INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

DO RIO GRANDE DO NORTE

NATHAN GOMES LINHARES COSTA

TIAGO MARQUES MACÊDO

**SISTEMA DE GERENCIAMENTO DE LABORATÓRIOS**

JOÃO CÂMARA/RN

2014

NATHAN LINHARES

TIAGO MARQUES MACÊDO

**SISTEMA DE GERENCIAMENTO DE LABORATÓRIOS**

Trabalho de Conclusão de Curso apresentado à Diretoria Acadêmica do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte – Campus João Câmara, como requisito necessário à obtenção do título de Técnico em informática.

Orientador: Prof. Me. Diego Oliveira

JOÃO CÂMARA/RN

2014

NATHAN GOMES LINHARES COSTA

TIAGO MARQUES MACÊDO

**SISTEMA DE GERENCIAMENTO DE LABORATÓRIOS**

Trabalho de Conclusão de Curso apresentado à Diretoria Acadêmica do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte – Campus João Câmara, como requisito necessário à obtenção do título de Técnico em informática.

Orientador: Prof. Me. Diego Oliveira

Aprovado em \_\_/\_\_/\_\_\_\_\_

BANCA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Me. Diego Oliveira

Orientador – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Me. Nickerson Fonseca

Coordenador do Curso Técnico Integrado em informática – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

**“Nada temos a temer quanto ao futuro, a menos que nos esqueçamos como Deus tem nos conduzido no passado” – Ellen White**

**Dedicamos esse trabalho ao nosso orientador Diego, aos professores Daniel e Lucas que deram uma colaboração importante ao nosso projeto, aos nossos pais e familiares que permitiram que chegássemos até aqui. E a Marcos Vinícius que cedeu o espaço da casa pra se reunirmos nas férias.**

**AGRADECIMENTOS**

Agradecemos ao nosso Deus por ter permitido a nós enfrentar esse último ano letivo com

esforço e persistência. Ao professor Odailson Fonseca pela ideia de projeto final de curso dado a nós, que mesmo não sendo professor de Sistemas de Informações deu importante colaboração para permitir desenvolvermos alguma temática em nosso trabalho de conclusão de curso.

Ao professor Diego por ter sido insistente conosco quanto a necessidade de agilização do trabalho final do curso e por ter ajudado a nós bastante na parte de desenvolvimento de nosso sistema em que decidiu tomar parte ativa no desenvolvimento do software, sendo decisivo para agilização do desenvolvimento do software.

Aos pais de Marcus Vinícius por terem permitido que se encontrássemos várias vezes, já que Nathan não tinha casa em João Câmara, sendo parte importante para possibilitar que conseguíssemos realizar reuniões do trabalho final e pudéssemos aproveitar as férias e contra turno para realizarmos tarefas.

Aos nossos pais por terem colaborado para que pudéssemos custear despesas necessárias a realização do trabalho de conclusão de curso e ser o elo de ligação do passado com o nosso sucesso presente.

**RESUMO**

F

O seguinte trabalho é a descrição das etapas de desenvolvimento de um software que propõe de uma proposta de desenvolvimento de um software para a COLAB, tendo por objetivo principal fazer um melhor controle de chaves com a automação por software em substituição dos registros físicos em papéis, auxiliando o cliente nas tarefas burocráticas diante de auditoria interna, solicitadas pela direção, produzindo um software com base em iterações com o cliente, buscando atende-lo além de atestar conhecimento prático das habilidades exigidas para se outorgar título de técnico em Informática.

# SUMÁRIO

[SUMÁRIO 7](#_Toc408943492)

[LISTA DE FIGURAS 10](#_Toc408943493)

[1. INTRODUÇÃO 1](#_Toc408943494)

[1.1 FUNÇÕES DESEMPENHADAS PELA COORDENAÇÃO DE LABORATÓRIOS 1](#_Toc408943495)

[1.2 PROBLEMAS A SEREM RESOLVIDOS NA COLAB 2](#_Toc408943496)

[1.2.1 Deslocamento 2](#_Toc408943497)

[1.2.1.1 Solução apresentada com a automação 2](#_Toc408943498)

[1.2.2 Ambiental e financeiro 3](#_Toc408943499)

[1.2.2.1 Solução apresentada com a automação 4](#_Toc408943500)

[1.2.3 Organização 4](#_Toc408943501)

[1.2.3.1 Solução apresentada com a automação 5](#_Toc408943502)

[1.2.4 Segurança das informações 5](#_Toc408943503)

[1.2.4.1 Solução apresentada com a automação 6](#_Toc408943504)

[1.2.5 Confiabilidade das informações 6](#_Toc408943505)

[1.2.5.1 Solução apresentada com a automação 7](#_Toc408943506)

[1.2.6 Agilidade 8](#_Toc408943507)

[1.2.6.1 Solução apresentada com a automação 9](#_Toc408943508)

[1.3 Requisitos de software 11](#_Toc408943509)

[1.3.1 Requisitos não-funcionais 11](#_Toc408943520)

[1.3.2 Requisitos funcionais 12](#_Toc408943532)

[2. FUNDAMENTAÇÃO TEÓRICA 13](#_Toc408943533)

[2.1 LINGUAGENS UTILIZADAS 13](#_Toc408943534)

[2.1.1 HTML 15](#_Toc408943536)

[2.1.1.1 História do HTML 18](#_Toc408943537)

[2.1.2 CSS 19](#_Toc408943538)

[2.1.2.1 Atributos de CSS 20](#_Toc408943539)

[2.1.2.2 História do CSS 24](#_Toc408943540)

[2.1.3](#_Toc408943541) *[JSP](#_Toc408943541)* [25](#_Toc408943541)

[2.1.3.1 Benefícios do JSP 26](#_Toc408943542)

[2.1.4 Java 27](#_Toc408943559)

[2.1.3.1 O surgimento do Java 27](#_Toc408943560)

[2.1.3.3 A existência da máquina virtual 28](#_Toc408943561)

[2.1.5 SQL 30](#_Toc408943562)

[2.1.5.1 Operações SQL 31](#_Toc408943563)

[2.2 SOFTWARES DESENVOLVIMENTO UTILIZADOS 34](#_Toc408943564)

[2.2.1 MySQL 34](#_Toc408943565)

[2.2.1.1 HeidiSQL 35](#_Toc408943566)

[2.2.2 Netbeans 35](#_Toc408943567)

[2.2.2.1 História do netbeans 35](#_Toc408943568)

[2.2.2.2 Primeiras versões do Netbeans 37](#_Toc408943569)

[2.2.2.3 Apache Tomcat 37](#_Toc408943570)

[3. METODOLOGIA 38](#_Toc408943571)

[3.1 SURGIMENTO DA IDEIA 38](#_Toc408943572)

[3.2 LEVANTAMENTOS INICIAL DE REQUISITOS 38](#_Toc408943573)

[3.2.1 Professores dos Laboratórios de Informática 39](#_Toc408943574)

[3.2.1.1 Utilidade das informações no processo de desenvolvimento 39](#_Toc408943576)

[3.2.2 Professores dos Laboratórios de Biologia 43](#_Toc408943577)

[3.2.2.1 Utilidade das informações no processo de desenvolvimento 44](#_Toc408943578)

[3.2.3 Alunos Usuários dos Laboratórios Durante as Aulas 44](#_Toc408943579)

[3.2.4 Aluno Bolsista da COLAB 45](#_Toc408943580)

[3.3 ORIENTAÇÕES TÉCNICAS PARA O DOCUMENTO DE VISÃO 45](#_Toc408943582)

[3.3.1 Reunião com Professora Alba 46](#_Toc408943583)

[3.3.1.1 Alterações que fizemos em nosso documento de visão 46](#_Toc408943584)

[3.3.2 Reuniões com professor Lucas 46](#_Toc408943585)

[3.3.2.1 Primeira reunião 46](#_Toc408943586)

[3.3.2.2 Segunda reunião 47](#_Toc408943587)

[3.3.3 Reuniões com o professor Daniel 48](#_Toc408943588)

[3.3.3.1 Primeira correção do documento de visão 48](#_Toc408943589)

[3.3.3.2 Segunda correção do documento de visão 50](#_Toc408943590)

[3.3.3.3 Terceira correção do documento de visão 50](#_Toc408943591)

[1.1 ENTREVISTAS COM ABRANTES 51](#_Toc408943592)

[1.1.1 Entrevista Inicial 52](#_Toc408943593)

[1.1.2 Entrevista após lançamento da primeira versão do sistema 54](#_Toc408943594)

[4. DESENVOLVIMENTO 57](#_Toc408943595)

[1.1 FRAMEWORKS 58](#_Toc408943596)

[1.2 Banco de Dados 59](#_Toc408943597)

[1.2.1 Tabela Laboratório 59](#_Toc408943598)

[1.2.2 Tabela Chave 59](#_Toc408943599)

[1.2.3 Telas da aplicação 59](#_Toc408943600)

[1.2.3.1 Tela de login 59](#_Toc408943601)

[REFERÊNCIAS 61](#_Toc408943602)

# LISTA DE FIGURAS

[Figura 1: Printscreen de exemplo de código JSP embutido no HTML 27](#_Toc409709380)

[Figura 2: Código nas linguagens que surgiram no mercado antes de Java são compilados para sistema específico 30](#_Toc409709381)

[Figura 3: Um único código compilado serve para todos os sistemas operacionais, pois há a máquina virtual como intermediador da execução 31](#_Toc409709382)

[Figura 4: A página de solicitação de serviços requer primeiramente o nome do solicitante, sua matrícula, o setor relativo ao qual quer manutenção 42](#_Toc409709383)

[Figura 5: A página de solicitação de serviços requer depois o nome o tipo de solicitação, descrição do pedido, prioridade, data e horário de conclusão 43](#_Toc409709384)

[Figura 6: Diagrama de casos de uso feito para ser apresentado a Abrantes, colocado no documento de Visão 52](#_Toc409709385)

[Figura 7: Tela de login ao ser preenchida e visualização dos campos com caracteres de senha escondidos 60](#_Toc409709386)

[Figura 8: Tela de empréstimo chaves 61](#_Toc409709387)

[Figura 9: Seleção do laboratório para o qual se reservará a chave 62](#_Toc409709388)

[Figura 10: Após apertar no botão OK, os dados são processados e executado o empréstimo de chave 63](#_Toc409709389)

[Figura 11: Tela que lista os laboratórios ocupados e disponíveis. De vermelho ocupados e verdes disponíveis 63](#_Toc409709390)

[Figura 12: Laboratório 2 se encontra ocupado e laboratório 3 se encontra disponível 64](#_Toc409709391)

[Figura 13: Clica-se no botão OK ir à página da retirada da chave, que será feita pelo sistema 65](#_Toc409709392)

[Figura 14: Tela com exibição de horários 66](#_Toc409709393)

[Figura 15: Seleção de horários é permida por meio de selects que fazerm parte de um formulário 66](#_Toc409709394)

[Figura 16: No menu lateral direito se dá a opção de aceder à tela de cadastro de pessoas que trabalham na COLAB 68](#_Toc409709395)

[Figura 17: Formulário que permite cadastrar um usuário que trabalha na COLAB 68](#_Toc409709396)

[Figura 18: Campos não preenchidos retornam uma advertência a ser atendida antes de prosseguir no cadastro 69](#_Toc409709397)

[Figura 19: Página para a qual o navegador é redirecionado, apenas para o caso de ter sido concluído o cadastro 69](#_Toc409709398)

[Figura 20: Essa outra tela de gerenciar laboratórios permite que os membros da COLAB editem os dados de usuários de seu sistema 69](#_Toc409709399)

[Figura 21: É permitido aos membros da COLAB editar as informações vinculadas ao usuário selecionado na tabela da página anterior 70](#_Toc409709400)

[Figura 22: O usuário é redirecionado para uma página que avisa que não foi recebido nenhum valor enviado por outra página. 70](#_Toc409709401)

[Figura 23: A página Confirmar Cadastro de Membro da COLAB avisa quando o cadastro é completado 71](#_Toc409709402)

[Figura 24: Estrutura da tabela professores 72](#_Toc409709403)

[Figura 25: Consulta foi utilizada para filtrar dados do banco de dados da aplicação por nós desenvolvida 73](#_Toc409709404)

[Figura 26: Código para request para incializar variáveis passadas enviadas por um formulário 74](#_Toc409709405)

[Figura 27: Código que cria várias linhas de tabela imprime dentro dessas células 75](#_Toc409709406)

# LISTA DE TABELAS

[Tabela 1: Requisitos não funcionais definidos no início do projeto 44](#_Toc409980234)

[Tabela 2: Requisitos não funcionais após correções iniciais 44](#_Toc409980235)

# INTRODUÇÃO

Esse projeto de software visou automatizar o processo de reserva de laboratórios do IFRN Campus João Câmara. O seguinte trabalho propõe uma solução automatizada para gerenciar os dados de solicitações de retirada de chaves dos laboratórios supervisionados pela Coordenação de Laboratórios.

Nesse TCC, nos textos abaixo, estaremos chamando os bolsistas ou servidores da COLAB de “membros da COLAB” ou “membros do setor”, enquanto aos usuários que dependem dos serviços da Coordenação de Laboratórios estaremos chamando eles de “usuários da COLAB” ou simplesmente se referindo como “usuário” ou como a pessoa que retira a chave do laboratório. A sigla COLAB representa o nome do setor em questão: a Coordenação de Laboratórios.

Procuramos compreender a missão da Coordenação de Laboratórios, que entrega a chave aos usuários, sempre que tenha sido reservado horário em acordo com a COLAB, feito exclusivamente por professor.

Procuramos também encontrar problemas existentes no trabalho do setor para o qual estamos desenvolvendo o software. Procuramos a solução de cada um desses problemas encontrados, que necessitavam de desenvolver métodos para serem resolvidos, utilizando a automação.

## FUNÇÕES DESEMPENHADAS PELA COORDENAÇÃO DE LABORATÓRIOS

A Coordenação de Laboratórios gerencia a marcação de horários dos laboratórios, controla a entrega de chaves para os professores, e bolsistas de professores, quando se entra em acordo para que os alunos possam ter acesso às chaves.

Entra em contato com os professores para reserva de horários regulares para aulas práticas, que determinam quais horários do laboratório serão utilizados para ministrar as aulas práticas, dentre os disponíveis.

Os professores da disciplina de informática são maioria dentre todos os usuários que estão submetidos à COLAB para desenvolverem suas atividades, já que os laboratórios de informática estão em maior número que laboratórios voltados a quaisquer outras áreas dentre os laboratórios supervisionados pela COLAB, como as áreas de línguas, química, manutenção ou biologia, que tem apenas um laboratório cada uma delas.

Existem sim outros laboratórios como no anexo e os na área externa da escola, mas esses laboratórios do Campus João Câmara são de uso dos cursos superiores de Licenciatura em Física e Energias Renováveis, portanto não estão entre os que fazem parte do domínio da COLAB e não tiveram seu uso estudado por nós.

O setor para qual o sistema está sendo desenvolvido utiliza um catálogo de assinaturas, que são o controle de empréstimo de chaves, que pode ser levada por um usuário dele, quando combinado com o coordenador do referido setor em um horário reservado para atividade.

Esses registros contem a assinatura da pessoa que porta a chave, o horário em que a pessoa retirou a chave do setor, e também o registro do horário de devolução, (quando devolvida a chave) seguido da assinatura do membro da COLAB, que ateste ter recebido a chave para guarda na sala da Coordenação de Laboratórios.

Cada uma dessas chaves emprestadas tem um número associado a elas, e existe uma relação que indica qual número de chave está relacionado a um laboratório específico. Cada laboratório tem pelo menos duas chaves, cada uma com sua numeração específica, sendo uma utilizada como a chave principal e a outra reserva.

As chaves de uso prioritário ficam normalmente acessíveis, em cima da mesa do membro do setor para o qual estamos desenvolvendo o software, responsável por liberar a chave, enquanto as chaves reserva estão guardadas num armário porta-chaves, onde serão utilizadas apenas se perdida ou inacessível a chave principal.

## PROBLEMAS A SEREM RESOLVIDOS NA COLAB

Nós pretendemos solucionar os problemas abaixo descritos em cada subtópico com a automação por software operando na web.

O primeiro problema, de deslocamento é um problema que atrapalha os membros da COLAB, por causa da distância das informações, que são armazenadas em registros físicos.

Há apresentação, também do problema financeiro e ambiental, em que explicamos sobre como os papéis podem ser maléficos em seu aumento contínuo de armazenamento, com dificuldade de organização deles e buscas nos registros.

Sobre o problema da organização, explica-se como o armazenamento das informações interfere na rapidez no procedimento de retirada da chave, e por não se usar automação erros de anotações de horário incorretas podem comprometer o cumprimento da função do registro.

Sobre a segurança de informações, há um subtópico esclarecendo como os dados registrados em papéis tem algumas formas de serem destruídos ou perdidos, sem chances de reparo, o que faz questionar sobre ser realmente seguro utilizar os atuais registros físicos do setor.

Sobre a confiabilidade das informações, comentamos como alguns dados, que podem ser vistos em auditoria, podem ser comprometidos quanto à sua validade e veracidade, pelo preenchimento manual que pode ser manipulado mais facilmente, enquanto que o automatizado é mostrado como solução para diversos problemas de confiabilidade das informações expostos no subtítulo.

O último tipo de dificuldade apontado é a questão da falta de agilidade em alguns processos, como quanto a correções necessárias, como a da questão de os registros físicos não conterem o nome de quem entregou a chave não poder ser resolvido pelos registros físicos, sem aumentar a lentidão no processo, que exigiria mais papéis e mais campos manuscrito. A utilização dos registros físicos tem a comparação com casos em que seria menos rápido utilizar os papéis ao software.

### Deslocamento

Atualmente, as informações a respeito de reservas de laboratório são prestadas fisicamente na COLAB, o que representa alguma perda de tempo, com as constantes idas e vindas ao setor. Se o professor estiver no Campus, esse problema pode não ser tão acentuado, porém se ele não estiver, torna-se maior a dificuldade da distância.

Os próprios membros do setor para o qual estamos desenvolvendo o software, ao momento em que se retiram da sala do referido setor, não tem acesso a mais nenhuma informação.

#### Solução apresentada com a automação

A automação permitirá que os membros da COLAB possam acessar essas informações mesmo sem estarem na sala do setor. Isso permitirá a eles saberem quais horários estão em uso no referido setor para cada atividade, como também permitirá que saibam responder quem está com a chave, mesmo quando estiverem resolvendo problemas noutra parte da instituição ou noutra atribuição de seu cargo ou função, dando as informações concernentes às reservas de chaves, mesmo estando noutro local, que não a sala do setor em questão, como computador de outro setor ou sala. Podendo, por exemplo, ao momento em que estiver fazendo uma manutenção em algum dos laboratórios que são coordenados pelo setor, gerenciar essas informações, simplesmente conectando a um dos computadores em que esteja disponível acesso à rede (presente em todos os laboratórios, por motivo de os locais contém ao menos um), sem ser necessário carregar papéis com essas informações, como seria obrigatório para realizar essa atividade noutro lugar.

Embora obviamente para ter acesso a chave, tenha que se ir fisicamente à COLAB, quando se disponibiliza o acesso a essas informações da retirada de chave independentemente de local, aumenta a comodidade na consulta de informações, que podem ser requeridas em necessidades de alguém saber onde está a chave imediatamente, já que com qualquer computador com acesso à internet será possível gerenciar o sistema.

O bom de prestar essas informações no sistema é inclusive a possibilidade de um membro do setor acompanhar a atividade à distância que o seu colega fez. Inclusive quando não estiver no Campus ele poderá realizar esse tipo de consulta.

Uma informação, portanto, pode ser dada por um servidor da COLAB à distância, como numa ligação telefônica. Algum usuário dos laboratórios pode ter interesse em saber com que outra pessoa está a chave, enquanto a sala da COLAB possa estar fechada para almoço e ele não encontra o servidor do setor que deu baixa nas informações.

Esse professor poderia entrar em contato com um outro funcionário da COLAB que poderia rapidamente verificar com quem está uma chave, e se ela foi devolvida na mão de algum funcionário da COLAB, ou se está ainda com algum servidor que realizou empréstimo dela. Assim passa a ser possível obter informações de importantes procedimentos da COLAB, sem perder a privacidade informações dos processos do setor, que permanecem restritas ao referido setor, não estando a vista de quem não está autorizado a realizar divulgação dos dados referentes ao controle dos laboratórios.

Ao mesmo tempo em que se mantém a segurança (devido não ser permitido acesso aos dados de trabalho da COLAB sem a devida autenticação), que até certo ponto também estava presente nos registros físicos, pois teoricamente os papéis não podem sair do referido setor, passa a não haver mais prejuízo de os dados contidos nos registros das atividades da COLAB não estarem acessíveis, caso não se esteja na sala do setor em questão. Não basta dar a segurança das informações, é necessário torná-las mais facilmente acessíveis a quem tem credenciais para isso, com uma maior agilidade no processo de registro de retirada da chave.

### Ambiental e financeiro

O consumo de papéis é necessário para a atual maneira de trabalhar da COLAB. O registro físico das entregas e devoluções existentes nos empréstimos de chaves embora seja uma maneira rápida de se fazer o registro, tem elementos negativos como o acúmulo de papel, (que fará necessário um descarte mais rápido de papel).

O problema do acúmulo de papel ocorre devido à quantidade gasta deles para anotar as assinaturas de quem retira a chave da COLAB, pois a grande quantidade de assinaturas demandará a longo prazo uma grande quantidade de material físico, já que os registros têm crescimento constante, e sendo feitos em anotações físicas, necessitarão tanto de mais material (papéis e canetas), como espaço físicos maiores para armazená-los.

A demanda por registros trará uma necessidade de fazer a compra de papéis reservada para esse propósito específico sempre necessária. Ela também tem o problema de ser uma despesa variável, pois depende de quanto se use o empréstimo de chaves do laboratório. Mesmo que sejam gastas poucas folhas ao mês, podem a longo prazo ter número expressivo.

O problema ambiental do gasto de papel vai além do malefício ao ambiente natural, que é atingido pela necessidade de derrubada de árvores para fabricação de novas folhas de papel. O ambiente artificial (sala da Coordenação de Laboratórios) também sofre desorganização. O ambiente necessita de se adaptar constantemente ao crescente volume de papéis e com compra de pastas.

#### Solução apresentada com a automação

O registro automatizado por software além de também ser ágil, tem rápida consulta ao acervo, devido à organização sistemática dos dados, que são armazenados dinamicamente, evitando a necessidade de aumentar o espaço físico de armazenamento dos dados.

A organização do espaço físico melhorará, pois o espaço físico para o armazenamento virtual é fixo, não adicionando para o caso da COLAB nenhum volume, pois mesmo as máquinas já estão instaladas no setor, portanto nenhum aumento de estrutura física e gasto de espaço haverá e nenhum contraste ou estranheza causará ao ambiente de trabalho para os bolsistas e servidores que trabalhem na sala, pois as máquinas já estão inseridas na rotina do setor em questão, e não haverá desorganização física do ambiente, portanto, por necessidade de computadores para acessar uma aplicação web.

A utilização adequada e racional do sistema, por sua vez, deixará organizados os dados virtuais. As informações que serão salvas no banco de dados do sistema, não são capazes de congestionar o servidor, nem mesmo os registros, que exigem salvamento de poucos dados.

Mesmo com o aumento do número de informações, a ação simplificada pelo sistema da que estamos fazendo para o referido setor, não gera uma quantidade enorme de dados digitais, sendo suportável ao servidor, e permitindo um uso contínuo uso do espaço do armazenamento no servidor para os registros digitais.

A automação pode tornar desnecessário o uso do papel para o registro da entrega e devolução da chave, propiciando maior organização, diminuindo a dependência do papel e também diminuindo a necessidade do gasto de dinheiro para a compra de papéis.

### Organização

É necessário organizar papéis. Na atual maneira de trabalhar da COLAB, além de se exigir fazer uma busca visual um a um dos registros de chaves antigos, se permite unicamente que se realize o registro de forma física, em papel, das retiradas de chave, o que deixa o ambiente de difícil uso ao usuário do laboratório que deseje retirar a chave.

Ao usuário do laboratório que necessite pegar a chave, é necessário procurar entre papéis em pastas a folha relativa ao laboratório que deseje pegar uma chave, do qual de direito, precise retirar, e verificar se a folha já está toda preenchida e sem espaço para sua assinatura. Caso já esteja, tem que procurar entre as folhas, a que se contenha os registros mais recentes. Caso não haja um papel impresso com espaços vagos para assinar, uma nova folha deve ser providenciada.

Verifica-se que um agravante é que o usuário procura a lista de assinatura numa pasta que muitas vezes contém fichas de registro de entrega de chaves de mais de um laboratório, com abreviaturas dos nomes e indicações dos laboratórios que por vezes são compreensíveis apenas às pessoas que trabalham no setor para o qual estamos desenvolvendo o software.

Após o usuário do laboratório que irá retirar a chave, conseguir assinar seu nome, o membro do setor em questão entrega a chave, porém imediatamente anota o horário em que a pessoa portadora da chave retirou-se do referido setor.

Diante dessa situação, o que pode causar descontrole no trabalho de quem entrega a chave é a questão de esquecimento de anotar o horário, que pode não ser um erro notado imediatamente podendo ser esquecido. Isso pode ocorrer várias vezes, assunto que falaremos mais à frente no subtópico de Segurança das Informações.

#### Solução apresentada com a automação

Com a automação por software, torna-se mais prática a maneira como se registra a retirada da chave, embora possa ser mais centralizada a retirada da chave nos membros do setor para o qual estamos desenvolvendo o software. O próprio trabalho dele será feito de maneira mais rápida e prática, do que à maneira antiga, que podia causar problemas com a procura de papéis.

O armazenamento é dinâmico. Para criação de um espaço para armazenamento de uma ficha de mais outro laboratório, o sistema automatizado permite criação dinâmica desse espaço, já que criação de arquivo virtual não necessita de muito espaço físico, não sendo necessário, principalmente no caso do setor do setor em questão ter que gastar com armazenamento de enorme espaço, já que o tamanho das informações não é tão grande, por serem informações de tamanho pequeno, portanto uma quantidade pequena a média de dados salvos estarão em uma base de dados em um servidor, não sendo necessário grande gasto.

A COLAB, principalmente, não deve precisar um alto investimento para organização de seu trabalho, principalmente porque poderá hospedar o serviço em um repositório no servidor do IFRN.

Com o arquivo virtual, não há necessidade de gastar bastante tempo procurando entre vários papéis para achar uma informação, porém apenas com uma busca rápida no Banco de Dados pode ser exibida.

Do laboratório do qual se queira retirar uma chave, portanto, poderá ser rapidamente consultado para ver se ela está em posse de alguém, podendo ser retirada apenas se o usuário tiver permissão para tal. Não há demora para encontrar qual o laboratório se deseja reservar. O usuário não terá dificuldades em efetuar a reserva, nem a mesma demora que antes existia na procura da pasta, do grupo de papéis, e. O processo é agilizado, portanto agora o membro da COLAB pode ele mesmo realizar praticamente todo o processo, não precisando ter que fazer o usuário efetuar sua retirada de chave sozinho.

