CHRISTIAN FERNANDES LOPES

Projeto de Tópicos em Desenvolvimento de Sistemas

Salvador/BA

2019

CHRISTIAN FERNANDES LOPES

Projeto de Tópicos em Desenvolvimento de Sistemas

Estudo de caso de criação de projeto de software como requisito para obtenção de nota para a disciplina Tópicos em desenvolvimento de sistemas na Universidade Ruy Barbosa

Docente: Heleno Filho

Salvador/BA

2019

SUMÁRIO

1. INTRODUÇÃO

Este trabalho dedicou-se a estudar a engenharia de software por completo, com isso, meu objetivo foi aprender os conceitos e técnicas de Engenharia de Software.

Para mim, futuro profissional da área de T.I, é imprescindível estar lado a lado com aliados que me permita ganhar tempo, e desempenho em meus projetos, com isso, além de ter a oportunidade de entender a fundo conceitos e técnicas, da engenharia de software, também poderemos executá-los, com a expectativa de extrair o máximo de conhecimento nessa área.

2. FASE DE PLANEJAMENTO

2.1. Simulação de uma empresa denominada “Global Security” que deseja criar um sistema de controle de acesso. A companhia atua no ramo de segurança e como forma de expandir os serviços quer oferecer aos clientes uma ferramenta completa capaz de monitorar e limitar os acessos nas instituições.

2.2. Desenvolver uma ferramenta completa leva bastante tempo e como forma de reduzir os riscos e entregar o programa no prazo utilizarei o modelo de prototipação para auxiliar no gerenciamento e organização dos requisitos, cada ciclo do modelo protótipo é composto de duas etapas onde é possível entregar uma nova versão para avaliação do cliente. Segue tabela com a relação das vantagens e desvantagens:

Quadro 1 - Vantagens e desvantagens do modelo espiral.

Vantagens	**Desvantagens**
Estimativas tornam-se mais realísticas	Muita ênfase a parte funcional
Mais versátil para lidar com mudanças	A avaliação dos riscos exige experiência
Melhora o tempo de implementação do sistema	É bem aplicado somente a sistemas de larga escala

3. ANÁLISE DE REQUISITOS

Iniciamos com uma entrevista informal. Durante a conversa com um dos representantes da Global Security foi possível destacar os seguintes requisitos Funcionais e Não-Funcionais:

REQUISITOS FUNCIONAIS

Um requisito funcional de um de software especifica uma função que o sistema deve ser capaz de executar. São requisitos que definem o comportamento do sistema, ou seja, o processo/transformação que o software efetua sobre as entradas para gerar as saídas. Os requisitos funcionais são as funcionalidades sob o ponto de vista do usuário.

Quadro 2 - Requisitos funcionais do sistema.

Nº	Ação	Descrição
01	Cadastrar	O sistema deve permitir o cadastro de pessoas
02	Cadastrar	O sistema deve permitir o cadastro de visitantes
03	Cadastrar	O sistema deve permitir o cadastro de credenciais (cartões RFID)
04	Relatórios	O sistema deve emitir um relatório com o total de ativos/inativos
05	Relatórios	O sistema deve emitir um relatório com o total de pessoas

REQUISITOS NÃO-FUNCIONAIS

Um requisito não funcional de um software é aquele que descreve não o que o sistema fará, mas como ele fará. Os requisitos não-funcionais são as qualidades que o sistema deve ter.

Quadro 3 - Requisitos funcionais do sistema.

Nº	Requisitos	Descrição
01	Externo	O sistema deve limitar apenas um cartão ativo por usuário
02	Externo	O sistema deve bloquear os cartões que passarem da validade
03	Organizacionais	O sistema deve bloquear os usuários inativos com mais de 3 meses
04	Produto	O sistema deve permanecer disponível mesmo após falhas
05	Produto	O usuário deve demorar no máximo 2 segundos para autenticar na catraca

Para realizar o levantamento de requisitos foram aplicadas 3 técnicas:

3.1. Entrevistas (Não estruturada - informal):

Foi realizada uma entrevista informal, com um dos representantes da Global Security sem a utilização de um vocabulário muito complexo, visando um bom entendimento. Durante a conversa foi possível destacar os seguintes pontos de exigência da empresa:

- Sistema simples – Deve ser possível cadastrar equipamentos pessoas e cartões em poucos campos.
- Acesso rápido – A verificação do cartão RFID na catraca não deve demorar mais que 2 segundos.
- Sistema a prova de queda – Verificar a possibilidade de funcionamento das catracas mesmo com falhas elétricas ou de conexão.

