Universidade de São Paulo
Instituto de Matemática e Estatística

ACHUSP

Serviço de achados e perdidos para a Universidade de São Paulo

Everton Topan da Silva
Orientador: Dr. Alfredo Goldman

11 de fevereiro de 2013

Agradecimentos

Agradeço ao professor Alfredo por me indicar o projeto e me ajudar a elaborá-lo, dando todo o apoio necessário para que o sistema possa ser utilizado pela comunidade USP. Agradeço também a Karina e Rafael, que me ajudaram a fazer o design do sistema. E finalmente agradeço minha família e amigos por todo o apoio que me deram durante o desenvolvimento desse trabalho.

Sumário

Resumo

A Universidade de São Paulo é uma das maiores universidades da América latina, e diversos objetos são perdidos em suas dependências todos os dias. Os sistemas de achados e perdidos existentes não são muito eficientes pois necessitam de um intermediário entre quem achou e quem perdeu o objeto, esse intermediário é geralmente o departamento de achados e perdidos de cada faculdade, fazendo com que as vezes as pessoas não saibam onde procurar seus pertences se não souberem onde os perderam.

O sistema desenvolvido neste trabalho pretende facilitar a devolução de itens perdidos ou encontrados, servindo como um local para centralizar todos os anúncios desse tipo dentro da Universidade de São Paulo, e oferecendo a opção de ser um intermediário entre as partes. Todos os serviços serão oferecidos em um site desenvolvido com Ruby on Rails, HTML5, coffee script e Sass no qual os usuários poderão cadastrar os objetos perdidos ou encontrados e se comunicar com outras pessoas que possam ter achado seus objetos.

1 Introdução

1.1 Motivação

É muito comum as pessoas perderem objetos pessoais, e esses serem encontrados por alguém que gostaria de devolvê-los. Para isso em diversos locais há o serviço de achados e perdidos. Dados da CPTM [1] mostram que no ano de 2011 cerca de 75 mil itens foram armazenados na sua central de achados e perdidos, desses, aproximadamente 35 mil foram devolvidos ao seus donos e 80% das devoluções acabam partindo dos próprios funcionários da CPTM, que identificam a pessoa através de dados nos pertences e entram em contato com ela para que ocorra a devolução. Isso mostra que muitas vezes quem perde um objeto acaba não tentando recuperá-lo.

Sistemas como esse da CPTM servem como uma ponte entre quem perdeu e quem achou o objeto e nesses casos o departamento de achados e perdidos necessita de um local para guardar os itens e pessoas para supervisionar o local, até mesmo quando o serviço é oferecido on-line como já acontece no metrô de São Paulo e nos correios.

Esse sistema tem a desvantagem da necessidade de um intermediário, pois nem sempre esse existe, ou está disponível. Aproveitando a web e sua usabilidade, foi pensado um site que pode eliminar esse intermediário, fazendo toda a comunicação entre as partes acontecer on-line. Esse sistema será implementado visando seu uso dentro da Universidade de São Paulo, onde é difícil localizar objetos perdidos, tanto que foi criado um grupo no Facebook para os alunos anunciarem objetos perdidos ou encontrados pelo campus.

1.2 Objetivo

O trabalho será a implementação de um sistema web, que possibilite o cadastro de itens perdidos ou encontrados. Além disso o site permitirá que os usuários se comuniquem para que os objetos sejam devolvidos, o sistema também permitirá que um usuário que perdeu algo, possa oferecer uma recompensa a quem encontrá-lo.

Para utilizar o sistema o usuário terá que fazer um cadastro no site, para que seja mais fácil das duas partes manterem contato. As informações necessárias para esse cadastro são: nome, email e uma senha de acesso.

As buscas no site serão otimizadas, pois nem sempre um usuário que esteja cadastrando um objeto, vê aquele item como o usuário que o perdeu, por exemplo uma pessoa pode cadastrar que perdeu um iphone e que achou o cadastra como um celular. Para resolver esse problema o sistema conta com tags, e com uma busca textual indexada, que devolverá ao usuário o máximo de informação relevante para sua busca.

O sistema será desenvolvido como um software livre, permitindo que desenvolvedores que gostarem da ideia possam modificar o código para usar o sistema em outros locais além da USP, ou implementar novas funcionalidades que agreguem valor ao sistema.

2 Metodologia de desenvolvimento

2.1 Métodos ágeis

Este trabalho foi feito utilizando princípios de métodos ágeis, que é uma metodologia de desenvolvimento de software que prega os seguintes valores:

> *Indivíduos e interações mais que processos e ferramentas*
> *Software em funcionamento mais que documentação abrangente*
> *Colaboração com o cliente mais que negociação de contratos*
> *Responder a mudanças mais que seguir um plano* [2]

Foi escolhido o uso de métodos ágeis pois o trabalho foi pensado incrementalmente, e diversas mudanças tiveram que ser feitas durante o desenvolvimento. Metodologias tradicionais tendem a pensar toda a estrutura da aplicação antes de escrever qualquer código, e por isso são pouco receptivos a mudanças. Como eram esperadas mudanças durante o desenvolvimento e esse trabalho não é um sistema crítico, métodos ágeis foram os mais adequados.

2.2 BDD

O BDD, behavior driven development, é uma forma de estruturar o programa pensando primeiramente no comportamento que esse deve ter. O uso de BDD foi escolhido pois é um modelo baseado em TDD, onde se faz os testes antes de escrever o código da aplicação, porém com BDD os testes são muito similares a histórias, visando aproximar clientes e desenvolvedores.