Com os dados exigidos somente, como matrícula e horário reservado para ele, o bolsista ou servidor da Coordenação de Laboratórios pode entregar chave, caso seja a pessoa responsável por utilizá-la no momento em se quer utilizá-la.

### Segurança das informações

Apesar da restrição de acesso da informação presente no papel, a segurança e a confiabilidade das informações armazenadas é questionável em alguns aspectos:

* Por estar constantemente sendo manipulados manualmente e de maneira constante, os papéis entram em contato com suor, sujeira, consequentemente diminuindo a durabilidade das informações presentes no registro da COLAB
* Os papéis estão sempre próximo das mãos de quem está trabalhando na COLAB, normalmente passando o dia todo em proximidade de onde a pessoa está sentada na mesa, dando chances a quem está trabalhando em algum momento derramar alguma bebida ou algum material, como cola ou outros materiais de escritório que podem apagar ou destruir os dados que a algum tempo vem sendo armazenados.
* Os dados gravados em papéis, com a entrada de pessoas que vem a sala podem ser rasurados e perdidos como alguma forma de vandalismo sem se saber quem foi responsável por fazer isso
* Os papéis em algum momento em que se porte eles e se coloquem em local que não são achados, se não tiverem alguma cópia de informações, há grande chance de perda de dados.

#### Solução apresentada com a automação

Com a automação por software os dados passam a ter maior dificuldade de perda, pois se evita que devido a fatores destruidores de informação possam se perder os registros. Suor e poeira teriam ínfima chance de danificar o HD onde estão os arquivos do servidor, pois esses dispositivos de armazenamento não necessitam de contato manual constante, portanto, evita-se problemas de permitir que esses elementos danificadores causem problemas às informações, pois não há mais utilização, com o estabelecimento do sistema, de arquivos físicos para armazenamento de informações.

Também, mesmo tendo riscos como em locais de trabalho a pessoa que esteja responsável por fazer a manutenção do computador que hospede os arquivos do sistema, deixe alguma bebida em sua mesa de trabalho, dificilmente isso estará perto da máquina e do HD do servidor que armazene as informações.

Pode haver erros quanto à manutenção periódica do computador servidor que hospede o sistema, já que são passíveis de erro procedimentos como esse, erros humanos, com atualizações necessárias na máquina ou razão de outro motivo.

Materiais de trabalho utilizado na manutenção da máquina servidor podem danificar o computador por procedimentos incorretos. Deve se ver, porém, que o computador que hospedará os arquivos do sistema, não tem manipulação tão constante quanto a sofrida pelas folhas de papel. Há um número pequeno de intervenções no hardware da máquina em relação ao que acontece em registros escritos.

A perda de dados também é algo que é bem dificultado com utilização de um software que esteja bem escrito e em perfeito funcionamento. O servidor tem os dispositivos de armazenamento pouquíssimo manuseados, ao contrário dos dados escritos fisicamente. O acesso às informações passa a ser uma busca aos dados armazenados no banco de dados, feita remotamente, à distância do local de trabalho, o que garante segurança dos dados daqueles, que poderiam ter interesse por essa informação, em tentativa de conseguir fazer algo prejudicial às regras do negócio do setor em questão.

As chances de vandalismo são grandemente diminuídas, já que várias vezes será pedida autenticação do usuário para evitar exclusão de dados. Pela autenticação para possibilitar a exclusão de dados, se diminui consideravelmente esse tipo de problema causador de perda de informações.

### Confiabilidade das informações

Apesar de ter um funcionamento bem definido e compressível, a falta de auditoria para descobrir quem alterou informações, a falta de ferramentas automatizadas que avisem erros comuns, como o anteriormente citado esquecimento de inserção do horário da retirada da chave, podem tornar o trabalho do registro de informações de pouca utilidade se as informações são registradas de maneira totalmente incorreta.

À forma não automatizada de trabalhar, o membro da COLAB pode salvar dados de maneira incorreta por não perceberem o erro na mesma hora. Sabemos que há um formulário em papel que armazena horário de entrega da chave, registra também o usuário responsável pela chave durante reserva concomitante para atividade no laboratório, seguido do horário de retirada, a assinatura, o nome do membro da COLAB que atesta a devolução do horário e o horário de devolução.

Os erros que podem ser cometidos no registro de papel, e ao não utilizar automação, podem ser principalmente:

* As assinaturas das pessoas responsáveis pelas chaves podem não ser compreensíveis ao servidor ou bolsista que esteja trabalhando, e caso a pessoa não tenha em mente quem lhe pegou a chave emprestada, isso pode causar problemas quanto à identificação
* O membro da COLAB responsável por entregar a chave pode solicitar a assinatura da pessoa, e talvez não anote o horário de retirada imediatamente porque deixe para analisar qual o horário atual após a assinatura. Não há problema no fato de o registro poder aparecer um pouco de tempo antes de ter se retirado a chave, caso ele anote antes de a pessoa que retira a chave assinar. Ou logo após o professor assinar, o membro da COLAB anote o horário. O problema é o membro do setor em questão esquecer de escrever o horário, e tendo que anotar bem depois, a hora fique com uma grande discrepância em relação à hora real do procedimento de entrega
* Parecido como o problema anterior, sendo a informação inversa: que no lugar de esquecer o horário de retirada da chave, que se esqueça de anotar a hora de saída, não permitindo depois que outro membro da COLAB saiba informar se determinado usuário está ou não entregando a chave no horário em que deve entregar.
* Há o problema de a folha de cadastro de entrada e saída de chave só registrar o servidor ou bolsista responsável por anotar a devolução da chave, enquanto não dá para saber quem entregou a chave à pessoa, no caso de a assinatura conter informação que não permita identificar a pessoa, já que a assinatura pode estar ilegível ou o professor que assinou tenha cometido um erro no momento da assinatura também cause dúvidas sobre quem emprestou a chave. Sabendo quem liberou a chave, permitiria saber com mais facilidade quem foi que saiu com a chave
* A necessidade de preencher manualmente as informações, falta de dados como o de que qual bolsista entregou a chave, e a dificuldade em inserir mais informações, devido o preenchimento das informações ser totalmente manual.

#### Solução apresentada com a automação

A automação pode resolver problemas especificamente causado pelo processo manual utilizado para fazer atividade de empréstimo na COLAB.

A automação por software resolve problemas de não conseguir visualizar assinaturas que que ficam ilegíveis, pois o sistema armazenará o nome da pessoa, caso ela esteja devidamente cadastrada no sistema. Ao se cadastrar um usuário no sistema, se evitará que o nome de uma pessoa esteja errado, dificultando a identificação do usuário que pediu empréstimo da chave do laboratório. Se porventura ainda ocorrer esse problema, será corrigido de maneira definitiva, já que o sistema relacionará aquele usuário à sua matrícula, e caso queira se substituir o nome entrado errado no sistema, a informação aparecerá correta em todos os registros de empréstimo de chave relacionados àquele usuário.

O bolsista ou servidor da COLAB pode acabar se esquecendo se deixar para depois quando faz registro manual do empréstimo de chave, porém, esse problema pode ser resolvido pela automação: passando a ser fixar o horário pelo horário atual, entrado pelo próprio sistema, evitando erros causados por esquecimento.

O mesmo ocorre também com a devolução de chaves, que passa a registrada junto com a confirmação de devolução por parte do membro da COLAB.

No formulário, algumas informações se encontram ausentes, como a informação de qual bolsista ou servidor da COLAB emprestou a chave, porém esse processo manualmente não está será mais lento de executar. Para resolver esse problema a automação poderá pedir as mesmas informações: nome do portador da chave empresta, horário de empréstimo, horário de devolução, nome de membro da COLAB que afirma a chave ter sido devolvida. Para efeito de mais perfeita auditoria do usuário, se colocaria quem entregou a chave a ele. Como já falamos nessa subseção, a solução de registrar e selecionar usuário registrado sistema da COLAB, reduziria grandemente os nomes estarem errados na lista de registros, e a falta de legibilidade do nome acabaria. Porém seria possível que o bolsista ou servidor da COLAB selecionasse o professor errado para empréstimo da chave

Para o caso de ocorrer o problema de selecionar usuário errado, a auditoria do usuário que recebeu a chave da COLAB, se daria sabendo qual membro da COLAB teria entregado a chave ao professor, verificando por meio da interface web os dados do banco que apontam quem da equipe da COLAB teria realizado o empréstimo. Esse dado, permitiria saber a quem foi entregue, se perguntando diretamente ao bolsista ou servidor da Coordenação de Laboratórios a quem ele teria entregue, caso se verificasse que a pessoa que ele tivesse registrado a chave não havia recebido. Essa estratégia resolverá certamente descobrir o problema se o usuário lembre quem foi a pessoa a quem entregou realmente a chave.

O fato de saber que o membro da Coordenação de Laboratórios que fez a entrega da chave foi realmente quem o sistema diz ser tem embasamento, pois o usuário necessitou logar, conhecendo senha e usuário.

A automação dá solução a esses problemas existentes no modelo manual de registro de atividades da COLAB.

### Agilidade

O processo extremamente manual decorrente de utilizar atualmente o papel para se registrar o empréstimo de chaves faz com que as informações, embora organizadas em apenas quatro colunas numa folha impressa (data e horário de empréstimo da chave, pessoa responsável pela chave, horário de devolução, nome do membro da COLAB) não sejam de difícil nem lento preenchimento, são não rapidamente encontrados devido a necessidade de se procurar entre vários papéis as informações, localizando primeiramente a pasta, depois encontrando na pasta encontrada quais folhas estão com os registros do laboratório de interesse, por fim olhando qual das folhas contém a informação do dia, horário mais recente para que seja continuada com a lista de assinaturas referentes a retirada de chaves para aquele laboratório. No caso de não haver folha para continuar assinando, (a mais recente, com dados relativos a retiradas de chave do laboratório, já esteja completamente preenchida) uma nova folha deve ser providenciada, para o caso de não haver mais espaço para assinaturas.

Da mesma forma pessoa que lhe interessa saber por questão de auditoria os procedimentos tomados por uma pessoa, como saber se tem regularmente entregue a chave ou se queira fazer uma avaliação geral dos dados registrados procurando saber quais chaves não foram devolvidas a considerável tempo, a missão será dificultada pela barreira física imposta: várias folhas de papel e busca manual, o que demonstra necessidade de encontrar soluções para o setor, que aqui sugerimos que venha da automação.

Verificamos a importância de se registrar a informação de forma digital de quem foi o membro do setor em questão que fez a entrega das chaves para o usuário de laboratório, para o caso de a assinatura com o nome da pessoa que retirou a chave estar ilegível. Sabendo quem entregou a chave, há maiores chances de ser informado do evento, perguntando ao membro do setor para o qual estamos desenvolvendo o software, quem entregou a chave, o nome de quem saiu da COLAB com ela.

Falamos isso pois, embora haja a necessidade de se registrar esse dado, ocorre de não haver registro dele – o de qual membro do setor em questão entregou a chave - a fim de verificar se ele entregou a chave a uma pessoa autorizada.

Porém utilizar mais um campo para preenchimento dados na tabela de preenchimento, escrito a mão, gastaria mais tempo. Portanto caso se necessite pedir mais dados, com certeza o processo ficará mais lento e menos prático, mesmo para a informações que consideramos necessária ser adicionada: nome do membro do setor em questão que liberou a chave.

Pode ocorrer também o problema de o próprio membro do setor para o qual estamos desenvolvendo o software colocar uma chave no local errado, recebendo-a após ser devolvida e não a guardar em local adequado, como o armário pertencente à sala. Nesse caso, é necessário identificar o membro do referido setor que recebeu a chave, o que faz muito importante manter essa informação registrada, que embora já esteja contida nos registros em papéis, fica de mais fácil acesso num sistema informatizado, evitando a dificuldade de achar entre papéis, que podem se tornar mais demorado se encontrar informações.

O registro de uma nova informação como essa, (de quem entregou a chave, ainda não registrada) é dificultado por questão do uso de anotações físicas que tornam o processo mais lento. Para adicionar essa informação, necessitaria de mais espaço na folha de papel, sendo necessário ao invés de uma linha, duas para adicionar esse dado, possivelmente, devido ao pouco espaço nas linhas do papel de controle de horário de entrada e saída dos laboratórios supervisionados pelo setor em questão.

Mesmo com um processo automatizado, seria necessário fazer o registro de quem autorizou a saída da chave pela necessidade de saber quem retirou a chave, caso também o membro do referido setor colocasse o nome da pessoa errado. Esse problema pode ocorrer, caso haja ilegibilidade do nome de quem entregou a chave ou então por erro na anotação, simplesmente.

Se acontecer de que o membro da COLAB saiba por ter em sua memória, o nome da pessoa a quem ela entregou a chave, o problema poderá ser resolvido,

#### Solução apresentada com a automação

Com a automação do processo de registro de reserva de chaves resolve-se o problema da lentidão do preenchimento dos formulários que levam o nome da pessoa que levou a chave, a data e hora de saída dela do referido setor, o nome do membro da COLAB que recebeu a chave e hora em que foi devolvida. Esse formulário passa a ser preenchido bem mais rapidamente e com avisos de erros, do sistema, que permitirá fazer salvamento apenas com a correção deles, que no processo de escrita manual talvez pudessem ser vistos muito tempo depois. Algo como uma informação obrigatória não preenchida, existente nesse formulário escrito, poderia ser ignorada, até inconscientemente ao membro do setor em questão esquecer de tomar nota do horário. Esse esquecimento, porém, o o software não permite, pois ele faz entrada do horário do sistema.

O que pode tornar mais rápido, além de seguro o registro das informações, é que a entrada de dados como a data e hora em que se entrega a chave passará a ser feita diretamente pela hora e deta atuaisdo sistema, o que tornará mais prático o processo de registro da informação. Também o nome do servidor ou bolsista da Coordenação de Laboratórios passará a ser entrado automaticamente, assim, não terá necessidade de preenchê-lo para cada registro, como se fazia em papel, já que o sistema pode verificar pelo logon quem que está manuseando o sistema que controla a retirada de chaves.

Assim mesmo adicionando um campo do registro da saída da chave do referido setor, nenhum tempo gasto se adiciona ao processo, porém se economiza, pois como dito anteriormente o nome do membro da COLAB seráfornecido ao registro do banco de dados pelo login do usuário do usuário que acessa o sistema, como também a data e hora do processo de retirada de chave serão a data e hora atual do servidor. Se retira a necessidade de preencher dois campos: hora e data (são preenchidos automaticamente) e o novo campo com nome da pessoa da equipe do setor em questão que os dados de entrega a chave, passando a ser registrado automaticamente após a pessoa logar no sistema primeira vez, não mais registrando manualmente a cada necessidade, tendo apenas o membro da COLAB, que controla a entrega de chaves, selecionar quem foi a pessoa e qual chave ela retirou, com seu respectivo laboratório.

Quanto à devolução de chaves, também não há necessidade de preenchimento manual de cada dado individualmente da hora e data, nome do membro do referido setor que entrega a chave, já que o servidor é capaz de fornecer a hora e data do sistema, juntamente com o nome do servidor logado.

## Requisitos de software

Para especificar o que deve ser desenvolvido na aplicação que resolva o problema do cliente, deve-se ver quais requisitos são necessários para atender a demanda do cliente. Por meio desses requisitos, define-se o que e como será desenvolvido.

Sobre o mapeamento dos requisitos:

A construção de um software, na maioria das vezes, é a união de várias fases de desenvolvimento não apenas técnico, mas de planejamento e estratégia. Saber o que é fundamental, baseado nos problemas que o sistema precisa resolver, pode fazer a diferença entre uma ferramenta de sucesso e outra defasada.

Assim, muitos empresários e fábricas de software acabam estabelecendo regras para que o levantamento e mapeamento dos requisitos de um software sejam voltados para o máximo aproveitamento dos recursos. A intenção desse tipo de metodologia é encontrar, documentar, organizar e rastrear as variáveis do sistema.

O resultado dessa estratégia é a definição de quais requisitos são funcionais e quais requisitos não são. Por requisito entende-se uma condição ou capacidade com a qual o sistema deve estar de acordo. Abaixo explicamos a diferença entre eles. (MAKESYS BLOG, 2013)

Como podemos ver, segundo o autor o principal objetivo de realizar criar regras para esse mapeamento dos requisitos de software é realizado com o objetivo de fazer como que “sejam voltados para o máximo aproveitamento dos recursos”. O melhor aproveitamento do software depende muito de seu planejamento, e esse tipo de mapeamento (de requisitos de software) é parte do planejamento do software, pois deve ser feito antes do início de seu desenvolvimento.

Já sobre a identificação de quais são os requisitos funcionais e não-funcionais, que faz parte dessa estratégia de definição de requisitos são abordados nas duas próximas subseções.

### Requisitos não-funcionais

Sobre os requisitos não funcionais e sua extrema utilidade em entrar no projeto, apesar de poder ser a parte mais técnica e de menos conhecimento por parte do cliente:

As aplicações Web de hoje em dia já possuem regras de negócio bastante complicadas. Codificar essas muitas regras já representam um grande trabalho. Além dessas regras, conhecidas como requisitos funcionais de uma aplicação, existem outros requisitos que precisam ser atingidos através da nossa infraestrutura: persistência em banco de dados, transação, acesso remoto, web services, gerenciamento de threads, gerenciamento de conexões HTTP, cache de objetos, gerenciamento da sessão web, balanceamento de carga, entre outros. São chamados de **requisitos não-funcionais**. (CAELUM, 2014c, **grifo do autor**).

Vários requisitos não-funcionais devem ser definidos no projeto de software. Embora muitos desses requisitos possam estar ocultos ao cliente, como por exemplo o banco de dados, o sistema operacional do servidor, eles são igualmente importantes aos conhecidos pelo cliente. Pois, estando ao acesso do conhecimento dele, ou não, são necessários ao adequado funcionamento do sistema.

Já quanto ao exemplo de alguns requisitos não funcionais, de existência necessária no documento de visão do sistema, permitindo melhor entendimento do que se trata nesse tipo de requisito:

Requisitos não funcionais são relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, disponibilidade, segurança e tecnologias envolvidas. Muitas vezes, os requisitos não funcionais acabam gerando restrições aos funcionais.

Com o crescente número de usuários vindos de dispositivos móveis, requisitos não funcionais como portabilidade e mobilidade tornam-se fundamentais em praticamente todos os sistemas web. Além deles, a preocupação com interoperabilidade, localização geográfica e alta disponibilidade também fazem parte do mapeamento e desenvolvimento de requisitos não funcionais.

Entender profundamente os tipos de requisitos no momento de definir seu software pode decidir o futuro dele. Por outro lado, também é preciso ter em mente que o software pode ser beneficiado se houver um pouco de flexibilidade no seu desenvolvimento. Assim, unindo os conceitos, é possível ter um sistema de qualidade sem um alto custo de aquisição. Esse tipo de mapeamento também facilita e otimiza a manutenção e correção de erros, além de prover uma visão clara do serviço que está sendo implementado. Não apenas do lado do cliente, mas também da empresa que está vendendo o software. (MAKESYS BLOG, 2013)

Os requisitos não-funcionais podem definir o campo em que poderá se trabalhar nos requisitos funcionais, limitando-os. Por isso é importante que se tenha eles em mente juntamente com os funcionais ao planejar o software. Tendo em mente que serão necessários requisitos como interoperabilidade, portabilidade e mobilidade, teremos que designar requisitos funcionais que não sejam de execução pesada para um navegador mobile.

Já sobre a necessidade “beneficiado se houver um pouco de flexibilidade no seu desenvolvimento” (MAKESYS BLOG, 2013), necessidade cujo motivo deve ser a busca por “ter um sistema de qualidade sem um alto custo de aquisição.” (MAKESYS BLOG, 2013) Fazer essa análise aprofundada também dos requisitos não-funcionais, que são parte “da arquitetura técnica, que pertence aos requisitos não funcionais” é importante conforme vimos, pois “[proveem] uma visão clara do serviço que está sendo implementado. Não apenas do lado do cliente, mas também da empresa que está vendendo o software,” (MAKESYS BLOG, 2013) representando uma melhor “manutenção e correção de erros” (MAKESYS BLOG, 2013), o que irá proporcionar uma melhor relação com o elo de ligação entre o cliente e o vendedor: o software.

### Requisitos funcionais

Os requisitos funcionais especificam as funcionalidades que o software proposto possui para resolver os problemas do cliente. Abaixo vemos a definição do que é um requisito funcional:

Um requisito funcional define uma função de um software ou parte dele. Ele é o conjunto de entradas, seu comportamento e sua saída, ou seja, envolve cálculos, lógicas de trabalho, manipulação e processamento de dados, entre outros. Dentro dos requisitos funcionais também encontram-se a arquitetura do aplicativo, diferentemente da arquitetura técnica, que pertence aos requisitos não funcionais.

Muitos autores ainda dividem os requisitos funcionais em três: evidente, escondida e friso. Requisitos evidentes é quando o usuário final do sistema está ciente do que está sendo executado. Escondida é quando uma função está sendo feita, mas é invisível ao usuário. Além das duas, ainda há a do tipo friso, quando a execução da funcionalidade não afeta outras funções do software.

Registrar a venda de produtos em lojas virtuais, calcular o total de uma venda, capturar a informação de um formulário, fazer o login com uma identificação e senha e exibir a descrição e preço de um item são exemplos de requisitos funcionais evidentes. Já dar baixa em um item, manter um log de vendas feitas e prover a integração com outros sistemas são requisitos escondidos. (MAKESYS BLOG, 2013)

Aqui como fala o autor acima, não são tratados nos requisitos funcionais a arquitetura técnica, mas a arquitetura do aplicativo, onde são definidas como o nome desse tipo de requisito sugere, as funções do software.

Há as funcionalidades visíveis aos clientes (requisitos funcionais evidentes), que são as funcionalidades que são permitidas ao usuário comum perceber e saber de sua existência durante a execução do software. Também há as funcionalidades ou requisitos funcionais escondidos, onde o usuário desconhece a execução na visualização da execução do software. A do outro tipo não, requisito funcional friso não foi utilizada.

# FUNDAMENTAÇÃO TEÓRICA

## LINGUAGENS UTILIZADAS

Quanto à escolha das linguagens, justificamos explicando rapidamente as características úteis das linguagens que as fazem ser soluções necessária ao desempenho do trabalho por nosso cliente, e também o histórico das linguagens web utilizadas no desenvolvimento do projeto.

Existem dois tipos de linguagens web quanto ao local de execução:

* Linguagens serve-side
* Linguagens client-side

As linguagens client-side são executadas do lado do cliente, ou navegador de internet, e as linguagens server-side são executadas do lado do servidor.

As linguagens server-side segundo o Portal Devmedia(2014) são:

Server-side [...] é o termo que representa o conjunto de tecnologias em que os processos são interpretados/processados diretamente no servidor, retornando como resultado a codificação client-side. Quando um cliente web(navegador) acessa uma página web, uma solicitação é enviada ao servidor através do protocolo http para que o servidor envie a resposta. O Servidor além de rodar os aplicativos, o lado servidor também é um repositório de páginas estáticas, que serão enviados ao cliente quando solicitado. Supondo que haja uma página JSP, esta será processada pelo servidor e encaminhado uma resposta ao cliente (Navegador).

Portanto, ao utilizarmos uma linguagem de programação web server-side, estamos procurando retornar “a codificação cliente-side” (PORTAL DEVMEDIA, 2014), que é assim chamada pois é interpretável pelo navegador.

Cabe agora definir o que são as linguagens client-side, que de acordo também com o Portal Devmedia (2014):

O cliente-side de uma aplicação é o local onde ela é processada, ou seja, no caso da web, executa no navegador do cliente que é o responsável por interagir com o Servidor HTTP. Entre as tecnologias cliente-side temos o HTML que é executado no navegador, o CSS é outra tecnologia cliente que serve para formatar páginas HTML, há ainda o javascript que permite desenvolver ou ampliar o poder ao lado do cliente.

Essas definições ajudam a identificar em que classificação se encontram as diferentes linguagens utilizadas no sistema. Dentre essas linguagens no parágrafo citadas pelo Portal Devmedia, utilizamos todas (CSS, Javascript, HTML e também para testes utilizamos o servidor virtual HTTP Apache Tomcat.

Vemos que no texto do Portal Devmedia o HTML figura entre as linguagens client-side, pois ele é executado totalmente no navegador.

Mesma característica é pertencente ao Javascript, que não é executado no servidor, porém apenas no navegador. É dada como ferramenta para “ampliar o poder ao lado do cliente."(PORTAL DEVMEDIA, 2014), já que a dinamicidade das informações exibidas não é característica principal do HTML e CSS, que tratam respectivamente da parte estrutural e visual da página web. O CSS, embora dê uma grande dinâmica visual à página, não é capaz de escrever estruturas HTML, como faz o Javascript, que pode manipular inclusive o estilo css, Por isso, é correto dizer, que do lado cliente a linguagem de script Javascript merece destaque como linguagem que dá “poder ao lado do cliente” (PORTAL DEVMEDIA, 2014).

Existe outra classificação também, quanto ao fato de a linguagem ser do nível de máquina ou ao nível de compreensão humana:

* Linguagens de alto nível:
* Linguagens de baixo nível:

As linguagens de alto nível são as linguagens que de estão mais próximas da linguagem humana, enquanto as de baixo nível estão próximas da linguagem de máquina.

A explanação do quem vem a ser linguagens de programação de baixo nível segundo o PORTAL INFOESCOLA (2014):

As linguagens de alto nível são mais fáceis de se trabalhar e de entender, as ações são representadas por palavras de ordem (exemplo faça, imprima, etc.) geralmente em inglês, foram feitos assim para facilitar a memorização e a lógica. Elas não são interpretadas diretamente pelo computador, sendo necessário traduzi-las para linguagem binária utilizando-se de um programa chamado compilador(**grifo do autor).**

Vemos aqui também outra definição dada em um fórum de discussão em resposta à pergunta de um usuário:

As linguagens de alto nível sãs as linguagens que se encontram mais próximas à linguagem natural que à linguagem máquina.

Trata-se de linguagens independentes da arquitetura do computador e dentro deste grupo pode-se citar: JAVA, FORTRAN.... (LINGUAGENS, 2014)

Já o mesmo site novamente faz uma explicação, dessa vez para o que é programação de baixo nível:

Os computadores interpretam tudo como números em base binária, ou seja, só entendem zero e um. As linguagens de **baixo nível** são interpretadas diretamente pelo computador, tendo um resultado rápido, porém é muito difícil e incômodo se trabalhar com elas. Exemplos de linguagens de **baixo nível** são a linguagem binária e a linguagem **Assembly** (PORTAL INFOESCOLA**,** 2014, **grifo do autor)**.

### CSS

O CSS é responsável pela parte visual das páginas web. Como se falou na seção sobre HTML, essa última linguagem não tem o poder necessário para gerar estilos de maneira prática, pois não há a dinamicidade presente no CSS na linguagem HTML e aplicabilidade de um mesmo documento de estilo escrito de uma única vez para diversas páginas.

O CSS foi criado para dar maior poder às aplicações web e dinamicidade com a possibilidade de escrever bem menos as regras de estilo. Vemos a seguir uma definição do CSS.

