliente e servidor.

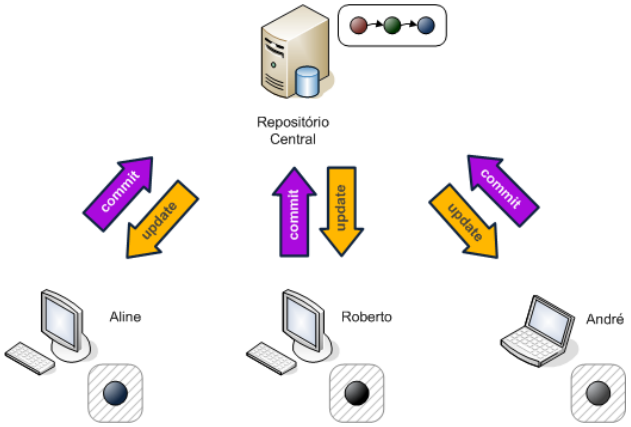

Figura 2 - Modelo Centralizado.

No modelo distribuído, ou descentralizado, existem vários repositórios autônomos e independentes, um para cada colaborador, e cada um desses repositórios possui uma área de trabalho acoplada a ele. Neste modelo as operações de commit e update acontecem localmente assim como mostra a Figura 3.

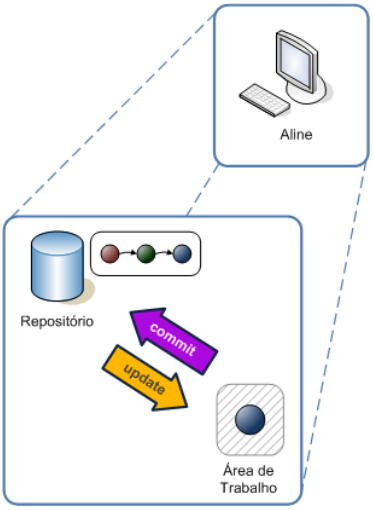

Figura 3 - No modelo distribuído cada colaborador possui um repositório próprio acoplado a sua área de trabalho.

1.1.1 Subversion

O Subversion é um SCV centralizado, entre suas características destaca se a portabilidade que é um de seus pontos forte, tanto o cliente como o servidor trabalham com os principais sistemas operacionais: Unix, Windows e Mac OS X, e sua portabilidade se deve ao subversion abstrair todas as diferenças entre os SO's para que sua utilização seja homogênea em todos eles, entre algumas divergências no modo de se trabalhar dos sistemas operacionais, podemos destacar o fato de plataformas Unix usarem permissões de sistemas de arquivo para determinar a executabilidade, enquanto no Windows são usadas as extensões no nome do arquivo, também são tratadas questões como a representação de finais de linhas diferentes entre Windows e Linux. Isso facilita o trabalho de colaboradores que precisam trabalhar em diferentes sistemas operacionais, a solução dos problemas causados por essas diferenças fez com que os comandos em linha de código da ferramenta fossem sempre os mesmos, independente do sistema operacional.

Diferentes tipos de protocolos de rede são suportados pelo Subversion, entre eles protocolos baseados em HTTP e HTTPS ou ainda seu próprio protocolo.

Permissões de acesso podem ser concedidas para diferentes partes do repositório remoto no Subversion, diferentemente de muitas outras ferramentas, que somente permitem definição de permissões para o repositório como um todo. Isso se deve ao fato do Subversion operar com permissões através do protocolo HTTP.

A cada commit, que é o envio das últimas alterações do código fonte ao repositório no Subversion é criado um changeset (conjunto de mudanças) diferente com informações como a

lista de arquivos alterados, quem realizou as alterações, mensagem de log e quando foi realizado. Cada operação de commit é atômica, ou seja, se a operação for interrompida pelo meio ela é desconsiderada. Isso permite, por exemplo, que em uma ocasião de queda de conexão com o servidor no momento em que era realizada uma operação de commit, o repositório não fique inconsistente.

Fluxo de trabalho

Os principais comandos do Subversion estão ligados ao seu fluxo básico de trabalho e fazem parte das seguintes operações:

Atualizar cópia de trabalho: através do comando svn update, que traz as alterações do repositório para a cópia de trabalho.

Realizar alterações: através dos comandos svn add, que adiciona arquivos, diretórios ou links simbólicos; svn delete, que exclui um item de uma cópia de trabalho ou do repositório; svn copy, que copia um arquivo ou diretório em uma cópia de trabalho ou no repositório e svn move , que move um arquivo ou diretório.

Verificar alterações: através dos comandos svn status, que exibe informações sobre o estado de arquivos e diretórios na cópia de trabalho e svn diff, que exibe as diferenças entre duas revisões ou caminhos.

Desfazer alterações: com o comando svn revert, que desfaz todas as edições locais.

Resolver conflitos: com os comandos svn update, que atualiza sua cópia de trabalho e svn resolved, que remove arquivos ou diretórios da cópia de trabalho do estado de “conflito”.

Submeter alterações: através do comando svn commit, que envia as alterações da cópia de trabalho local para o repositório.

A Figura 4 mostra o diagrama de representação do modelo de compartilhamento do Subversion.

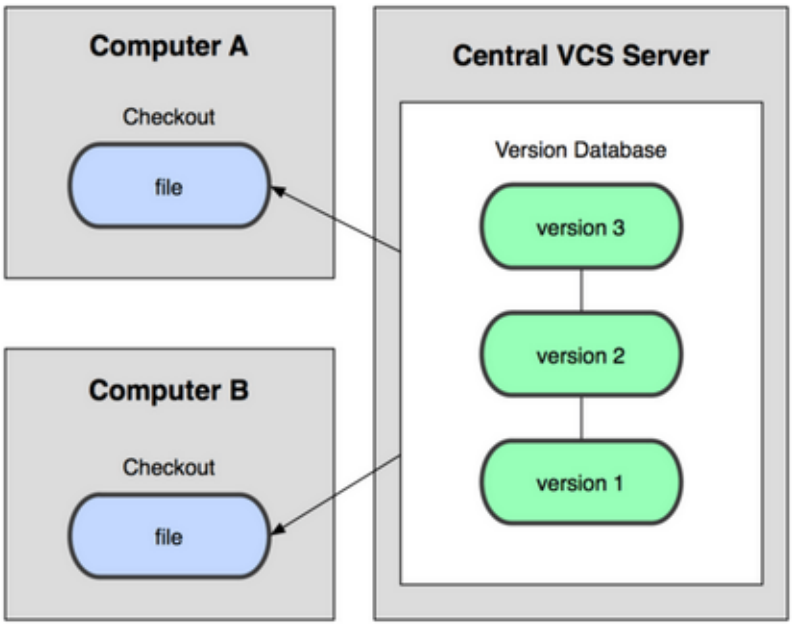

Figura 4 - Diagrama de representação do modelo de compartilhamento do Subversion.

Licença

O Subversion seguia os termos de licença do Debian Free Software Guidelines (DFSG), que define que qualquer pessoa é livre para baixar o código da ferramenta, modificá-lo, e redistribuí-lo conforme lhe convier; não sendo necessário pedir permissão aos seus criadores. Com o apoio da Apache Foudation, o software passou a adotar os termos da licença Apache. A licença Apache é uma licença de software livre de autoria da Apache Software Foundation (ASF). Ela requer preservação do aviso de direitos autorais e aviso legal, porém, permite o uso do código fonte para o desenvolvimento de software proprietário, bem como software livre e open source.

1.1.2 GIT

O GIT é um SCV distribuído, entre suas principais características estão, velocidade, design simples, suporte robusto a desenvolvimento não linear (branches paralelos, ramificações do projeto) e capacidade de lidar eficientemente com grandes projetos como o kernel do Linux (velocidade e volume de dados).

Ao contrário de sistemas de controle de versão centralizados, a natureza distribuída do GIT lhe permite ser muito mais flexível na forma como os desenvolvedores podem colaborar em projetos. Cada desenvolvedor pode contribuir para outros repositórios e ao mesmo tempo pode manter um repositório público em que outros possam basear seu trabalho e fazer

contribuições, isso abre uma vasta gama de possibilidades de fluxo de trabalho para equipes colaborativas.

Com o GIT é possível trabalhar com vários protocolos de redes diferentes, como por exemplo, rsync, ssh, HTTP, HTTPS ou pode trabalhar também com seu próprio protocolo. O protocolo git procura otimizar a largura de banda utilizada e por isso torna operações de atualizações rápidas e eficientes.

O GIT permite definir permissões tanto para o repositório como um todo, quanto para diretórios e arquivos. O controle dessas permissões pode ser feito de várias maneiras, como por exemplo, através do protocolo ssh, que permite criar chaves para controle de autenticação ou através do procotolo HTTP, que permite definir permissões de leitura e escrita para cada arquivo ou diretório.

Assim como no Subversion, no GIT a operação de commit é atômica, ou seja, se uma operação é interrompida, ela é desconsiderada e o repositório não fica em um estado inconsistente.

Fluxo de trabalho

Os principais comandos do GIT estão ligados ao seu fluxo básico de trabalho. O ciclo básico de trabalho pode ser definido com as operações comuns realizadas por um colaborador em seu dia a dia. Fazem parte do fluxo de trabalho do GIT as seguintes operações:

Atualizar cópia de trabalho: através do comando git fetch, que faz o download de alterações do repositório desejado, seguido do comando git merge, para mesclar as alterações baixadas com o seu repositório, ou através do comando git pull, que atualiza seu repositório e sua área de trabalho com as alterações de outro repositório.

Fazer alterações: através dos comandos git add, que adiciona arquivos ao index; git rm, que exclui um item de uma cópia de trabalho ou do repositório e git mv, que move ou renomeia um arquivo ou diretório.

Verificar alterações: através dos comandos git status, que exibe informações sobre o estado de arquivos e diretórios na cópia de trabalho e git diff, que exibe as diferenças entre duas revisões ou caminhos.

Desfazer alterações: com o comando git revert, que desfaz todas as edições locais.

Resolver conflitos: com o comando git mergetool, que executa ferramentas de resolução de conflitos.

Submeter alterações: através do comando git commit, que envia as alterações de sua cópia de trabalho para o repositório.

Propagar alterações: através do comando git push, que envia as alterações do repositório local (origem) para outro repositório (destino).

A Figura 5 mostra o diagrama de representação do modelo de compartilhamento do GIT.

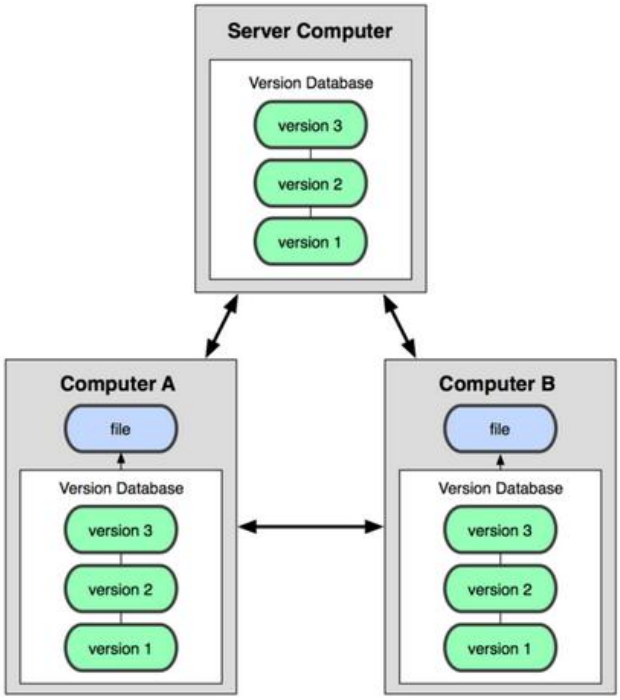

Figura 5 - Diagrama de representação do modelo de compartilhamento do GIT.

Licença

O GIT segue os termos da licença General Public License (GPL). A GPL foi criada por Richard Stallman em 27 de setembro de 1983 para o projeto GNU, um projeto de software livre. Existem três versões para a licença GPL. O GIT segue os termos da segunda versão.