- Revisão automática de acesso – O sistema deve ser capaz de verificar e bloquear cartões e pessoas automaticamente de acordo com a utilização ou por tempo de validade.
- Relatórios – O sistema deve gerar relatórios sobre os acessos: quantas pessoas passaram por dia/semana/mês. Quantos usuários estão ativos e quantos foram bloqueados na última semana/mês.

3.2. Montagem de cenários

Ainda durante a entrevista como metodologia primordial da nossa empresa, foi realizada uma montagem de cenários como objetivo de coletar mais detalhes sobre como o sistema deve reagir a determinadas situações, portanto foram elaborados os seguintes cenários:

- Visitantes
    - A empresa determinou que os números de matrículas não precisam obrigatórias, no caso de visitantes o sistema deve cadastrar com o número do CPF.
- Como o sistema deve reagir para visitantes sem documento?
    - A empresa determinou que O CPF também não precisa ser obrigatório, no caso de visitantes sem documento o sistema deve cadastrar com uma foto.
- Segunda via de cartão
    - Os usuários que perderem o primeiro cartão devem solicitar online um novo cartão.
    - O cartão antigo precisa ser bloqueado no momento da solicitação do novo cartão.
    - Os usuários não podem ter mais de um cartão ativo.

4. PROJETO DE SOFTWARE

Segue relação detalhada dos diagramas utilizados no projeto.

4.1. Diagrama de Atividade

Descreve as atividades a serem executadas para a conclusão de um processo.Concentra-se na representação do fluxo de controle de um processo

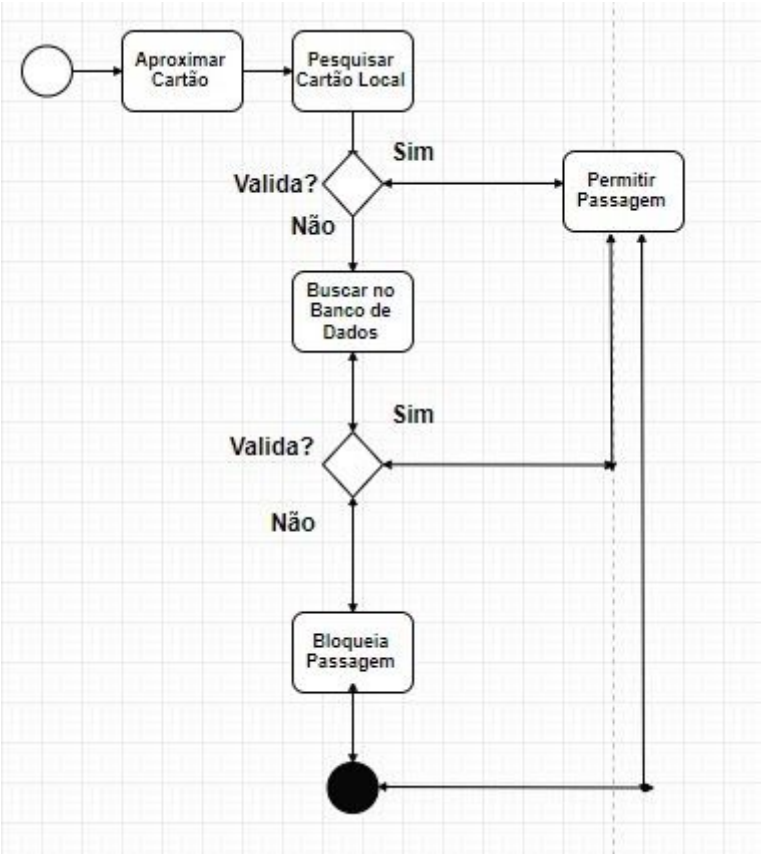


Figura 1: Diagrama de atividade

Diagrama de Caso de Uso

O diagrama de caso de uso descreve a funcionalidade proposta para um novo sistema que será projetado.

Vamos utilizar como exemplo o sistema utilizado pela empresa Global Security, no qual foram elaborados dois cenários, o principal no qual o cliente possui o cartão de acesso, e o alternativo, no qual será necessário a elaboração do cartão de acesso.