CSS, é uma tecnologia que nos permite criar páginas web de uma maneira mais exata. Graças às CSS somos muito mais donos dos resultados finais da página, podendo fazer muitas coisas que não se podia fazer utilizando somente HTML, como incluir margens, tipos de letra, fundos, cores...CSS são as siglas de *Cascading Style Sheets*, em português Folhas de Estilo em Cascata. (ALVAREZ, 2004a, *grifo nosso*)

O maior poder proporcionado por essa ferramenta ocorre pelo motivo de a parte visual e a disposição da estrutura da página ser arbitrado pela folha de estilo, e estando essas propriedades de estilo num arquivo à parte, se tem uma melhor organização, não existindo mais a necessidade de procurar no texto do arquivo HTML o que é estilo e o que é estrutura, quando são habilmente usados os recursos de CSS.

O poder da linguagem aumenta a produtividade do programador web, por evitar que escreva várias vazes as configurações de estilo das estruturas definidos por HTML. A produtividade é um bom argumento para se tornar adepto ao uso dessa ferramenta de estilo, a linguagem CSS. (

O CSS3 agora permite coisas antes impossíveis como elementos com cor ou fundo gradiente, sombras e cantos arredondados. Antes só era possível atingir esses resultados com o uso de imagens e às vezes até com um pouco de JavaScript.

A redução do uso de imagens traz grandes vantagens quanto à performance e quantidade de tráfego de dados necessária para a exibição de uma página. (CAELUM, 2015)

#### Atributos de CSS

Para inserir negrito e cor de fontes vemos abaixo da descrição de como executar o processo:

[Esse é um] atributo que se pode utilizar em quase todas as etiquetas HTML: style.   
  
**Exemplo:**  
<p style="color:green;font-weight:bold">O parágrafo sairá com a cor verde e em negrito</p>   
Dentro do atributo style se devem indicar os atributos de estilos CSS separados por ponto e vírgula (;). [...] Os dois primeiros que vimos aqui são:   
**Color**: indica a cor do conteúdo, na etiqueta onde estivermos utilizando, geralmente indica a cor do texto.   
**Font-weight**: indica a grossura do texto. Bold serve para colocar em negrito. (ALVAREZ, 2004a, **grifo do autor**)

Os atributos de cor e estilo da fonte, que no exemplo do autor foi definido em negrito, pode ser da maneira que ele exemplifica, declarado inline, diretamente na página HTML, embora essa não seja a prática mais utilizada.

Esses dois atributos são separados por ";" dentro da tag de abertura do elemento no qual queiramos aplicar o estilo desejado

Aqui já se demonstra como colocar cor nos links:

Com HTML definimos a cor dos links no <body>, com os atributos link, vlink e alink. Isto nos permite mudar a cor dos links para todo o documento, mas e se quisermos mudar a cor de um link em concreto, para que tenha outra cor que a definida na etiqueta <body>?  
Para fazer isto utilizaremos o atributo style dentro do link:  
<a href="meulink.html" style="color:red">  
Assim sairá o link na cor vermelha, independentemente da cor definida para todo o documento. (ALVAREZ, 2004a)

Os links passam a poder ser manipulados na tag de abertura de um link (<a>), o que mostra uma das formas pelas quais é possível manipular a cor do elemento, atribuindo a cor no atributo "color" atribuído à tag exemplo "<a></a>" e definindo a cor pelo nome da cor em inglês entre "". Essa forma já é uma alternativa a outro método de inserção da cor, que pode ser a representação um número específico da cor.

Para definir espaçamento entre linhas:

Com CSS podemos definir o espaço entre cada linha do documento, utilizando o atributo line-height. Por exemplo, podemos definir que para todo um parágrafo o espaço entre cada uma de suas linhas seja 25 pixels: 

<p style="line-height: 25px;">

Um parágrafo normal no qual cada uma das linhas está separada 25 pixels da outra. Há de colocar suficiente texto como para que se vejam 2 linhas, assim sairão separadas   
</p> (ALVAREZ, 2015a)

Aqui vemos outra forma definir atributos de estilo CSS, que permite formatar o texto, definindo pelo atributo da tag de parágrafo, "style", que será aplicado um estilo para esse parágrafo que dará distância de 25 pixels entre as linhas dele. Vemos que é mais um exemplo de como pode ser facilitado manipular o estilo dos elementos HTML, ao declararmos atributos CSS na definição de estilo da etiqueta, que no exemplo é a "<p></p>".

Para se definir espaçamento entre caracteres:

Pode-se definir também o espaço entre cada caractere. Isto se faz com o atributo de CSS letter-spacing. Vejamos um exemplo:   
<p style="letter-spacing:12cm">(*sic*)   
Este parágrafo tem as letras espaçadas por 1 centímetro.   
</p>

Este atributo, assim como ocorre com muitos outros de CSS, não está suportado para todos os navegadores. Netscape por exemplo, em sua versão 4 ainda não o inclui. (ALVAREZ, 2015a)

Vemos que nesse código, embora se tenha um erro diretamente retirado do texto do autor, que se define um centímetro de espaço entre as letras por meio do atributo CSS "letters-spacing" que poderá ser usado também para outras tags HTML, onde haja inserção de texto. Algo interessante é que mesmo à tag "<body></body>" se pode definir também esses estilos CSS.

Para definir que os links não devem ficar em sublinhado:

Um dos efeitos mais significativos e fáceis de realizar com CSS é eliminar o sublinhado dos links de uma página web. Existe um atributo que serve para definir a decoração de um texto, se está sublinhado, riscado, ou se não tem nenhuma destas "decorações". É o atributo text-decoration, neste caso indicaremos em um link que não queremos decoração:

<a href="minhapagina.html" style="text-decoration:none">   
(ALVAREZ, 2015a)

Vemos que nesse código, embora se tenha um erro diretamente retirado do texto do autor, que se define um centímetro de espaço entre as letras por meio do atributo CSS "letters-spacing" que poderá ser usado também para outras tags HTML, onde haja inserção de texto. Algo interessante é que mesmo à tag "<body></body>" se pode definir também esses estilos CSS.

Sobre definir estilos para todo um site web:

Uma das características mais potentes da programação com folhas de estilo consiste em definir os estilos de todo um website. Isto se consegue criando um arquivo onde simplesmente colocamos as declarações de estilos da página e linkamos todas as páginas do site com esse arquivo. Deste modo, todas as páginas compartilham uma mesma declaração de estilos e, portanto, se a mudamos, todas as páginas serão mudadas. (ALVAREZ, 2015a)

Esse era o modo de acelerar o processo de desenvolvimento de sites, pelo qual se dizia anteriormente deixar mais rápida a atividade de atribuir estilos aos elementos HTML.

Vejamos agora todo o processo para incluir estilos com um arquivo externo no no HTML:

**1- Criamos o arquivo com a declaração de estilos**  
É um arquivo de texto normal, que pode ter qualquer extensão, apesar de que lhe podemos atribuir a extensão .css para lembrarmos de que tipo de arquivo é. O texto que devemos incluir deve ser escrito exclusivamente em sintaxe CSS, é um pouco diferente da sintaxe que utilizamos dentro do atributo style. Estaria errado incluir código HTML neste arquivo: etiquetas e outras. Podemos ver um exemplo a seguir:   
  
P {   
font-size : 12pt;   
font-family : arial,helvetica;   
font-weight : normal;   
}   
H1 {   
font-size : 36pt;   
font-family : verdana,arial;   
text-decoration : underline;   
text-align : center;   
background-color : Teal;   
}   
BODY {   
background-color : #006600;   
font-family : arial;   
color : White;   
} (ALVAREZ, 2015a, **grifo do autor**)

O que vemos nesse texto acima é que o CSS está definindo que manteremos intactas as tag HTML, não sendo necessário necessário definir um atributo HTML style, nenhum atributo class para importar o estilo, somente ao atribuí-lo no head, já haverá o efeito de estilo pretendido para o parte da estrutura HTML a qual se deseje alterar.

Aqui se fala algo importante que o arquivo com estilo por convenção possui a extensão .CSS, mas outras extensões podem ser utilizadas, embora isso vá atrapalhar a identificação do arquivo de estilo.

Há alguma diferença, nessa atribuição da que é feita inline, já que ela está entre "{" e "}". Fora da chave de abertura, temos identificador que contém o nome da classe CSS, se ela for uma classe que deve ser atribuída a uma tag, deve simplesmente conter o nome da tag. Se quiser criar uma nova classe, deve-se utilizar "."+ nome da classe desejada, sendo atribuído ao elemento dentro de uma tag de abertura conforme este exemplo: "<p class='qualqueruma'>Texto</p>".

Além das formas inline de se atribuir estilo CSS, essas duas, atreladas à etiqueta de abertura da tag CSS, ou definição do atributo "class" da estiqueta de abertura da tag escolhida com o recebimento de um nome de uma classe, que pode ser declarada em arquivo externo, ou inline, dentro da tag "<style></style> normalmente aninhada dentro da tag "<head></head>", o que não é obrigatório, mas é a convenção mais utilizada.

Abaixo vemos o exemplo com método adotado pelo autor, o qual identificamos nos parágrafos anteriores perto deste que utiliza uma classe CSS que tem como nome atribuído o próprio nome da tag, o que como mostra abaixo é o método mais simples, onde todas as tags conforme definidas abaixo, receberão o estilo pretendido pelo desenvolvedor da página.

**2- Linkamos a página web com a folha de estilos**  
  
Para isso vamos colocar a etiqueta <LINK> com os atributos

* rel="STYLESHEET" indicando que o link é uma folha de estilo.

type="text/css" porque o arquivo é de texto, em sintaxe CSS.

* href="estilos.css" indica o nome do arquivo fonte dos estilos.

Vejamos uma página web inteira que linka com a declaração de estilos anterior:   
  
<html>   
<head>   
<link rel="STYLESHEET" type="text/css" href="estilos.css">   
<title>Página que lê estilos</title>   
</head>   
<body>   
<h1>Página que lê estilos</h1>   
<p>   
Esta página tem no cabeçalho a etiqueta necessária para linkar com a folha de estilos. É muito fácil.   
</p>   
</body>   
</html> (ALVAREZ, 2015a, **grifo do autor**)

Vemos que se utiliza um atributo HTML da etiqueta de abertura da tag link como "stylesheet" para se indicar que vai se importar uma folha de estilo, que é indicada pelo por tipo CSS ("type='text/css'"). O atributo href, dentro da etiqueta de abertura ainda da mesma tag indica o endereço absoluto ou relativo do qual se importa a folha de estilo com as classes CSS. Se indica utilizar o endereço relativo para indicar o local da folha de estilo, pois se o projeto for depois removido para um outro repositório no computador, ou mesmo na web, não haverá problema na importação e apresentação do estilo pretendido ocorrerá problemas.

O endereço absoluto tem que conter o endereço completo do arquivo, porém se for salvo um endereço no atributo href com algo igual a "C:/varios\_projetos/projeto/css/index.css", ao transferir a pasta "css" para outro repositório no computador local ou na nuvem se perderá o estilo da página relacionado ao arquivo de folha de estilo importado pelo HTML.

Já o endereço relativo utiliza "." para substituir por uma declaração que explicite que a mesma pasta ("projeto", pois estamos considerando o mesmo exemplo do parágrafo anterior e importando o mesmo CSS) usada para hospedar o HTML poderá conter uma subpasta chamada "css" utilizada para salvar o arquivo de folha de estilo, sendo declarado como: "<link rel="STYLESHEET" type="text/css" href="./css/index.css">, o que permitiria ao eu remover para "F:/" a pasta "varios\_projetos", o projeto tivesse todas as suas propriedades CSS de "index.css" mantidas.

Posso também utilizar outro modo de indicar endereço relativo. Por inserção de ".." na url indicada de endereço local ou remoto. Esse método de indicação do endereço é usado para indicar que o arquivo está em uma pasta um nível acima da que o arquivo HTML, está. Considerando o mesmo exemplo de arquivo HTML dos dois últimos parágrafos do texto, que está no endereço absoluto "C:/varios\_projetos/projeto/", ao qual chamamos "index.html", suponhamos que além do arquivo "index.css" na pasta um nível abaixo ("css"), temos um arquivo um nível acima chamado index2.css, que daria na pasta "varios\_projetos", se declararia assim: "<link rel="STYLESHEET" type="text/css" href="../index2.css">.

#### História do CSS

Essa linguagem CSS veio melhorar o já existente HTML, que tinha sido criado para divulgação científica por Tim, mas tinha muitos problemas de estilo percebidos ao ser mais utilizado o HTML, como fato de ser cansativo aplicar estilo a HTML, a seguir se explica como surgiu essa demanda:

Vejamos o começo de tudo, a linguagem de marcação surgindo para solucionar o problema de um cientista que queria uma forma de divulgar seus artigos na rede, para isso, ele criou algo simples, restrito a comunidade científica e usado apenas para estrutura o conteúdo e exibir um texto de forma coerente.

Mas a linguagem usada pelos cientistas se tornou padrão da Internet, e com o tempo, passou a ser mais testada. Todos queriam imagens, cores e designer avançado. E o HTML antigo não suportava isso, então foram adicionadas novas tags. Isso solucionou o problema por um tempo, mas começou a surgir outro grande problema. Com os novos recursos, designers e desenvolvedores de websites, abusavam da criatividade e enchiam suas paginas (*sic*) de fontes e estilos visuais, mas para alterar uma linha até que era mais simples, porem (*sic*) quando se queria mudar a cor de um link, por exemplo, e esse link estava em 300 paginas (*sic*) diferentes, era trabalho manual mesmo, um por um, tag por tag.

Misturar estilo e estrutura não era mais interessante, e foi assim que em 1995, Håkon Wium Lie e Bert Bos apresentaram a proposta do CSS (Cascading Style Sheets) que logo foi apoiada pela W3C. A idéia (*sic*) geral era, utilizar HTML somente para estruturar o website e a tarefa de apresentação fica com o CSS disposto em um arquivo separado .css ou no proprio (*sic*) HTML demarcado pelas tags. (PEREIRA, 2015)

O problema inicial que o criador Tim teve foi um passo inicial para o CSS, como o autor corretamente falou, mas com a popularização, várias páginas foram sendo acumuladas e as exigências por uma exibição gráfica mais bonita foi pedida. O HTML permitiu que fosse configurado estilo por meio dele, porém aos usuários da linguagem perceberem a dificuldade em alterar o visual para todas as páginas, verificaram a existência de uma tarefa exaustiva, por isso a proposta de Håkon Wium Lie e Bert Bos foi logo acatada, pois dinamizou muito alterar o visual de centena de páginas de uma vez, quando se permitiu que de uma vez só se configurasse o estilo das várias páginas por um único arquivo .CSS que as páginas da web passaram a aplicar, com o aceite do CSS pelo W3C.

O que em extremo facilita é que quando se aplica um padrão CSS a uma página da web, pode se fazer que ao aplicar uma tag no HTML, automaticamente se receba o visual pretendido.

### *JSP*

JSP ou Java Server Pages é uma linguagem de programação WEB que endossa códigos da linguagem de programação Java, da linguagem de scripts Javascript, e da linguagem de marcação HTML. Uma de suas principais características é facilitar o desenvolvimento com linguagem Java para Web, pois não é necessário com ela, escrever manualmente os arquivos servlet, os códigos Java passam a ser escritos dentro de um arquivo .JSP que contém códigos Java embutidos na estrutura HTML dentre as balizas “<%” e “%>”.

“Java Server Pages [..], integrado ao HTML, permite dar mais dinamismo às páginas web” (PORTAL DEVMEDIA, 2014, grifo do autor).

Quanto ao fato de a linguagem ser “integrado às páginas web” (PORTAL DEVMEDIA, 2014), e a falta de necessidade de ser compilada fala também o Portal Devmedia (2014):

O JSP necessita de servidor para funcionar por ser uma linguagem Server-side script, o usuário não consegue ver a codificação JSP, pois esta é convertida diretamente pelo servidor, sendo apresentado ao usuário apenas codificação HTML.

Uma página JSP possui extensão .jsp e consiste em uma página com codificação HTML e com codificação Java, inserida entre as tag´s, denominada scriptlets e funcionando da seguinte forma: o servidor recebe uma requisição para uma página JSP, interpreta esta página gerando a codificação HTML e retorna ao cliente o resultado de sua solicitação. A página JSP que foi interpretada pelo servidor não precisa ser compilada como aconteceria com um servlet java por exemplo, esta tarefa é realizada em tempo real pelo servidor. É necessário apenas desenvolver as páginas JSP e disponibilizá-las no Servlet Container (Tomcat, por exemplo).

Essa linguagem de programação contém três diretivas utilizadas para informações especiais dentro de páginas (PORTAL DEVMEDIA, 2014):

Na codificação JSP, têm-se algumas diretivas, as quais são utilizadas para informações especiais dentro de páginas, sendo dividido em três tipos:

@include: utilizado para inserir os códigos de arquivos à página corrente;

@page: responsável por trazer informações sobre a página JSP;

@taglib: responsável por habilitar uma biblioteca de tags personalizada (item que será abordado em outro artigo com mais detalhes) (PORTAL DEVMEDIA, 2014).

O JSP é considerada uma linguagem de programação por conter laços de repetição.

#### Benefícios do JSP

“O objetivo da linguagem JSP não é só o desenvolvimento de páginas dinâmicas para Internet. Com ela é possível desenvolver sistemas inteiros para Internet. Além disso, existem diversos benefícios em se utilizar a linguagem JSP” (PORTAL DEVMEDIA, 2014).

A linguagem JSP possui várias características que podem defini-la como uma boa escolha para desenvolver um sistema Web:

* Custo
* JSP é embutido no HTML
* Aperfeiçoamento de recursos utilizando Java em Servlets;
* Manipulação de arquivos como texto, PDF, DOC etc.;
* Criptografia de dados;
* Utilização de cookies e sessões;
* Manipulação de arquivos XML;
* Suporte a diversos bancos de dados como: MySQL, SQL Server, Oracle, Informix, etc.;
* Suporte a sistemas de relatórios como o JasperReport entre outros (PORTAL DEVMEDIA, 2014).

Sobre o custo:

O JSP não tem custo de licença. Isto significa que pode ser utilizado em qualquer máquina, para qualquer número de usuários sem violar nenhum direito autoral. Claro que isto depende do servidor escolhido, o Tomcat é um servidor livre, licenciado sob a licença da Apache Foundation, de alta qualidade e sem custo de licenciamento (PORTAL DEVMEDIA, 2014).

Já sobre o JSP ser embutido no HTML, “é simples de se utilizar, podendo gerar o HTML ou ainda estar embutido dentro do HTML” (PORTAL DEVMEDIA, 2014). Dá para se ver o exemplo de código que demonstra a capacidade de conter código Java embutido dentro do HTML, característico da linguagem de programação:

| Figura 1: Printscreen de exemplo de código JSP embutido no HTML |
| --- |
|  |
| Fonte: *PORTAL DEVMEDIA, 2014* |

### Java

Java é uma linguagem de programação que dá suporte a interfaces web, desktop

#### 2.1.3.1 O surgimento do Java

A linguagem Java em meados da década de 90 tinha alguns problemas a época de seu surgimento a resolver:

* Ponteiros
* Gerenciamento de memória
* Organização
* Falta de bibliotecas
* Ter de reescrever parte do código ao mudar de sistema operacional
* Custo financeiro de usar a tecnologia[...] (CAELUM, 2014a).

O Java queria atingir o mercado pensando em oferecer ser uma linguagem de programação client-side, que porém fez sucesso ao rodar aplicações server-side¸conforme podemos verificar no relato dado pelo site CAELUM (2014a):

A linguagem Java resolve bem esses problemas, que até então apareciam com frequência nas outras linguagens. Alguns desses problemas foram particularmente atacados porque uma das grandes motivações para a criação da plataforma Java era de que essa linguagem fosse usada em pequenos dispositivos, como tvs, videocassetes, aspiradores, liquidificadores e outros. Apesar disso a linguagem teve seu lançamento focado no uso em clientes web (browsers) para rodar pequenas aplicações (applets). Hoje em dia esse não é o grande mercado do Java: apesar de ter sido idealizado com um propósito e lançado com outro, o Java ganhou destaque no lado do servidor.

Houve um movimento de mudança de foco da Sun Microsystens da produção da linguagem cliente-side para server-side em 1992 para passar a rodar aplicações no navegador de internet(CAELUM, 2014):

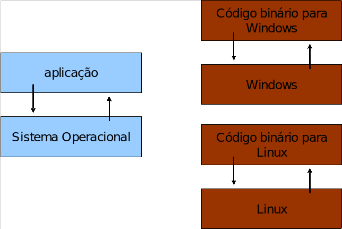
Com o advento da web, a Sun percebeu que poderia utilizar a ideia criada em 1992 para rodar pequenas aplicações dentro do browser. A semelhança era que na internet havia uma grande quantidade de sistemas operacionais e browsers, e com isso seria grande vantagem poder programar numa única linguagem, independentemente da plataforma. Foi aí que o Java 1.0 foi lançado: focado em transformar o browser de apenas um cliente magro (*thin client* ou terminal burro) em uma aplicação que possa também realizar operações avançadas, e não apenas renderizar html (CAELUM, 2014b, *grifo nosso*).

O fato de a linguagem possuir “ações [...] representadas por palavras de ordem (exemplo faça, imprima, etc.) geralmente em inglês” apontados pelo Portal Infoescola (2014) como o print (imprimir), utilizado no famoso System.out.print (“olá mundo”); fazem dela uma linguagem de baixo nível.

#### 2.1.3.3 A existência da máquina virtual

Uma das mudanças mais positivas que a linguagem Java trouxe ao mercado foi que um código não necessitava mais ser produzido para um sistema operacional específico, como ocorria em outras linguagens como C, em que para cada sistema operacional, tinha que ser recompilado o código (CAELUM, 2014a; CAELUM, 2014c).

Figura 2: Código nas linguagens que surgiram no mercado antes de Java são compilados para sistema específico



Fonte: *CAELUM, 2014c*

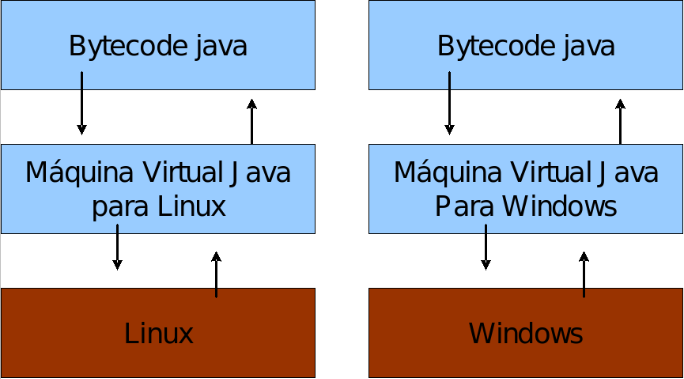
O que ocorre com essas linguagens antecedentes do Java é o seguinte:

Temos um código executável para cada sistema operacional. É necessário compilar uma vez para Windows, outra para o Linux, e assim por diante, caso a gente queira que esse nosso software possa ser utilizado em várias plataformas. Esse é o caso de aplicativos como o OpenOffice, Firefox e outros.

 A [...]aplicação se utiliza das bibliotecas do sistema operacional, como, por exemplo, a de interface gráfica para desenhar as "telas". A biblioteca de interface gráfica do Windows é bem diferente das do Linux: como criar então uma aplicação que rode de forma parecida nos dois sistemas operacionais? (CAELUM, 2014c)

A linguagem Java responde a esse tipo de necessidade com a criação da máquina virtual que vem a ser (CAELUM, 2014) “uma camada extra responsável por ‘traduzir’ - mas não apenas isso - o que sua aplicação deseja fazer para as respectivas chamadas do sistema operacional onde ela está rodando no momento:” (CAELUM, 2014)

Figura 3: Um único código compilado serve para todos os sistemas operacionais, pois há a máquina virtual como intermediador da execução



Fonte: *CAELUM, 2014*

Pontos importantes a serem observados são os expostos na apostila da CAELUM (2014):

Dessa forma, a maneira com a qual você abre uma janela no Linux ou no Windows é a mesma: você ganha independência de sistema operacional. Ou, melhor ainda, independência de plataforma em geral: não é preciso se preocupar em qual sistema operacional sua aplicação está rodando, nem em que tipo de máquina, configurações, etc.

Uma máquina virtual é como um "computador de mentira": tem tudo que um computador tem. Em outras palavras, ela é responsável por gerenciar memória, threads, a pilha de execução, etc.

Sua aplicação roda sem nenhum envolvimento com o sistema operacional! Sempre conversando apenas com a Java Virtual Machine (JVM).

### SQL

Sobre o que é a linguagem SQL, utilizada para o desenvolvedor persistir dados em sua aplicação:

SQL (Structured Query Language) é a linguagem padrão universal para manipular bancos de dados relacionais através dos SGBDs. Isso significa que todos os SGBDRs (Sistema de Gerenciamento de Banco de Dados Relacionais) oferecem uma interface para acessar o banco de dados utilizando a linguagem SQL, embora com algumas variações. Logo, saber o que é SQL e como utilizá-la é fundamental para qualquer desenvolvedor de softwares.

A ‘Linguagem Estruturada de Consultas’ (SQL, traduzida para o português) é utilizada para interagir com o SGBD e executar várias tarefas como inserir e alterar registros, criar objetos no banco de dados, gerenciar usuário, consultar informações, controlar transações, etc. Todas as operações realizadas no banco de dados podem ser solicitadas ao SGBD utilizando esta linguagem. (FURTADO, 2015)

Com essa linguagem, que controla o mecanismo de persistência, podemos fazer diversas operações desde consultas, exclusões de dados, alterações e novas entradas, que são todas realizadas por meio dessa linguagem, que em geral, tem sintaxe diferente quanto ao uso de símbolos simples como " ou ' estão diferentes para indicar strings de consulta, o que no final de conta exigirá uma simples adequação às regras do SGBD, que não serão executadas, de maneira alguma, se o indivíduo que esteja utilizando o banco não souber o básico de uma consulta SQL padrão.

Utilizamos o SGBD MySQL que possui algumas variações da linguagem SQL. Dependemos dessa linguagem para se comunicar por diversos comandos que são enviados diretamente pelo workbench, que no nosso caso é o HeidiSQL,

Quanto às subdivisões da linguagem SQL, veremos nessas próximas subseções quais elas são e as atividades que são desempenhadas por cada uma delas.

#### Operações SQL

Existe uma subdivisão dos grupos de comandos presentes no SQL em quatro tipos de operações, que são abaixo apresentadas:

A linguagem SQL é dividida em 4 agrupamentos de acordo com o tipo de operação a ser executada no banco de dados. A saber, DML (Data Manipulation Language, ou Linguagem de Manipulação de Dados e português), DDL (Data Definition Language, ou Linguagem de Definição de Dados em português), DCL (Data Control Language, ou Linguagem de Controle de Dados em português) e DTL (Data Transaction Language, ou Linguagem de Transação de Dados em português). Alguns autores classificam também uma divisão da linguagem para consultas, a DQL (Data Query Language, Linguagem de Consulta de Dados), que tem apenas um comando (SELECT), porém é mais comum encontrar este comando como integrante da DML, juntamente com os comandos INSERT, UPDATE e DELETE. (FURTADO, 2015)

Os comandos da linguagem de comunicação com o banco de dados, que tem diferentes funções: manipulação, definição, controle e consulta de dados. Abaixo estaremos apresentando as subseções que tratam de comando presentes na subdivisão da linguagem referente a cada função específica da linguagem SQL.