2. OUTRAS FERRAMENTAS DE GERENCIAMENTO DE REPOSITÓRIOS

Foi realizado um levantamento de ferramentas que possuem o mesmo propósito do CollabDev e realizadas comparações e propostas de melhorias, as ferramentas encontradas se encaixam em dois grupos, que são: ferramentas proprietárias e ferramentas *open source* (distribuídas sob licença de software livre).

Dentre as ferramentas proprietárias destacam-se: XP-Dev, Bitbucket, GitHub e Assembla. Entre as ferramentas open source destacam-se: SVN-Access-manager, SVN Manager e GitLab.

Este capítulo irá apresentar o levantamento das características e funcionalidades de cada uma das ferramentas para critérios de comparações com a ferramenta proposta.

2.1 FERRAMENTAS PROPRIETÁRIAS

Ferramentas proprietárias são aquelas cujas cópias, redistribuição ou modificação são restritas pelo seu criador ou distribuidor e exigem algum tipo de pagamento através da compra ou plano de assinatura mensal, trimestral ou anual. Algumas disponibilizam planos gratuitos com restrição de uso, como no caso das ferramentas analisadas a quantidade de repositórios que podem ser gerenciados, o número de colaboradores por repositório, a possibilidade de se manter um repositório privado e entre outras, a fim de possibilitarem que os usuários possam conhecer as funcionalidades da ferramenta antes de efetivarem uma assinatura.

2.1.1 XP-Dev

Xp-Dev é uma ferramenta de gerenciamento de projetos e repositórios que possui suporte para os SCV's Subversion, GIT e Mercurial, o gerenciamento dos repositórios é feito através de ambiente web, oferece suporte a acesso com segurança SSL multiusuário com níveis de perfis. O repositório fica hospedado em servidores externos de domínio da empresa responsável pela ferramenta, esse servidores são dedicados com HD's criptografados e com backups em data center redundantes e independentes para garantir a consistência e integridade das informações hospedadas. Na Figura 6 é demonstrado a interface da ferramenta XP-Dev.

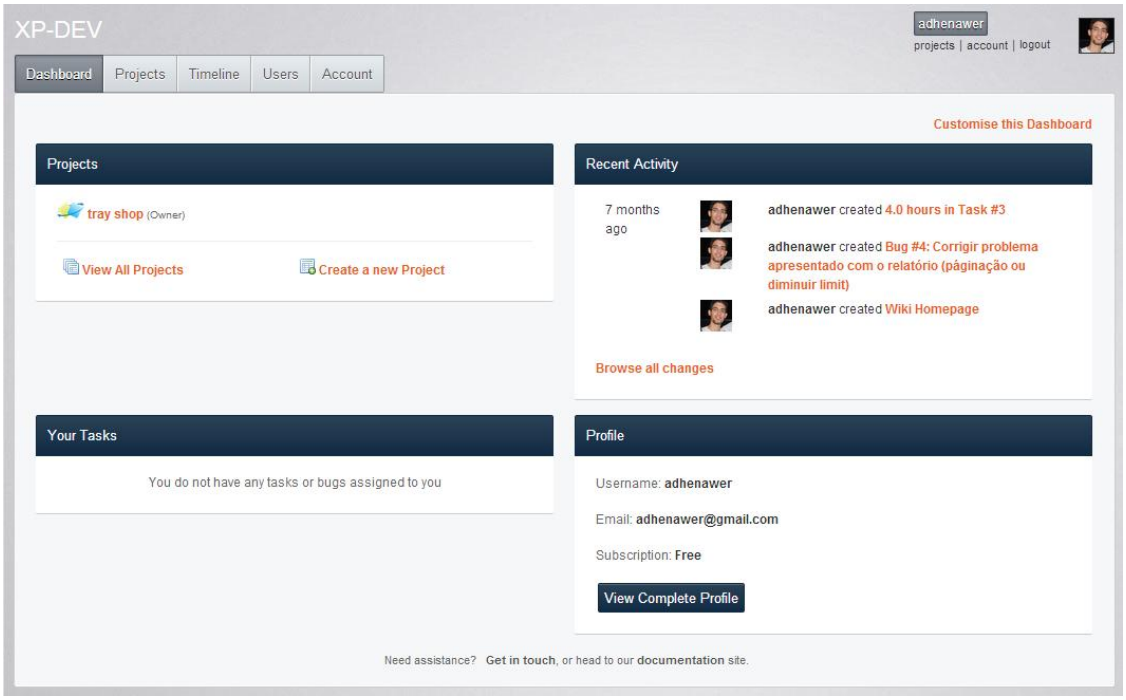

Figura 6 - XP-Dev.

Principais características e funcionalidades

Entre suas principais características e funcionalidades podemos destacar algumas como a gestão de repositórios com acesso através de interface web com segurança SSL, possui controle de usuários, grupos e suas permissões por repositório, interface para revisão de código, suporte à comunicação com blogs, fóruns, comentários e páginas de wiki. E suporte para sistemas de controle de versão Subversion, GIT e Mercurial.

2.1.2 Bitbucket

O Bitbucket por sua vez possui suporte para os SCV's GIT e Mercurial e além de possuir suporte para gerenciamento de repositórios através de ambiente web, possui clientes desktops para as plataformas Windows e Mac. Na Figura 7 é demonstrado a interface web da ferramenta Bitbucket.

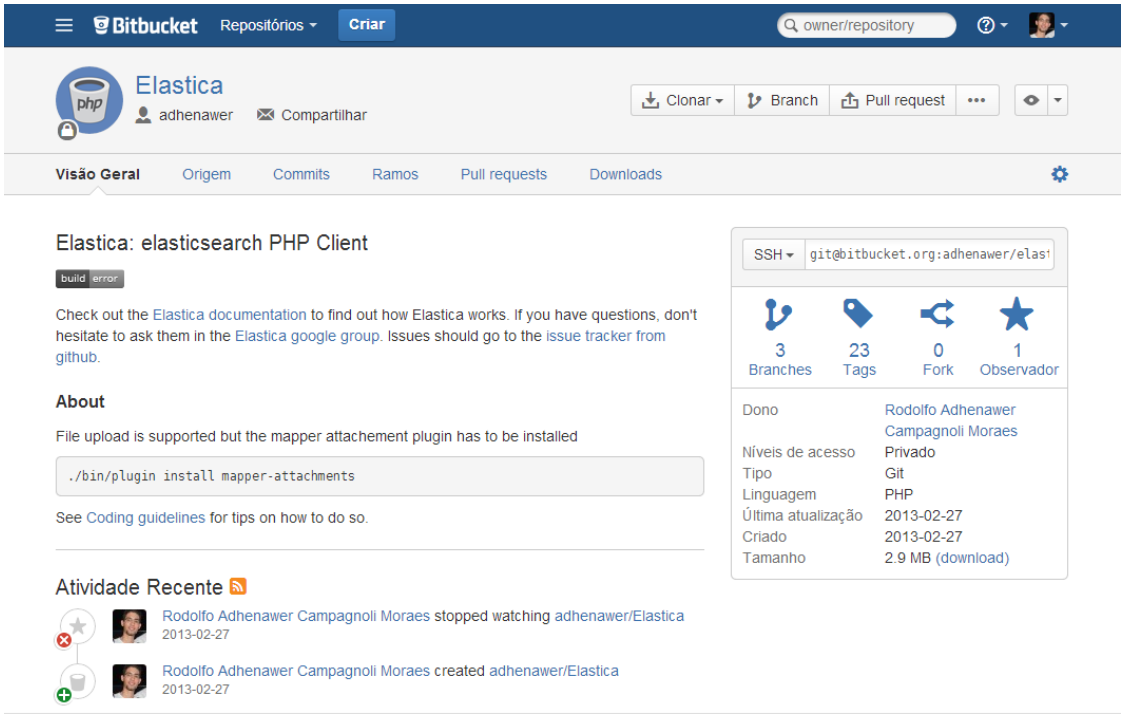

Figura 7 - Bitbucket interface web.

Principais características e funcionalidades

Entre suas principais características e funcionalidades podemos destacar algumas como a gestão de repositórios com acesso através de interface web com segurança SSL, a ferramenta disponibiliza uma versão para ambiente desktop com funcionalidades semelhantes ao do ambiente web, possui controle de usuários, grupos e suas permissões por repositório, interface para revisão de código. E suporte para sistemas de controle de versão GIT e Mercurial.

2.1.3 GitHub

O Github possui suporte para o SCV GIT, é gratuito para a hospedagem de repositórios públicos onde tudo o que é desenvolvido fica público para visualizações, downloads e colaborações. Assim como o Bitbucket possui cliente para desktop para as plataformas Windows e Mac. Na Figura 8 é demonstrado interface web da ferramenta Github.

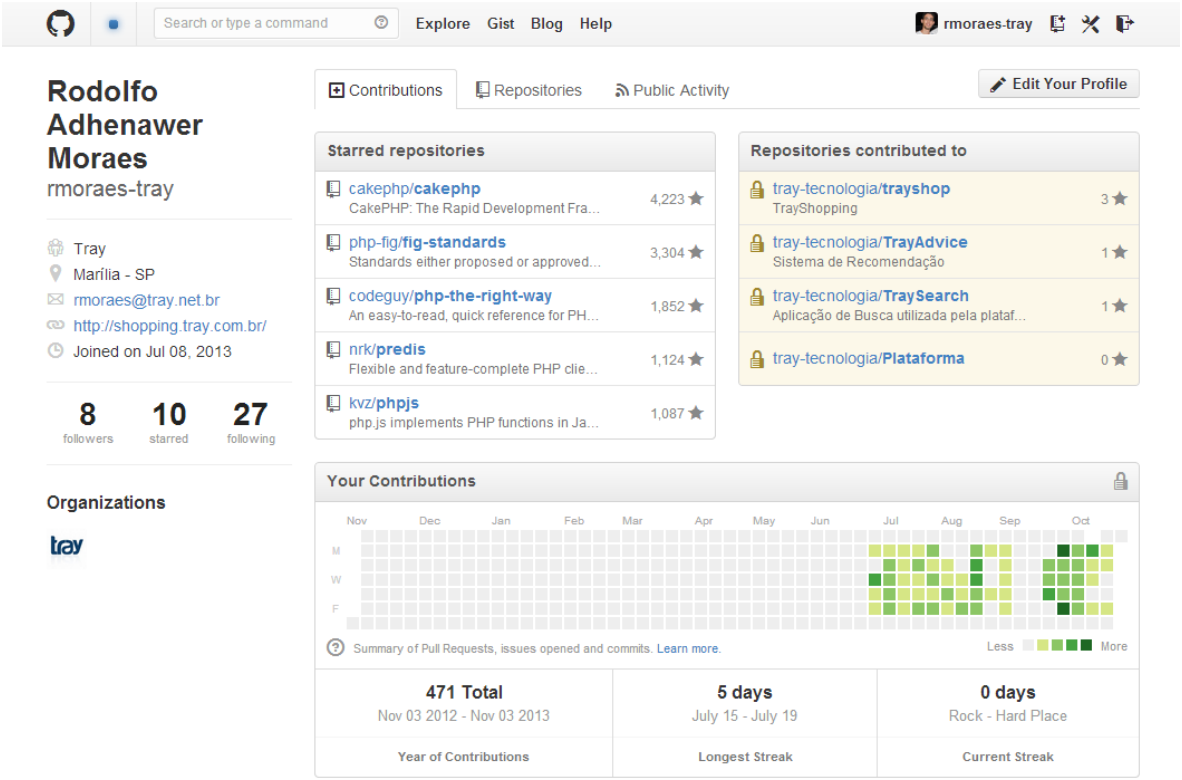

Figura 8 - Github interface web.