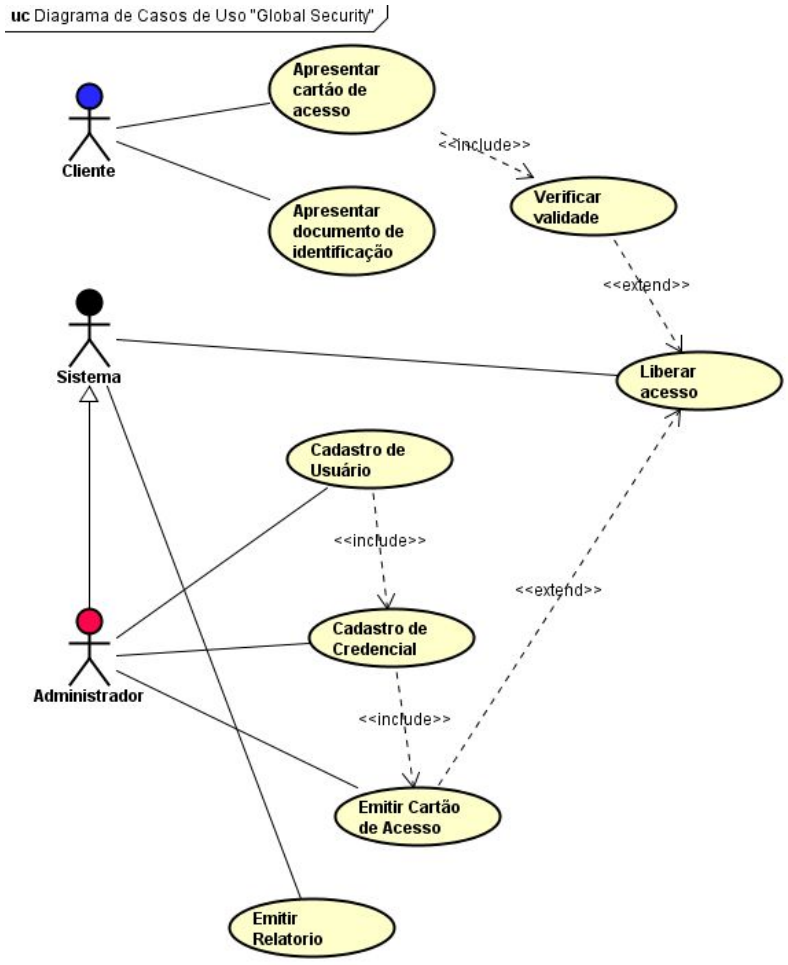


Figura 2: Diagrama de caso de uso

Cenário principal:

1. Cartão de acesso é apresentado
2. Acesso liberado automaticamente pelo sistema

Atores: Cliente, Sistema

Cenário Alternativo:

1. Documento de identificação é apresentado ao administrador
2. O administrador realiza o cadastro do cliente
3. É realizado o cadastro da credencial
4. O cartão de acesso é emitido
5. O acesso é liberado

Atores: Cliente, Administrador

4.2. Diagrama Entidade Relacionamento

Um diagrama entidade relacionamento (ER) é um tipo de fluxograma que ilustra como “entidades”, p. ex., pessoas, objetos ou conceitos, se relacionam entre si dentro de um sistema.