##### DML – DATA MANIPULATION LANGUAGE

Esses DML (Linguagem de Manipulação de Dados) é o subconjunto mais utilizado da linguagem SQL, pois é através da DML que operamos sobre os dados dos bancos de dados com instruções de inserção, atualização, exclusão e consulta de informações. Os comandos SQL desse subconjunto são:

* **INSERT**: utilizado para inserir registros (tuplas), em uma tabela.
  + Exemplo: INSERT into CLIENTE(ID, NOME) values(1,’José’);
* **UPDATE**: utilizado para alterar valores de uma ou mais linhas (tuplas) de uma tabela.
  + Exemplo: UPDATE CLIENTE set NOME = ‘João’  WHERE ID = 1;
* **DELETE**: utilizado para excluir um ou mais registros (tupla) de uma tabela.
  + Exemplo: DELETE FROM CLIENTE WHERE ID = 1;
* **SELECT**: O principal comando da SQL, o comando select é utilizado para efetuar consultas no banco de dados.
  + Exemplo: SELECT ID, NOME FROM CLIENTE; (FURTADO, 2015, **grifo do autor**)

Os comandos acima especificados consultam, salvam, atualizam e apagam informações. Cláusulas SQL, como SELECT podem ser utilizados conjuntamente com outras cláusulas como WHERE:

* Exemplo:
  + SELECT U.NOME AS 'MEMBRO DA COLAB',P.NOME AS 'PROFESSOR', L.NOME AS 'LABORATÓRIO', H.DATA AS 'DATA', H.HORARIO AS 'HORÁRIO'

FROM HISTORICO AS H, PROFESSORES AS P, USUARIO AS U, LABORATORIOS AS L

WHERE P.ID\_PROFESSOR=H.ID\_PROFESSOR AND H.ID\_USUARIO=U.ID\_USUARIO AND L.ID\_LABORATORIO=H.ID\_LABORATORIO;

Esse exemplo é de uma consulta utilizada no sistema desenvolvido por nós, em sua primeira versão. Porém, falando da parte teórica envolvida aí, podemos verificar que para consultas que utilizem várias tabelas, podemos usar o recurso do alias, onde no exemplo acima ao invés de se selecionar "USUARIO.NOME" selecionamos "U.NOME", devido termos utilizado a declaração de alias "USUARIO AS U". Como a linguagem é inglês, pode passando para português entender-se melhor: "USUARIO COMO U".

##### DDL – DATA DEFINITION LANGUAGE

Podemos ver abaixo essa subdivisão das operações SQL, que permitem que alteremos, excluamos e criemos objetos do Banco de dados:

**DDL** (**Linguagem** de Definição de Dados) é o subconjunto da **SQL** utilizado para gerenciar a estrutura do banco de dados. Com a DDL podemos criar, alterar e remover objetos (tabelas, visões, funções, etc.) no banco de dados. Os comandos deste subconjunto são:

**CREATE**: **utilizado** para criar objetos no banco de dados.

Exemplo (criar uma tabela): **CREATE** TABLE CLIENTE ( ID INT PRIMARY KEY, NOME VARCHAR(50));

**ALTER**: utilizado para **alterar** a estrutura de um objeto.

Exemplo (adicionar uma coluna em **uma** tabela existente):ALTER TABLE CLIENTE ADD SEXO CHAR(1);

**DROP**: utilizado para remover um objeto do banco de dados.

Exemplo (remover uma **tabela**): DROP TABLE CLIENTE; (FURTADO, 2015, **grifo do autor**)

Não alteramos mais com esses comandos acima explicados as tabelas do banco de dados e seu conteúdo, - linhas (ou tuplas) - mas alteramos os objetos hierarquicamente abaixo do bando de dados: a própria tabela, permitindo excluir colunas das tabelas, inserir colunas nelas, também.

Também a própria tabela é criada e excluída pelas cláusulas desse grupo de operações SQL (DDL), sendo também possível remover funções do banco de dados, conforme explicado pelo autor citados acima.

##### DCL – DATA CONTROL LANGUAGE

Esses comandos da linguagem têm acesso tem acesso a definir regras de acesso ao banco de dados por parte dos usuários registrados no banco de dados, dizendo quem poderá executar determinadas funções do banco de dados, de acordo com o que vemos sendo explicado abaixo:

**DCL** (Linguagem de Controle de Dados) é o subconjunto da **SQL** utilizado para controlar o acesso aos dados, basicamente com dois comandos que permite ou bloqueia o acesso de usuários a dados. Vejamos estes comandos:

**GRANT**: Autoriza um usuário a executar alguma operação.

Exemplo (dar permissão de consulta na tabela cliente para o usuário carlos): GRANT select ON cliente TO carlos;

**REVOKE**: Restringe ou remove a permissão de um usuário executar alguma operação.

Exemplo (não permitir que o usuário carlos crie tabelas no banco de dados): REVOKE CREATE TABLE FROM carlos; (FURTADO, 2015, **grifo do autor**)

Esse tipo de comando, já é utilizado para fazer que o usuário não tenha acesso a alguma operação ou permiti-lo a realizar alguma ação. Portanto, chegamos à parte que controla a segurança do banco, utilizando uma cláusula da linguagem SQL.

Embora tenhamos conhecimento, não criamos esse tipo de controle de acesso do usuário do banco, pois usamos um apenas.

##### DTL – DATA TRANSACTION LANGUAGE

Esse subgrupo de operações permite o controle das transações, permitindo recuperação de dados salvos no banco, segundo:

**DTL** (Linguagem de controle de transações) é o subconjunto da **SQL** que fornece mecanismos para controlar transações no banco de dados. São 3 comandos: iniciar uma transação (BEGIN TRANSACTION), efetivar as alterações no banco de dados (COMMIT) e cancelar as alterações (ROLLBACK). (FURTADO, 2015, **grifo do autor**)

Essas cláusulas permitem recupera dados de transações feitas no banco, as quais são feitas pelos outros tipos de operações SQL. Isso é interessante, devido assim ser atendido a necessidade, quando utilizado esse recurso, de retornar ao estado anterior com os dados salvos anteriormente no banco, propiciando mais confiança na execução de atualizações no banco de dados, evitando que se perca tudo, podendo-se recuperar o que se fez antes da transação.

## SOFTWARES DESENVOLVIMENTO UTILIZADOS

Esses softwares aqui listados foram utilizados para o desenvolvimento de nosso sistema, o motivo pelo qual foram escolhidos para a atividade de programação por nós executada é explicado aqui.

Falamos sobre o MySQL, seu histórico, sua utilidade para o processo de desenvolvimento de nosso software.

Já sobre o Netbeans, tratamos do uso e imortan

### MySQL

MySQL foi o SGBD utilizado como mecanismo de persistência de nossa aplicação, salvamento de dados entrados no sistema.

Ele não foi inicialmente a escolha para o sistema, mas o postgres, devido à mudança de orientador do projeto, e por encontrar no próprio instituto, no Campus João Câmara pessoas que utilizavam o banco de dados, sendo ele opção da maioria.

No meio acadêmico, ele é muito utilizado, devido a ser considerado o mais didático, pela existência de uma versão gratuita para utilização, além do amplo conteúdo presente na internet de ajuda para uso de suas versões de cláusulas SQL, que como ocorre nos demais SGBDs que estão presentes como liderança no mercado da informática, tem suas especificidades no uso de várias funções da linguagem SQL, se dividindo do puro padrão SQL, mas o aprendizado que exige para quem tem conhecimento do padrão SQL é fácil e simples, devido não possuir várias personalizações com diferença grande de procedimento em relação à outros SGBDs.

O banco de dados MySQL é um sistema de gerenciamento (SGBD) que usa a Linguagem de Consulta Estruturada (SQL – Structured Query Language) como interface. A flexibilidade, versatilidade e a facilidade encontrada em seu uso fizeram do MySQL um dos bancos de dados preferidos. Usado por algo em torno de dez milhões de instalações no mundo todo. O MySQL é usado pela NASA; pela Friendster; pelo Bradesco; pela Nokia; pela HP; pela Sony; pela Lufthansa e por diversas outras corporações das mais variadas pátrias e pessoas físicas de várias cidadanias.

Sobre a relevância do uso desse SBD PRDA

Facilmente integrado com linguagens de programação [...], com presença quase unânime nos planos de hospedagem de sites para Internet, o banco de dados MySQL suporta Unicode, Full Text Indexes, replicação, Hot Backup, GIS, OLAP e muitos outros recursos como a portabilidade (é aceito por quase todas as plataformas usadas atualmente); compatível com drives ODBC, JDBC e .NET e tem módulos de interface para as linguagens de programação mais comumente utilizadas como a Delphi, a Java, a C/C++, o Phyton, a Perl, o PHP, o ASP e a Ruby; isso assegura uma fantástica performance e garante grande estabilidade na operação dos servidores; necessitando de poucos recursos de hardware. Simples de usar é gratuito e tem licenciamento do tipo GPL; permitindo o uso de vários Storage Engines como MyISAM, InnoDB, Falcon, BDB, Archive Federated, CSV, Solif e etc; suportando controle transacional, triggers, Cursors (Non-Scrollable e Non-Updatable), Stored Procedures e Functions, possui replicação configurada de forma descomplicada e extremamente facilitada e interfaces gráficas. (ANCONRADOR, 2015)

Como podemos ver, ele É utilizado para armazenar informações por parte de várias empresas importantes economicamente, com dados que notoriamente necessitam grande segurança, o que ressalta o valor desse SGBD, e sua importância no mercado.

Podemos verificar também o quanto ele é aceito por desenvolvedores de várias linguagens de programação, conectando-se com os mecanismos de persistência de várias linguagens com C#, Java, C++, Delphi.

Como nossa linguagem de programação escolhida foi Java - é a linguagem de programação embutida nas páginas JSP, e o que é responsável pela geração de conteúdo dinâmico nas páginas.

Seu uso por pessoas usuárias de outras linguagens de programação, que não a nossa – já que utilizamos JSP – fazem com que o uso desse software possa ser mais seguro, por ser mais fácil trocar informações sobre com alguém que tenha conhecimento sobre isso e pesquisar assuntos discutidos em tópicos de discussão web, já que é a preferência de vários usuários de diversas linguagens de programação.

É também utilizado para armazenar informações por parte de várias empresas importantes e com dados que necessitam grande segurança, o que ressalta a importância do uso desse SGBD.

Possui grande performance, extrema robustez e realiza o trabalho de forma perfeita em qualquer ambientação (Windows, linux, solaris, Sun OS, SGI, etc), essas características garantiram ao MySQL a fama de ser uma grande sacada para aqueles que precisam gerenciar bancos de dados de qualquer tamanho e de qualquer volume de requisições. Desde as aplicações simples e destinadas apenas a um único usuário, até as mais complexas e gigantescas rotinas destinadas às aplicações das grandes corporações multinacionais que fazem uso dele sem sobressaltos.

#### -História do MySQL

Sobre o surgimento do MySQL e contextualização no tempo de seu surgimento no mercado dos bancos de dados:

O MySQL nasceu na Suécia e foi criado por David Axmark, Allan Larsson e o finlandês Michael “Monty” Widenius. Eles trabalham em conjunto desde os anos de 1980. Além disso, em torno do desenvolvimento do MySQL, trabalham um número superior a quatrocentas pessoas que atuam de localidades diversas ao redor do mundo. Esse número não considera os mais de mil responsáveis pelos testes do software e análise da sua integração com as várias versões das aplicações e produtos ou que criam conteúdo sobre ele. (ANCORADOR, 2015)

O MySQL não tem surgimento relacionado às empresas tradicionais de informática do vale do silício, mas tem hoje uma grande importância no mercado e confiabilidade, devido ao seu projeto de de desenvolvimento com forte participação de vários desenvolvedores.

Sobre o surgimento da versão 5 do MySQL: “A versão MySQl 5 surgiu primeiramente em (março de 2005) e, apenas em outubro daquele ano, passou a ser comercializado de fato com os avanços significativos no diagnóstico e no rastreio da performance do sistema.” A versão 5.6 do MySQL foi utilizada por nós para realizar os testes do projeto de desenvolvimento, embora durante esse período foi lançada uma atualização para a versão 5.7 para o servidor de banco de daods

### Netbeans

O Netbeans é uma das IDEs mais utilizadas no meio acadêmico e profissional. Ele tem como principal concorrente no mercado o Eclipse. Aqui se apresenta como foi desenvolvido esse software tão importante para desenvolvedores.

NetBeans IDE é uma aplicação de código aberto feita para auxiliar os desenvolvedores na criação de aplicativos em diferentes plataformas, utilizando tecnologia Java para isto. Nela você encontra todas as ferramentas necessárias para projetos profissionais em Java, C/C++ e Ruby on Rails.

[...]

NetBeans IDE oferece o recurso GUI Builder que é uma ferramenta utilizada para criação de design e posicionamento dos componentes da interface, já que apresenta para o desenvolvedor todos os espaços e alinhamentos corretos automaticamente.

[...]

Também é possível criar, testar e eliminar os erros de aplicações em desenvolvimentos para celulares. O NetBeans IDE oferece ferramentas com suporte de criação em todos os componentes Java EE, incluindo EJBs, serviços web, entre outros. é permitida a modelagem de visuais UML, com vários tipos de diagramas, bem como desenhar e analisar as aplicações destes modelos.

[...] [É] possível também utilizar os projetos já desenvolvidos [...] nas linguagens [disponíveis], e aproveita-los para novas criações.

NetBeans IDE oferece todo o suporte para criação de interfaces gráficas de maneira visual, bem como o desenvolvimento de aplicações para a internet, celulares e colaborações entre usuários, apresentando funcionalidades para soluções mais completas em programações com Java. (NETBEANS, 2014)

No Netbeans é possível se programar em várias linguagens, escolhidas no momento da instalação. Foi criado pensando no desenvolvimento web, embora seja muito útil para criar aplicações desktop.

Ele possui integrado em seu ambiente servidores virtual web que permitem acessar facilmente o projeto, principalmente em computadores de uma rede pessoal, mostrando no navegador páginas da web do ambiente da aplicação desenvolvida.

Utilizamos ele para desenvolvimento web, com a linguagem de programação JSP.

#### História do netbeans

O netbeans foi um projeto, que começou no âmbito acadêmico e foi um IDE que teve grande participação na popularização da linguagem Java, se tornando um IDE bastante popular para a atividade da produção de softwares com código Java.

O relato abaixo comenta como iniciou a história do IDE:

O NetBeans começou como um projeto estudantil (originalmente denominado Xelfi) na República Tcheca em 1996. O objetivo era escrever um Java IDE similar ao Delphi no Java. O Xelfi foi o promeiro (*sic*) Java IDE (Ambiente Integrado de Desenvovimento) escrito em Java, com sua pré-versões em 1997.

O Xelfi [...] [entrou num] território inexplorado naquela época. O projeto atraiu bastante interesse por estes estudantes, e quando se graduaram, decidiram que o poderiam o comercializar como um produto comercial. Pedindo recursos de amigos e parentes para um espaço na Web, eles formaram uma empresa em torno do mesmo

Logo depois, eles foram contratados por Roman Stanek, um empreeendedor (*sic*) que já tinha se envolvido em diversas empresas embrionárias na República Tcheca. Ele estava procurando por uma boa idéia (*sic*) na qual investir e descobriu o Xelfi. Ele se encontrou com os fundadores, eles se entenderam, e nascia um negócio.

O plano de negócios original era o de desenvolver componentes Java ativados para rede. Jarda Tulach, que projetou a arquitetura básica do IDE, teve a idéia (*sic*) do nome *Net*Beans para descrever o que os componentes fariam. O IDE seria o meio para entregar os mesmos. Quando a especificação para o Enterprise Java Beans surgiu, fez mais sentido trabalhar com o padrão para tais componentes do que competir com ele- mas o nome colou. (NETBEANS, 2015)

A interessante ideia de criação do IDE, veio de um país europeu, a República Tcheca e queria seguir o modelo do IDE de uma linguagem de programação concorrente (Delphi) para arquitetar um software do tipo para programar em Java.

Esse software dependeu da participação de patrocinadores, o que o empreendedor Roman Stanek fez. O Xelfi, que hoje veio que com o passar do tempo veio receber o nome que perdura até hoje, Netbeans, além de investimento precisou da ajuda de se pensar um nome atrativo comercialmente, como de servir para desenvolvimento web, que tinha surgido a não tanto tempo e estava, como até hoje ainda está em crescimento, naquela época percentualmente mais explosivo.

O Enterprise Java Beans foi utilizado como padrão de componentes com os quais trabalhar, e o nome Netbeans quis transparecer essa ideia.

É o Enterprise Java Beans:

“O Entertprise Java Bean (EJB) é um componente server-side que encapsula a lógica de negócio de uma aplicação. A lógica de negócio é o código que preenche os objetivos da aplicação. Ocupa a camada web em em um container EJB. [...]

Simplificam o desenvolvimento de aplicações grandes e distribuídas. Primeiro, porque o container EJB fornece serviços de nível de sistema a elas. Assim sendo, o desenvolvedor pode se concentrar em resolver problemas do negócio. O container EJB é responsável por serviços como gestão de transações e autorizações de segurança. Segundo, porque são os EJBs que contêm a lógica de negócios, não os clientes. Assim sendo, o desenvolvedor da aplicação cliente pode se concentrar na apresentação, não tendo que implementar regras de negócio ou bancos de dados de acesso. Como resultado, clientes tornam-se mais leves, executáveis em máquinas menos poderosas.” (JUNIOR, 2015)

O Netbeans ao invés de concorrer, recorreu ao padrão Java, utilizando o componente server-side, quem permite aos desenvolvedores se preocuparem mais com a regra de negócio, podendo destinar mais tempo no desenvolvimento ao que interessa ao cliente: a regra de negócio.

#### Primeiras versões do Netbeans

O início desse atualmente grandemente popular IDE foi no ano de 1999 com uma versão oficial do software disponível para os desenvolvedores, sendo uma opção de importância que estava surgindo no mercado.

Na primavera de 1999, o [NetBeans DeveloperX2](http://www.internetnews.com/dev-news/article.php/75561) foi lançado, suportando o Swing. Os aprimoramentos no desempenho vindos com o JDK 1.3, lançado no outono de 1999, tornou o NetBeans uma escolha viável para ferramentas de desenvolvimento. No verão de 1999, a equeipe (*sic)* estava trabalhando duro em rearquitetar o DeveloperX2 para o NetBeans mais modular que forma a base do software hoje.

Algo mais estava surgindo no verão de 1999: a [Sun Microsystems](http://www.sun.com/) queria ter melhores ferramentas de desenvolvimento Java e se interessou pelo NetBeans. Era um sonho se tornando realidade para a equipe do NetBeans: o NetBeans iria se tornar o conjunto de ferramentas estrela do próprio criador do Java!

Durante a aquisição, os jovens desenvolvedores que tinham estado envolvidos em projetos de código-fonte aberto na maioria de suas carreiras de programação, mencionaram a idéia de tornar o NetBeans em código-fonte aberto. Movendo rápido adiante menos de seis meses depois, foi tomada decisão que o NetBeans seria de código-fonte aberto. Embora a Sun tenha contribuído com considerável quantidade de código para projetos de código-fonte aberto ao longo dos anos, este foi o primeiro projeto de código-fonte aberto patrocinado pela Sun, um projeto para o qual a Sun pagaria pelo site e pelo manuseio da infraestrutura. (NETBEANS, 2015)

#### Apache Tomcat

O Apache Tomcat foi o Servlet container utilizado pelo nosso sistema, que é servidor responsável por interpretar os códigos presentes nas páginas .jsp presentes em nosso sistema.

Ele é totalmente gratuito, e sobre sua criação e funções principais diz o Portal Devmedia (2014):

O Apache Tomcat foi desenvolvido pela Apache Software Foundation, o Tomcat é um servlet container de código aberto, ou seja, uma aplicação que interpreta e processa servlets (java servlets) e JSP (Java Server Pages). O servidor está disponível livremente na Internet sem a necessidade de pagamento de licenciamento.

Após ser instalado, se exige do Tomcat que seja configurado para já poder ser utilizado:

Com o Tomcat instalado pode-se iniciar o desenvolvimento JSP. Para iniciar o desenvolvimento deve-se criar um diretório easyjava em “C:\Program Files (x86)\Apache Software Foundation\Tomcat 7.0\webapps (no Windows) e /usr/java/apache-tomcat-7.0 (no Linux)”. No diretório easyjava deve-se criar os diretórios src, web, WEB-INF e WEB-INF/lib.

•src = diretório onde fica armazenado o código fonte dos servlets;

•web = diretório raiz da aplicação web;

•WEB-INF = diretório que armazena o descritor da aplicação web (web.xml), bem como outros arquivos de configuração. Este diretório é invisível ao usuário;

•WEB-INF/lib = bibliotecas necessárias para a aplicação.

Colocamos aqui esse subtópico, pois conhecemos a utilização do Apache TomCat em uma instalação conjunta com o Netbeans, como um complemento dele, utilizando-o para fazer os testes necessários para corrigir o desempenho da aplicação.

# METODOLOGIA

Essas informações colhidas permitiram saber o que queriam nossos clientes. E foi isso mesmo que permitiu-nos concluir que esses problemas diversos deveriam ser tratados em sistemas diferentes. O que nos pareceu mais importante, por ser mais abrangente foi um sistema, que dentre essas necessidades, solucionasse o problema da reserva de horários e da entrega das chaves às pessoas que utilizadoras dos horários dos laboratórios.

## SURGIMENTO DA IDEIA

A primeira etapa foi o surgimento da ideia, que teve participação de um professor do Campus João Câmara, Odailson, que nos falou da necessidade de reserva de materiais e horários. Consideramos necessário, portanto, procurar os possíveis clientes de um sistema que resolvesse problemas do gênero, que chegamos à conclusão que seriam os membros da COLAB e os usuários deles. Fizemos uma consulta a professores utilizadores dos laboratórios gerenciados pelo setor em questão (a maioria dos que usam os laboratórios lecionam para as turmas de informática).

## LEVANTAMENTOS INICIAL DE REQUISITOS

Começamos a procurar professores que utilizavam os laboratórios, a maioria professores do curso de informática, já que a maioria dos laboratórios da instituição tem essa finalidade principal. Procuramos também os professores que lecionavam disciplinas e realizavam atividade de pesquisa nos outros laboratórios.

Entramos em contato com os professores fazendo um questionário espontâneo onde perguntávamos se seria realmente necessária a automação das reservas dos laboratórios, já que consideramos como um princípio real um aprendido na disciplina de Engenharia de Software que nem sempre será por um software que resolverá o problema de um cliente[[1]](#footnote-2). Aceitamos, assim, desde o princípio, que a nossa ideia pudesse ser rejeitada como boa alternativa para resolver os problemas dos usuários dos laboratórios.

Os professores não se posicionaram contra a implementação do sistema, porém não deixaram de aproveitar a oportunidade para declarar o que seria necessário para o efetivo funcionamento do sistema. Essas entrevistas foram feitas ainda em fevereiro de 2014.

### Professores dos Laboratórios de Informática

Dentre os professores do curso de informática, questionamos a alguns professores como Augusto Montalvão, Felipe Dantas, Alba Lopes, Valério Medeiros, que seriam atingidos pela ação de automatizar o controle de reservas dos laboratórios o que consideravam ser necessário a um sistema que automatizasse a COLAB, como se eles julgavam ele ser necessário.

O professor Valério Medeiros foi a favor de implementar o sistema e sugeriu alguns requisitos para o sistema. Ele falou ser importante o requisito de registrar solicitação de manutenção, onde se pediria para realizar algum conserto no hardware ou software de um computador. Numa tela de pedido se selecionaria qual o PC estaria com defeito dentre registrados no respectivo laboratório onde a máquina estivesse presente. Se deveria dar notificações por e-mail ou na própria tela do sistema. Deveria se implementar também a reserva automática, para que, segundo compreendemos, se o professor precisasse dar as aulas por todo o semestre, não fosse necessário ele semanalmente solicitar.

Ele tinha convicção de que o setor responsável pelo suporte dessas máquinas seria o setor de Tecnologia da Informação (TI), e na conversa com ele, sua afirmação foi dizer que o sistema poderia permitir ao professor pedir diretamente ao setor de TI, algo que creio que o professor Valério considerava um avanço, porque ele afirmou que a responsabilidade de solicitar suporte das máquinas seria da COLAB. No caso de a solicitação realmente não poderem ser mandadas diretamente de um usuário ao setor responsável pelo suporte das máquinas, perguntamos se não poderiam ser encaminhadas mesmo à COLAB, para reencaminharem para a TI, ao que ele nos respondeu afirmativamente.

O professor Augusto, ao ser entrevistado, concordou com a ideia de necessidade de um sistema de reserva de horários, e já sobre a questão de reserva de materiais, cuja necessidade foi negada por outros professores, ele afirmou que o professor pudesse talvez precisar pedir seria data show, apagador, ar condicionado, troca de máquina e cabos. Falamos que a reserva de materiais seria para aulas práticas, como o professor Odailson queria para as suas aulas práticas de eletricidade e manutenção. Também propomos que para as aulas práticas tivesse na solicitação de horário um campo que selecionasse que ela era uma aula prática, e que depois o professor pudesse emitir um relatório dizendo quantas aulas práticas deu em determinado período.

#### Utilidade das informações no processo de desenvolvimento

Desses dados coletados em entrevistas juntamente com os professores, foi importante adquirir essas informações, embora não tenham sido atendidas todos os requisitos inicialmente definidos no software.

Primeiramente, quando a validade da entrevista como sendo de professores que representem as demandas presentes na instituição para o setor para o qual estamos desenvolvendo o software: expomos aqui que os professores não se encontram mais no campus João Câmara, tendo se desligado da instituição ou se remanejado. O professor Odailson foi citado aqui na seção dos laboratórios de informática, pois na entrevista quisemos saber se eles tinham a mesma demanda que os usuários do laboratório de manutenção, apenas. Porém o mesmo ainda é ligado ao campus. Entretanto, independentemente do fato exposto, de terem se retirado do campus os professores de informática, usuários da COLAB entrevistados enquanto realizávamos a análise de requisitos, mesmo assim essas opiniões são válidas pois representam usuários de mesmo perfil e que também tem a mesma necessidade de conserto de máquinas, solicitar horários ou reserva de materiais.

Sobre o que o professor Valério disse de ser necessário, de realizar solicitação de manutenção de hardware ou software de computador, pelo menos para hardware, atualmente pode ser feita uma solicitação para o Coordenação de Serviços Gerais e Manutenção (COSGEM) do Campus João Câmara. Conforme podemos ver descrito nesse texto:

Visando uma maior eficiência no atendimento ao público, a Coordenação de Serviços Gerais e Manutenção do Campus João Câmara (COSGEM) disponibilizou nesta segunda-feira (29/07), no site da Instituição, um Link específico para a solicitação de seus serviços.