Principais características e funcionalidades

Entre suas principais características e funcionalidades podemos destacar algumas como a gestão de repositórios com acesso através de interface web com segurança SSL, a ferramenta disponibiliza uma versão para ambiente desktop com funcionalidades semelhantes ao do ambiente web, possui controle de usuários, grupos e suas permissões por repositório, tem interface para revisão de código, suporte à comunicação com fóruns, comentários e páginas de wiki. E suporte para o sistema de controle de versão GIT.

2.1.4 Assembla

Assembla é um gerenciador de repositórios Subversion e GIT através de seu painel web é possível fazer o controle de atividades de um time colaborativo através de tickets e tarefas com controle de mensagens por usuários e grupos. Na Figura 9 é demonstrado a interface web da ferramenta Assembla.

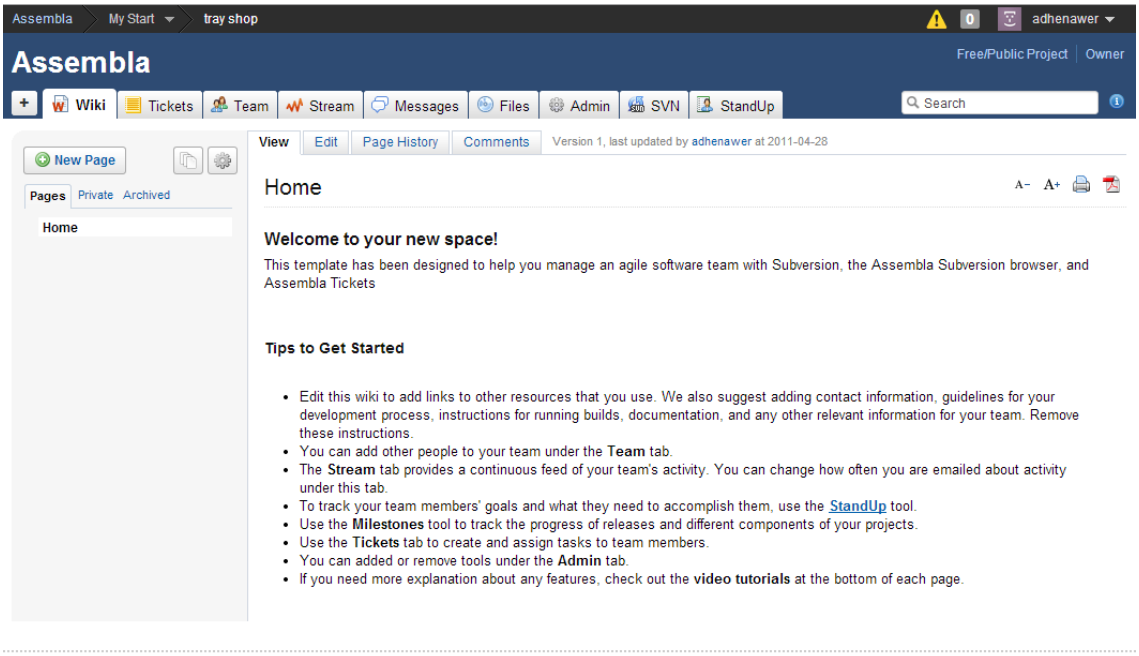

Figura 9 - Assembla interface web.

Principais características e funcionalidades.

Entre suas principais características e funcionalidades podemos destacar algumas como a gestão de repositórios com acesso através de interface web com segurança SSL, possui controle de usuários, grupos e suas permissões por repositório, tem interface para revisão de código, páginas de wiki e gestão simplificada de projetos com painel de tarefas. E suporte para o sistema de controle de versão GIT.

2.2 FERRAMENTAS OPEN SOURCE

Ferramentas são consideradas *open source* quando podem ser usadas, copiadas, modificadas e redistribuídas sem nenhuma restrição. Uma ferramenta é considera open source quando atende aos quatro tipos de liberdade para os usuários, que segundo a *Free Software Foundation* são:

- A liberdade de estudar e adaptar o programa.

- Acesso ao código fonte.

- A liberdade de aperfeiçoar o programa.

- A liberdade de redistribuir cópias exatas ou modificadas do programa.

2.2.1 SVN access manager

SVN Access manager é uma ferramenta de gerenciamento de grupos colaborativos de desenvolvimento, através de sua interface web é feito o controle de usuários, grupos e repositórios que suporta exclusivamente o sistema de controle de versão Subversion, o desenvolvimento da ferramenta é baseado na linguagem de programação PHP, possui um único perfil de usuário, que é o de administrador, que consiste em ter total acesso as funcionalidades da ferramenta para gerir o processo de criação e manutenção dos repositórios e equipes.

A ferramenta é distribuída livremente através da licença GNU, o que possibilita ter posse do código fonte e hospedá-lo em servidores internos para que os artefatos gerados durante o desenvolvimento não fique em servidores de terceiros. Na Figura 10 é demonstrado a interface web da ferramenta SVN Access manager.

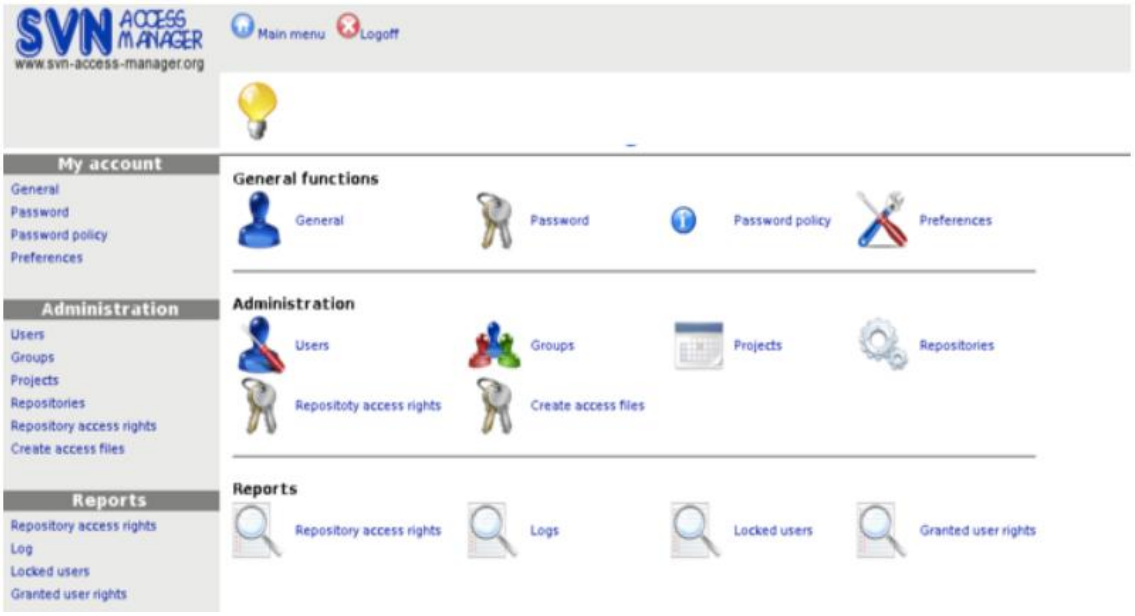

Figura 10 - SVN-Access-manager.

Principais características e funcionalidades

Entre suas principais características e funcionalidades podemos destacar algumas como a gestão de repositórios com acesso através de interface web, possui controle de usuários, grupos e suas permissões por repositório e por diretório, a autorização é feita via módulo do apache (mod_authz) e permissões via ACL (lista de controle de acesso) SVN. E suporte para o sistema de controle de versão Subversion.

Requisitos de instalação para o SVN Access Manager

- Sistema operacional Windows ou Unix.
- Apache 2 ou superior com módulos dav-svn, mod_authz.
- Subversion
- Java
- PHP 4 ou superior
- Banco de dados MySQL 4 ou superior

Avaliação

Conforme consta em sua página de distribuição, a ferramenta SVN Access manager, possui avaliações e classificações de usuários que utilizaram a ferramenta, destas avaliações, 10% foram negativas e 90% positivas, de um total de 30 usuários que avaliaram a ferramenta. O que indica um bom nível de aceitação na utilização da ferramenta.

2.2.2 SVN Manager

Embora o SVN manager seja mais simples do que a ferramenta anterior, também possui gerenciamento através de interface web e é baseada na linguagem de programação PHP, gerencia as funcionalidades essenciais do subversion para se prover um ambiente de desenvolvimento colaborativo de forma simples. Na Figura 11 é demonstrado a Interface web da ferramenta SVN Manager.

Figura 11 - SVN Manager.

Principais características e funcionalidades

Entre suas principais características e funcionalidades podemos destacar algumas como a gestão de repositórios com acesso através de interface web, possui controle de usuários, grupos e suas permissões por repositório, importação e exportação de repositórios, suporte à comunicação com envio de emails para usuários para a criação de conta no servidor. E suporte para o sistema de controle de versão Subversion.

Requisitos de instalação para o SVN Manager

- Sistema operacional Windows ou Unix.
- Apache 2 ou superior com módulos dav-svn, mod_authz.
- Subversion
- Banco de dados MySQL ou SQLite.
- PHP 5 ou superior + Pear.

Avaliação

Conforme consta em sua página de distribuição, a ferramenta SVN Manager, possui avaliações e classificações de usuários que utilizaram a ferramenta, destas avaliações, 25% foram negativas e 75% positivas, de um total de 24 usuários que avaliaram a ferramenta. O que indica um nível satisfatório de aceitação na utilização da ferramenta.

2.2.3 GitLab

O GitLab assim como as outras ferramentas open sources possui interface web, é distribuída livremente sob a licença MIT, seu desenvolvimento foi baseado na linguagem de programação Ruby apoiado pelo framework Rails, a ferramenta gerencia exclusivamente o sistema de controle de versão GIT, suas principais funcionalidades permitem criar projetos, repositórios e gerenciar usuários e suas permissões de acesso ao código fonte que é gerenciado através da ferramenta, uma funcionalidade que também se destaca dentro da ferramenta é o suporte à comunicação que é provido aos membros de uma equipe colaborativa, que pode ser feito através de criação de tarefas, listas de discussões e suporte à criação de páginas wiki por projeto. O GitLab possui uma comunidade altamente ativa e colaborativa, releases de versões são lançadas constantemente com atualizações e correções. Na Figura 12 é demonstrado a Interface web da ferramenta GitLab

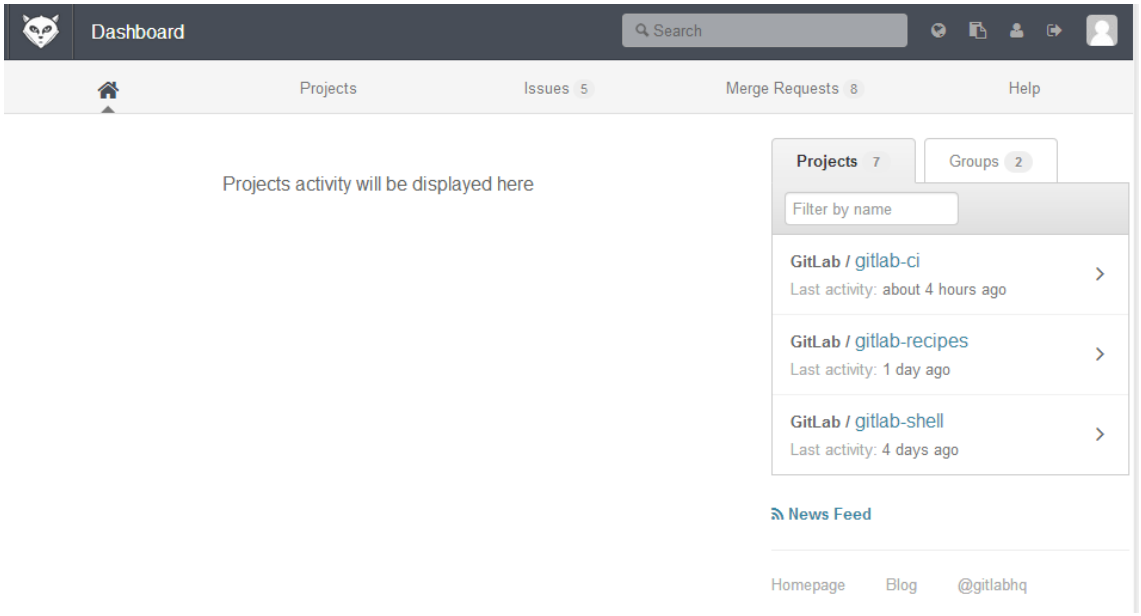

Figura 12 - GitLab.