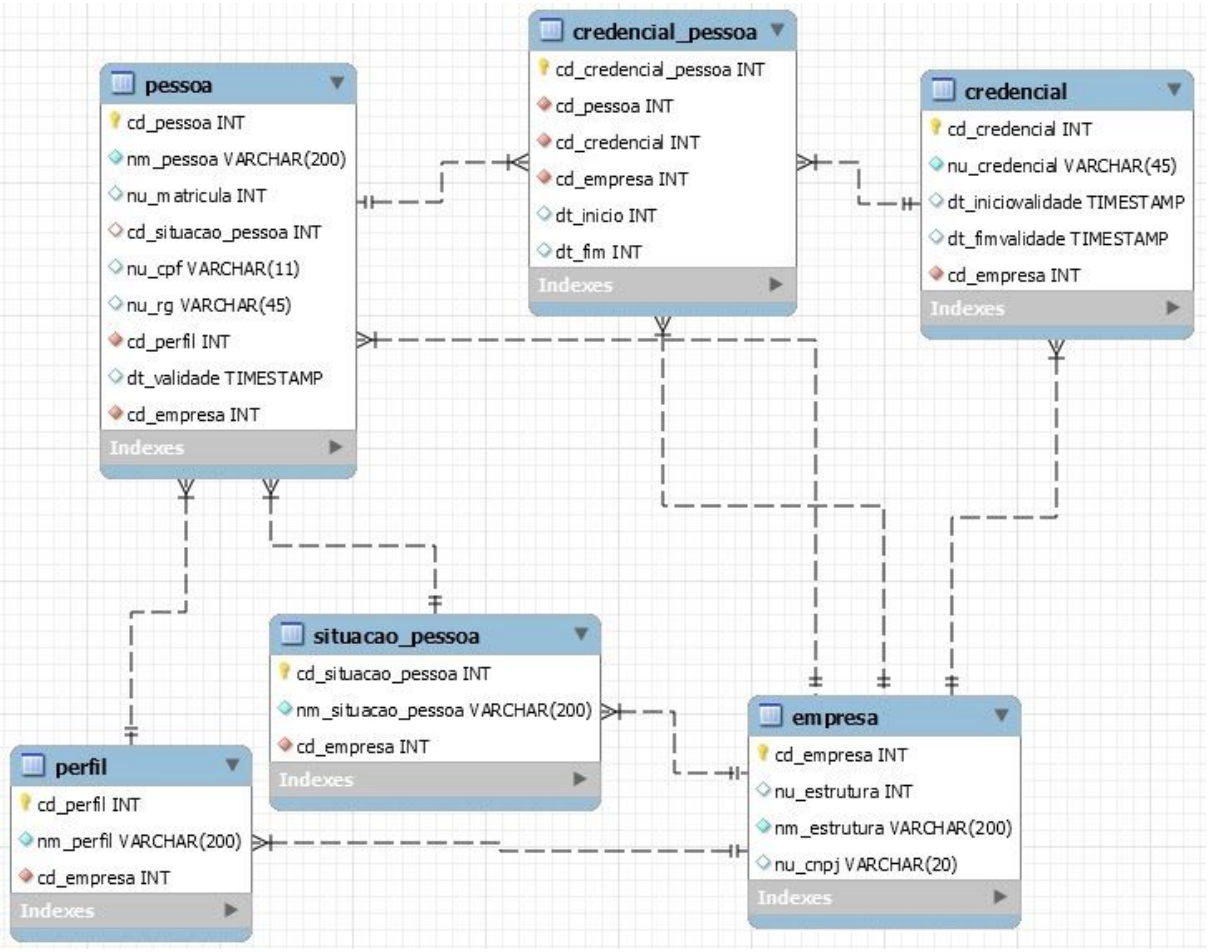


Figura 3: Diagrama entidade relacionamento

No diagrama entidade relacionamento acima, temos as principais entidades do sistema (credencial, pessoal, empresa etc) e as linhas mostram os relacionamentos e dependências entre as entidades. É possível notar técnicas de modelagem como a criação de uma entidade N:N (credencial pessoa) que permite associar mais de uma credencial para a mesma pessoa, que faz parte da regra de negócio definida na etapa de levantamento de requisitos.

5. EVOLUÇÃO DO SOFTWARE

> *Segundo a tradução do artigo:* ***Rules and Tools for Software Evolution Planning and Management*** *(2001), "O fenômeno da evolução do software se manifesta na necessidade comum, desde o início da computação eletrônica, de manutenção contínua e atualizações periódicas de software usado em aplicações do mundo real.".*

Conforme visto na primeiro lei de Lehman, um software que não é constantemente adaptado se torna não satisfatório e obsoleto, portanto, após a conclusão e entrega do software procuramos novamente o cliente perguntando se haviam surgido novas demandas.

Também de acordo com a segunda lei de Lehman, os novos recursos irão aumentar a complexidade do sistema, sendo necessário alterar a modelagem do banco de dados e criar novas telas na aplicação.

EVOLUÇÕES

- A empresa determinou que o sistema de controle de acesso também deve funcionar como um registro de ponto para funcionários, calculando as horas de trabalho e gerando relatórios mensais para os gestores.
- A empresa determinou que o sistema passe a aceitar cadastramento biométrico de digital para os usuários, visando aumentar a segurança pois assim somente pessoas cadastradas teriam acesso à instituição, já que uma das desvantagens do cartão RFID é que pode ser usado por outra pessoa
- A empresa ofereceu feedback dos erros encontrados após a entrega do programa e determinou a correção dos bugs.