O procedimento é muito simples: na parte inferior da página do IFRN/JC há um link intitulado COSGEM. Ao clicá-lo, será aberta a página da referida Coordenação, onde constará os links para solicitação de serviços e agendamento de transportes.

Feito isso, deve-se clicar no link desejado, preencher os dados correspondentes ao seu pedido e, em seguida, na opção "ENVIAR". Ao final, sua demanda estará registrada e, respeitadas as prioridades do serviço, será atendida. (IFRN/JC, 2013)

Abaixo estão as imagens capturadas do referido sistema:

Figura 4: A página de solicitação de serviços requer primeiramente o nome do solicitante, sua matrícula, o setor relativo ao qual quer manutenção



Fonte: *print screen* no sistema operacional Windows 8.1 de *COSGEM, 2015*

Figura 5: A página de solicitação de serviços requer depois o nome o tipo de solicitação, descrição do pedido, prioridade, data e horário de conclusão



Fonte: *print screen* no sistema operacional Windows 8.1 de *COSGEM, 2015*

Embora o sistema seja da COSGEM, quem faz o suporte das máquinas é o setor de TI. Definimos que nosso cliente seria a COLAB. Não conhecemos o mecanismo de troca de dados do entre setor que gerencia o sistema de planilhas da Google e o setor que dá suporte aos computadores, mas sabemos que não é automatizado, já que não existe sistema específico de troca de dados entre a COSGEM e a TI. Não está sendo criada a dificuldade por esse texto aqui, existe esse problema na realidade. Aqui especificamos qual seria a solução: criar um sistema de troca de dados entre a COSGEM e a TI, o que não está em nosso escopo, pois temos apenas a COLAB por cliente. Se criaria um sistema de troca de dados entre os dois outros setores da instituição, citados nesse parágrafo, pra depois ser possível, de fato pensar e executar uma funcionalidade que trabalhasse em conjunto com o método atual de fazer chamado de suporte aos computadores.

Portanto, levando em conta a conjuntura atual, não se propicia desenvolver essa funcionalidade de chamado para suporte das máquinas dos laboratórios de informática. Nem mesmo na primeira versão de nosso documento de visão, se pensou em colocar pedir suporte às máquinas como requisito funcional. Foi automaticamente descartado, e nunca mais se pensou em desenvolvê-lo.

Dentre as entrevistas com os alunos, se permitiu por João Victor saber da existência desse sistema, conforme se verá descrito na seção com clientes alunos.

Já quanto a ideia com a qual o professor Augusto concordou, de fazer reserva para aulas prática de material, já encontramos aqui primeiramente a realidade de que contar quantas aulas práticas são dadas, é um controverso o uso desse conceito para aulas de informática, porque não existiria bem o conceito de aula prática e teórica em informática, já que a ideia geral poderia ser de que praticamente todas são práticas laboratoriais.

Considerando, porém, que as aulas dadas sejam práticas, encontramos no desenvolvimento de um sistema de reserva de materiais o problema de que não é fácil, nem automático o cumprimento do pedido de colocar materiais como apagadores, projetores, se não existirem no laboratório, primeiramente porque há necessidade de licitação para conseguir materiais de consumo como apagadores, lápis de quadro branco, com limite de gastos, assim como ocorre com os projetores, que são materiais permanentes e de mais difícil licitação, segundo nos falou Abrantes. Não podíamos fazer, portanto, segundo o professor Augusto sugeriu, com dificuldade quanto a conseguir esses materiais, que foi considerado necessário por poucos professores de informática.

Sobre o requisito de reserva de material não ser feito se observará mais adianta na subseção de “Reuniões com o professor Lucas”, que ele considerou isso parte de outro sistema específico, pois ele se enquadra mais para uso dos professores que dão aula no laboratório de manutenção, para o qual a COLAB possui muito material de consumo, como resistores, transistores. Portanto não foi desenvolvido.

O que originou a documentação de requisitos de software atual foi a ideia da reserva de laboratórios, que ficou nas discussões com o cliente na maior parte do tempo como principal funcionalidade, mas que com o tempo, foi apontada como secundária, já que se tratava de um processo específico, e menos geral, sendo diminuída a prioridade de entrega, já que são feitas poucas vezes pedidos avulsos de horário, e mudam várias vezes os horários permanentes, o que devido ao tempo existente para fazer a implementação dos códigos, impede de se fazer uma aplicação mais generalista, incluindo o requisito de reserva de horários, que gerou a necessidade de entregar e receber chaves relativas aos seus específicos horários. Como a função da COLAB de entrega e recebimento de chaves é mais utilizada do que a reserva de horários de laboratórios, mais complexa, com várias mudanças regularmente existentes no início dos semestres, preferiu-se implementar o controle de chaves, devido ao tempo existente, ficando operante essa ferramenta.

### Professores dos Laboratórios de Biologia

Dentre os professores que utilizam o laboratório de Biologia, falamos com Airton, que era usuário do laboratório e coordenador de pesquisa do campus na época da entrevista.

O professor Airton foi questionado sobre se a reserva de laboratórios seria necessária e falou-nos então que seria necessário ao professor que fizesse a reserva no laboratório de Biologia escolher durante a reserva se o laboratório seria usado para atividade de pesquisa ou extensão no horário em que ele escolhesse. Já ao perguntarmos se seria necessário utilizar um módulo de solicitação de materiais para reservar materiais com antecedência para aulas práticas, requisito que dissemos a ele ter sido pedido por Odailson para suas aulas, Airton disse que não, já que segundo Airton, os bolsistas utilizavam mais para atividades de pesquisa e não seria adequado pedir a um técnico da COLAB que pudesse nem ter conhecimento dos reagentes que estariam em uso nos experimentos, prover materiais para a pesquisa, solicitados por meio do sistema, por não ser ideal os assistentes administrativos utilizarem os equipamentos do laboratório. Porém ele indicou que seria bastante útil se houvesse um sistema que agendasse avisos que informasse quando um certo experimento deveria receber alguma intervenção, levando em conta o tempo estipulado pelo responsável pelo pesquisador para determinado evento ocorrer, ou ações terem que ser executadas em momento precisos. O que seria algo como: “preste atenção, necessita-se deixar estufa desligada após 10 horas”.

#### Utilidade das informações no processo de desenvolvimento

Ao falarmos com ele, tivemos maior convicção que não seria possível aplicar para todos a ideia de reservas de materiais para aulas práticas de professores de todas as disciplinas, pois não seria isso uma necessidade de todos, por trabalharem com diferentes especificidades.

O que o professor Airton nos afirmou, negando a necessidade de reserva de materiais para a aula corroborou para a noção de que a reserva de materiais para os laboratórios seria uma necessidade mais específica dos professores que utilizavam o laboratório de manutenção, embora não só Odailson, mas também o professor Augusto, de informática, teria como requisito necessário reserva de materiais, na parte da entrevista em que se discutia sobre esse requisito.

Pelo que Airton falou, para sanar os problemas do laboratório de ciências por meio da automação, teriam uma especificidade diante de outros laboratórios na automação por software.

Sobre o que Airton falou, sobre não ser bom um técnico da COLAB mexer nos reagentes por não ter conhecimento sobre eles, na época em que ele entrevistou era realidade, porém hoje tem mais técnicos nos laboratórios, chegando para suprir as necessidades da instituição, uma técnica em química, que tem esse conhecimento dos reagentes trabalha no setor em questão, sendo, então mais viável que algum técnico pudesse colaborar nas atividades do laboratório e utilizar um sistema para ajudar nessa tarefa, porém não obrigatoriamente na reserva de materiais, mas nas necessidades de auxílio na realização do experimentos, no passo a passo, como dando lembretes e telas de gerenciamento das tarefas.

Porém, chegamos à conclusão que não seria bom colocar nesse mesmo software a funcionalidade reserva de materiais. Melhor afirmar que isso seria corretamente executado por um software ou específico para reserva de materiais, que poderia futuramente ser integrado às demais funcionalidades do sistema, e mais útil ao laboratório de manutenção do que ao de biologia, como apropriadamente falou o professor Airton.

Quanto às necessidades desse laboratório (de biologia), não se trata das necessidades gerais de todos os usuários do setor para o qual estamos desenvolvemos o software.

### Alunos Usuários dos Laboratórios Durante as Aulas

Dentre os professores do curso de informática, questionamos a alguns professores como Augusto Montalvão, Filipe Dantas, Alba Lopes, Valério Medeiros, que seriam atingidos pela ação de automatizar o controle de reservas dos laboratórios.

O professor Valério Medeiros foi a favor de implementar o sistema e sugeriu alguns requisitos para o sistema. Ele falou ser importante o requisito de registrar solicitação de manutenção, onde se pediria para realizar algum conserto no hardware ou software de um computador. Numa tela de pedido se selecionaria qual o PC estaria com defeito dentre registrados no respectivo laboratório onde a máquina estivesse presente. Se deveria dar notificações por e-mail ou na própria tela do sistema. Deveria se implementar também a reserva automática, para que, segundo compreendemos, se o professor precisasse dar as aulas por todo o semestre, não fosse necessário ele semanalmente solicitar.

Ele tinha convicção de que o setor responsável pelo suporte dessas máquinas seria o setor de TI, e na conversa com ele, sua afirmação foi dizer que o sistema poderia permitir ao professor pedir diretamente ao setor de TI, algo que creio que o professor Valério considerava um avanço, porque ele afirmou que a responsabilidade de solicitar suporte das máquinas seria da COLAB. No caso de a solicitação realmente não poderem ser mandadas diretamente de um usuário ao setor de TI, perguntamos se não poderiam ser encaminhadas mesmo à Coordenação de Laboratórios, para reencaminharem para a TI, ao que ele nos respondeu afirmativamente.

O professor Augusto, ao ser entrevistado, concordou com a ideia de necessidade de um sistema de reserva de horários, e já sobre a questão de reserva de materiais, cuja necessidade negada por outros professores, ele afirmou que o professor pudesse talvez precisar pedir seria data show, apagador, ar condicionado, troca de máquina e cabos. Falamos que a reserva de materiais seria para aulas práticas, como o professor Odailson queria para as suas aulas práticas de eletricidade e manutenção. Também propomos que para as aulas práticas tivesse na solicitação de horário um campo que selecionasse que ela era uma aula prática, e que depois o professor pudesse emitir um relatório dizendo quantas aulas práticas deu em determinado período. Não podíamos fazer, porém, segundo Augusto, uso desse conceito, porque não existiria bem o conceito de aula prática e teórica em informática, já que a ideia geral poderia ser de que todas são sim práticas.

### Aluno Bolsista da COLAB

Falamos com um bolsista que estava no setor da Coordenação de Laboratórios. Perguntamos a ele sobre se se tinha necessidade de fazer o sistema para controlar os horários. Ele nos falou que deveria poder ver os horários que estivessem disponíveis para reserva ao ser ver a tabela de horários. Ele falou que por meio da matrícula se deveria criar a reserva, e que o sistema deveria ser WEB.

Nós tentamos em nossa tentativa de contato com os clientes do software, falar com Abrantes, que é coordenador da COLAB, porém estava de férias, assim conseguimos apenas falar com o bolsista que estava trabalhando.

## ORIENTAÇÕES TÉCNICAS PARA O DOCUMENTO DE VISÃO

Procuramos a orientação dos professores da área de Sistema de Informação do Instituto, como Lucas e Alba que nos auxiliaram na formalização do projeto por meio do Documento de Visão. Após algum tempo registramos na coordenação de extensão do campus nosso projeto, tendo como nosso orientador o professor Daniel Aguiar, que nos direcionou no projeto de desenvolvimento de nosso software até seu desligamento da instituição e indicação do professor Diego Oliveira para nos guiar na etapa da prática de desenvolvimento.

Iniciamos entrevistas e colhimento de informações na tentativa de formular o escopo de nosso projeto, suas funcionalidades, e voltamos a criar a percepção de quem seriam nossos clientes: que concluímos que seriam os professores, a Coordenação de Laboratórios, que é representada diretamente pelo Coordenador do setor, Abrantes, que foi quem nos trouxe conhecimento de como funciona o trabalho da COLAB, e o procedimento de reserva um horário por parte de um professor.

### Reunião com Professora Alba

Procuramos ao dia 9 de maio de 2014 orientação técnica da professora Alba, ao momento professora do curso de informática no campus João Câmara.

A professora Alba pediu para que pudesse nos sugerir alterações para nosso projeto de software que lhe mostrássemos um documento de visão que permitisse conhecimento do que seria o sistema. Definimos na primeira versão de nosso documento de visão que nosso sistema teria como requisitos “agendamento de laboratórios”, “reserva de material” e “visualização de horários”, porém ao consultarmos a professora Alba, ela nos notificou a ausência de dois requisitos funcionais: “gerenciamento de usuários” e “cadastro de laboratórios”, o que viemos a corrigir muito tempo depois criando “gerenciar laboratórios”.

#### Alterações que fizemos em nosso documento de visão

Com a orientação da professora Alba, que foi simplesmente correção de nosso documento de visão, inserimos na segunda versão do nosso documento de visão dos dois requisitos solicitados por ela, “gerenciamento de usuários” e “cadastro de laboratórios”, apenas o primeiro, por termos esquecido de colocar “o cadastro de laboratórios”, erro corrigido apenas meses depois, com a criação do requisito “gerenciar laboratórios”.

Também entre nós, por sugestão de Nathan inserimos professores do PRONATEC como clientes do nosso sistema.

### Reuniões com professor Lucas

As reuniões com professor Lucas ainda não aconteceram após a oficialização, que se deu com a procura do professor Daniel, que oficializou o projeto que orientou por um período e antes de sair da instituição pediu para Diego continuar a orientação. Essas reuniões foram uma consulta técnica, a fim de o professor Lucas apontar quais os erros a serem corrigidos no documento de visão, apontando nele possíveis requisitos ausentes, necessários ao sistema, que considerasse necessários e ideias mal apresentadas em nosso projeto.

#### Primeira reunião

Após as férias e ao fazermos nossos levantamentos iniciais, resolvemos procurar um de nossos professores do curso para nos orientar no registro de requisitos levantados para o sistema.

O professor Lucas indicou a altura dessa primeira reunião que inseríssemos gerenciamento de usuários como requisito funcional, algo que confirmou o pedido de Alba o que citamos a ele

##### Alterações que fizemos em nosso documento de visão

Alteramos na segunda versão, conforme ele nos pediu, os requisitos funcionais, adicionando o requisito funcional Gerenciamento de Usuários, que havia sido ignorado primeiramente.

Inserimos também o controle de chaves, por espontânea vontade Gerenciamento de Chaves, que era algo que percebemos estar faltando ser adicionado em nosso projeto, devido a demanda por gerenciamento automatizada, por ser uma maneira pela qual um sistema informático pode agilizar mais as atividades do setor, que faz grandes gastos de papel, e faz busca mais lenta que uma consulta num bando de dados, e uma atividade que precisa de mais intervenção manual. Alterações que fizemos em nosso documento de visão.

#### Segunda reunião

Na segunda reunião, além de o professor fazer correções no texto de significado e ortografia existentes, nos orientou na estruturação do documento de visão, que continha falhas, como explicitar qual local estaríamos visando automatizar por meio de um software, o que seriam os “laboratórios do IFRN Campus João Câmara”. Também um erro quanto ao local correto no texto de fazer conhecida ao leitor a sigla “COLAB”; que devia ser ao início do texto, na primeira vez que citássemos a Coordenação de Laboratórios. Também pediu que a gente no parágrafo de Cenário, Problemas e Justificativa, usássemos o conector “atualmente”, de maneira que deixássemos claro a situação da COLAB antes do sistema, para convencer o cliente quanto a melhoria depois do software, deixando mais claro o argumento do produto que estávamos oferecendo naquele documento de visão.

O professor também sugeriu correções do ponto de vista técnico, essas mais importantes, quais foram: o banco de dados deveria seria “relacional”, e não “SQL”, como descrevemos inicialmente e deveríamos segundo ele pode ver inserir mais um requisito funcional, “cancelamento de reserva”, já que em nenhum requisito existia essa função necessária para o bom uso do software.

A essa altura tínhamos quatro requisitos funcionais: “controle de chaves”, “gerenciamento de usuários”, “visualização de horários”, “reserva de material”, “agendamento de laboratório”. Mesmo existindo um requisito “controle de chaves”, o professor falou que era necessário inserir o “cancelamento de reserva”, pois na descrição do requisito, não pomos que o controle de chaves incluiria a possibilidade de exclusão de reserva, pelo contrário, ele tratava de “controle de horário de entrada e saída de chaves relacionado ao nome da pessoa que a retirou do setor, possibilitando também exibição de quais chaves foram retiradas e quais outras estão disponíveis no setor COLAB”. Portanto, esse requisito ainda não tinha sido criado no documento de visão, já que entrada nem cancelamento de reservas tinham sido propostos nos requisitos existentes.

Conversando com ele sobre o nosso documento de visão nessa reunião, foi comentado sobre os pontos que escrevemos em nosso documento de visão, e também os pontos que houveram divergência por parte dos professores entrevistados no levantamento de requisitos inicial: como necessidade ou não necessidade de utilizar reserva de materiais dentre as funcionalidades do sistema.

Ao expormos ao professor a opinião das pessoas a quem consultamos, lembrando especificamente dos comentários da maioria dos professores entrevistados serem contrários à ideia de inserir a reserva de material, perguntando a ele sobre isso, nos passou a posição de que reserva de materiais seria um sistema à parte, portanto, deveria sair dos nossos requisitos funcionais. Segundo professor a reserva de materiais poderia ser ideia para construção de um sistema dedicado a gerenciar essa tarefa, assim como também deveria ocorrer com a reserva de horários para os laboratórios, que seria um sistema considerado necessário pela maioria dos professores que entrevistamos.

Foi pedido também pelo professor Lucas que fizéssemos alteração do requisito que definia em que lugar se executaria nosso sistema, onde lhe afirmamos que seria um sistema web, ele solicitou que disséssemos quais navegadores executariam o sistema de nosso projeto.

##### Alterações que fizemos em nosso documento de visão

Nós dissemos conforme pedido pelo professor Lucas de onde o software visava a “automatização do processo de reserva”, “de laboratórios do IFRN Campus João Câmara”, onde antes dizia apenas “de salas e materiais”, não inserindo o local do qual se visava executar o projeto de software.

Quanto a sigla COLAB, pusemos referência dela logo ao início do texto, “Coordenação de Laboratórios (COLAB)” onde antes só apresentávamos no segundo parágrafo do documento.

Substituímos também a descrição do requisito não funcional de “SQL”, para “relacional”, embora não fosse algo obrigatório de estar presente dentre os requisitos.

Quanto à criação do requisito “cancelamento de reserva”, devido ao esquecimento, inserindo-o somente na quarta versão do documento de visão, com o nome de “reserva e controle”.

Inserimos por autocrítica os requisitos “reserva recursiva”, “gerar relatório” e “aviso de reservas pendentes para horário selecionado”.

Quanto ao cancelamento do requisito reserva de materiais, obedecemos, portanto, à orientação do professor, deixando à parte, o requisito de reserva dos materiais, ficando fora do nosso sistema, que passou a tratar de solicitação e cancelamento de reserva de chaves, já que como o professor falou, esse requisito seria parte de outro sistema, caso se desejasse implementar.

### Reuniões com o professor Daniel

Além do professor Lucas, para a criação da terceira versão do documento de visão, contatamos o professor Daniel, que permaneceu como nosso orientador até o desligamento da instituição, quando indicou a substituição dele por Diego.

#### Primeira correção do documento de visão

Na nossa primeira reunião, ele olhou basicamente os requisitos funcionais na primeira correção do documento de visão. Segundo Daniel, seria indispensável o sistema ter como requisito, como qualquer outro sistema, gerar relatórios ao usuário, que permitisse, por exemplo, o professor visualizar qual a carga horária reservada nos laboratórios, ou o Coordenador de Laboratórios visualizar quais chaves estão em uso, quais horários estão reservados, permitindo uma formatação dos dados do banco para visualização do usuário.

O professor também ordenou que revisássemos os requisitos não-funcionais que não estavam colocados adequadamente, onde colocamos coisas erradas e outras, necessárias.

Tínhamos assim definido:

Tabela 1: Requisitos não funcionais definidos no início do projeto

|  |  |  |
| --- | --- | --- |
| **Cod.** | **Nome** | **Descrição** |
| NF01 | Banco de Dados | SQL |
| NF02 | SGBD |  |
| NF03 | Linguagem Web | HTML5 |
| NF04 | Estilo | CSS3 |
| NF05 | Scripts | Javascript |

Fonte: produção própria

Foi ordenado que seguíssemos um modelo correto, e colocássemos coisas que realmente fossem requisitos não-funcionais, como vimos em um modelo de um documento correto, Segurança, Sistema Operacional, Ambiente Operacional, Base de Dados, Mobilidade e Plataforma de Desenvolvimento.

O professor falou da necessidade de aprendermos a fazer diagramas de caso de uso para apresentarmos ao nosso cliente, Abrantes e representar os papéis dos usuários no sistema.

##### Alterações que fizemos em nosso documento de visão

Alteramos os requisitos não-funcionais, que o professor tinha dito que não eram apropriadamente requisitos não-funcionais, os que antes estavam presentes no documento de visão.

Assim passou a se apresentar os requisitos não-funcionais:

Tabela 2: Requisitos não funcionais após correções iniciais

|  |  |  |  |
| --- | --- | --- | --- |
| **Cod.** | **Nome** | **Descrição** | **Categoria** |
| NF01 | Banco de Dados | Relacional |  |
| NF02 | Ambiente Operacional | Google Chrome, Mozilla, Firefox, Opera, Safari | Desejável |
| NF03 | Sistema Operacional do Servidor | Windows | Desejável |
| NF04 | Mecanismo para persistência | JBDC | Desejável |
| NF05 | Plataforma de desenvolvimento | Netbeans. |  |
| NF06 | Base de Dados | MySQL | Desejável |
| NF07 | Mobilidade | O usuário poderá acessar em computador em qualquer lugar com acesso à internet | Obrigatório |
| NF08 | Segurança | Apenas usuários devidamente autenticados poderão reservar horários e/ou gerenciá-los | Obrigatório |

Fonte: produção própria

As alterações que fizemos excluíram os requisitos não-funcionais estilo, scripts, linguagem web e adicionou outros requisitos não palpáveis. Colocamos Banco dados como relacional devido uma observação do professor Lucas, porém isso também que fizemos não era algo necessário.

Adicionamos ambiente operacional como todos os navegadores web mais conhecidos, porém deixamos isso como algo desejável, pois depois saberíamos se de fato haveria compatibilidade com eles. Já como sistema operacional do servidor colocamos Windows, porém, por falta de conhecimento, por não ser obrigatoriamente necessário. Por termos usado como plataforma de desenvolvimento o Netbeans, poderia muito bem o nosso sistema rodar em Linux, sem problema algum. O sistema de persistência escolhido foi o JBDC, pois é a maneira conhecida de conectar a aplicação com código Java ao banco de dados. A base de dados foi definida como MySQL, que ao momento tínhamos conhecimento. Cremos que a mobilidade fosse necessária, por se tratar

Ampliamos o subtópico Cenário, Problemas e Justificativa da Introdução, começando a explicar que apesar do fato de o professor estar ou não no Campus, com o sistema, ele poderia reservar um horário e ter acesso ao sistema.

Falamos do problema de as reservas de horário serem feitas por e-mail, o que tornava difícil gerenciar as mensagens e tornava lento o processo.

#### Segunda correção do documento de visão

Dessa vez, o professor mandou a gente melhorar a explicação presente em nosso documento de visão, ordenando que gente rapidamente o que é a COLAB, explicar os problemas que há nela, na e que também falássemos da perda de materiais e de chaves subseção da Introdução, Cenário, Problemas e Justificativa, para que assim se descrevesse a realidade que mostraria a utilidade do desenvolvimento de software, que evitaria de ocorrer essas perdas, por meio da automação do controle das chaves, auxiliando na busca por devolução das chaves, diante da demora delas serem devolvidas.

Daniel pediu que detalhássemos todas as funcionalidades presentes nos requisitos funcionais do escopo, e que criássemos o diagrama de casos de uso para inserir no documento e exibir ao cliente.

Mandou que passássemos todas as primeiras palavras do requisito funcional para verbos no infinitivo, onde tinha o nome dos requisitos **visualização** de horários, **gerenciamento** de usuários, **reserva** recursiva, **agendamento** de laboratório e **controle** de chaves.

Dessa vez o professor ordenou que fizéssemos outro documento acessório de nosso projeto, o documento de ambiente, que devia ser iniciado juntamente com as alterações que deviam ser feitas no documento de visão.

##### Alterações que fizemos em nosso documento de visão

Adicionamos o requisito reserva e controle para os requisitos dentre os requisitos funcionais.

Não corrigimos os nomes dos requisitos funcionais, que deviam ter nomes no infinitivo na outra versão do documento de visão.

O diagrama de casos de uso foi criado na outra versão do documento do documento de visão, e além disso tiramos o gerar relatórios, que foram

O documento passou uma nova edição muito rápida, e não foram concluídas as alterações, devido à ordem do professor de começarmos a utilizar um novo modelo para edição do documento de visão.

#### Terceira correção do documento de visão

O professor nos ordenou que alterássemos para outro modelo, e pediu que praticamente recomeçássemos a edição e formatação do documento, e consequentemente fazer as alterações de colocar os verbos dos requisitos funcionais no infinitivo.

A ampliação do Cenário, problemas e Justificativa, explicando melhor o papel da COLAB deveria voltar a ser feito, já que não tinha dado tempo concluir. Foi reordenado fazer o diagrama de casos de uso, atividade que deveria ser o foco, para ser o elo facilitar a conversa com o cliente sobre a parte técnica do sistema.

O professor examinou e viu que a definição de cada requisito funcional foi muito ampliada, o que o fez solicitar que fosse resumida, como próprio para esse documento, que representa o contrato com o cliente.

##### Alterações que fizemos em nosso documento de visão

Foi feito o diagrama de casos de uso, que foi apresentado a Abrantes para ele opinar sobre os atores que poderiam usar certos casos de uso – que tinha relação direta com os requisitos funcionais, e fizemos uma tabela, que indicava a relação dos requisitos funcionais com os não funcionais.