Principais características e funcionalidades

Entre suas principais características e funcionalidades podemos destacar algumas como a gestão de repositórios com acesso através de interface web, possui controle de usuários, grupos e suas permissões por repositório, interface para revisão e mesclagem de código, suporte à comunicação através de tarefas, comentários e páginas de wiki. E suporte para o sistema de controle de versão GIT.

Requisitos de instalação para o GitLab

- Sistema operacional Ubuntu/Debian
- Ruby 1.9.3 ou superior.
- Rails
- GIT 1.7.10 ou superior.
- Redis 2.0 ou superior.
- MySQL ou PostgreSQL

Avaliação

Apesar da página de distribuição da ferramenta não possuir suporte para votação de níveis de classificação de satisfação do usuário é possível perceber que a ferramenta é muito bem aceita na comunidade e atualmente possui mais de vinte e cinco mil organizações

utilizando a ferramenta, dentre elas destacam-se instituições como a universidade do estado norte americano de Michigan e empresas como a Canadense BlackBerry.

3. PROPOSTA

Propõe-se o desenvolvimento do software CollabDev de acordo com as premissas da licença open source GNU versão 2 que segundo a *Free Software Foundation* garantem a liberdade para executar o programa, estudar e mudar o código-fonte do programa, redistribuir cópias exatas e distribuir versões modificadas.

A ferramenta gerencia dois tipos de sistemas de controle de versão diferentes, que são eles dos tipos centralizados e distribuídos, Subversion e GIT respectivamente.

As principais características e funcionalidades do software são:

- Software livre (Licença GNU v2).
- Gerenciar repositórios com acesso através de interface web.
- Gerenciar a manutenção de usuários, grupos e suas permissões por repositório.
- Suporte para o SCV Subversion.
- Suporte para o SCV GIT.
- Arquitetura preparada para a adição de novos SCV's.

O CollabDev faz interações diretas com o SCV para efetuar a criação, modificação ou exclusão de repositórios, administra usuários, grupos e suas respectivas permissões, cada usuário terá permissões de acesso e interação somente com repositórios que estiverem previamente cadastrados.

Existem três perfis de usuários diferentes no Collabdev, eles são subdividos da seguinte maneira: Administrador, Gerenciador e Colaborador. Cada grupo possui responsabilidades distintas que são descritas nos parágrafos a seguir.

O grupo Administrador é responsável pela configuração e manutenção do CollabDev no servidor em que será hospedado, definição dos caminhos de diretórios onde serão armazenados os repositórios, usuários e permissões do controle de versão, assim como os tipos de SCV's suportados pelo CollabDev.

O grupo Gerenciador é responsável pelo cadastro de usuários, repositórios e criação e manutenção de equipes colaborativas, uma equipe colaborativa no CollabDev é constituída por usuários de um determinado repositório que interagem no desenvolvimento de um projeto comum entre eles, o CollabDev será responsável pela manutenção do repositório do grupo fazendo as interações necessárias com o SCV.

O CollabDev possui uma url de cadastro aberta para que o próprio usuário tenha a possibilidade de se cadastrar, todos os usuários que se cadastram a partir desta tela se tornam do tipo Colaborador. Caso o usuário tenha sido cadastrado através de um usuário do tipo Gerenciador ele poderá alterar seus dados de cadastro no painel administrativo do gerenciador.

No CollabDev existem dois tipos de interações, do usuário com o gerenciador e do gerenciador com o controle de versão que executa todos os comando necessários para prover um repositório em que seja possibilitada a colaboração.

O gerenciador faz interações diretas com o controle de versão no momento em que se cria uma equipe colaborativa relacionando usuários e repositório, são disparadas ações no gerenciador que fazem interação diretamente com o controle de versão através da execução de comandos específicos, que fazem a criação do usuário no controle de versão e suas respectivas permissões.

Através da ferramenta proposta toda a administração de repositórios no SCV será facilitada, as interações necessárias para o gerenciamento de repositórios, usuários e suas respectivas permissões serão feitas através de interface gráfica em ambiente web conforme ilustrado na Figura 13.

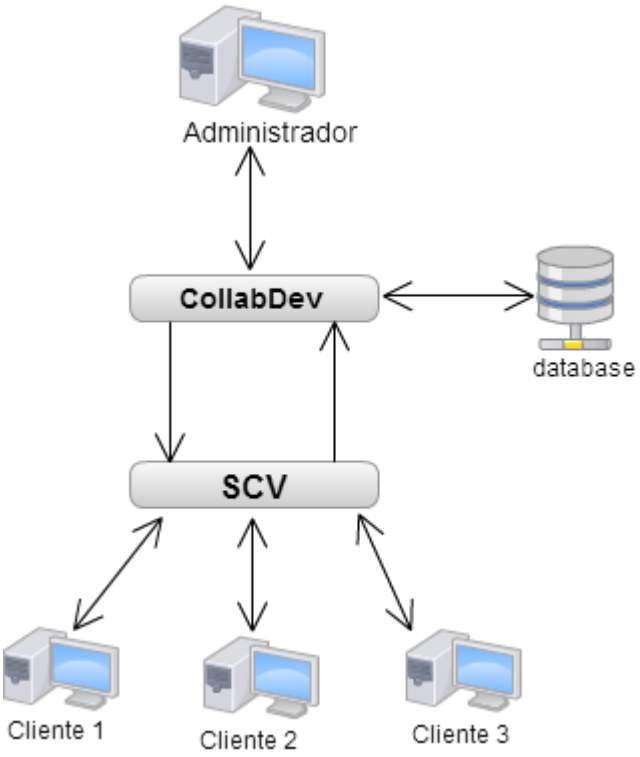

Figura 13 - Modelo de interação.

O software poderá ser hospedado em instituições de ensino, empresas, em nuvens públicas ou privadas com uma arquitetura básica de configuração. Com isto, todo código desenvolvido poderá ser armazenado e gerenciado dentro dos servidores da própria instituição, e não em servidores de terceiros como acontecem em outras ferramentas já citadas anteriormente.

3.1 REQUISITOS DE INSTALAÇÃO PARA O COLLABDEV

- Sistema operacional Unix.
- Apache 2.2.5 ou superior com módulos dav-svn, mod_authz e mod_rewrite.
- PHP 5.3.x ou superior
- MySQL 5.1 ou superior
- Subversion
- GIT 1.7.1 ou superior

3.2. DESENVOLVIMENTO

No processo de desenvolvimento foram especificados requisitos e desenvolvido diagramas seguindo os padrões de especificação de engenharia de software para guiar o desenvolvimento da proposta, foram gerados diagramas de caso de uso, atividade, classe e diagrama ER para modelagem de dados, Pressman (1995) define que "a engenharia de software é a união da engenharia de sistemas com o uso de hardware, abrangendo três elementos fundamentais: métodos, ferramentas e procedimentos, possibilitando o controle do desenvolvimento e o aumento da qualidade produtiva".

Os métodos demonstram os detalhes de como se deve fazer o software, envolvendo as etapas de planejamento e estimativa de projeto, análise de requisitos de software e de sistemas, projeto da estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste e manutenção.

As ferramentas proporcionam apoio aos métodos, sendo que existe um tipo específico de ferramenta para cada método anteriormente citado. Quando uma única ferramenta reúne vários métodos, disponibilizando a informação gerada por um método a outro subsequente, cria-se a chamada Engenharia de Software Auxiliada por Computador (CASE - Computer-Aided Software Engineering). O CASE combina software, hardware e um banco de dados de engenharia de software, criando uma estrutura que irá conter informações sobre análise, projeto, codificação e testes.

Todo o processo de análise de sistemas faz parte de uma teia de sub-processos, sendo muito difícil fazer uma distinção clara entre eles. A um nível de abstração grande, um processo de análise de sistemas pode ser descrito como a obtenção, definição, análise, negociação, documentação e validação de requisitos.

Utilizando-se das técnicas de engenharia de software para o desenvolvimento do CollabDev se obtém como resultado um software padronizado e de melhor compreensão para futuras expansões que poderão vir da comunidade open source.

Como o CollabDev suporta controles de versões distintos, sua arquitetura de desenvolvimento é estruturada para facilitar a integração de diferentes SCV's e permitir que a ferramenta possa evoluir de forma transparente e atender a diferentes tipos de necessidades, em sua versão inicial o CollabDev irá suportar dois tipos de sistemas de controle de versão diferentes, o Subversion e o GIT.

Como são de tipos diferentes, centralizado e distribuído possuem características e particularidades específicas de cada um, dentre elas podemos citar a diferença de como são aplicadas as permissões, no SVN as permissões dos repositórios são controlados através de uma lista (ACL) que contém os dados de grupos de usuários, repositórios e suas respectivas permissões dentro de cada repositório. Já no GIT tem-se a possibilidade de gerir as permissões de acordo com as permissões concedidas no sistema operacional que o usuário possui no diretório do servidor no qual o repositório está hospedado.

Para suportar SCV’s de diferentes tipos foi elaborado o seguinte diagrama de classe conforme ilustra a Figura 14. No qual Actions é uma interface com métodos comuns entre SCV’s para gerir um repositório colaborativo, classes que o implementarem terão que obrigatoriamente implementar esses métodos conforme as classe SVN e GIT o fazem.

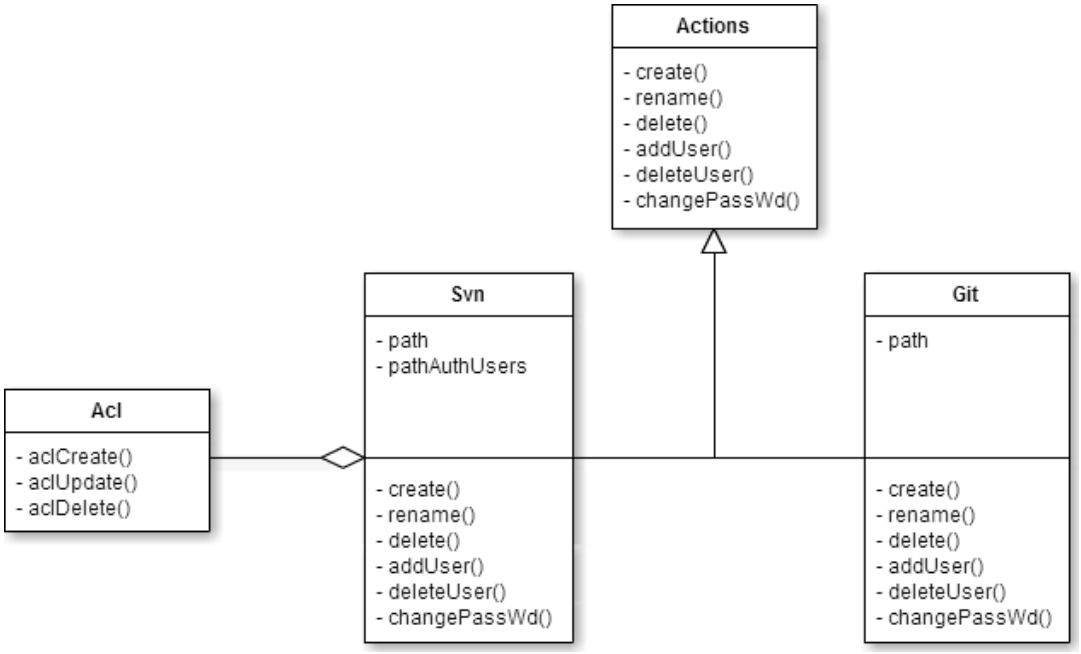

Figura 14 - Diagrama de classe.