6. MANUTENÇÃO DO SOFTWARE

A manutenção de software é o processo de alteração realizado depois que um programa é liberado para uso, segundo Pressman, a manutenção engloba quatro atividades: manutenção corretiva, manutenção adaptativa, manutenção evolutiva e manutenção preventiva.

Neste projeto utilizamos as seguintes manutenções:

6.1. Manutenção Corretiva

- Correções de erros que não foram identificados na parte de testes. Como por exemplo um erro com o formato da data ao ler os registros no banco de dados. O erro acontecia ao gravar os milissegundos e o equipamento aguardar um retorno de tamanho menor, sem os milissegundos.

6.2. Manutenção Adaptativa

- Adição de novas tabelas como por exemplo o tipo de credencial (RFID, Biometria, códigos de barra).
- Alteração no hardware das catracas adicionando leitores biométricos de digitais e leitores de códigos de barra.
- Adição de novas telas e novos relatórios.

6.3. Manutenção Evolutiva

- Adição de novas funcionalidades que não estavam no planejamento inicial como por exemplo, utilizar o software como controle de ponto de funcionários.
- Alteração da documentação original para abranger todas as novas funcionalidades

6.4. Manutenção Preventiva

- Adição de funções de limpeza de logs antigos, para evitar que o arquivo de log no banco de dados ocupe muito espaço e pare a aplicação por falta de espaço em disco.

7. CONTROLE DE VERSÃO

- O controle de versão é o ato de gerenciar variações de um dado documento que pode ou sofre alterações ao decorrer do tempo e pode-se ser utilizado dois métodos de controle mais destacados sejam eles:

7.1. Controle de Versão Centralizado

- O sistema de versão centralizado foi desenvolvido com o intuito de facilitar o trabalho em equipe entre os desenvolvedores.O Controle funciona como um um dado computador como servidor centralizado que armazena todos os dados e atualizações realizadas ,e pode-se resgatar todo o histórico de suas versões. A falha achada nesse método é que caso o computador com os dados arquivos ficasse inoperável poderia haver a perda ou corrupção dos arquivos.