Esse foi o primeiro diagrama de casos de uso, que foi apresentado a Abrantes, para análise dele das necessidades indicadas para o sistema:

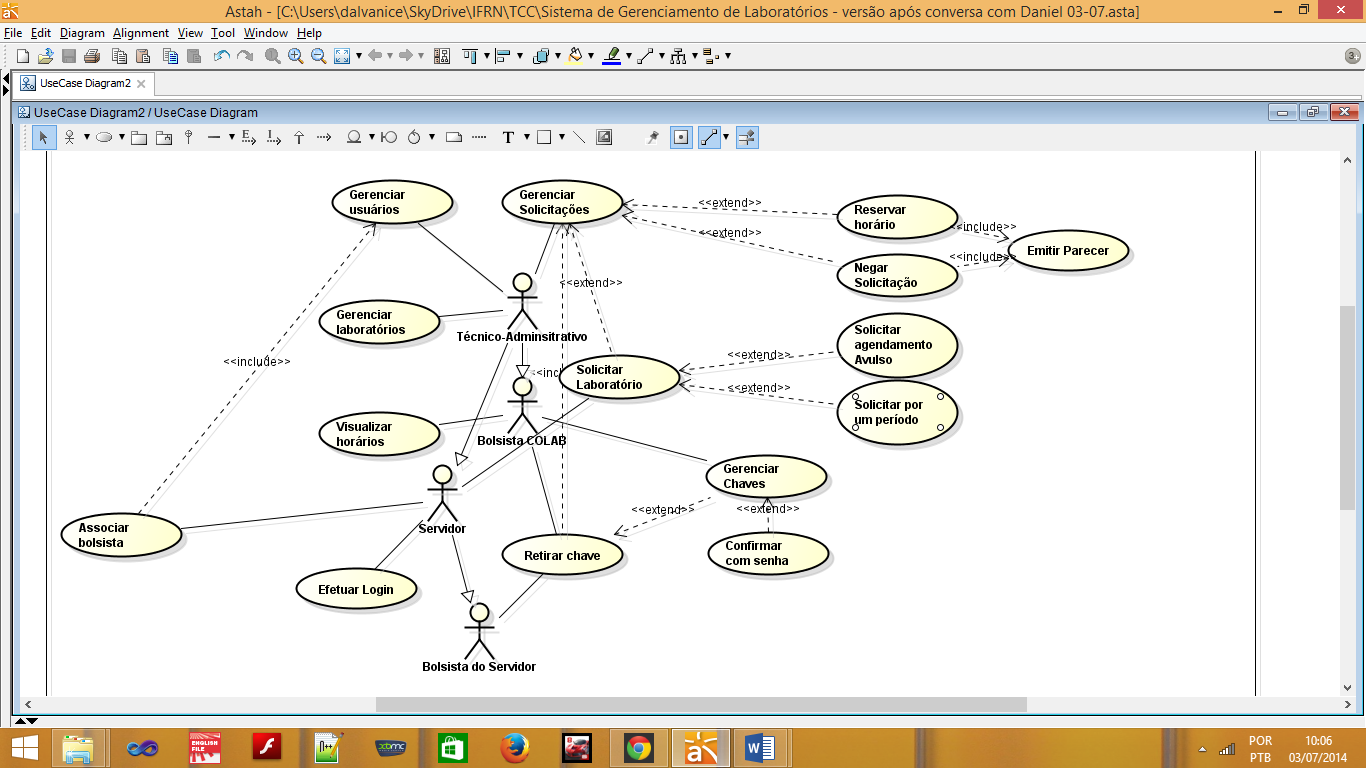


Figura 6: Diagrama de casos de uso feito para ser apresentado a Abrantes, colocado no documento de Visão

.

Fonte: feito a partir de *printscreen* de programa Astah

Os requisitos **Gerar relatório** e **Aviso de situação de reservas pendentes para horário selecionado** foram excluídos.

Os requisitos **Agendamento de Laboratório**, **Visualização de Horários**, **Gerenciamento de Usuários**, **Controle de Chaves**, **Reserva recursiva** tiveram seus nomes corrigidos e substituídos respectivamente por: **Solicitar agendamento avulso**, **Visualizar horários**, **Gerenciar usuários**, **Gerenciar chaves**, **Solicitar por um período**,

**Gerenciar laboratórios**, **Emitir parecer**, **Gerenciar solicitações**, **Associar Bolsista**, **Efetuar login**, **Retirar chave**, **Negar solicitação**, **Solicitar laboratório**, **Reservar horário** e **Confirmar com senha** foram criados.

Para todos esses requisitos novos e os que tiveram os nomes substituídos (por estarem fora de convenção) foi arbitrado que teriam um caso de uso de mesmo nome que representaria as ações, para facilitarmos a associação entre os dois elementos do projeto.

Os atores criados para representar todas essas ações foram: **Técnico-administrativo**, **Bolsista COLAB**, **Servidor** e **Bolsista do servidor**, que tiveram todas as suas ações relacionadas.

Já quanto aos requisitos não-funcionais, o requisito não-funcional **Ambiente operacional** foi renomeado para **Compatibilidade**. O requisito **Banco de Dados** teve sua categoria em branco alterada para obrigatório. **Sistema Operacional do Servidor** foi excluído. **Base de dados, Mobilidade**, **Segurança** e **Mecanismo para persistência** foram mantidos, enquanto os requisitos não-funcionais **Plataforma de desenvolvimento** (Netbeans), **Tecnologias de desenvolvimento** (JSP) e **Usabilidade** foram criados.

Todos os requisitos funcionais, com exceção de emitir parecer foram explicados no escopo.

##### Alterações que fizemos em nosso documento de visão

Fizemos mais outra versão ainda movidos pela mesma orientação do professor e ao falar-nos sobre a necessidade de escreve o TCC, verificamos e percebemos que esquecemos de colocar a descrição do requisito funcional emitir parecer no escopo, o que foi corrigido.

## ENTREVISTAS COM ABRANTES

Entrevistamos Abrantes, e consideramos ele nosso cliente direto, já que ele é o Coordenador da COLAB, coordenando o grupo de bolsistas e servidores que gerenciam as tarefas dos laboratórios. Os professores que deram nas entrevistas anteriormente relatadas, são clientes da COLAB, em uma vinculação institucional, portanto, indiretamente a clientela de nosso software, que consideramos importante ser consultado, antes de chegarmos com alguma proposta a Abrantes, de maneira de já ter obtido mais conteúdo e chegar mais rápido nos contatos com ele à solução que se devia atingir, discutindo como quem já sabia qual o serviço que ele realiza no setor para o qual estamos desenvolvendo o software.

Sabemos que por meio dele se coloca a implementação de soluções para o serviço do setor, e portanto foi necessário falar com ele, e com os dados que tínhamos coletados, os quais serviram para tratar do que prometemos que debateríamos com ele – quais as funcionalidades necessárias ao sistema.

Os dados levantados com os professores foram importantes para entender o serviço para o referido setor, onde ele fez apenas confirmar ou negar a veracidade das impressões que tivemos dos contatos professores, como sendo realmente a forma de trabalho do setor em questão.

Tivemos por objetivo apresentar os requisitos já levantados a partir do contato com os professores, um bolsista da COLAB, e perguntar quais eram as funcionalidades que ele considerava importante ser adicionado ao sistema e quais não.

Apresentamos a ele o diagrama de casos de uso presente no documento de visão e perguntamos se o papel de cada usuário do sistema que identificamos estava correto ou não, apontando os erros e quais alterações no diagrama de casos de uso resolveria os problemas, que ocasionalmente fossem identificados.

Após isso, na segunda entrevista, já apresentamos um parte do software pronto, ainda incompleto, perguntando se havia ainda alguma coisa que faltava ser implementada ao modo de ver dele como um dos administradores do sistema.

### Entrevista Inicial

No momento em que contatamos Abrantes, descobrimos que as reservas são feitas a partir de uma planilha online do Google Drive, por parte dos professores de informática, que são compartilhadas normalmente por envio de e-mail entre o coordenador da COLAB e os professores. Também disse reservar além de por essa planilha, por mensagens de e-mail, e conforme vai recebendo e-mails ou pedidos presenciais do professor que se dirija até ele ou outro técnico da COLAB, vai centralizando essas informações na planilha. Com isso, o fechamento da reserva ocorre conforme ele fosse lendo as informações em seu e-mail e descobrindo qual horário foi solicitado pelo professor, registrando enfim, em sua planilha.

Das reuniões que expressam o consenso dos professores de informática sobre em quais horários estarão sob responsabilidade de cada professor os laboratórios, são enviadas informações por e-mail ou pela planilha a qual Abrantes tem poder de editar para a concretização da marcação de horários.

Vimos que os horários podem ser reservados apenas por servidores da instituição e professores do PRONATEC, mas podendo ser abertos por alunos desses servidores sob acordo com a COLAB.

Quanto aos horários marcados para alunos bolsistas realizarem suas tarefas, somente o professor pode solicitar esses horários na Coordenação dos Laboratórios, sendo permitido a partir daí o relacionamento do aluno com a COLAB, por meio da retirada de chave, a partir do momento em que a COLAB reserva o horário solicitado pelo professor responsável pela orientação das atividades que seus bolsistas desenvolvem nos laboratórios.

Quanto à liberação da chave, para o servidor pegar sua chave, deve ser verificado pelo bolsista da COLAB ou pelo servidor do mesmo setor se há horário marcado. Mas isso é feito, assinando em listas de papel seu nome, tendo o controle do pessoal que vem ao laboratório, e verificando se o usuário tem direito a pegar a chave, e verificando se o usuário tem direito a pegar a chave para verificar a existência de reserva de horários em tabela listando atividades nos laboratórios, verificando se o usuário é bolsista de algum professor ou não, caso seja um aluno que peça à COLAB a chave. De início essa tabela de horários causava grandes problemas, principalmente para os alunos, que não podem pegar a chave, mesmo se ele estiver livre. O Coordenador da COLAB quando demorava a chegar, ao ainda não ter impresso o arquivo, deixava sem atendimento os alunos bolsistas que vinham aqui para pegar a chave do laboratório, para estudar, pois o bolsista da COLAB não tem a permissão de usar o e-mail institucional pertencente a ele (devido a pessoalidade do endereço eletrônico).

Mas isso é feito, assinando em listas de papel seu nome, tendo o controle do pessoal que vem ao laboratório, e verificando se o usuário tem direito a pegar a chave para verificar a existência de reserva de horários, verificando se o usuário é bolsista de algum professor ou não.

Esse problema se resolve, pelo que observamos, apenas quando Abrantes imprimia o horário, porém enquanto não se fazia isso, era obrigatório chegar o Coordenador COLAB para que se pudesse, por meio de e-mail, verificar o horário.

Entre os fatos que citamos um deles (as informações estarem na caixa de entrada do Coordenador) geram dois problemas visíveis:

• Primeiro, o bolsista do professor utilizador do laboratório pode ficar esperando desnecessariamente, e ao mesmo tempo, sem ter acesso ao laboratório, caso o Coordenador da COLAB não tenha impresso a tabela de horários, já que outra pessoa afora Abrantes não tem acesso à caixa de entrada, por ser de uso pessoal.

• Depois, no pedido de reserva de horários feita de maneira não automatizada, pode ficar complicado gerenciar diversas reservas efetuadas por e-mail, onde a diversidade de assuntos das mensagens de e-mail, pode atrapalhar o gerenciamento das reservas, por estarem misturadas com mensagens que tratem de solicitações diversas.

### Entrevista após lançamento da primeira versão do sistema

Nessa entrevista mostramos a primeira versão do sistema rodando e executando a função de retirada de chave, pois essa função já estava pronta para exibição.

Ao perguntar novamente como seria a regra de negócio para a COLAB, perguntamos o que ele esperava do Sistema de Gerenciamento de Laboratórios (SGL), que estava em sua primeira versão para apreciação e avaliação por parte dele.

Falamos a ele que a função de retirada de chave estava funcionando corretamente, ao que ele falou que realmente quanto a função de reserva de horários era bastante complexa, segundo ele mesmo, sendo mesmo o coordenador do setor, que os horários eram bastante variáveis.

Essa função qual anteriormente ele demonstrava dúvidas em conversas informais se funcionaria de maneira mais prática ou não e se teria mesmo que colocar um por um, e escrever novamente os horários que os professores já submetiam no sistema de planilhas do Google Drive.

Falando sobre a lógica de funcionamento do setor, obrigatória de ser contemplada no software, Abrantes indicou que para reserva do horário o professor e bolsista do professor – os quais convencionamos chamar de usuários utilizadores ao momento da conversa - deveriam cadastrar uma senha para que o usuário administrador – outra convecção para conseguirmos mais rápidas conclusões na conversa que se referia aos servidores e bolsistas ou membros da COLAB – deveriam ter login e senha para identificar suas ações permitindo que se monitorasse o tráfego de dados no sistema e quem faria as entregas da chaves relacionadas no sistema.

Tanto o bolsista administrador do sistema quanto o servidor administrador do sistema – teriam acesso – ao cadastro de usuários mediante presença do professor, ou pedido por e-mail, essa última forma que unicamente Abrantes seria legitimamente autorizado a fazer, que não teria controle por meio do sistema, mas um controle humano e hierárquico do setor pelo para o qual estamos desenvolvendo o software.

Sobre a forma como seriam instruídos a se cadastrar os diferentes usuários utilizadores da COLAB, o servidor, o bolsista e aluno utilizadores, ficou assim definido:

* **O servidor utilizador:** Ele teria que se dirigir à COLAB e fazer seu cadastro por meio de qualquer um dos bolsistas ou servidores que estejam responsáveis ao momento de gerenciar o software. Caso ele não desejasse ir à COLAB, se dá a opção dele entrar em contato por meio eletrônico por meio de Abrantes, que se o servidor opte fazer dessa maneira, unicamente Abrantes, que recebeu a correspondência, poderia cadastrar. Mesmo o servidor de outro setor que usasse ocasionalmente o laboratório teria que fazer o cadastro, que é simples, e essa seria uma medida adotada, pois segundo o Coordenador da COLAB, é necessário haver responsabilidade de alguém sobre a chave, pois se houverem objetos perdidos durante o uso do laboratório tem que se pedir que quem usou o laboratório preste contas sobre o que aconteceu.
* **O bolsista utilizador:** Ele para poder se cadastrar tem que se cadastrar com a presença do professor, ou se dirigir até a COLAB, mesmo sem a presença do professor, cadastrando Abrantes esse aluno ao verificar existência de e-mails do professor solicitando o cadastro do referido bolsista, pelo qual ele é responsável. O Coordenador do setor em questão falou que é necessário absolutamente a presença do bolsista para que os membros da COLAB, saibam quem é, tanto para a criação de sua senha pessoal, que representa sua responsabilidade com a chave.  
  Abrantes falou que há uma corresponsabilidade da chave, quando um servidor autoriza um bolsista a retirar uma chave do laboratório, mas isso Abrantes disse que não era necessário cadastrar no sistema, preferindo acompanhar isso por uma lista impressa que mostraria a relação entre servidores e bolsistas utilizadores da COLAB, já que com o tempo, nem precisariam ver a lista, mas automaticamente saberiam se nunca um aluno tinha vindo ao laboratório, como também saberiam quem era o servidor responsável por aquele bolsista, sobrando a lista, quando isso fosse esquecido.

Só seria necessário o cadastro do aluno como bolsista, se fosse cadastro por um professor, que reserve atividade regular nos laboratórios.

Quanto ao cadastro de saber a qual área o bolsista pertence para realizar suas atividades no laboratório específico, Abrantes afirmou não ser necessário, pois isso pode e ele considera melhor não fazer o controle automatizado, mas ele prefere ser o próprio controle disso. A informação de se poderia ou não realizar o empréstimo de chave de determinado laboratório a um bolsista, seria descoberta por saber quem é o orientador desse bolsista, associado a atividade que o laboratório está reservado na tabela de horários ser realizada. Caso não se lembrasse de quem seria o orientador do bolsista, a folha relacionando os orientadores aos seus respectivos bolsistas poderia ser verificada.

* **O aluno avulso:** Esse é o grupo dos alunos que ocasionalmente tenham necessidade de usar o laboratório, e queiram desenvolver uma atividade. Porém, isso pode ser conseguido apenas com o pedido do servidor por reserva, que seja quem realize a solicitação pelo horário (pois como dissemos somente os professores podem realizar essa tarefa). O aluno não é cadastrado, pois não é usuário constante. O servidor é unicamente responsável por esse usuário, portanto, para o servidor melhor seria autorizar os alunos a irem pegar chave nos laboratórios se fosse para atividade mais duradoura, que exigiria que ele fosse cadastro como bolsista e se permitisse maior controle sobre as atividades do aluno.

Quando foi apresentada a funcionalidade de histórico, Abrantes na conversa que foi realizada após a apresentação da funcionalidade, afirmou que seria necessário que a página de histórico, listasse os dados de quem pegou a chave e quem autorizou, qual o laboratório, qual o horário de entrega e o de devolução (se devolvido).

Dessa funcionalidade, já estava em operação no momento a realização do registro do nome de quem autorizou a chave, seguido do horário de entrega, nome da pessoa a quem se entregou a chave. Faltavam portanto implementar o horário em que se devolveu a chave e qual membro da COLAB que a havia recebido de volta.

Ele comentou que a fim de se saber em caso de facilitação indevida na entrega de chaves por meio dos bolsistas administradores do sistema, se deveria registrar o nome do bolsista que deixou a pessoa se retirar a chave. Para ele ser punido, se, por exemplo, se aproveitasse da autorização de acessar o sistema para fazer login no sistema e autorizar uma chave a um bolsista colega fora de seu horário de trabalhar no referido setor.

Essa ação indevida se verificaria pelo histórico de retiradas de chave, que se feitas fora do horário de entrega da chave, se notaria como atitude errada do bolsista, quando Abrantes verificasse o histórico de chaves no sistema e verificasse um registro fora do horário de atuação do bolsista administrador do sistema.

Já o registro do nome de quem entrega as chaves tem a utilidade, segundo tivemos informação na entrevista, de saber quem foi que saiu com a chave durante o sumiço de um objeto, ou quem estava responsável por uma chave não devolvida.

Quanto ao cadastro de quem recebeu uma chave, o coordenador do setor para o qual estamos desenvolvendo o software, relatou que seria necessário para saber quem foi que deu baixa na chave, caso se tenha perdido ela, mesmo que registrado que ela estava disponível.

Sobre a importância de se manter as informações desse histórico, Abrantes disse ser importante para rapidamente saber quem foi que não devolveu a chave em outro turno, e também ter noção do que fizeram os seus colegas de outro turno quanto a esse gerenciamento de chaves, para ele poder ter facilidade de compreendendo o que os outros membros da COLAB fizeram, saber o que fazer para concluir as tarefas não completadas, como a devolução de chave.

Sobre quem vai solicitar a chave, além dos campos adicionais de hora de devolução da chave e nome de qual membro da COLAB recebeu a chave devolvida, ele pediu que no momento da reserva tivesse um campo para selecionar qual a atividade desenvolvida (pesquisa, extensão, reunião, prática profissional, aula extra), amarrando a algumas atividades fixas, algo que Abrantes destacou, bem provavelmente preocupado que se fugisse para outras tarefas não adequadas ao propósito para o qual existem os laboratórios.

Sobre a discussão de ele e os demais servidores poderem realizar uma autorização especial, logo voltou atrás, ao pensar no acúmulo de tarefas, que segundo ele, são de grande responsabilidade pelo fato de gerenciar 16 laboratórios.

Ele bem respondeu a própria questão, dizendo que na verdade seria mesmo melhor cadastrar um servidor utilizador do laboratório, atividade essa – o registro desse servidor utilizador – às quais ele e os demais membros da COLAB tinham acesso.

Ele perguntou sobre uma página para cadastro de novos laboratórios, algo que lembrou de me perguntar ao ver que novos laboratórios poderiam surgir e me dizer que haviam 16 laboratórios.

Teve dúvidas a respeito de se poderiam ou não acessar os membros da COLAB que trabalhavam distante da sala da Coordenação dos Laboratórios no prédio anexo, ao que respondi que sim, se conectados devidamente a rede, pois estaria o sistema hospedado no servidor do Campus Natal Central.

Ele disse com convicção que todas as salas já tinham acesso a rede no novo prédio, e a preocupação que lhe restava era se, pelo fato de estar hospedado no Campus Central, os professores do Campus Central poderiam acessar o sistema, ao que, prontamente respondi que não, pois estaria o sistema protegido por senha e os dados depois poderiam ser criptografados – ao ele ver o Workbench do banco dados listando as tabelas, ele percebeu que as senhas estavam visíveis no banco de dados, ao que precisei garantir que os dados seriam criptografados.

Sobre o fato dos usuários administradores do sistema, surgiu a discussão do que fazer com os bolsistas da COLAB que se deligassem do IFRN, e, consequentemente, da COLAB. Ele pensou em excluí-los da tabela de usuários, mas eu comentei que isso geraria problemas ao tentar filtrar no histórico dados de movimentações recentes, e se fosse necessário saber quem foi que ficou com uma chave entregue pelo recém saído bolsista, não seria isso mais possível, portanto o aconselhado seria desativar (retirar a permissão do usuário de gerenciar as chaves, porém isso é uma tarefa que exige direitos administrativos, algo que até então Abrantes tinha descartado, mas a partir da necessidade de desativação de usuários, se fez algo necessário.

# DESENVOLVIMENTO

A etapa de desenvolvimento de nosso projeto de software contou com o auxílio do professor Diego, já definido como nosso orientador de TCC durante o período da implementação do software.

Durante o início da implementação do software, tínhamos definido que iríamos mudar a visão de desenvolvimento do software, que passaria a ser desenvolvido com outras prioridades que foram aceitas pelo cliente, e na verdade consideradas como um software melhor atendesse a COLAB, pelo que foi falado com Abrantes, com quem nosso orientador, Diego, conversou, e nós também.

A nós, ele falou que seria melhor atender a necessidade de substituir os registros de papéis do que focar na implementação de um sistema de solicitação de horários, uqe é algo menos frequentes, visão defendida pelo nosso orientador e o cliente, o coordenador da COLAB, já que é algo que é menos frequente, e ocorre apenas no início do semestre letivo, e quanto a utilização de automação para fazer a solicitação desses horários, isso é algo até já feito por meio de utilização de algum software, pois, pelo menos entre a coordenação do curso de informática e os servidores da COLAB, está combinado utilizar o sistema de planilhas do Google para trocar as informações de quais são os horários desejados, tendo o servidor Abrantes, conforme tinha nos dito, acesso à edição da planilha online, podendo fazer exclusão caso não tenha como atender aos horários que seja pretensão os professores utilizarem.

O acordo, portanto da solicitação não é o maior problema, já que o acordo se faz por meio eletrônico, não existindo o mesmo problema ambiental, financeiro e organizacional existente no processo de retirada de chaves dos laboratórios, que é frequente, e obrigatório para concretizar todas as operações em que se realizem atividades nos laboratórios: se necessita se assinar em papéis, não sendo informação tão organizada como os horários, que embora mudados frequentemente às primeiras semanas dos semestres, são algo menos frequente, e que não trazem tão constantemente à tona a problemática que queríamos antes resolver pelo software, sendo racionalmente visível ser mais prioritário substituir a pilha de papéis, que atrapalha o processo de busca de informações, que devem ser regularmente prestadas à direção, sobre quem utilizou os laboratórios no período no qual a direção quiser se informar sobre as atividades supervisionas pelo pessoal do setor para o qual estamos desenvolvendo o software.

O cliente antes até pensava em descartar o registro de retirada de chaves dentre os requisitos funcionais do aplicativo web, provavelmente por crer não ser possível nem prático se fazer esse tipo de tarefa no sistema, o que talvez, ele julgasse ser a forma mais lenta de executar essa atividade, não seria prático, ou que não seria possível conter todos os dados, como a limitação da impossibilidade de adicionar a assinatura da pessoa existente na folha de papel, em substituição da tal, sugerimos colocar no software, um recurso de assinatura digital, uma senha para acesso da chave, à qual seria de caráter pessoal, imputando responsabilidade pela chave que fosse retirada do sistema, por meio dessa autenticação.

## Início da implementação do software

O esboço da primeira versão incompleta do software foi desenvolvido principalmente por Nathan em companhia de um de monitor de informática da instituição, Carlos Henrique ainda estando inoperante e sem conexão ao banco de dados, tendo praticamente implementada a parte visual do sistema, as folhas de estilo para definição do design do software, como a estrutura em HTML, das ainda poucas páginas da primeira versão do software.

Após isso para inserir funcionalidades a esse esboço, Diego Oliveira, orientador do projeto iniciou as atividades juntamente conosco concretizando a primeira versão operante do software, inserindo ele mesmo linhas de código para a conexão da aplicação ao banco de dados, em nossa presença, implementando já um exemplo de como fazer com que dados capturados das telas JSP, pudessem ser tratadas como objetos Java, manipulados por métodos das classes do sistema e salvos no banco de dados também por métodos de inserção no banco de dados.

As telas implementadas e deixadas em perfeito funcionamento foram a retirada de chave, que já salvava no banco de dados da aplicação (o que é solicitado pela interface web), selecionando primeiramente qual o professor que iria retirar a chave e depois o laboratório que ele solicitava a chave para usar.

Foi realizado o salvamento das informações no banco de dados conforme os parâmetros enviados pela aplicação web, de qual usuário do laboratório que retirava a chave do, de qual membro da COLAB teria autorizado a chave, a gravação do horário atual da máquina servidor do sistema.

Além disso foi feita a exibição do ID do membro da COLAB logado.

Para captura dos dados de usuário do sistema que retirava a chave, foi criada um tag <SELECT></SELECT> do HTML, que tem como seu conteúdo impresso vários elementos de tag <OPTION></OPTION>, em mesmo número que o dos professores com seus nomes salvos no banco de dados.

Cada um dos <OPTION></OPTION> foram desenhados por uma repetição de um laço for do Java que repetia o mesmo número de vezes que o número de elementos ArrayList, que continha os mesmos elementos que o banco de dados possuía, e mesmo número, também.

Um método de retorno ArrayList de objetos das classes Professor e Laboratório de nomes idênticos aos das tabelas do banco de dados, eram criados em cada um dos métodos que retornava para o servlet da página, (gerado automaticamente, pois se estava desenvolvendo com scriptlet) os ArrayList de objetos de Professor (ArrayList<Professor>) e ArrayList de objetos de Laboratório (ArrayList<Laboratorio>) por meio dos respectivos métodos: public ArrayList selectAllProfs () e public ArrayList selectAllLabs.

Ao ocorrer de o servlet da página receber esses valores, pela declaração de duas variáveis ArrayList no scriptlet da página JSP, que recebem o retorno dos métodos correspondentes ao retorno dos dados dos professores e dos laboratórios, se escreveu o restante da estrutura da página (os SELECTS E OPTIONS anteriormente citados) que conteriam os dados escritos por scriptlets com códigos de impressão das variáveis de instância dos objetos contidos no ArrayList retornado por seu respectivo método de leitura do banco de dados.

O primeiro laço do for que escrevia dentro do SELECT do nome dos professores por meio do método getName() definido para retorno da variável de instância “nome” de objetos Professor: o nome do professor contido no primeiro objeto, que seria o primeiro elemento do ArrayList a ser percorrido e escrito pelo comando contido no scriptlet.

Cada laço escrevia um option com nome de exibição com o nome do professor selecionado. A mesma lógica era usada para escrita no nome do laboratório.

Porém, além de exibir esse dado, era também escrito um atributo do <OPTION></OPTION>, o value que era igual ao ID do professor, pois, no banco de dados, o salvamento do histórico de retirada de chaves não tem o nome do professor salvo na tabela do histórico, mas o campo “id\_professor”, que salva um número associado a ele, pois seu nome poderia não ser único, mas o ID é único e exclusivo. Essa referência pelo ID é mais que suficiente para retornar os dados do professor no histórico requerido pela função de histórico, pois por meio do ID como chave estrangeira, é possível retornar o valor do campo nome da tabela que contém como chave primária o “id`\_professor”.