O uso do BDD começa nas histórias do usuário. A partir delas, escreve-se um teste, onde o desenvolvedor já está pensando no comportamento que o sistema terá ao executar essa tarefa. Depois é escrito o código que confirma tal comportamento, dando confiança ao desenvolvedor para fazer alguma modificação depois. Tais testes são como documentos da intenção do sistema, e são muito valiosos como documentação do software.

3 Conceitos Estudados

3.1 MVC

"MVC consiste de três tipos de objetos. O modelo é o objeto da aplicação, a Visão é a tela de apresentação e o Controlador define o modo como a interface reage a entradas do usuário. Antes do MVC, o design de interfaces tendiam a juntar esses objetos. O MVC visa separá-los de acordo com a sua funcionalidade, diminuindo o acoplamento e aumentando a flexibilidade e o reuso." [12]

MVC (Model/View/Controller) é um modelo de desenvolvimento muito usado em aplicações Web, a divisão do sistema nessas três partes permite que mudanças que ocorram em uma camada da aplicação, não afetem as outras camadas, o que deixa o sistema muito mais fácil de manter.

3.2 Sistema de busca inteligente

"Dado uma consulta - uma palavra chave ou uma frase - busca é o processo de recuperar documentos que são relevantes à consulta. A lista de documentos é geralmente devolvida na ordem de quão relevantes eles são para a consulta" [10]

Com a grande quantidade de informação contida na web, ferramentas de busca inteligentes são cada vez mais importantes nos sistemas. Essas ferramentas visam recuperar e organizar dados de forma a transformá-los em informação relevante. Isso é feito levando em consideração alguns critérios como a frequência das palavras, o posicionamento delas no documento ou a quantidade de vezes que o termo aparece.

Para tornar as buscas mais rápidas e eficientes são criados índices que são listas de palavras chave, onde cada item dessa lista tem apontadores para documentos que contenham a palavra chave, geralmente as ferramentas ignoram as "stop words" que são termos comuns no idioma ou domínio pesquisados.

Nesse sistema a busca inteligente é necessária, para que a partir de uma descrição bem feita do objeto, o usuário possa receber o máximo de dados relevantes em para sua pesquisa. Isso é possível com o uso da plataforma de busca Apache Solr, que juntamente com o Lucene [1], permite que os objetos sejam indexados e buscados de acordo com a necessidade da aplicação, sendo possível definir quais critérios devem ser levados em conta e qual o peso de cada critério quando o usuário realiza uma busca.

3.3 Sistema de recomendação

Ainda pensando em filtrar a grande quantidade de informações da internet e oferecer itens relevantes para o usuário, muitos sistemas usam ferramentas de recomendação. Essas ferramentas mostram itens interessantes para o usuário dentro do seu contexto.

O sistema de recomendação implementado nesse trabalho visa ajudar os usuário a encontrarem itens relevantes para o que estão procurando, complementando o sistema de busca. Para um usuário que cadastrou um item perdido, por exemplo, é mostrado itens encontrados parecidos com esse. E no caso de um usuário estar apenas navegando por itens, serão recomendados objetos do mesmo tipo (encontrado ou perdido) que pareçam com o item que estão vendo. Isso facilita ainda mais os usuários do sistema a receberem o máximo de informações relevantes.

[1] Lucene é uma ferramenta de recuperação de informação escalável e de alta performance, que permite adicionar indexação e busca à sua aplicação

Para fazer a recomendação foi usado um algoritmo de similaridade de texto onde todas as palavras são postas em um vetor de termos. Cada termo nesse vetor tem um peso que é menor para quanto mais vezes esse termo aparece no texto. Isso é usado para retirar o peso de palavras comuns e aumentar o peso de palavras especificas. Com esse vetor é calculada a similaridade de coseno[2] do vetor e é devolvido um valor de 0 a 1 sendo 0 nada similar e 1 um texto idêntico. Desse modo é possível recomendar para o usuário itens parecidos com o que ele se interessou a partir da descrição do item.

Para a construção do vetor foi usado um algoritmo para fazer a lematização dos termos, que é a tentativa de reduzir a palavra à sua raiz. Pois desse modo o algoritmo obtém melhores resultados.

[2]Algoritmo que calcula o produto interno de dois vetores e devolve o cosseno do ângulo entre eles

4 Tecnologias utilizadas

O sistema foi implementado utilizando a linguagem Ruby juntamente com o arcabouço Rails para o back end, o front end foi feito com HTML5 e Sass. Nessa seção serão explicadas as principais ferramentas utilizadas no sistema.

4.1 Ruby

"Ruby é uma linguagem de programação dinâmica com uma complexa, mas expressiva gramática e uma biblioteca de classe integrada com uma rica e poderosa API. O Ruby se inspira no LISP Smalltalk e Perl, mas usa uma gramática que é fácil para os programadores de C e Java aprenderem." [11]

A linguagem escolhida para fazer o trabalho foi Ruby, pois é uma linguagem com uma gramática simples, open source, orientada a objetos e é bastante presente na comunidade de software livre. Além disso o uso do arcabouço Rails ajudou muito no desenvolvimento do trabalho, pois esse oferece uma arquitetura MVC eficiente e de fácil uso.

4.2 RubyGems

O RubyGems é um gerenciador de pacotes do Ruby, que provê um padrão para a distribuição de aplicações e bibliotecas em Ruby. Esses pacotes são chamados Gems e contém os arquivos e informações necessários para serem instalados no sistema. As Gems tem uma grande importância em aplicações que utilizam Ruby, pois elas podem ser facilmente usadas para estender ou modificar a funcionalidade dos programas, evitando que os desenvolvedores "reinventem a roda", e diminuindo a duplicação.