A modelagem assim como o diagrama de classes possibilita a criação de tipos de repositórios de SCV's diferentes conforme mostra a Figura 15, onde a entidade repositórios é relacionada com o tipo de repositório que inicialmente serão SVN e GIT. Os tipos de repositórios existentes no CollabDev são previamente cadastrados e configurados no momento de sua instalação, para que no momento da criação de repositórios colaborativos seja possibilitada a escolha de um tipo de SCV desejado pelo usuário.

No "Apêndice E – Código fonte" é demonstrada a implementação destas classes no CollabDev. O código fonte completo pode ser acessado a partir de http://github.com/adhenawer/collabdev.

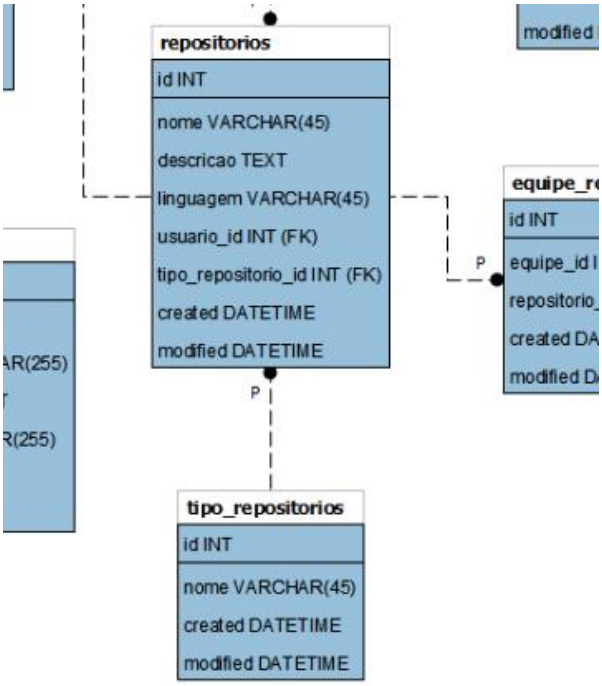

Figura 15 - Diagrama entidade relacionamento.

3.3 INTEGRANDO UM NOVO SCV NO COLLABDEV

Conforme visto durante este capítulo, atualmente o CollabDev suporta dois tipos de sistema de controle de versão, que são eles o Subversion e GIT e o fato do CollabDev ser open source deixa aberto a possibilidade da ferramenta receber contribuições no futuro, neste item será abordado o tema sobre a integração de um novo SCV ao CollabDev, para os exemplos aqui demonstrados será utilizado o Mercurial como SCV padrão a ser integrado.

O CollabDev foi desenvolvido de forma que seja facilitada a integração de novos SCV's de modo que para acrescentar um novo SCV são necessárias poucas modificações no código fonte do sistema, como pré-requisito para a implementação do tema abordado neste item é necessário que o SCV esteja previamente instalado no servidor que o CollabDev estará hospedado.

Inicialmente o que se tem a fazer é criar uma nova classe que implemente a interface Actions que contém os métodos genéricos de interações com os repositórios conforme demonstrado na figura 16 a seguir.

Os métodos contidos na interface são essenciais e implementa-los garante a interação básica com SCV, como criação de um repositório, adição de um novo usuário ao grupo entre outras ações indispensáveis para se gerir um repositório.

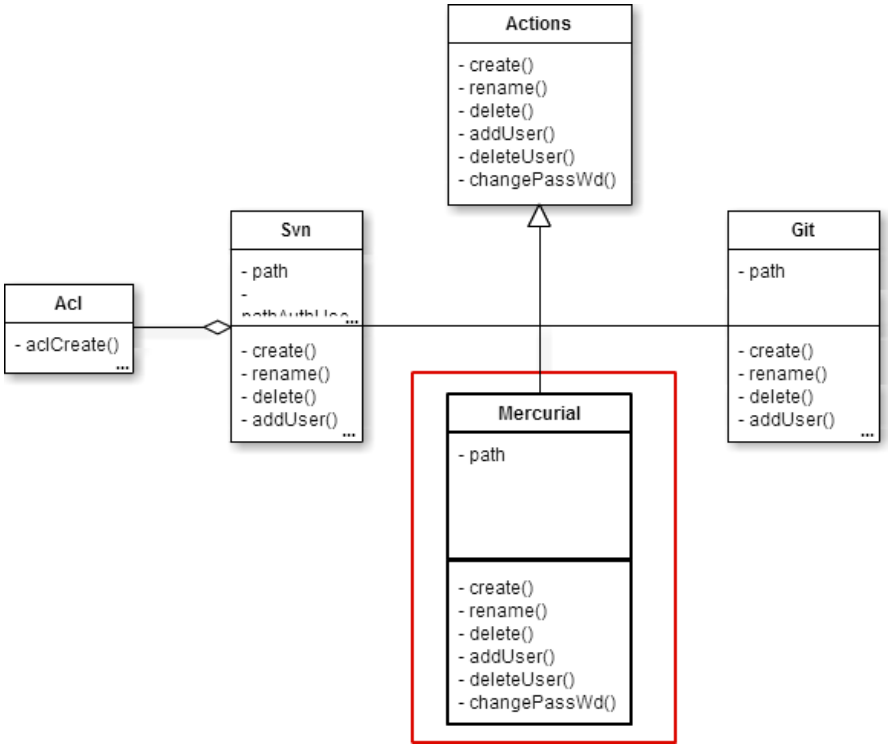

Figura 16 - Adicionando novo SCV.

Após implementar a interface na nova classe o próximo passo que deve ser feito é criar o tipo de repositório através da interface do próprio CollabDev, assim como exemplificado na figura 17.

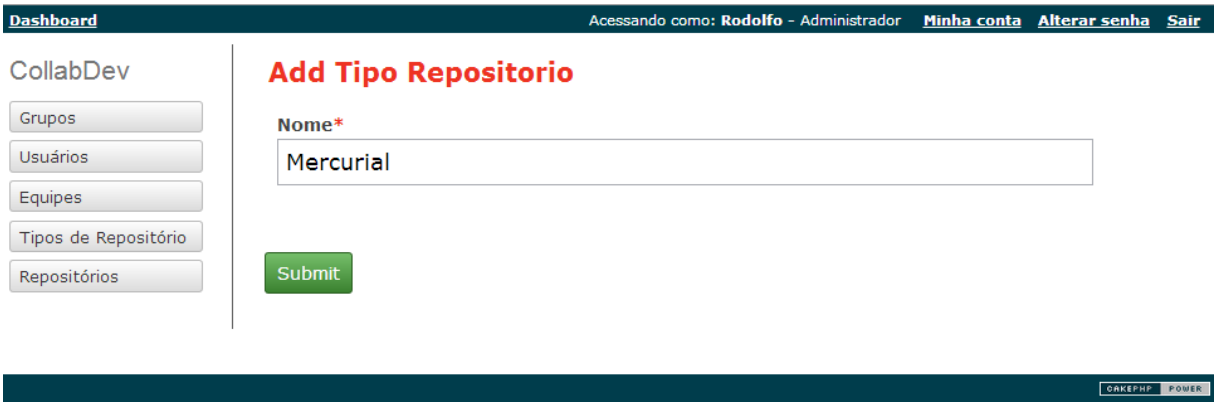

Figura 17 - Criando novo tipo de repositório.

No cadastro de repositórios todos os tipos de controle de versão aceitos são listados e podem ser selecionados na opção de tipo de repositório, após confirmação do cadastro o CollabDev irá instanciar a classe referente ao controle de versão que este repositório pertence e então todas as interações são feitas através desta classe. A figura 18 mostra a criação de um novo repositório no CollabDev com o exemplo de um novo SCV implementado.

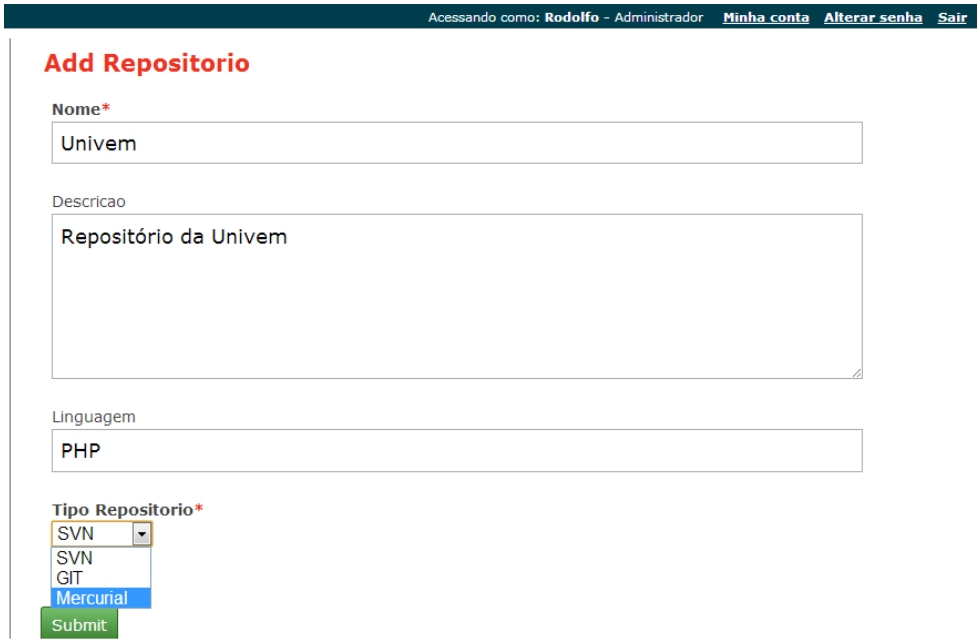

Figura 18 - Criando um repositório no CollabDev.

4. AVALIAÇÕES E RESULTADOS

Neste capítulo é feito uma análise das avaliações e resultados obtidos pela ferramenta CollabDev e comparado com as ferramentas proprietárias e open sources que foram listadas nos capítulos anteriores, serão apontados vantagens e desvantagens da ferramenta de acordo com cada uma de suas funcionalidades e características.

4.1 COMPARAÇÃO COM FERRAMENTAS PROPRIETÁRIAS

Na Tabela 1 foram listadas as principais características de cada uma das ferramentas proprietárias e realizada uma comparação com a ferramenta desenvolvida e quais funcionalidades estão disponíveis somente com planos pagos.

Tabela 1. Tabela de comparações ferramentas proprietárias

Características	XP-Dev	Bitbucket	Assembla	Github	CollabDev
Repositório suportado	SVN/Git/Mercurial	Git/Mercurial	SVN/Git/Mercurial	Git	SVN/GIT
Importa/Exporta repositório	✔	✔	✔		
Páginas Wiki	✔	✔	✔	✔	
Controle de acesso/Permissões	✔	✔	✔ *	✔ *	✔
Gestão de tarefas	✔		✔	✔	
Open source					✔
Repositórios ilimitados	✔ *	✔ *	✔ *		✔
Usuários ilimitados	✔ *	✔ *		✔ *	✔

✔ Possuí característica
* Plano pago

4.1.1 Vantagens

Como vantagem sobre as ferramentas proprietárias, destaca-se o fato de que o CollabDev pode ficar hospedado em servidores internos e de domínio de quem o utilizará e todo código desenvolvido nos repositórios por ele gerenciado não ficará em servidores de terceiros.

E até mesmo a integridade e segurança do código fonte diante de seus próprios colaboradores, já que todas as ferramentas proprietárias funcionam na internet e o acesso aos repositórios pode ser feito através de qualquer lugar podendo ser feita uma cópia do projeto do repositório sem o consentimento da empresa ou instituição que o mantém, com o CollabDev a empresa ou instituição tem a possibilidade de manter o acesso aos repositórios somente através da intranet.