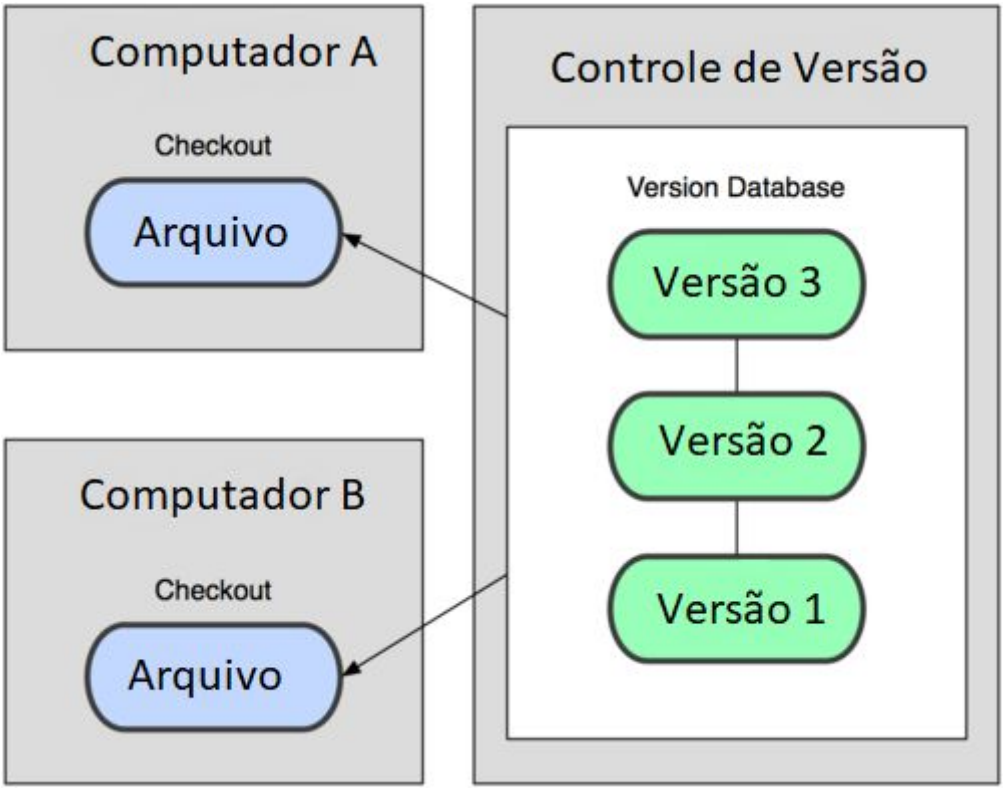


Figura 4: Controle de versão centralizado

7.2. Controle de Versão Distribuído

- O sistema de Versão Distribuído surgiu logo após o problema de dependência do servidor no estilo de Centralizado, o modelo é capaz de distribuir as versões do sistema em computadores dos próprios desenvolvedores tornando assim o possível acesso remotamente, cada cópia existente é salva completamente fazendo com que os próprios clientes não consigam acesso para cópias não autorizadas do arquivo versionado.

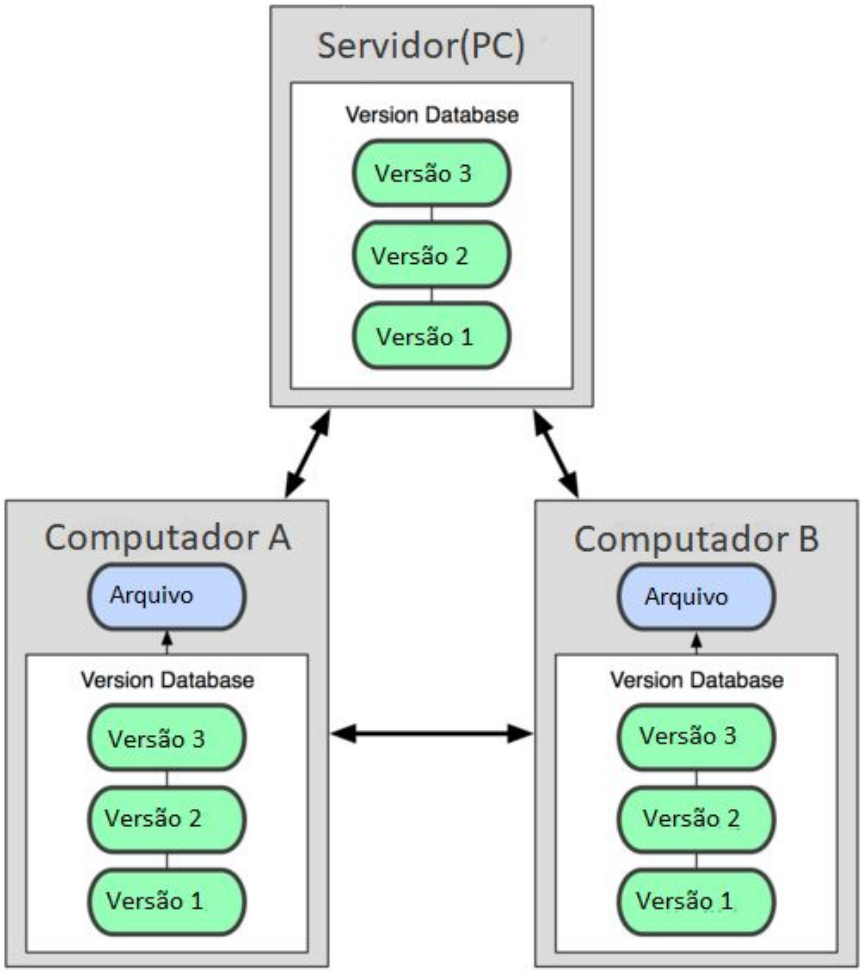
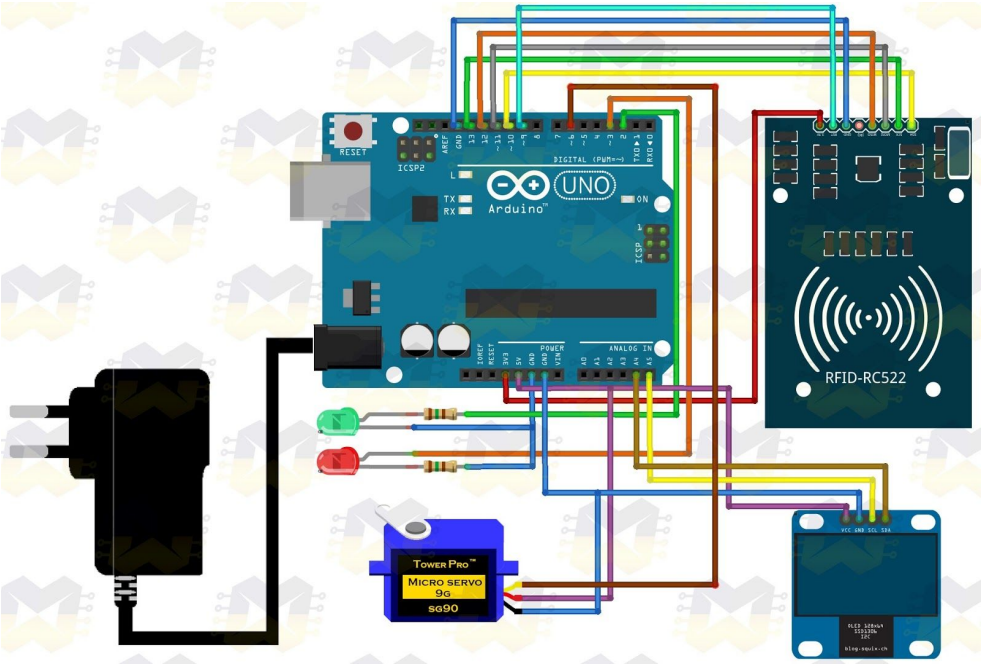