4.3 Rails

Rails é um arcabouço que torna mais fácil desenvolver, instalar e manter aplicativos web. Todos os aplicativos Rails são implementados utilizando a arquitetura MVC, onde cada fragmento de código e todas as partes do seu aplicativo interagem de uma maneira padrão. Além disso o Rails oferece um ótimo suporte a testes.[13]

O design do Rails foi baseado em alguns conceitos como DRY [3] e convenção sob configuração. Usando as convenções é possível escrever um aplicativo Rails com menos código do que um feito em java, por exemplo, pois não será necessário vários arquivos de configuração XML.

Porém o principal fator que incentivou o uso do Rails foi que ele promove o desenvolvimento ágil, a documentação é gerada automaticamente a partir do seu código, e muitas vezes o código é a documentação. O fato do Rails honrar o principio DRY faz com que alterações na sua aplicação tem menos impacto sobre o código. E a ênfase em testes junto com o suporte fornece a segurança para mudanças e refatorações.

4.4 Devise

O devise é uma Gem que oferece o sistema de autenticação flexível. Essa solução é bem modular, o que permite que o sistema use apenas o que for necessário e possui uma ótima integração com o Rails, pois também é baseado em MVC. Essa Gem

[3] Don't repeat yourself (Não se repita)

salva todos os dados de senhas criptografados, e permite que sejam recuperadas, além disso possui gerenciamento de sessão, e muitas outras funcionalidades que não estão sendo utilizadas no sistema.[3]

O sistema delega ao devise toda a parte de autenticação, ou seja, login, cadastro de usuários, recuperação de senhas e validação dos dados. Além disso essa ferramenta também oferece vários métodos para lidar com as sessões do usuário, sendo possível recuperar data do ultimo login, endereço ip utilizado e várias outras informações importantes para colher estatísticas de uso do sistema.

4.5 Cancan

O Cancan é uma Gem de autorização, que utiliza um sistema de papéis para usuários e restringe quais ações cada usuário pode fazer no sistema. Por exemplo, quando um usuário tenta editar um item que não foi ele quem cadastrou, o cancan lança uma exceção de acesso negado. O sistema então recebe essa exceção e exibe uma mensagem para o usuário informando que ele não pode executar essa ação.[4]

4.6 Paperclip

O Paperclip [5] é uma ferramenta que facilita o uso de arquivos na aplicação, desde o upload até o modo que eles são salvos no banco de dados. O paperclip permite que arquivos da sua aplicação possam ser tratados como qualquer outros atributos dos seus objetos. Além disso essa Gem permite a geração de thumbnails se usado junto com o ImageMagick.[4]

4.7 Cucumber

Cucumber é uma ferramenta que executa testes de aceitação automatizados escritos no padrão do BDD. O cucumber roda os testes de aceitação simulando ações que o usuário faria na aplicação. Todos os testes são escritos em linguagem de alto nível, e portanto podem ser usados como documentação do sistema.[6]

4.8 Acts_as_taggable_on

Gem usada para gerenciar as tags do sistema, essa ferramenta permite que as tags sejam tratadas como uma string, e oferece diversos métodos para ajudar na manipulação das tags como recuperar todos os itens com determinada tag, ou gerar uma tag cloud.

As tags no sistema são importantes para categorizar os itens no sistema, como o conjunto de objetos que pode ser perdidos é muito grande as tags ajudam a separar objetos, juntamente com a busca indexada as tags tornam muito mais fácil para os usuários encontrarem um objeto cadastrado. Além disso o site possui um sistema de auto completar no momento de inserir tags ao item, incentivando o usuário a usar tags que já estão cadastradas no sistema ao invés de criar uma nova.

4.9 Sunspot

O sunspot é uma gem que faz a interação do Ruby com a ferramenta solr. O sunspot oferece uma maneira simples de indexar e buscar por objetos da sua aplicação, além de permitir a definição de critérios para realizar as buscas.

[4]ImageMagick é um software que permite criar, editar e converter imagens.

As buscas por itens são feitas em 3 atributos: título, descrição e tags. Os campos de título e descrição são indexados pelo solr, para uma busca textual inteligente, pois esses campos são os mais importantes para o calculo da relevância dos itens para a busca.

Para o calculo da relevância, o título é o atributo mais importante, então quando o termo buscado aparece no título do anuncio esse item aparece primeiro para o usuário. O segundo atributo em importância é a descrição e por ultimo as tags.

Buscas feitas por termos com mais de uma palavra tem uma relevância baseada em campos que tenham todas as palavras chave, seguidas por quão próximas as palavras aparecem nos atributos, e por último itens que possuam apenas parte das palavras no termo. Essa configuração foi escolhida, pois enquanto o sistema não tiver um grande número de itens é importante que o usuário tenha o máximo de informação que o sistema possa oferecer por cada busca. Uma vez que o sistema tenha vários itens cadastrados, provavelmente será necessária uma redefinição desses parâmetros.

4.10 Páginas dinâmicas com JQuery e Ajax

JQuery[7] é uma biblioteca Javascript que simplifica a busca por elementos HTML, manipulação de eventos e requisições com Ajax. Essa biblioteca foi usada para fazer o carregamento das informações dos itens no perfil do usuário dinâmico, fazendo com que o usuário não fique com diversas informações que não são do seu interesse na tela. O JQuery também foi utilizado para a implementação da ferramenta de preenchimento automático no campo de inserção de tags, que serve para ajudar o usuário a escolher uma tag já cadastrada no sistema em vez de criar uma nova, o que melhora a categorização e a busca dos itens.