Outro fato importante a se destacar são as limitações das ferramentas proprietárias no quesito de quantidade de repositórios, usuários e armazenamento suportado, sendo que os limites são baixos e sempre que é preciso um número maior de usuários ou repositórios é necessário migrar para um plano superior ou aderir a um plano de valor elevado que possuem capacidades ilimitadas, sendo que com o CollabDev as limitações ficam a critério do servidor que está sendo utilizado para hospeda-lo, assim diminuindo as barreiras de limitações de capacidades físicas que as ferramentas proprietárias impõem a fim de obrigar a migração de plano e o pagamento de uma mensalidade superior.

Outra vantagem do CollabDev sobre as ferramentas proprietárias é a disponibilidade de acesso aos repositórios. Apesar da maioria das ferramentas proprietárias garantirem disponibilidade de 24x7, ou seja, todos os dias da semana de disponibilidade, hora ou outra ocorrem problemas de indisponibilidade. Foram constatados no uso contínuo das ferramentas Github e Bitbucket alguns períodos onde os repositórios ficaram inacessíveis para o uso. Com o CollabDev apesar de não estar imune a possíveis problemas de rede interna, esses problemas podem ser resolvidos localmente pela equipe de infra estrutura da empresa ou instituição que é responsável pela manutenção da rede não ficando a mercê de terceiros.

4.1.2 Desvantagens

Como desvantagens do CollabDev perante as ferramentas proprietárias analisadas destaca-se alguns pontos e funcionalidades, a começar pelo fato de que o CollabDev requer uma instalação manual e de que a empresa ou instituição tenha um administrador de TI que faça sua configuração e de todos os recursos listados na seção 3.1 do capítulo "REQUISITOS DE INSTALAÇÃO PARA O COLLABDEV" para que funcione em sua rede intranet.

Funcionalidades existentes na maioria das ferramentas proprietárias que se destacam sobre o CollabDev são as de interação social entre os desenvolvedores dentro do ambiente online onde ficam os repositórios, funcionalidades de páginas wiki utilizadas para descrever detalhes do projeto que está sendo feito no repositório, controle de tarefas para a organização do desenvolvimento do projeto para se analisar algumas métricas como tempo de desenvolvimento de determinada atividade, qual desenvolvedor responsável por determinada funcionalidade dentro do projeto e ainda controle de defeitos encontrados. Outra funcionalidade que se destaca também é a revisão de código, funcionalidade esta que só existe nas ferramentas proprietárias onde após o desenvolvimento de determinada tarefa do projeto o desenvolvedor submete o código que foi desenvolvido para que alguém responsável pelo

projeto possa analisar e revisar o código gerado a fim de evitar algum erro de programação que o desenvolvedor tenha cometido.

4.2 COMPARAÇÃO COM FERRAMENTAS OPEN SOURCE

Na Tabela 2 foram listadas as principais características de cada uma das ferramentas open source e realizada uma comparação com a ferramenta desenvolvida.

Tabela 2. Tabela de comparações ferramentas open source

Características	SVN Access Manager	SVN Manager	GitLab	CollabDev
Repositório suportado	SVN	SVN	GIT	SVN/GIT
Importa/Exporta repositório	✔			
Páginas Wiki			✔	
Perfil de usuário				✔
Permissões por diretório	✔			
Controle de acesso/Permissões	✔	✔	✔	✔
Gestão de tarefas			✔	

✔ Possuí característica

4.2.1 Vantagens

Como vantagens sobre as ferramentas *open source* encontradas destaca-se o fato de que o CollabDev é o único entre as ferramentas a possuir suporte de gerenciamento a mais de um tipo de repositório diferente, que são eles SVN e GIT e ainda o que possui a maior facilidade de expansão a outros tipos de repositórios, sendo que sua arquitetura de desenvolvimento foi criada para facilitar tal tipo de implementação conforme descrito na seção 3.3 do capítulo "INTEGRANDO UM NOVO SCV NO COLLABDEV".

Outra vantagem sobre as demais ferramentas *open source* é o suporte a perfil de usuários do CollabDev, onde é possível a criação de três tipos de perfis diferentes, que são eles administrador, gerenciador e colaborador, característica importante para que se tenha delegação de responsabilidades dentro da ferramenta, para que o processo de gestão de repositórios, grupos e permissões não fique preso a uma única pessoa ou equipe de TI por exemplo, com este recurso é possível ter vários gerenciadores onde cada um poderia ser responsável pela gestão de sua equipe de desenvolvimento dentro do CollabDev.

4.2.2 Desvantagens

Como desvantagem do CollabDev perante as ferramentas *open source* analisadas pode-se destacar alguns pontos e funcionalidades, como por exemplo a importação e exportação de repositórios, funcionalidade que permite a migração de repositórios sem perder o histórico de *commits* que foram feitos no projeto ao longo do tempo, para caso exista a necessidade de mudança de ferramenta ou controle de versão não seja necessário começar um projeto desde do início novamente como se fosse um projeto novo.

Outra funcionalidade existente entre as ferramentas *open source* que o CollabDev não suporta é a possibilidade de configuração de permissões por diretório, onde em alguns casos pode ser útil para quando existem diferentes equipes trabalhando em um mesmo projeto, onde cada equipe é responsável por determinada área do desenvolvimento onde o código fonte fica separado por diretórios, por exemplo uma equipe responsável pelo design que tenha que fazer alterações nos diretórios "js" e"css" e outra equipe responsável pelas regras de negócio tenha que fazer alterações nos diretórios "*controller*" e "*model*", assim cada equipe poderia interagir somente com os diretórios no qual tem permissão.

CONCLUSÕES

Ainda hoje, existem instituições de ensino e empresas que não utilizam nenhum gerenciador de repositórios para ambientes de desenvolvimento colaborativo, onde toda a manutenção desses repositórios são feitas manualmente através de linha de comando e configurações específicas de servidores e de particularidades de cada sistema de controle de versão.

Visando facilitar essa interação com os sistemas de controles de versão foi desenvolvido o CollabDev que gerencia ambientes colaborativos de desenvolvimento, deixando transparente a interação do usuário com o controle de versão, seja ele qual for, onde o comando para criação de um repositório no SVN ou no GIT seja através da mesma interface no CollabDev, abstraindo a complexidade dos comandos sendo executado em qualquer um dos controles de versão. E ainda facilitando futuras colaborações com uma arquitetura de desenvolvimento flexível e de fácil expansão para adicionar novas funcionalidades ou um novo sistema de controle de versão suportado pelo CollabDev.

Para avaliar os resultados obtidos foi feito um levantamento de ferramentas disponíveis atualmente no mercado, prioritárias e *open source* e foi feito um comparativo com a ferramenta desenvolvida a fim de mostrar as características, vantagens e desvantagens de cada uma, onde foi possível constatar que o CollabDev possui as principais funcionalidades para se gerir repositórios para ambientes colaborativos de desenvolvimento.

REFERÊNCIAS

ARAUJO, J. B. Trabalho Colaborativo e Groupware <http://www.uninove.br/PDFs/Publicacoes/cadernos_posgraduacao/cadernos_v2/cdpv2_josebraz_miltonabreu.pdf> Acesso em 13 Mai 2013.

ASSEMBLA. Disponível em: <https://www.assembla.com> Acesso em 06 mar 2013.

Bitbucket. Disponível em: <https://bitbucket.org> Acesso em 05 mar 2013.

CHACON, SCOTT (2009). Git Book. Disponível em: <https://github.s3.amazonaws.com/media/progit.en.pdf> Acesso em 20 Ago 2013.

CHAGAS, I. Trabalho Colaborativo. Condição Necessária para a Sustentabilidade das Redes de Aprendizagem. Disponível em: <http://www.educ.fc.ul.pt/docentes/ichagas/ticc/cnetrabalhocolaborativo.pdf> Acesso em 13 Mai 2013

DIAS, A. F. Conceitos Básicos de Controle de Versão de Software-Centralizado e Distribuído.2011. Disponível em <http://www.pronus.eng.br/artigos_tutoriais/gerencia_configuracao/conceitos_basicos_controle_versao_centralizado_e_distribuido.php> Acesso em 13 Mai 2013.

GITHUB. Disponível em:<https://github.com> Acesso em 01 mar 2013.

GITLAB. Disponível em: < http://gitlab.org> Acesso em 16 ago 2013.

JAMIL, G. L. Repensando a TI na Empresa Moderna. Rio de Janeiro: Axcel Books, 2001.

MASON, M. Pragmatic Version Control: Using Subversion (The Pragmatic Starter Kit Series). Pragmatic Bookshelf, 2006

MCLAGAN, P.; NEL, C. A Nova Era da Participação. Rio de Janeiro: Campus, 2000.

OPEN SOURCE INITATIVE. The Open Source Definition. Disponível em: <http://opensource.org/osd> Acesso em 01 mar 2013.

PARRILA, A.; DANIELS, H. Criação e desenvolvimento de grupos de apoio para professores. São Paulo: Loyola, 2004.

SCHOOLER, E. (1996). Conferencing and collaborative computing. Multimedia Systems, 4, 210-225.

SUSSMAN, W. FITZPATRICK e PILATO (2002). The Subversion Handbook. Disponível em: <http://code.google.com/p/svnbook-pt-br/downloads/list> Acesso em: 18 fev 2013.

SVN ACCESS MANAGER. Disponível em: <http://indico.cern.ch/getFile.py/access?contribId=112&sessionId=10&resId=0&materialId=slides&confId=119169> Acesso em 16 ago 2013.

SVN MANAGER. Disponível em: < ttp://svnmanager.sourceforge.net/index.php?page=intro> Acesso em 16 ago 2013.

XP-DEV. Disponível em: <https://xp-dev.com> Acesso em 16 mar 2013.