Sendo o atributo value definido no <OPTION></OPTION>, o name é definido no <SELECT></SELECT>, onde era possível pelo nome e valor os dados os dados serem enviados pelo formulário em que os dois SELECTS estavam presentes por método POST á própria página, sendo executada uma consulta SQL para inserir os valores enviados pelo campo do formulário.

O formulário envia os dados à própria página de empréstimo, sendo eles salvos e executados os métodos de inserção do histórico no banco de dados, se eles não forem nulos.

Além de ser implementada a tela de empréstimo, foi também feita por nós, juntamente com o professor Diego, a tela de login, por meio do envio do atributo dois campos de dois campos, um <INPUT type=”text”> e outro <INPUT type=”password” pelo seu respectivo formulário à própria página, onde uma variável definida em seu scriptlet salvava uma variável com o ID do membro da COLAB, impresso na página de solicitação do sistema, e colocado como entrada no formulário de empréstimo como <INPUT type=”hidden”>, servindo para salvar o valor do ID do membro da COLAB na coluna histórico.

O campo de senha e matrícula do membro do setor para o qual estamos desenvolvendo o software era verificado, e também salvo uma variável acessível a qualquer outra classe, que poderia ser usada para um scriptlet chamar para imprimir dentro de um formulário, para poder se referir a quem se deseje atribuira as ações do sistema, como emprestar chaves e dar baixa na devolução.

Além dessas duas funções, que aqui se referimos, não havia outras funções do sistema em funcionamento.

Depois, realizamos a implementação da funcionalidade de histórico de reservas que por meio também de método retornamos um ArrayList<Historico>, cujos objetos da classe histórico continham o nome do usuário de laboratório que retirou a chave, nome do membro da COLAB que retirou a chave, laboratório, data e horário de entrega da chave.

O histórico escrevia dentro das várias tags TD os valores das variáveis de instância dos objetos de Historico presentes no ArrayList.Cada <TD> possuía um a impressão de um valor por meio dos métodos de acesso da classe histórico.

Cada <TD> era impresso por um código semelhante a este:

* out.println("<td value=" + historicos.get(i).getUsuario() + ">" + historicos.get(i).getUsuario() + "</td>");

No exemplo acima demonstrado vemos um código que estava dentro do scriptlet presente da tag <TABLE></TABLE>, que imprima um <TR></TR>, com 5 dessses TDS, cada um retornando uma informação relativa ao histórico.

Esse exemplo acima é um dos 5 valores que era impresso na tabela histórico, que no caso era o nome do usuário da COLAB que era definido na classe Historico como a variável de instância “usuario”, acessada pelo método getUsuario. O “ArrayList<Historico> historicos”, como é um Array de objetos, está retornando do objeto número i (conforme o número da iteração), somente a variável de instância usuário para impressão no TD que colocamos na COLUNA que exibe o nome dos membros da COLAB.

Criamos, executando também o método println os demais TDS com dados relativos às demais colunas impressas, conforme o nome de seu método de acesso *get* definido pela classe os chamamos: professor, laboratório, data e horário do empréstimo.

O menu da aplicação foi implementado utilizando um arquivo JSPF, um fragmento de JSP, que foi inserido acima do nome em letras garrafais do sistema, SGL (Sistema de Gerenciamento de Laboratórios).

Essa prática foi necessária, devido ao fato de sermpre se inserir uma página, e como chegava ao ponto de aplicação estar aumentando o número de páginas, o novo botão do menu tinha que ser digitado de página em página, fazendo com que ele

## Segunda versão do software

Na segunda versão do sistema, para tentativa de entrega em tempo hábil ao cliente, e para poder apresenta-lo a tempo de defender o trabalho e apresentar o software à banca, o professor Diego nos auxiliou, mesmo durante as férias, completando algumas funcionalidades.

Nessa segunda versão do software foi feita a devolução de chaves

A devolução de chaves teve sua ação implementada no HTML pela criação da estrutura da tag <SELECT></SELECT>, que semelhantemente à funcionalidade de empréstimo escrevia elementos das variáveis de instância do ArrayList, por meio da impressão escrevendo scriptlets com métodos de retorno das variáveis de instância de cada objeto. Cada <OPTION></OPTION> tinha como texto para exibição o nome do laboratório e valeu a ser mandado pelo formulário com o ID do laboratório. Como a forma de escrita desse <SELECT></SELECT> foi igual ao da tela de empréstimo, já basta foi suficiente o falado como foi escrita a tag SELECT na página JSP.

O ID do laboratório é mandado para um método, o qual fazia um UPADATE de um campo criado na tabela “professores” no banco de dados, que contém o valor de “id\_laboratorio” igual ao do laboratório para o qual se queira devolver a chave, e se definia ele como “0”, indicando na definição de nosso software que o usuário estaria sem nenhuma chave.

Além de atualizar esse campo também se atualizava o campo “status” da tabela “laboratórios”, de “ocupado” para “disponível”, como também um campo “id\_professor” que contém o ID do professor para 0.

Devido à implementação dessa devolução de chave, agora passou a ser necessário na tela de empréstimo a verificação de os laboratórios estarem disponíveis ou ocupados.

Agora, então o método de retorno “ArrayList<Professor>” deveria passar a filtrar somente os resultados onde a coluna “id\_laboratorio” possuísse valor igual a 0, representando que esse professor poderia retirar uma chave, já que teria devolvido a anterior.

Também na tela de empréstimo, passaria a haver a necessidade de o laboratório ser filtrado e retornado o objeto correspondente às suas informações, se ele não possuísse chave retirada, o que em nossa aplicação acontece se o laboratório for marcado em sua tabela do banco de dados como “disponível” na coluna “status” e com o campo “id\_professor” com valor igual a 0.

## Terceira versão do software

Foi a partir dessa versão que o cadastro de usuários pela interface se tornou disponível, e assim se criaram as primeiras entradas digitadas na aplicação, sendo diretamente inseridas no no banco de dados. Criou-se também a autenticação por senha exigida por parte do usuário do laboratório no empréstimo, ou seja, houve uma nova atualização na operação de empréstimo, devido a implementação da desativação do usuário, requisito que não havia antes se conseguido tempo para se implementar.

### Gerenciar Usuário

Para desativar o usuário e editá-lo, podendo acessar configurações como senha, criou-se a tela de Gerenciamento de Usuário dos Laboratórios, onde se desenhou numa tabela, semelhante ao feito com a tabela de Histórico das Chaves, a exibição dos usuários de laboratórios que estão cadastrados no sistema, colocando

Houve a criação de um método para verificar se o usuário estaria ou não desativado para exibir a lista de empréstimo, colocando em métodos println a chamada dos métodos de retorno de variáveis que instância de um objeto Professor, de objetos salvos num ArrayList<Professor>, que contém todos os professores sem qualquer filtro de estar com chave ou não, listando simplesmente todos os usuários presentes no sistema.

Em cada <TD></TD> desenhado pelo método println, colocamos um retorno de variável para ser exibido na tela como conteúdo visível do TD. Depois de estar funcionando a visualização do nome dos usuários, projetamos os botões de edição e desativação/ativação dos usuários.

Esses botões eram do tipo <INPUT type=”submit”>, e enviavam os dados por meio de um <FORM> criado dinamicamente.

Sobre os <INPUT type=”submit”>, colocamos eles dentro dos TDS, sendo obviamente não visualizáveis, e neles colocamos todos os dados de cadastro de usuário.

Como não podem ser criados formulários dentro da tabela, conforme vimos num fórum da internet que “um **form** não é permitido ser um elemento filho de uma **table**, **tbody** ou **tr**. Você pode ter uma tabela inteira dentro de um formulário. Você pode ter um **form** dentro de uma célula do **tabel**. Você não pode ter parte de uma **tabel** dentro de um **form**.” (FORM, 2014, Tradução nossa, grifo nosso)

Portanto, para colocarmos uma linha da tabela com um form dentro não seria possível, devido não poder um <FORM> não poder conter um recorte de um <TABLE>, como uma linha (<TR>) dessa tabela, então tínhamos que achar outras soluções para colocar esses vários <INPUT type=”submit”> e <INPUT type=”hidden”> dentro de cada linha da tabela, como pertencentes a um <FORM> diferente cada.

Utilizamos, uma solução encontrada no mesmo fórum da internet em que se indicou o problema:

Use o atributo "form", se você quiser guardar a sua marcação:

<form method="GET" id="my\_form"></form>

<table>

<tr>

<td>

<input type="text" name="company" form="my\_form"></td>

<button type="button" form="my\_form">ok</button>

</td>

</tr>

</table>

Portanto, para implementar o botão de desativar o Colocamos o atributo “form”, que permite se inserir um elemento de um formulário dentro de uma tabela.

Isso fez criamos os elementos com a seguinte sintaxe:

* <input style="visibility: hidden;"name="id" form="id<%=i%>" type="hidden" class="form-control" value="<%=profs1.get(i).getId\_professor()%>">

Colocamos o atributo **form** com o nome do formulário que ele pertence como “id<%=i%>”, devido a estar dentro de um laço **for** que desenha uma linha da tabela a cada ciclo de repetição do laço. Sendo assim o **form** atribuído aos <input type=”hidden”> e ao <input type=submit> da primeira linha seria “form1”, aos da segunda linha “form2”, e assim sucessivamente.

Foi necessário também criar um laço **for** além do que criava a tabela, para criar os diversos **forms**. Como o laço for escrevia linhas de acordo com o tamanho do ArrayList, que o Java retorna com um método padrão size, já existente na linguagem Java, chamamos o método para calcular o tamanho do Array para definir as repetições de nosso laço for, chamando no nosso chamado como “profs1.size” (já que era ArrayList de objetos de Professor), criado diversos <FORMS>.

Por esse código criamos os **forms** relacionados aos inputs declarados dentro dos TRS do <TABLE>:

<form id="id<%=i%>" action="editarCadastroUsuario.jsp" method="post" accept-charset="iso-8859-1,utf-8"></form>

Com esse código criamos **forms** criados e depois chamados conforme os nomes nessa sequência: “id1”, “id2”, “id2”. Esse são os **forms** que são atribuídos aos inputs de cada linha da tabela, assim garantimos que todos

Já os atributos “name” de todos os **inputs** era igual, quer estivessem na primeira linha ou última da tabela, o que não causava dúvida sobre qual dos nomes de professor mandar, se o da primeira linha, se o da segunda, se o da terceira, já que estavam atribuídos os formulários por ID, não confundido de maneira alguma o <INPUT type=”submit” qual dos nomes” devia mandar. Testamos e exatamente os nomes corretos eram passados relativos aos dados do mesmo usuário para a outra página, tanto usando método GET, como POST. Porém, por estarmos passando campo de senha para edição e salvamento noutra página, consideramos adequado mandar por POST.

Foram mandados os campos do **form** que enviavam valor do ArrayList<Professores> profs1 por método POST para a Página de Edição de Dados do Usuário, que recebe esses dados exibe em caixas de texto de um formulário que permite edição e salvamento. Criaram-se variáveis tipo String criadas em scriptlet existente na página de Edição do Usuário, atribuindo os valores enviados por POST, recebendo-os usando o método getParameter.

Esses dados sendo ou não editados, ao serem submetido com o botão <INPUT type=”submit”> da página de Edição do Usuário, são enviados por POST para outra página, a Página de Confirmação de Gravação de Alterações, pelo fato de os **inputs** conterem o mesmo mesmos valores de atributo “name”, enviando-os para outra página para evitar falhas e manter o código das páginas do sistema organizado.

### Cadastro de usuário

O cadastro de usuários foi implementado de maneira parecida com a tela de edição e alteração dos usuários do sistema, mas diferente do que lá ocorre, o método que trata os dados enviados por POST (pelo salvamento da senha), os dados são enviados por POST para a própria página, já que não tem como haver nenhum conflito, por essa página não ter nenhuma variável estar esperando recebimento de dados de outra página, não tendo como cometer erro com a rotina com a qual proceder.

Os dados sendo enviados por método post, são recebidos por método **getParameter**, sendo então enviado por parâmetro a um método que cria uma conexão com o banco de dados, e os salva, realizando um INSERT na tabela de usuários, que salva os membros da COLAB. Segue-se essa rotina também com a inserção de valores na tabela “laboratórios” e “professores”.

### Autenticação por senha

A autenticação por senha foi na verdade uma inclusão ao processo de empréstimo de uma assinatura digital, que confirma que a chave foi conscientemente pega pelo usuário de laboratório.

Para implementar essa tarefa foi implementado num aninhamento de laços IF-ELSE que recebe ao ser feito o **submit** do formulário da página de empréstimo a matrícula e senha da pessoa e excuta testes, primeiramente, verificar se a entrada dos dados é nula (o que pode acontecer se alguém conseguir pegar o link direto para a página) para poder avançar até o salvamento dos dados no banco. Depois verifica o login e senha do usuário, e se a senha do usuário estiver errada, se reencaminha para a página Senha Incorreta, que tem opção de voltar a página de empréstimo e ir para Página de Gereciamento de Usuários, de onde se edita todos os dados do sistema, inclusive senha. Depois, se verifica se o usuário não está ativado, e caso não esteja, será reencaminhado para a página de Usuário Desativado, onde existem as mesmas opções de da página de Senha Incorreta. Finalmente, não sendo considerado desativada ou com senha incorreta, a autenticação é aceita.

## FRAMEWORKS UTILIZADOS

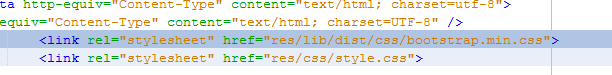
Algumas ferramentas já possuem soluções prontas para agilizar o desenvolvimento de softwares. No nosso caso, os frameworks foram utilizados principalmente na parte da parte visual do sistema.

### Bootstrap

Esse site disponibiliza uma série de folhas de estilo CSS e arquivos Javascript com predefinições prontas para ser utilizadas. No site do Bootstrap, encontramos exemplos de como utilizar as opções de estilo existentes no site.

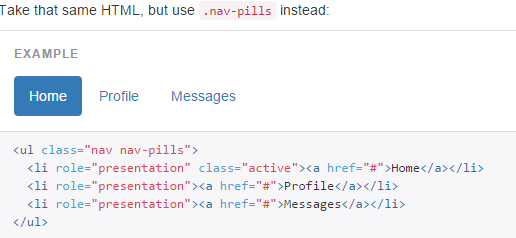
A pessoa que deseje utilizar o Bootstrap tem apenas que baixar e importar os arquivos CSS e Javascript baixados do site do Bootstrap.

Figura 7: Demonstração de importação de arquivo de folha de estilo CSS contendo class do Bootstrap



Fonte: *printscreen* de aplicação Netbeans, no sistema operacional *Windows 8.1*

Figura 7: Exemplo de uso de uma classe utilizada no nosso projeto



Fonte: *BOOTSTRAP, 2015*

Esse é um exemplo de um menu, que foi utilizado por nós em nossa tarefa de desenvolver o software. Esse estilo está salvo na folha de estilo do Bootstrap com o nome “nav nav pills”, e simplesmente colocando o arquivo CSS no computador e importando-o às nossas páginas JSP é possível utilizar todos os diversos elementos com o visual apresentado no site de exemplos de uso do Bootstrap.

### CSS Table Generator

Essa ferramenta é útil para desenhar tabelas, onde ela permite definir cores de contraste das linhas tabela, que é um recurso que ajuda o usuário a ler as linhas sem confundir os dados de uma linha com as outra.

Já para o desenvolvedor, o lado bom é evita-se para quem está produzindo o software de gastar tempo pesquisando os códigos das cores e propriedades CSS para aplicar na tabela.

O site dá a opção de salvar os códigos de estilo em código HTML, ou CSS.

Essa aplicação web também consegue gerar uma classe CSS com estilo para ser gerado

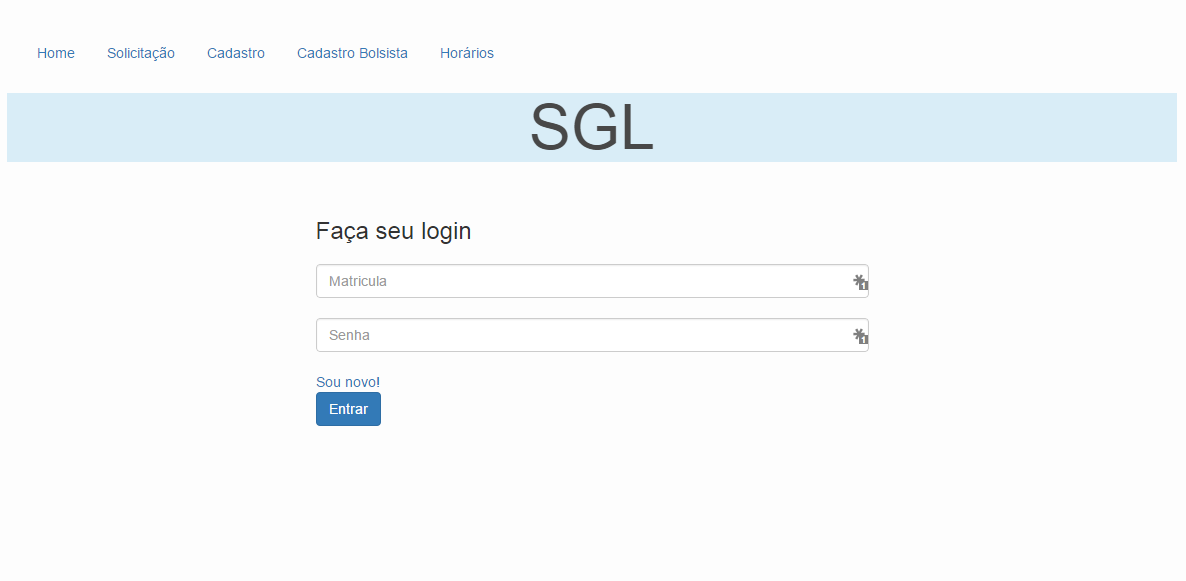
|  |
| --- |
| Figura 7: Opções de ediçãopresentes no site CSS Table Design para contruir a parte visual de tabelas |
|  |
| Fonte: *CSS TABLE GENERATOR, 2015* |

## Telas da aplicação

As telas que desenvolvemos da aplicação foram desenvolvidas num processo gradual, tendo o aperfeiçoamento após as iterações com o cliente. Registramos elas em torno das principais funcionalidades do software. São as funções de Login, Empréstimo de chave, Verificação de Status dos Laboratórios, Histórico de Chaves. Cadastros de Usuários, Gerenciamento de Usuários do Laboratório e Logout, as quais tem nas próximas subseções suas funções explicadas a nível do cliente.

### Tela de login

Figura 9: Tela de login de nosso sistema



Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

Essa tela possui um <INPUT type=”text”> e um <INPUT type=password>, presente num formulário que envia os dados para a própria a página por método POST os valores dos campos.

Essa página em seu código JSP tem uma chamada a um método que verifica a senha e redireciona à página de empréstimo, o que ocorre caso o membro da COLAB acerte a senha. Porém, se ele errar a senha, irá ser redirecionado a uma página que avisa que ele digitou a senha incorreta.

O tipo de senha embute o valor digitado, e o método post, evita rastros muito fáceis de levar a se descobrir de outro usuário, conforme vê-se digitado abaixo na captura de tela do formulário:

Figura 7: Tela de login ao ser preenchida e visualização dos campos com caracteres de senha escondidos

Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

### Empréstimo

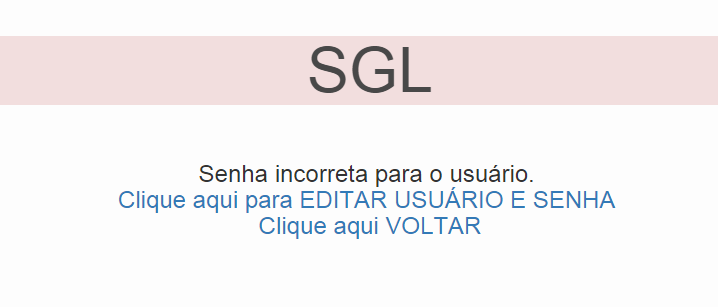
O usuário do laboratório para acesso a chave necessita digitar a senha que seria uma espécie de assinatura digital e substituiria a assinatura em papel.

Figura 8: Tela de empréstimo chaves

Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

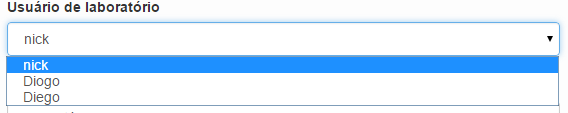
Caso ele cometa algum erro na digitação da senha ele é reencaminhado para uma página de erro, em que informa que ele tem as opções de voltar à página para tentar digitar novamente. Isso é feito, é uma prática de segurança para evitar erros, já que ao clicar no botão de voltar no sistema os valores definidos por método POST por request já estarão zerados.

Figura 9: Tela de erro para a senha de acesso a asos laboratórios



Essas opções dadas pela aplicação dão chance de bem facilmente o membro da COLAB localizar o que provavelmente seria seu desejo, como recuperar a senha do usuário

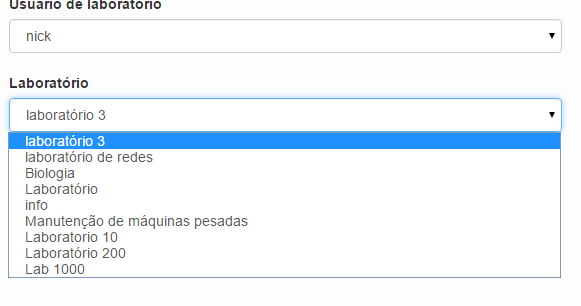
Figura 10: Seleção de usuário de laboratório que irá fazer a retirada de chave



Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

Essa seleção inclui apenas professores que estejam ativados. Usuários desativados nem mesmo aparecerão nessa lista, embora a autenticação desses usuários de laboratório não seja permitida, assim se procede para evitar que se perca tempo com um procedimento não permitido, como tentar emprestar chave a um usuário que tenha sido proibido de pedir chaves.

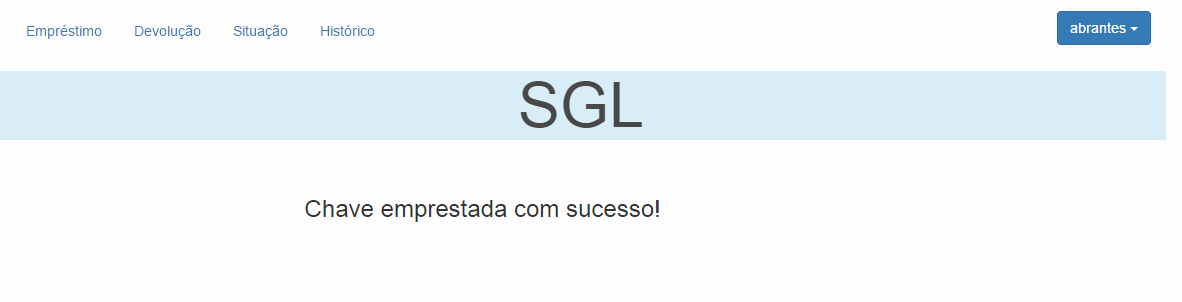
Figura 10: Seleção do laboratório para o qual se reservará a chave



Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

Dentre os laboratórios disponíveis para seleção e retirada de chave nessa lista, estão os laboratórios que não tiveram chaves emprestadas para nenhum usuário.

Figura 11: Após apertar no botão OK, os dados são processados e executado o empréstimo de chave



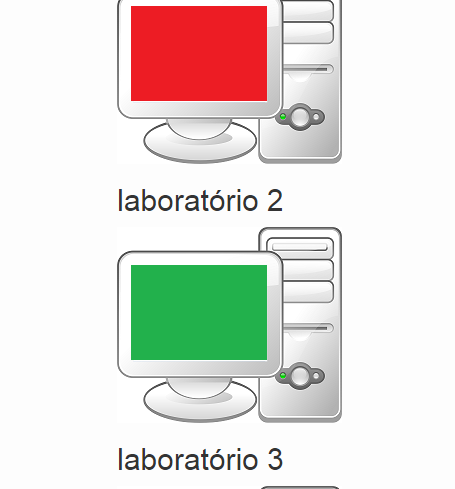
Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

### Situação dos Laboratórios

Essa é parte da aplicação que permite a leitura mais amigável do status dos laboratórios (ocupado ou desocupado), isso porque por meio de imagens se demonstra se o laboratório está disponível ou não.

A cor vermelha na no ícone do computador presente na tela capturada abaixo, representa que o laboratório está em uso por alguém. Já a cor verde que ele está disponível.

Figura 13: Laboratório 2 se encontra ocupado e laboratório 3 se encontra disponível



Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

### Histórico de chaves

É exibido um histórico de chaves que retorna por padrão das datas mais recentes às mais antigas de maneira geral, onde não se filtram as datas

Figura 15: Tela com exibição de horários



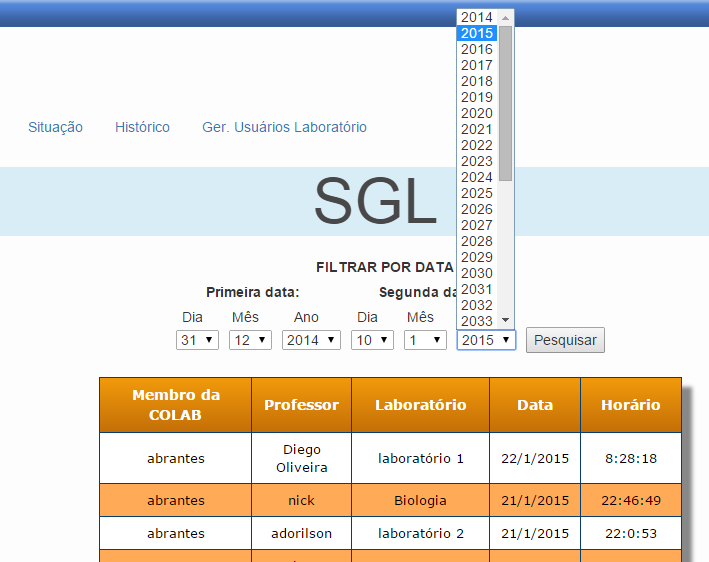
Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

Aqui está disponível para o membro da COLAB, um histórico de empréstimos de chave para o pessoal do setor ser capaz de gerar rapidamente um relatório, quando pedido que façam isso. Essa funcionalidade atende uma demanda, que é o pedido por parte da direção de históricos de pesquisas de uso da chave durante um período específico.

O pedido da direção desse histórico de pedidos de chave, é algo que foi citado por Vitoriano, enquanto foi entrevistado como o Coordenador da COLAB, enquanto ocorriam as férias de Abrantes.

Para filtrar o histórico, se deve informar as datas inicial e final como parâmetros de busca. Essa consulta retorna os 300 primeiros resultados, já que com o tempo a busca ficaria enorme podendo chegar a processar milhares de registros para serem impressos na tabela.