5 Desenvolvimento

O início do desenvolvimento se deu em junho, após o estudo das ferramentas que seriam utilizadas no trabalho, todo o desenvolvimento foi iterativo, com as histórias sendo criadas conforme fosse necessário, começando com algumas histórias principais do sistema que permitissem sua utilização na USP. Por falta de experiência no desenvolvimento desse tipo de sistema e no uso de algumas ferramentas, esse primeiro conjunto de histórias acabou atrasando, o que levou o sistema a não ser lançado no prazo esperado, atrasando a coleta de feedback. Apesar desse atraso, ao continuar o trabalho e conhecendo melhor todas as ferramentas o desenvolvimento foi fluindo cada vez mais fácil, o que levou novas funcionalidades pensadas depois a serem implementadas numa velocidade muito maior do que no início, levando o sistema a ser colocado em produção no endereço `http://valinhos.ime.usp.br:50480` para ser utilizado pelos alunos. Após a divulgação do sistema, as pessoas começaram a usá-lo e fornecer feedback ajudando a melhorar a usabilidade.

5.1 Mudanças Realizadas

Todos os sistemas estão sujeitos a mudanças e com esse trabalho não foi diferente. Um dos benefícios que o modo de desenvolvimento ágil trouxe ao projeto foi quanto as mudanças, pois quando essas ocorreram, foi necessário apenas criar um teste, e desenvolver o código para essa mudança e as vezes realizar algumas refatorações que ficam muito mais simples com um código bem escrito e com testes. Em modos tradicionais de metodologias, como o modelo cascata, a cada mudança no sistema seria necessário um ciclo para fazer a modelagem (UML, use cases, diagrama entidade relacionamento), depois implementá-las e finalmente testá-las, o que em geral demora mais tempo.

A maior mudança que ocorreu no sistema foi a resolução do problema das buscas. A principio foi pensado em usar uma ontologia pronta para que o sistema usasse os dados para obter resultados similares ao que o usuário buscou. Por exemplo, ao procurar por iphone, o sistema faria um acesso a ontologia e devolveria resultados de telefones celulares. Porém essa abordagem não seguiu como era esperado, pois não foi encontrada uma ontologia que cobrisse o domínio da aplicação. Para substituir as ontologias, foi usado o sistema de busca atual.

Outra mudança que tiveram que ocorrer no sistema foram a inclusão de um formulário de contato, para que os usuários pudessem enviar uma mensagem para o administrador do sistema, com alguma dúvida ou sugestão sobre o sistema, a inclusão de uma listagem de casos solucionados. Além disso o feedback fornecido pelos usuários incentivaram a implementação de notificações para mensagens recebidas.

5.2 Arquitetura da aplicação

Em um sistema de tamanho médio, como é o caso dessa aplicação, é necessário dividir o sistema em serviços relacionados e o Rails é o responsável por estabelecer o controle e e a comunicação dos subsistemas, e isso auxilia no reuso e desacoplamento. A arquitetura da aplicação seguiu o padrão proposto pelo Rails, pois é uma arquitetura muito bem pensada para sistemas web, onde o desenvolvedor passa a maior parte do tempo lidando com módulos de alto nível.

A estrutura da aplicação é baseada no modelo MVC, e o Rails tem um módulo para lidar com cada componente, os principais são:

- **Active Record** - É a camada de mapeamento objeto-relacional fornecida

pelo Rails. Essa camada é responsável pelo modelo, e permite que tabelas do banco de dados sejam mapeadas para classes, linhas para objetos, e colunas para atributos. Além disso o Active Record fornece diversos métodos que auxiliam a manipular os dados, lidar com validações e fazer relacionamentos entre tabelas.

- **Action View** - Módulo que encapsula toda a funcionalidade necessária para renderizar templates , mas comumente gerando HTML, XML ou Javascript, para o usuário. Ou seja ele cuida da parte da visão do MVC.
- **Action Controller** - É o módulo de controlador do Rails. Ele juntamente com o Action View fornecem suporte ao processamento das requisições recebidas, e geram respostas para o usuário, de acordo com as informações do controlador.

5.3 Padrões de projeto

Padrões de projeto vieram de problemas de design que algum desenvolvedor já enfrentou e desenvolveu uma solução reutilizável, geralmente os problemas resolvidos com padrões de projeto são comuns em projetos de software. Nessa seção será mostrado quis padrões foram utilizados nesse sistema e quais problemas eles resolveram.

- **Observer** - Define uma dependência entre objetos de modo que quando um objeto muda de estado, seus dependentes são notificadas. O observer foi utilizado juntamente com o Active Record para definir ações que deveriam ser feitas antes ou depois de alguma ação do usuário. Como por exemplo criar uma mensagem de log toda vez que um usuário é criado.
- **Factory** - Define uma interface para criar um objeto, permitindo que a classe decida qual objeto instanciar. Esse padrão permite que uma classe escolha passar a instanciação para as subclasses. Esse padrão foi usado na criação de mensagens.
- **Composite** - Padrão que permite que objetos individuais ou composições de objetos sejam tratados por uma mesma interface. Esse padrão foi utilizado nas visões do sistema, onde são utilizadas páginas dentro de páginas, possibilitando reuso de código, como cabeçalho e rodapé, presente em todas as telas.
- **Facade** - Fornece uma interface simplificada para facilitar a utilização de um subsistema. Esse padrão foi utilizado no sistema para lidar com as datas, já que o padrão de datas do Rails é americano.