APÊNDICE A - DIAGRAMA DE ATIVIDADE

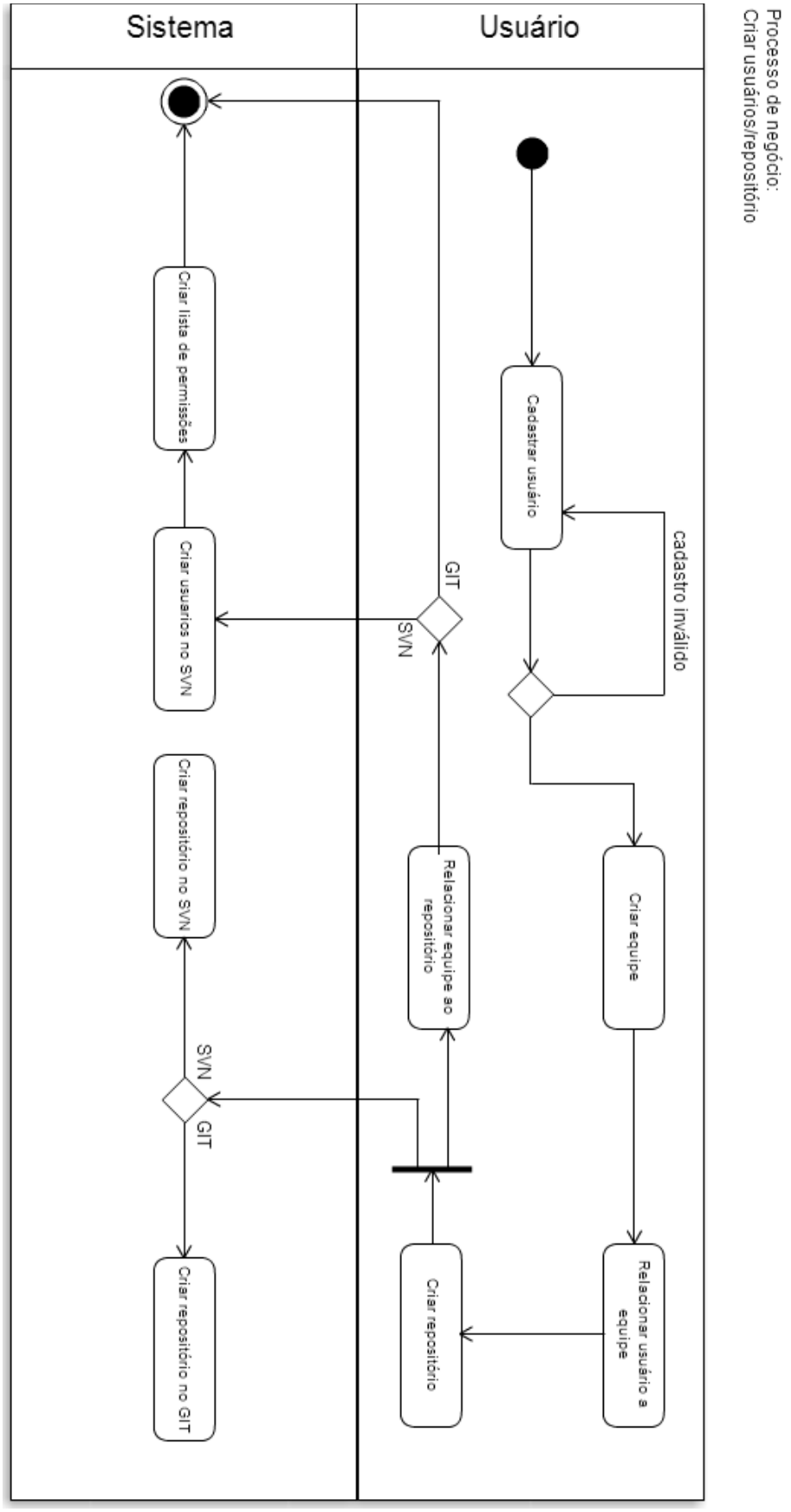

Figura 1 A. Diagrama de atividade

APÊNDICE B - DIAGRAMA DE CASO DE USO

Figura 1 B. Diagrama de caso de uso

APÊNDICE C - DIAGRAMA DE ENTIDADE E RELACIONAMENTO

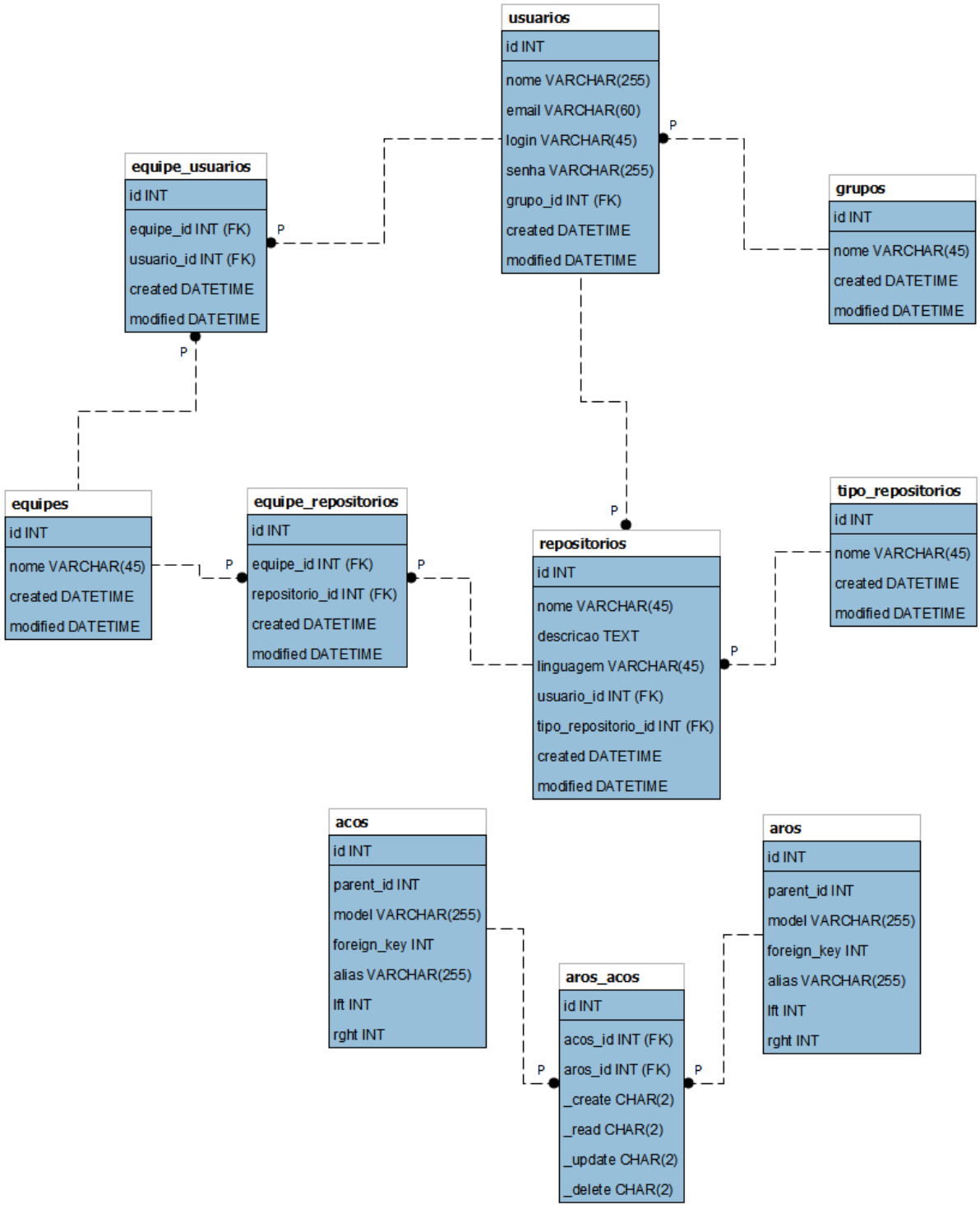

Figura 1 C. Diagrama de entidade e relacionamento

APÊNDICE D - APRESENTAÇÃO DO COLLABDEV

Figura 1 D. Dashboard

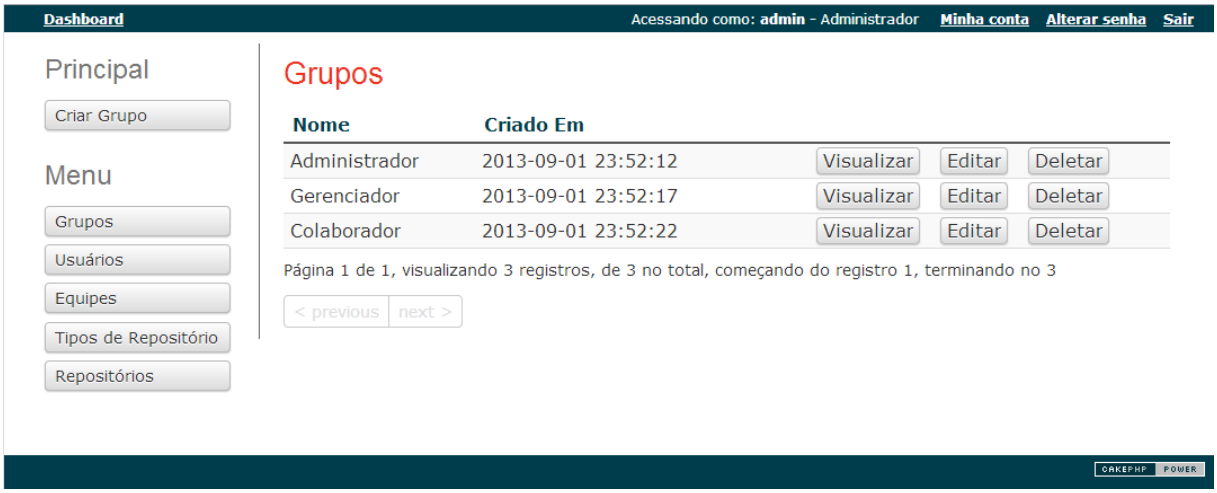

Figura 2 D. Index grupos

Figura 3 D. Index usuários

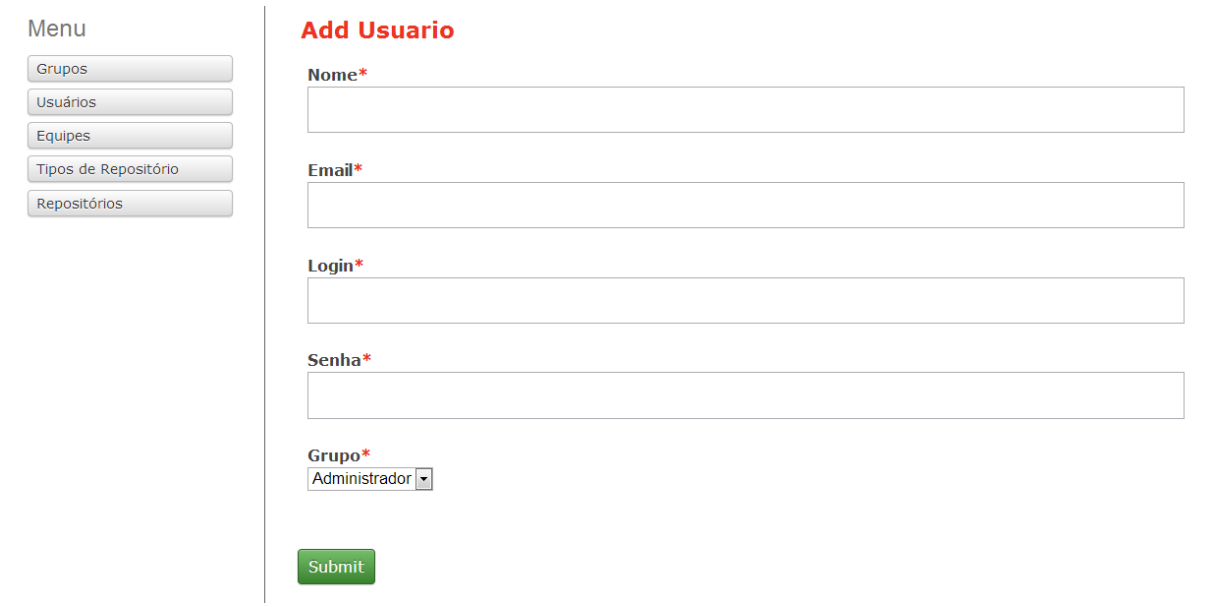

Figura 4 D. Adicionar usuário.

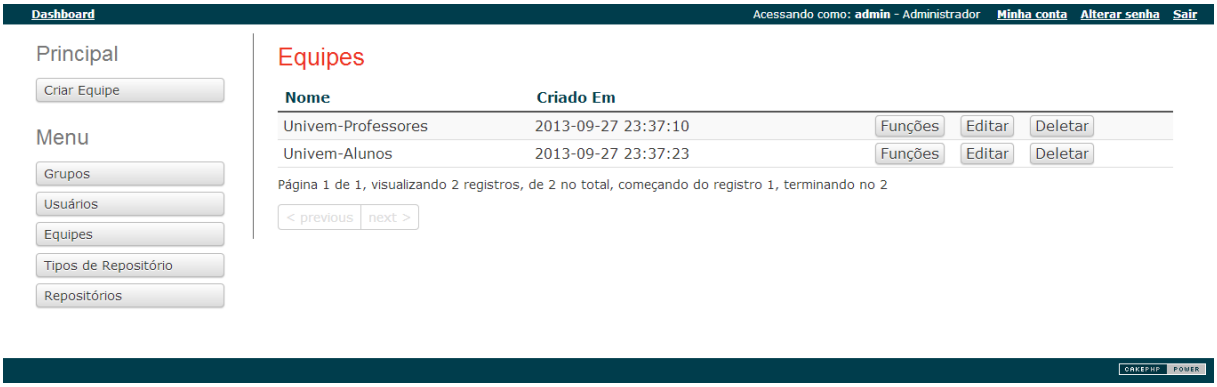

Figura 5 D. Index equipes.

Figura 6 D. Criação de equipes.

Figura 7 D. Index repositórios.