Figura 5: Controle de versão distribuído

O modelo adotado por nós na Global security apesar dos problemas apresentado foi o Controle de Versão Centralizado pois facilita o encaminhamento e o uso dos servidores pois sempre há o check up das catracas e do modelo dos cartões passados nas catracas viabilizando caso ocorra uma atualização da versão que possa indisponibilizar o uso do cartão.

8. EXEMPLO DE SISTEMA DE PROTÓTIPO (RFID)



9. CONCLUSÃO

Engenharia de Software envolve o uso de modelos abstratos e precisos que permitem ao engenheiro especificar, projetar, implementar e manter o sistema de software, avaliando e dando garantia de sua qualidade.

Ao fim desse trabalho, pude compreender ainda mais a importância que a Engenharia de Software tem, como um todo, no processo de desenvolvimento, manutenção e avaliação de um software

Portanto, todas as expectativas e objetivos foram alcançados, além de aprender conceitos e técnicas de Engenharia de Software, pude executar muitas dessas técnicas e comprovar na prática a grande melhoria que a utilização correta e em conjunto das mesmas trazem a um projeto.

10. REFERÊNCIAS

RULES and Tools for Software Evolution Planning and Management. *In*: JUAN F. RAMIL, Juan F.; LEHMAN , Meir. Rules and Tools for Software Evolution Planning and Management. [*S. l.*], 1 nov. 2001. Disponível em: https://link.springer.com/article/10.1023/A:1012535017876 . Acesso em: 24 nov. 2019.

PRESSMAN, Roger S. Engenharia de Software: Uma Abordagem Profissional. 8. ed. [*S. l.*: *s. n.*], 2016.

METODOLOGIAS Clássicas. [S. l.], 1 jul. 2019. Disponível em: http://metodologiasclassicas.blogspot.com/p/blog-page.html. Acesso em: 17 set. 2019.

MODELO em Espiral. [*S. l.*], 16 maio 2019. Disponível em: https://pt.wikipedia.org/wiki/Modelo_em_espiral. Acesso em: 17 set. 2019.

ENGENHARIA de Software: O Modelo Espiral. [*S. l.*], 2 mar. 2007. Disponível em: http://engenhariadesoftware.blogspot.com/2007/03/o-modelo-espiral.html. Acesso em: 17 set. 2019.

REQUISITOS não funcionais. [*S. l.*], 1 fev. 2001. Disponível em: https://www.devmedia.com.br/artigo-engenharia-de-software-3-requisitos-nao-funcionais/9525. Acesso em: 17 set. 2019.

A ESSENCIALIDADE da Engenharia de Software. [*S. l.*], 1 mar. 2012. Disponível em: https://www.devmedia.com.br/a-essencialidade-da-engenharia-de-software/24833. Acesso em: 17 set. 2019.