5.4 Código

Na programação código bem escrito é uma qualidade importante no seu programa. Apesar disso não agregar valor ao seu programa na visão da maioria dos clientes, um código sem acoplamento, duplicação e com bom uso dos padrões de programação orientada a objetos é muito mais fácil de manter e programas bem escritos também são muito mais fáceis para inserir funcionalidades.

Para analisar o código foi usada ferramentas de métrica, que mostram estatísticas do seu código e também apontam alguns problemas como código duplicado. Algumas estatísticas do projeto são:

- 1034 linhas de código
- Cobertura de testes de 85.8%
- 95 commits
- 15 classes
- 54 métodos

Além das estatísticas as ferramentas devolveram alguns problemas, e isso permitiu que fossem feitas refatorações pontuais no projeto. Apesar de muitos dos resultados dessas ferramentas serem bem precisos e úteis há casos onde erros são apontados, mas sua refatoração não trará nenhum beneficio ao código, por isso é necessário analisar os resultados e utilizá-los quando isso melhorar a qualidade do código. alguns dos problemas apontados foram:

- Código duplicado. A métrica não só mostra códigos duplicados, mas também parecidos.
- Ausência de índices no banco de dados. A ferramenta também mostrou que faltavam índices em algumas chaves estrangeiras nas tabelas do banco. tais índices são necessários pois na estrutura do Rails e do Active record, esse dados são muito utilizados, fazendo com que o uso de um índice melhore consideravelmente a performance das transações.
- Construção com parâmetros complexos. As vezes na aplicação é necessário construir um modelo complexo a partir dos parâmetros e fazer isso no controlador não é uma boa prática, portanto nesses casos é recomendado o uso do padrão Factory.

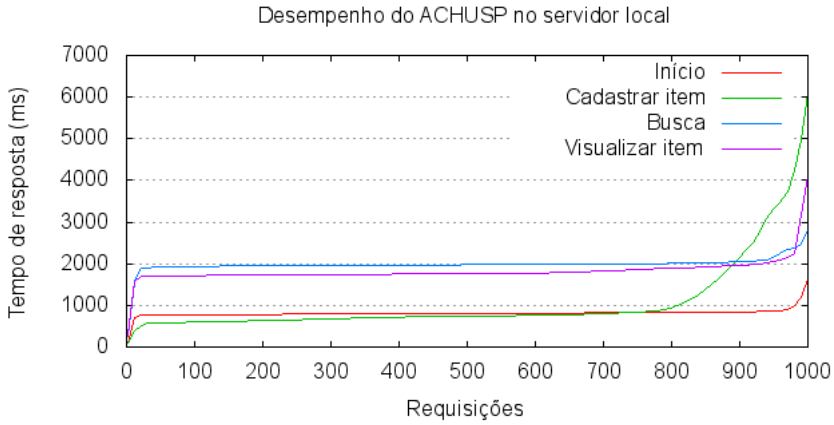
O código da aplicação é livre e está disponível no github em `https://github.com/evertonts/usp-lost-and-found`. Lá é possível obter o código, reportar bugs e criar forks, que são novas frentes de desenvolvimento do projeto, para outras pessoas implementá-lo modificando como quiserem.

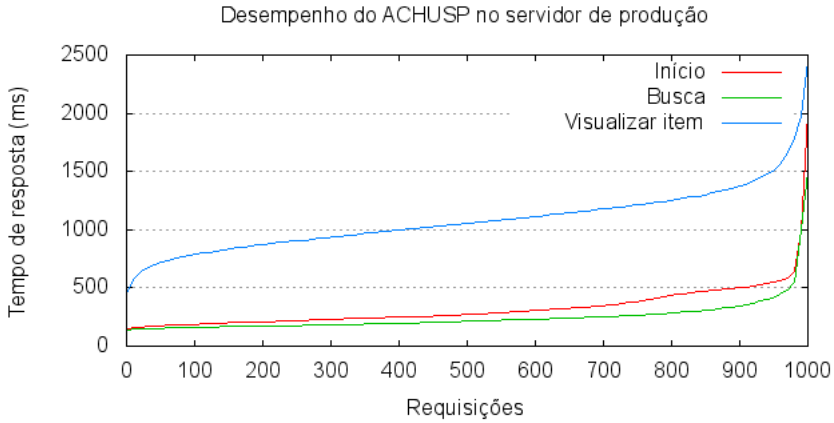
5.5 Desempenho

Uma grande preocupação dos desenvolvedores em sistemas web é o desempenho que o site terá quando vários usuários estiverem acessando-o ao mesmo tempo. Por isso testes de desempenho são importantes. Tais testes são usados principalmente para descobrir gargalos no site, ajudando os desenvolvedores a focarem o trabalho nesses gargalos.

Nesse sistema os testes foram feitos no servidor de desenvolvimento, que roda na mesma máquina em que os testes foram feitos, e no de produção, que roda na máquina valinhos do CCSL. Esses testes foram feitos usando a ferramenta apache bench [5] simulando 1000 requisições sendo feitas por 30 usuários simultaneamente. Estes foram os resultados:

[5] `http://httpd.apache.org/docs/2.2/programs/ab.html`





Os gráficos mostram o tempo que o sistema levou para processar as 1000 requisições, cada linha representa a ação feita na requisição. No gráfico do servidor remoto não há a ação de cadastro de item, pois como o sistema já está em produção, não seria possível cadastrar diversos itens para realizar o teste.