Figura 16: Seleção de horários é permida por meio de selects que fazerm parte de um formulário



Fonte: *printscren* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

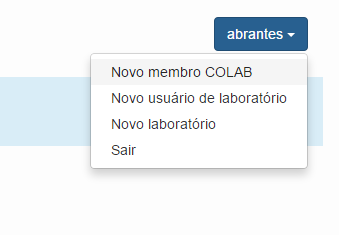
Essas consultas trazem um recurso que faz se tornar bastante útil se dar relatórios sobre as atividades feitas pela COLAB dentro de um período de tempo, dando a possibilidade de o setor atender pedidos emergenciais de relatório de atividades quando solicitado pela direção.

### Cadastro de usuários

A funcionalidade de cadastro de usuários pode ser acessada, clicando no menu superior direito de qualquer uma das páginas principais quando um membro da COLAB estiver logado.

Se deve selecionar “Novo membro COLAB para cadastrar novo usuário que gerencie o sistema e clicar “Novo usuário de laboratório” para usuário de laboratório que se queira autorizar para retirar chaves.

Figura 17: No menu lateral direito se dá a opção de aceder à tela de cadastro de pessoas que trabalham na COLAB



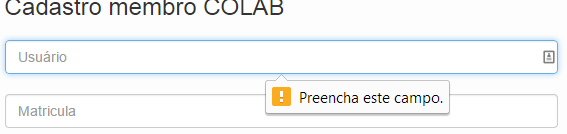
Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

Será aberto um formulário noutra página, em que se deverá inserir nome, matrícula, email e senha, sendo todos dados de inserção obrigatória.

|  |
| --- |
| Figura 18: Formulário que permite cadastrar um usuário que trabalha na COLAB |
| Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat |

A verificação de se os dados foram preenchidos ou não ocorre devido ao final da tag input colocarmos o atributo <REQUIRED>, que torna obrigatória a inserção dos dados assim marcados.

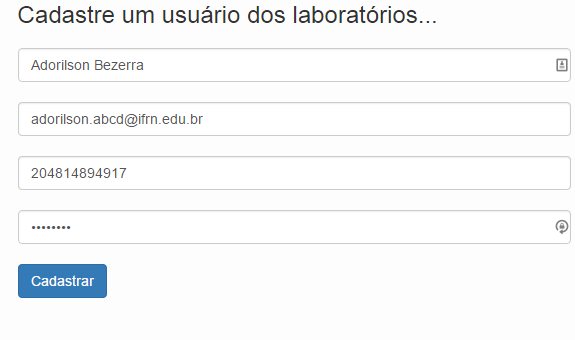
Figura 19: Campos não preenchidos retornam uma advertência a ser atendida antes de prosseguir no cadastro



Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

De maneira semelhante ocorre o cadastro dos usuários de laboratórios, que são usuários frequente e podem ser servidores da instituição ou bolsistas, por padrão.

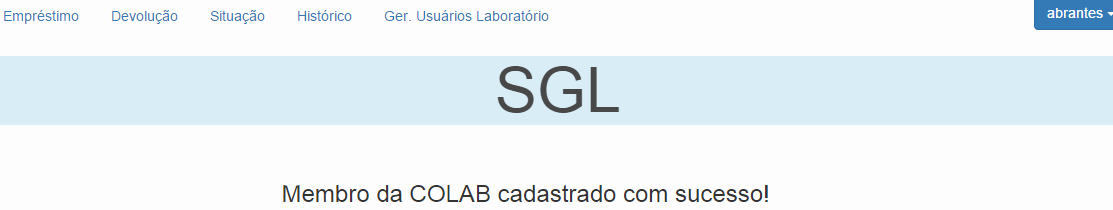
Figura 20: Cadastro de usuários de laboratório



Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

De maneira semelhante ocorre o cadastro dos usuários de laboratórios, que são usuários frequente e podem ser servidores da instituição ou bolsistas, por padrão.

Figura 21: Página para a qual o navegador é redirecionado, apenas para o caso de ter sido concluído o cadastro



Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

Se todas as informações forem preenchidas corretamente, o usuário terá seu cadastro efetivado.

É importante ressaltar que após efetivado o cadastro, ele está efetivado e disponível para realizar as respectivas atividades de seu perfil. Essa funcionalidade de cadastro foi pensada com algo a ser feito de maneira a possibilitar uso imediato da conta de usuário de laboratórios para retirada de chaves. Caso após algum tempo se queira ativar esse usuário, isso é possível indo na funcionalidade de gerenciar usuários de laboratório.

### Gerenciamento de usuário do laboratório

Essa parte da aplicação é necessária, pois é necessário desativar usuários que não possam mais ter acesso às chaves, como também eles possam ter suas informações editadas como senha, quando, por questão de esquecimento, for necessário editá-la, como também possa haver erros no cadastro do usuário.

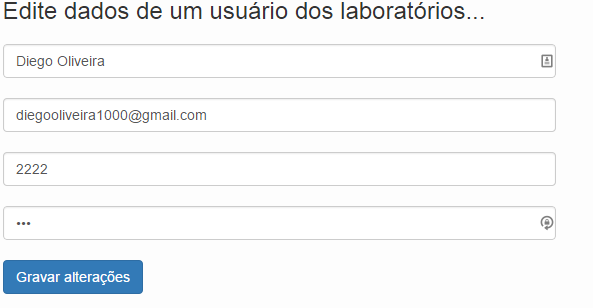
Figura 22: Essa outra tela de gerenciar laboratórios permite que os membros da COLAB editem os dados de usuários de seu sistema



Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

Os dados de um usuário de edição permitida, são exibidos. Senha e ID pertencentes a cada usuário, não poderão ser visualizados nessa página. O usuário clicará no botão editar corresponde ao usuário ao qual se deseje efetuar alteração.

Figura 23: É permitido aos membros da COLAB editar as informações vinculadas ao usuário selecionado na tabela da página anterior



Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

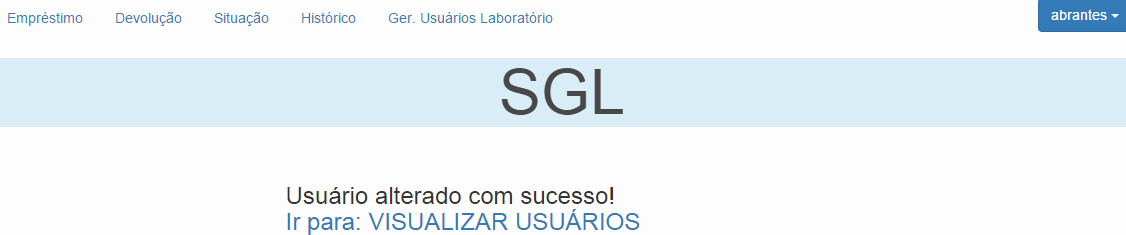
Caso o usuário tente editar sem submeter sem todos os dados estarem preenchido, não poderá utilizar o botão Gravar alterações, mas uma mensagem será exibida, travando o uso da gravação se o membro da COLAB apagar algum campo durante a edição.

Como se pode prestar atenção todos os dados já se encontram preenchidos na tabela de edição (são mandados da página de Gerenciamento de Usuários de Laboratório por método post), cabendo ao membro do setor para o qual estamos desenvolvendo o software deverá selecionar o campo que será editado. Portanto, não seria necessário redigitar todos os dados do usuário novamente para poder submetê-los ao banco de dados.

Para alterar um nome, adicionando um sobrenome, o membro do setor em questão deve selecionar o campo de nome e simplesmente adicionar o dado que seja necessário inserir, concretizando a alteração ao clicar em Gravar Alterações.

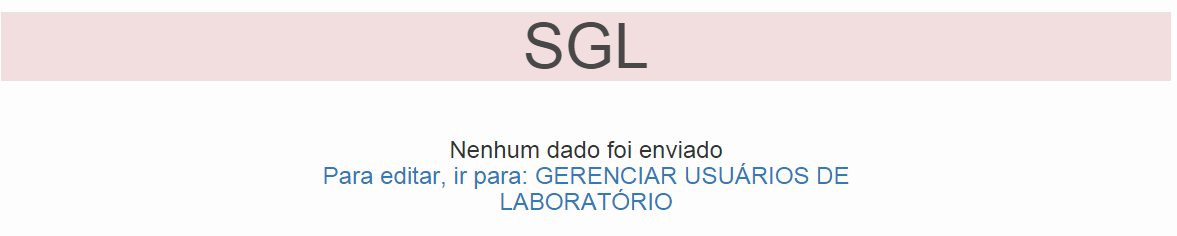
Ao clicar em Gravar Alterações, os dados são enviados por POST para outra página, que informa se os dados foram por meio de um teste escrito em código JSP, que verifica se foi recebido algum valor. Caso tenha sido recebido, ela confirma o recebimento dos dados.

Figura 24: Confirmação de alteração do usuário



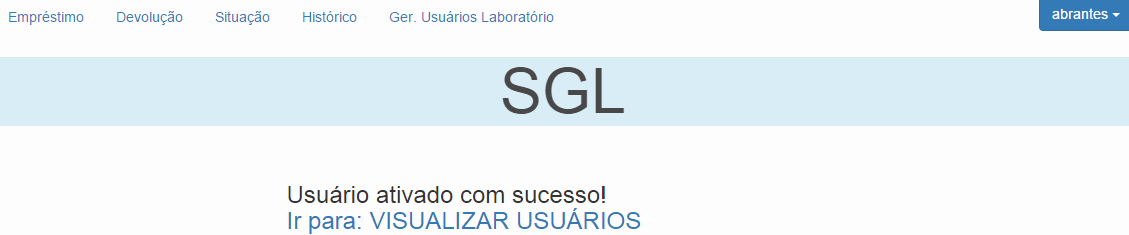
Essa tela demonstra

Figura 25: O usuário é redirecionado para uma página que avisa que não foi recebido nenhum valor enviado por outra página.

Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

Só se permite o acesso automatizado à página para que se evitem graves erros de inserção no banco de dados, já que dados na página como o id do usuário são selecionados automaticamente.

Figura 26: A página Confirmar Cadastro de Membro da COLAB avisa quando o cadastro é completado



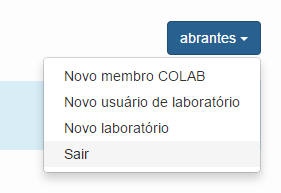
Fonte: *printscreen* feito no sistema Windows 8.1 com aplicação rodando a partir do servidor Apache TomCat

Aqui se dá a opção de voltar a tela que gerencia os usuários, permitindo que edite o registro de outro usuário, utilizando-se da página de Gerenciamento de Usuários.

### Função de logout

Para sair do sistema, o usuário deve apertar no botão azul no lado superior direito, que contém o nome do usuário.

Figura 27: Seleção do botão para deslogar do sistema, feito para uso dos usuários da COLAB



Esse botão redireciona ou usuário para a página logout.jsp, que nem aparece por tempo suficiente para permitir ser visualizada, de tão rápido rápido que o Java executa um código que “zera” uma variável que armazena os dados do login, deixando-a nula e permitindo que qualquer página com as funcionalidades do sistema que tente ser acessada não seja exibida, ao tentarem acessá-la com login, evitando que os usuários não logados

## Banco de Dados

Fizemos várias tabelas e campos para se adaptar às necessidades do sistema. Quatro tabelas principais foram criadas no Banco de Dados: Histórico, Professores, Usuário e Laboratórios, para armazenar e dar persistência à aplicação.

### Tabelas do banco de dados

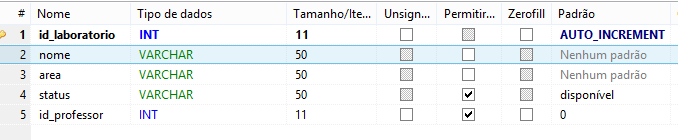
Utilizamos algumas tabelas, conforme abaixo citado para persistência da aplicação. As tabelas armazenam os membros que trabalham na COLAB, como os usuários dos laboratórios, os próprios laboratórios, e o histórico das operações de empréstimo de chaves.

As informações foram gerenciadas pelo banco de dados MySQL, e as entradas foram testadas no workbench HeidiSQL para visualização do resultado da inserção das queries, antes de coloca-las no sistema, e depois, no caso de erros terem sua existência notada na aplicação.

#### Tabela Laboratórios

O atributo área foi escolhido para representar se o laboratório o qual se estará utilizando será um laboratório de alguma área como de línguas, química, manutenção ou biologia.

Figura 28: Campos da tabela SQL “laboratórios”

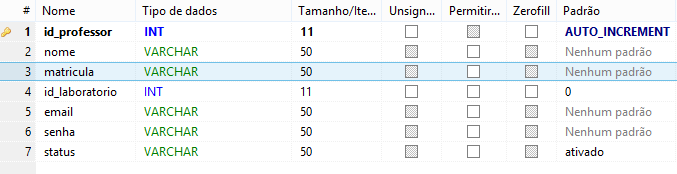


Fonte: feito a partir de *printscreen* de programa HeidiSQL, no sistema operacional Windows 8.1

#### Tabela professores

A seguinte tabela foi criada com os seguintes sete campos, listados na primeira coluna da tabela capturada no programa.

Figura 29: Estrutura da tabela professores

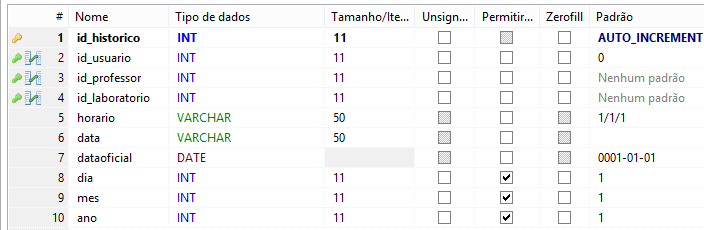


Fonte: feito a partir de *printscreen* de programa HeidiSQL, no sistema operacional Windows 8.1

O campo “id\_professor” que é a chave primária, por isso recebeu auto-increment, para que em momento nenhum a aplicação possa inserir o “id\_professor”, o que tende a provocar erros, já que deixar a aplicação com essa responsabilidade pode fazer que ocorram várias exceções

#### Tabela Histórico

Figura 31: Tabela histórico exibindo seus campos e seus tipos



Fonte: feito a partir de *printscreen* de programa HeidiSQL, no sistema operacional Windows 8.1

### Operações importantes realizadas no banco de dados

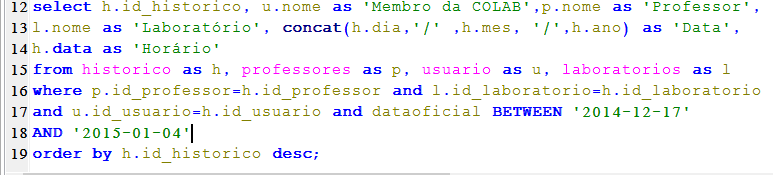
Realizamos várias consultas utilizando a linguagem SQL, que é uma linguagem de consultas em banco de dados universal. As consultas foram bastante utilizadas, utilizando métodos de conexão existentes nas classes de nosso projeto a fim de retornar a saída pretendida.

Listamos exemplos de uso da linguagem em nosso projeto.

#### Seleção de usuário de históricos de chaves

Uma consulta idêntica a essa foi utilizada para retornar os objetos requeridos pela aplicação:

Figura 30: Consulta foi utilizada para filtrar dados do banco de dados da aplicação por nós desenvolvida



Fonte: feito a partir de *printscreen* de programa HeidiSQL, no sistema operacional Windows 8.1

Nessa consulta temos por objetivo filtrar os dados contidos em uma data específica, que como apresentado na subseção de Telas de Aplicação, Histórico de Chaves, é inserida pelo usuário.

Na aplicação data é definida pelo valor de uma variável text que concatena dia, mês e não enviadas por parâmetro pela interface web.

Onde vemos a utilização da cláusula “as”, vemos implementados os aliases, que são apelidos usados para reduzir o tamanho do código, que deve ser digitado para executar a query e sendo bastante importante para o caso “duas ou mais colunas são combinadas em conjunto” (W3SCHOOLS, 2015, Tradução nossa)

### 

### 

# CONCLUSÃO

O que podemos verificar como produtivo nesse Trabalho de Conclusão de Curso é que a melhor maneira de trabalhar na programação de um software existente é estar a par da problemática enfrentada pelo cliente, manter um contato próximo dele e saber administrar com responsabilidade e agilidade o cumprimento dos compromissos quanto a cumprir os requisitos do software, para que o atraso não gere mais atraso e impossibilite a entrega do software.

Há sempre a necessidade de obter além do conhecimento da área de Tecnologia da Informação para desenvolver um bom software, sendo necessário ter o conhecimento teórico das tecnologias a serem utilizadas para resolver os problemas do cliente, porém sempre fazer isso ocorrer de maneira a ser o mais próximo do universo da vida prática para que o entendimento seja facilitado entre o cliente e o desenvolvedor.

Uma atualização dos conhecimentos pessoais sobre as linguagens computacionais é de bastante importância, para se ter certeza de que se está utilizando a solução mais eficiente e eficaz o exercício da programação.

A escolha das ferramentas utilizadas para o desenvolvimento é algo crucial para definir seu rumo em um projeto. A constante mudança de alguns requisitos não-funcionais da aplicação desenvolvida pode ser de efeito mais negativo mesmo que uma mudança de requisitos funcionais da mesma.

O colhimento das ideias antes da entrevista com seu cliente é tão importante quanto o trabalho de um repórter que antes entrevistar alguma pessoa, pesquisa sobre ela, para que mostre que sabe o que falar. O cliente achará menos produtivo o profissional de informática que trabalha na área de desenvolvimento, que não saiba captar e compreender bem os pensamentos do cliente direto de um produto.

A importância de aprender além de nosso mundo de conhecimento teórico é que sem o conhecimento prático, acaba-se por perder o teórico, sendo importante o profissional que entre nesse tipo de serviço tenha uma formação mais ampla e tenha o senso de descobrir como resolver problemas, que é a lógica mais importante para alguém se interessar por seu trabalho.

A produção do software com iterações no processo de desenvolvimento deu ao cliente maior engajamento na produção de ideias mais relevantes para o desenvolvimento de nossa função. O fato de ter algo em mãos para mostrar ao cliente, mesmo antes de tudo completo, é uma forma de agilização do processo do desenvolvimento, mostra um desenvolvedor aberto à cobrança, e ao contrário do pensamento de que seja uma forma de enrolar o cliente, querendo dar um ar de satisfação antes do software concluído, é uma chance de mostrar que você está trabalhando.

A linguagem de programação Java é muito complexa e possui várias úteis ferramentas, mesmo as mais interessantes ferramentas não devem nos fazer estimular-se à animosidade da curiosidade pelas novidades, devemos unicamente obter mais conhecimento da linguagem para nos auxiliar a focar no nosso propósito.

Um conhecimento mais aprofundado da linguagem SQL é muito importante para não gastar enorme quantidade de linhas de código para filtrar informações, quando o desenvolvedor do SGBD já fez isso por nós. Devemos se lembrar que quando trabalhamos para o cliente, por maior que seja nosso interesse de explorar mais da linguagem de programação de preferência para a realização de nosso trabalho, devemos ter primazia pelas escolhas corretas para a maior agilidade e qualidade na entrega de nosso software.

A virtualização de servidores é uma ótima maneira de testar seus aplicativos web. O Apache Tomcat, servidor de nossa escolha nos auxiliou a desenvolver com mais certeza do resultado da aplicação, é também uma boa maneira de apresentar a aplicação ao cliente, e utilizado junto com uma IDE como a de nossa escolha, o Netbeans, pode ser uma maneira bem rápida de testar a aplicação, onde a implantação automática dos arquivos JSP e recarregamento automático, tende a ser um bom passo para economia de tempo na criação da aplicação. Não se deve ter como de pouca importância as ferramentas desenvolvidas para acelerar o trabalho do técnico em informática.

É necessário

# REFERÊNCIAS

1. ALVAREZ, Miguel Angel. **O que é CSS**: Comentamos brevemente o que são as folhas de estilo em cascata e explicamos uma série de efeitos rápidos e interessantes que se pode fazer com elas. Disponível em: <<http://www.criarweb.com/artigos/173.php>>. [S.I.]:2004. Acesso em: 13 jan. 2015.
2. ALVAREZ, Miguel Angel. **O que é HTML**: Chegou o momento de falar sobre HTML, a linguagem com que se criam as páginas web. [S.I.]:2004. Disponível em: <<http://www.criarweb.com/artigos/7.php>> Acesso em: 06 jan. 2015.
3. ANCORADOR. **Saiba Como Surgiu o MySQL e Suas Funcionalidades**. Disponível em: < http://www.ancorador.com.br/internet/hospedagem/saiba-como-surgiu-o-mysql-e-suas-funcionalidades>. Acesso em: 24 jan. 2015.
4. BAIXAKI. **NetBeans IDE**: Ferramenta completa para desenvolvimento Java em serviços web e aplicações para celulares. [S.I.]:2014. Disponível em: <<http://www.baixaki.com.br/download/netbeans-ide.htm>>. Acesso em: 11 jan. 2015.
5. BOOTSTRAP. **Components – Bootstrap.** Disponível em: <http://getbootstrap.com/components/#btn-dropdowns>. Acesso em 25 jan. 2015.
6. CAELUM. **APOSTILA DESENVOLVIMENTO WEB COM HTML, CSS E JAVASCRIPT**. Disponível em: <<http://www.caelum.com.br/apostila-html-css-javascript/css-avancado/>>. CAELUM. **APOSTILA DESENVOLVIMENTO WEB COM HTML, CSS E JAVASCRIPT**. Disponível em: <http://www.caelum.com.br/apostila-java-web/javaserver-pages/#6-1-colocando-o-html-no-seu-devido-lugar>. Acesso em: 22 jan. 2015.
7. CAELUM. **APOSTILA JAVA E ORIENTAÇÃO A OBJETOS.** Disponível em: <http://www.caelum.com.br/apostila-java-orientacao-objetos/>[<http://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java/#2-1-java>](file:///C:\\Users\\Tiago\\AppData\\Roaming\\Microsoft\\Word\\%3chttp:\\www.caelum.com.br\\apostila-java-orientacao-objetos\\o-que-e-java\\) Acesso em 28 dez. 2014.
8. CAELUM. **APOSTILA JAVA E ORIENTAÇÃO A OBJETOS.** Disponível em: [<http://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java/#2-2-uma-breve-historia-do-java>](file:///C:\\Users\\Tiago\\AppData\\Roaming\\Microsoft\\Word\\%3chttp:\\www.caelum.com.br\\apostila-java-orientacao-objetos\\o-que-e-java\\) Acesso em 28 dez. 2014.
9. CAELUM. **APOSTILA JAVA E ORIENTAÇÃO A OBJETOS.** Disponível em: [<http://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java/#2-3-maquina-virtual>](http://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java/" \l "2-3-maquina-virtual) Acesso em 28 dez. 2014.
10. COSGEM. **Solicitações de Serviços**. João Câmara: [s.n.], 2013. Disponível em: <<https://docs.google.com/forms/d/1uRokeFpAX7iKCuac_wrD64Ck_vRzKomK80H6eC0crUo/viewform>>. Acesso em: 11 jan. 2015.
11. **CSSTABLEGENERATOR**. Disponível em <http://www.csstablegenerator.com/>. Acesso em 25 jan. 2015.
12. FURTADO, Gustavo. **Você precisa saber o que é SQL!** [S.I.]:Dicas de Programação. Disponível: <http://www.dicasdeprogramacao.com.br/o-que-e-sql/>. Acesso em: 12 jan. 2015
13. http://www.caelum.com.br/apostila-java-web/javaserver-pages/#6-1-colocando-o-html-no-seu-devido-lugar
14. IFRN/JC. **COSGEM implanta procedimento eletrônico para solicitação de seus serviços**: Solicitações de serviços inerentes ao Câmpus e os agendamentos de transporte passam a ser realizados pelo site do IFRN/JC. João Câmara:2013. Disponível em: <<http://portal.ifrn.edu.br/campus/joaocamara/noticias/cosgem-implanta-procedimento-eletronico-para-solicitacao-de-seus-servicos>>. Acesso em: 11 jan. 2015.
15. JUNIOR, Pasteur Ottoni de Miranda. Enterprise Java Beans. Disponível em: <<http://www.tesestec.com.br/pasteurjr/ejb.pdf>>. Acesso em 11 jan. 2015.
16. MAKESYS BLOG. **DEFININDO SEU SOFTWARE: O QUE SÃO REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS?** [S.I.]:MAKESYS BLOG,2013. Disponível em: <<http://blog.makesys.com.br/definindo-seu-software-o-que-sao-requisitos-funcionais-e-nao-funcionais>>. Acesso em: 09 jan. 2015.
17. NETBEANS. **Um breve histórico do NetBeans**. Disponível em: <<https://netbeans.org/about/history_pt_BR.html>>. Acesso: em 11 jan. 2015.
18. PAULA, Rodrigo de. **HTML SURGIMENTO E HISTÓRIA**. Uberlândia: 2014. Disponível em: <https://www.youtube.com/watch?v=9lwqCK4Bklc>. Acesso em: 06 jan. 2015
19. PEREIRA, Altieri. **A origem do CSS, um pouco da história**: Descubra um pouco da história de uma das maiores melhorias no desenvolvimento de paginas para internet. Disponível em: <<http://www.devmedia.com.br/a-origem-do-css-um-pouco-da-historia/15195>>. Acesso em: 13 jan. 2015.
20. PORTAL DEVMEDIA. **Introdução ao Java Server Pages – JSP.** Disponível em: <http://www.devmedia.com.br/introducao-ao-java-server-pages-jsp/25602#ixzz3N8hxi000> Acesso em: 27 dez. 2014.
21. PORTAL INFOESCOLA. **O que são Linguagens de Programação**. Disponível em: < http://www.infoescola.com/informatica/o-que-sao-linguagens-de-programacao/>[<http://www.infoescola.com/informatica/o-que-sao-linguagens-de-programacao/>](http://www.infoescola.com/informatica/o-que-sao-linguagens-de-programacao) Acesso em 28 dez 2014.
22. Quentin. **FORM inside a table.** Fórum de discussão sobre programação. Mantindo por Stack Overflow. [S.I.]:2014 Disponível em: <http://stackoverflow.com/questions/5967564/form-inside-a-table>. Acesso em: 25 jan. 2015.
23. Sara Alvarez. **LINGUAGENS de alto nível:** Fórum de discussão de discussão sobre desenvolvimento web. Mantido pelo site Criar Web. Disponível em: <<http://www.criarweb.com/faq/linguagens-alto-nivel.html>> Acesso em 06 jan. 2015.
24. Vladimir. **FORM inside a table.** Fórum de discussão sobre programação. Mantindo por Stack Overflow. [S.I.]:2014 Disponível em: <http://stackoverflow.com/questions/5967564/form-inside-a-table>. Acesso em: 25 jan. 2015.
25. W3SCHOLLS. **SQL Aliases.** Disponível em: < http://www.w3schools.com/sql/sql\_alias.asp>. Acesso em: 22 jan. 2015.

1. Adorilson Bezerra, aula dada na disciplina Engenharia de Software, do curso de Técnico em Informática, do IFRN, no dia 05 jun. 2014. [↑](#footnote-ref-2)