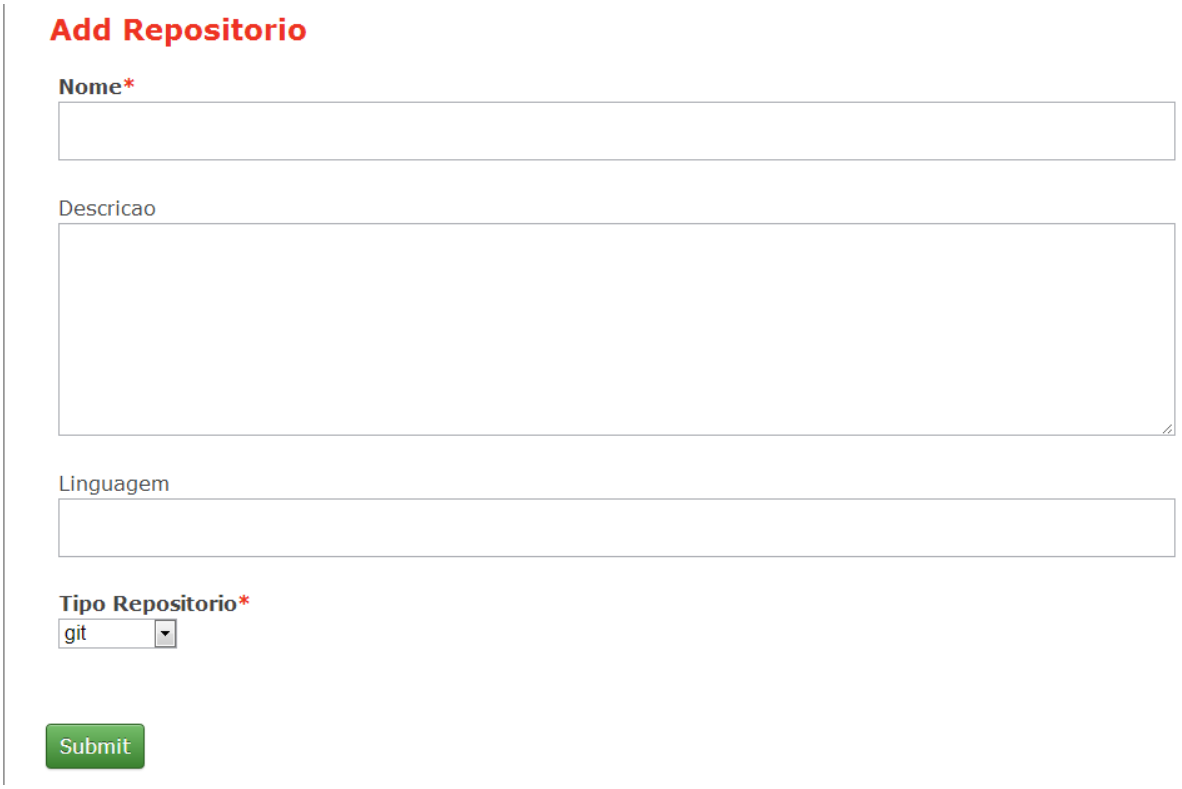

Figura 8 D. Adicionar repositório.

APÊNDICE E - CÓDIGO FONTE

Este apêndice contém apenas trechos parciais do código fonte do CollabDev, o código fonte completo encontra-se disponível no endereço:

- https://github.com/adhenawer/collabdev

```php
<?php
App::uses('AppController', 'Controller');
/**
 * Acl Controller responsável por controlar o nível de permissão dos usuários.
 *
 * @property Acl $Acl
 */
class AclController extends AppController {

    public $uses = array('Usuario');

    public function beforeFilter() {
        parent::beforeFilter();
        $this->Auth->allow('initDB');
    }

    public function initDB() {
        $group = $this->Usuario->Grupo;
        //Administrador
        $group->id = 1;
        $this->Acl->allow($group, 'controllers');

        //Gerenciador
        $group->id = 2;
        $this->Acl->deny($group, 'controllers');
        $this->Acl->allow($group, 'controllers/Usuarios/index');
        $this->Acl->allow($group, 'controllers/Usuarios/add');
        $this->Acl->allow($group, 'controllers/Usuarios/edit');
        $this->Acl->allow($group, 'controllers/Usuarios/view');
        $this->Acl->allow($group, 'controllers/Usuarios/delete');
        $this->Acl->allow($group, 'controllers/Usuarios/dashboard');
        $this->Acl->allow($group, 'controllers/Usuarios/conta');
        $this->Acl->allow($group, 'controllers/Usuarios/alterarSenha');

        $this->Acl->allow($group, 'controllers/Repositorios/index');
        $this->Acl->allow($group, 'controllers/Repositorios/add');
        $this->Acl->allow($group, 'controllers/Repositorios/edit');
        $this->Acl->allow($group, 'controllers/Repositorios/view');
        $this->Acl->allow($group, 'controllers/Repositorios/delete');
        $this->Acl->allow($group, 'controllers/Repositorios/sincronizar');

        $this->Acl->allow($group, 'controllers/Equipes/index');
        $this->Acl->allow($group, 'controllers/Equipes/add');
        $this->Acl->allow($group, 'controllers/Equipes/edit');
        $this->Acl->allow($group, 'controllers/Equipes/view');
        $this->Acl->allow($group, 'controllers/Equipes/delete');
        $this->Acl->allow($group, 'controllers/Equipes/relacionarUsuario');
        $this->Acl->allow($group, 'controllers/Equipes/removerUsuario');
        $this->Acl->allow($group, 'controllers/Equipes/relacionarRepositorio');
        $this->Acl->allow($group, 'controllers/Equipes/removerRepositorio');

        //Colaborador
        $group->id = 3;
        $this->Acl->deny($group, 'controllers');
        $this->Acl->allow($group, 'controllers/Usuarios/dashboard');
        $this->Acl->allow($group, 'controllers/Usuarios/conta');
        $this->Acl->allow($group, 'controllers/Usuarios/alterarSenha');

        echo "all done";
        exit;
    }

}
```

Figura 1 E. Controle de acesso.

```php
/**
 * loadVersionador method
 *
 * @param string $versionador ex: [svn|git]
 * @return void
 */
    private function loadVersionador($versionador){
        $versionador = ucfirst(strtolower($versionador));
        if (!class_exists($versionador)) {
            throw new Exception(__('Classe "'.$versionador.'" inexistente!'));
        }
        $this->versionador                = new $versionador();
        $this->versionador->path          = Configure::read(
            'path.'.strtolower($versionador).'.repo'
        );
        if ($versionador == 'Svn') {
            $this->versionador->pathAuthUsers = Configure::read(
                'path.'.strtolower($versionador).'.auth-users'
            );
        }
    }
}
```

Figura 2 E. Método responsável por carregar o versionador (SVN/ GIT) dinâmicamente.

```php
/**
 * sincronizar method, sincroniza os usuários da equipe com o controle de versão.
 *
 * @param string $repositorioId
 * @param string $equipeId
 * @return void
 */
    public function sincronizar($equipeId, $repositorioId){
        $repositorio = $this->Repositorio->find('first', array(
            'conditions' => array(
                'Repositorio.id' => $repositorioId
            )
        ));
        $this->loadVersionador($repositorio['TipoRepositorio']['nome']);
        $usuarios = $this->getUsuariosEquipe($equipeId);
        if ($usuarios) {
            foreach ($usuarios as $value) {
                $this->versionador->addUser(strtolower($value), Configure::read('user.default.senha'));
            }
        }
        $equipe = $this->EquipeUsuario->Equipe->find('first', array(
            'conditions' => array(
                'Equipe.id' => $equipeId
            )
        ));
        $this->Svn->svnAcl($usuarios, $equipe, $repositorio);
        $this->Session->setFlash(
            __('Repositório sincronizado com sucesso.'),
            'default',
            array('class' => 'notification success')
        );
        return $this->redirect($this->referer());
    }
```

Figura 3 E. Método responsável por sincronizar com o controle de versão.

```php
<?php
/**
* Actions interact with subversion.
*/
interface Actions
{
    /**
    * Create repository
    */
    public function create($repo);

    /**
    * Rename repository
    */
    public function rename($oldName, $newName);

    /**
    * Delete repository
    */
    public function delete($repo);

    /**
    * Add user
    */
    public function addUser($user, $pass);

    /**
    * Delete user
    */
    public function deleteUser($user);

    /**
    * Change password user
    */
    public function changePassWd($user, $pass);
}
```

Figura 4 E. Interface Actions.

```php
<?php
class Svn implements Actions
{
    /**
     * Repository path.
     */
    public $path = null;

    /**
     * Path svn-auth-users.
     */
    public $pathAuthUsers = null;

    public function create($repo)
    {
        return $this->runCmd('cd '.$this->path.'; svnadmin create '.$repo);
    }

    public function rename($oldName, $newName)
    {
        return $this->runCmd('cd '.$this->path.'; mv '.$oldName.' '.$newName);
    }

    public function delete($repo)
    {
        shell_exec('chmod 777 -R '.$this->path.'/'.$repo);
        return $this->runCmd('cd '.$this->path.'; rm -Rf '.$repo);
    }

    public function addUser($user, $pass)
    {
        return $this->runCmd(
            'htpasswd -b '.$this->pathAuthUsers.'/svn-auth-users '.$user.' '.$pass.' 2>&1'
        );
    }

    public function deleteUser($user)
    {
        return $this->runCmd('htpasswd -D '.$this->pathAuthUsers.'/svn-auth-users '.$user.' 2>&1');
    }

    public function changePassWd($user, $pass){
        return $this->addUser($user, $pass);
    }

    private function runCmd($command){
        if ($this->path == NULL || $this->pathAuthUsers == NULL) {
            throw new NotFoundException('Path is not defined!');
        }
        return shell_exec($command);
    }
}
```

Figura 5 E. Classe SVN implementando interface Actions.

```php
<?php
/**
 * Component reponsável por manipular  arquivo de permissões do SVN  'svn-access-control'.
 *
 */
class SvnComponent extends Component {

/**
 * svnAcl method
 *
 * @param array $usuarios, type find list. Caso passado array() empty grupo de permissões é apagado.
 * @param array $grupo, type find all.
 * @param array $repo, type find all.
 */
    public function svnAcl($usuarios, $grupo, $repo){
        $filename = Configure::read('path.svn.access-control').'/svn-access-control';
        $idGroup = $repo['Repositorio']['id'].$grupo['Equipe']['id'];
        $handle = @fopen($filename, "r");
        if ($handle) {
            $this->svnAclDelete($handle, $idGroup, $filename);
            if(!empty($usuarios)){
                $this->svnAclCreate($usuarios, $grupo, $repo);
            }
            fclose($handle);
        } else {
            $this->svnAclCreate($usuarios, $grupo, $repo);
        }
    }

    private function svnAclCreate($usuarios, $grupo, $repo){
        $string = "\n";
        $idGroup = $repo['Repositorio']['id'] . $grupo['Equipe']['id'];
        $filename = Configure::read('path.svn.access-control').'/svn-access-control';
        $grupo['Equipe']['nome'] = strtolower($grupo['Equipe']['nome']);
        $string .= "###" . $idGroup . "###\n";
        $string .= "[groups]\n";
        $string .= $grupo['Equipe']['nome'] . ' = ';
        foreach ($usuarios as $value) {
            $string .= $value. ',';
        }
        $string .= "\n\n";
        $string .= "[" . $repo['Repositorio']['nome'] . ":/]\n";
        $string .= "@" . $grupo['Equipe']['nome'] . ' = rw';
        $string .= "\n###" . $idGroup . "###\n";
        $fp = fopen($filename, 'a');
        fwrite($fp, $string);
        fclose($fp);
    }

    private function svnAclDelete($handle, $idGroup, $filename){
        $buffer = '';
        while (!feof($handle)) {
            $buffer .= trim(fgets($handle, 4096))."\n";
        }
        $buffer = ereg_replace("###" . $idGroup . "###(.|\s)+.?###" . $idGroup . "###+",'', $buffer);
        unlink($filename);
        $fp = fopen($filename, 'a');
        fwrite($fp, trim($buffer));
        fclose($fp);
    }
}
```

Figura 6 E. Classe responsável por manipular o arquivo de permissões do SVN.