Podemos ver que o servidor de produção tem um desempenho bem melhor que o servidor local, em todas as ações testadas nos dois, e também vemos que ambos tem um grande aumento de tempo quando se aproxima de 1000 requisições mostrando que quando o sistema começar a ter uma essa carga de requisições, talvez seja necessária uma melhora no hardhare do servidor ou uma revisão no código ou na configuração da aplicação. Além disso vemos no primeiro gráfico que a ação mais demorada do sistema é o cadastro de itens que chega a levar 6 segundos para completar as 1000 requisições e isso é esperado pois o cadastro é feito por uma

requisição POST, em que há verificação da autenticidade do usuário enquanto as outras ações são requisições GET que não passam por nenhuma verificação.

O melhor desempenho da aplicação em produção é principalmente devido ao servidor HTTP, pois no sistema em produção é usado o Apache, uma ferramenta robusta e preparada para lidar com varias requisições simultâneas, enquanto a máquina de desenvolvimento está rodando o webrick que é um servidor web simples usado para rodar aplicações Rails facilmente.

Esses dados mostram que o sistema tem um bom desempenho, e pode lidar com várias requisições simultâneas, pois para termos um parâmetro, o sistema foi em média apenas três vezes mais lento que o google, e duas vezes mais lento que o facebook, que são os dois sites mais acessados da internet[8].

6 Atividades Realizadas

Seguindo princípios de extreme programming as atividades foram criadas em formato de histórias. As histórias são muito parecidas com use cases, porém são usadas também para estimar o tempo das tarefas e servem juntamente com os testes como documento do sistema [9].

6.1 Estrutura das histórias

Para escrever as histórias foi usada uma estrutura que é muito parecida com os testes feitos com cucumber. Elas são compostas de títulos, narrativas e cenários.

6.1.1 Título

A história deve ter um título que a descreva, tornando fácil descobrir o que ela agrega ao sistema.

6.1.2 Narrativa

A narrativa deve incluir um papel, uma funcionalidade e um beneficio. O template "como um [papel] eu quero [funcionalidade] para que [beneficio]" é muito usado é bem explicativa e inclui todos os elementos necessários para uma boa descrição.

6.1.3 Cenário

A história pode ter vários cenários, e os cenários tem um título e uma descrição. O título do cenário deve mostrar o que mudou, ou seja, deve ser possível, lendo apenas o título saber a diferença de um cenário e outro da sua história. A descrição dos cenários deve conter uma entrada uma ação e uma saída.

6.2 Histórias

6.2.1 Registro de usuários

```
Como um usuário do site
Eu quero ser capaz de me cadastrar
Para ter acesso a partes protegidas do sistema

Cenário: Usuário se cadastra com dados válidos
  Dado que eu não tenho uma conta no site
  Quando eu insiro meus dados no sistema
  Devo ser capaz de fazer login no site

Cenário Usuário se cadastra com dados inválidos
  Dado que eu não tenho uma conta no site
  Quando eu me cadastro com dados inválidos
  Devo receber uma mensagem de erro
```

6.2.2 Usuário edita seus dados

Como um usuário registrado do site
Eu quero ser capaz de editar meu perfil
Para poder trocar meus dados

Cenário: Usuário logado edita sua conta
Dado que eu estou logado no site
Quando eu atualizo meus dados no sistema
E insiro minha senha
Devo ver meus dados alterados

6.2.3 Usuário faz login

Como um usuário registrado do site
Eu quero ser capaz de fazer login
Para poder acessar minha conta e ter acesso a todo site

Cenário: Usuário faz login com dados válidos
Dado que eu não estou logado no site
Quando eu faço login com dados válidos
Devo estar autenticado no sistema

Cenário: Usuário faz login com dados inválidos
Dado que eu não estou logado no site
Quando eu faço login com dados inválido
Devo ver uma mensagem de erro

6.2.4 Usuário faz logout

Como um usuário registrado do site
Eu quero ser capaz de fazer logout
Para proteger minha conta de acesso não autorizado

Cenário: Usuário faz logout do site
Dado que eu estou logado no site
Quando faço logout
Devo não ter mais acesso a minhas informações.

6.2.5 Mandar mensagens para outros usuários

Como um usuário registrado do site
Eu quero ser capaz de mandar mensagens referentes a algum item para quem o criou
Para poder devolver ou recuperar um objeto

Cenário: Usuário manda mensagem
Dado que eu estou logado no site
Quando eu mando uma mensagem através de um item
Devo poder acompanhar essa mensagem no meu perfil

6.2.6 Reportar um item encontrado

Como um usuário registrado do site
Eu quero ser capaz de cadastrar objetos encontrados
Para poder achar seu dono

Cenário: Usuário cadastra um objeto encontrado com dados válidos
Dado que eu estou logado no site
Quando eu cadastro um objeto com dados válidos
Devo ver esse objeto no meu perfil
E receber mensagens referentes a ele

Cenário: Usuário cadastra um objeto encontrado com dados inválidos
Dado que eu estou logado no site
Quando eu cadastro um objeto com dados inválidos
Devo ver uma mensagem de erro

6.2.7 Reportar um item perdido

Como um usuário registrado do site
Eu quero ser capaz de cadastrar objetos perdidos
Para poder recuperá-los

Cenário: Usuário cadastra um objeto perdido com dados válidos
Dado que eu estou logado no site
Quando eu cadastro um objeto com dados válidos
Devo ver esse objeto no meu perfil
E receber mensagens referentes a ele

Cenário: Usuário cadastra um objeto perdido com dados inválidos
Dado que eu estou logado no site
Quando eu cadastro um objeto com dados inválidos
Devo ver uma mensagem de erro

6.2.8 Editar item

Como um usuário registrado do site
Eu quero ser capaz de editar itens cadastrados por mim
Para inserir ou arrumar informações desse item

Cenário: Usuário edita item
Dado que eu estou logado no site
Quando eu atualizo um item com novas informações
Devo ver esse objeto atualizado

6.2.9 Dar um item como recuperado

Como um usuário registrado do site
Eu quero ser capaz de marcar um item como recuperado
Para que outros usuários saibam que esse item foi recuperado
E eu não receba mais mensagens referentes a esse item

Cenário: Usuário marca um item como recuperado
Dado que eu estou logado no site
Quando eu marco um item como recuperado
Devo ver uma mensagem de item recuperado

6.2.10 Buscar por itens

Como um usuário do site
Eu quero ser capaz de buscar por itens
Para que eu veja se alguém já cadastrou o item de meu interesse

Cenário: Usuário busca por um termo
Quando eu busco por um termo
Devo ver uma lista de itens relevantes à pesquisa

7 Resultados e produtos obtidos

Nessa seção serão mostradas as principais telas do sistema.

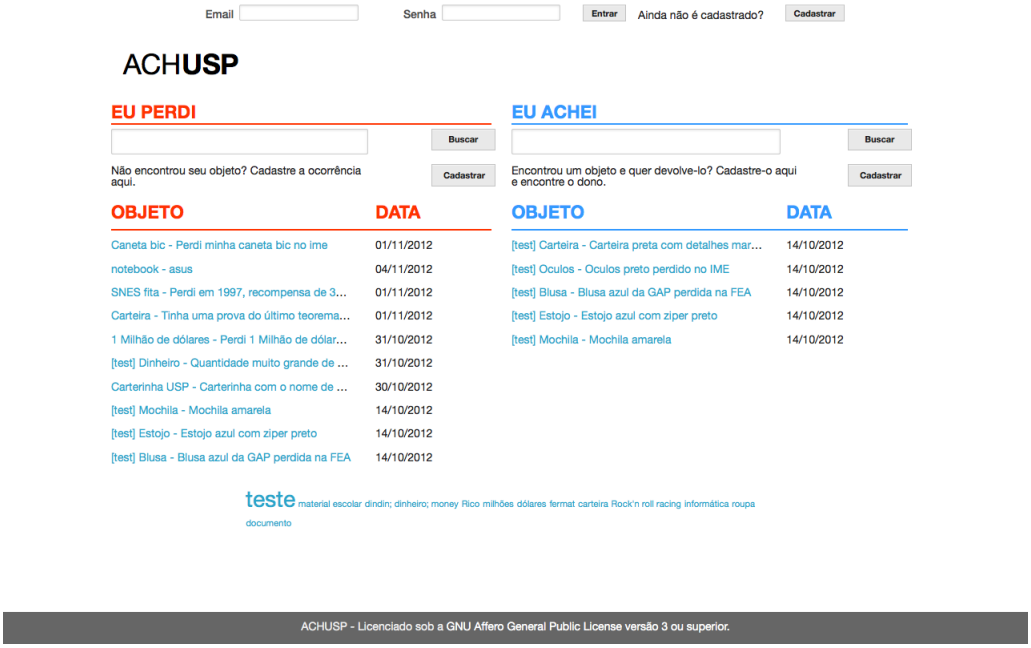


Figura 1: Tela inicial

Tela inicial e principal do sistema, nela o usuário vê os últimos itens cadastrados no sistema separados por itens encontrados e perdidos. Além disso essa tela tem links para o usuário cadastrar novos itens no sistema e na parte inferior tem um tagcloud que mostra as tags mais importantes do sistema com o tamanho da fonte delas dependendo de quantos itens usam a tag.

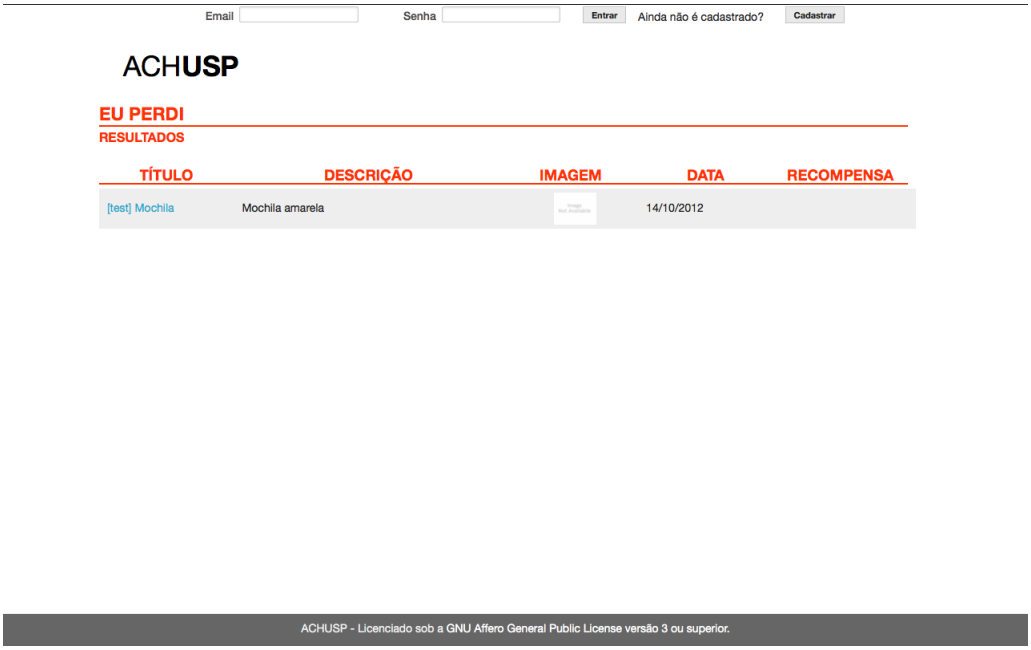


Figura 2: Tela de resultados da busca

Tela mostra os itens relevantes para a busca do usuário.

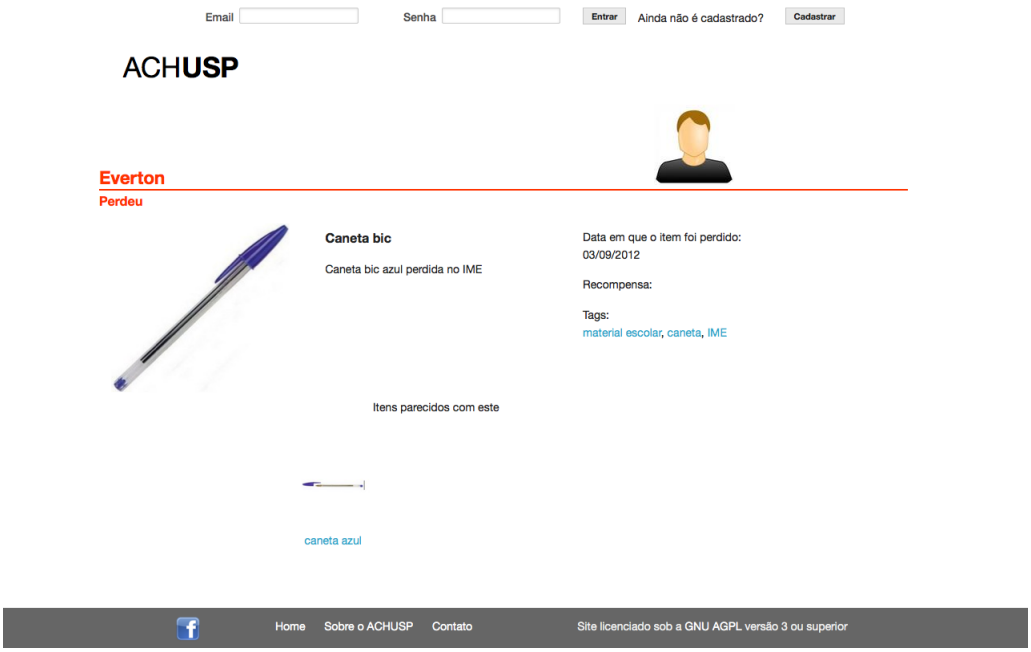


Figura 3: Tela de item

Tela que mostra os dados dos itens e do usuário que cadastrou esse item. As informações do usuário são necessárias para que a devolução dos itens possa ser realizada, mesmo quando o usuário não quiser passar outras informações. Além disso nessa tela é possível que outros usuários mandem uma mensagem referentes ao objeto para quem o cadastrou. Na parte inferior, são mostrados itens semelhantes a esse no sistema, se existir algum.

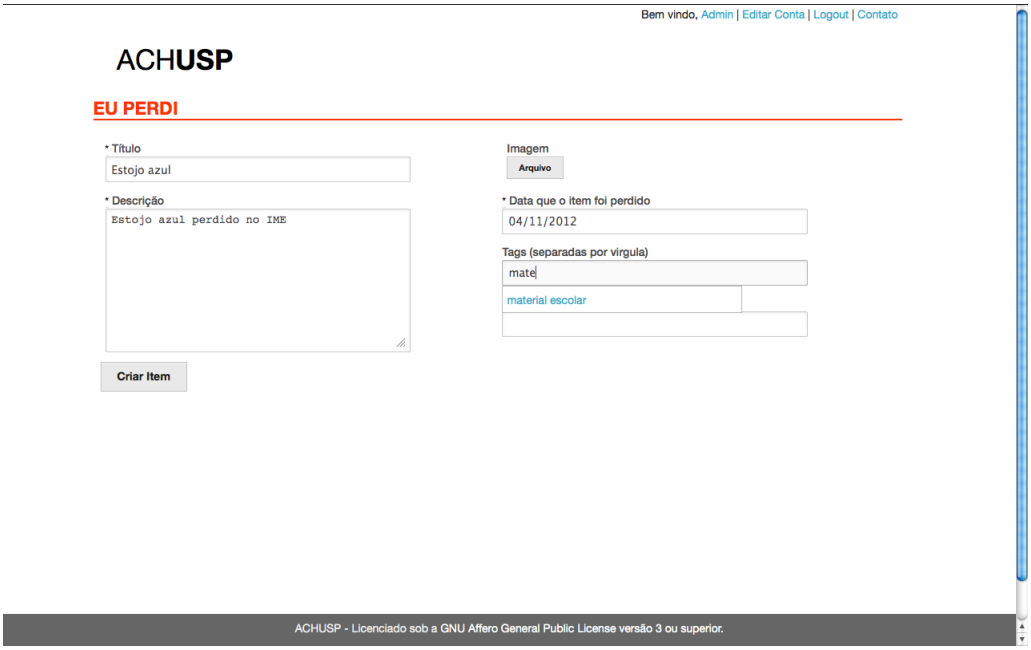


Figura 4: Tela de cadastro

Tela usada para cadastrar itens, os itens tem campos de título e descrição obrigatórios, mas além deles é possível incluir uma imagem do item, tags e uma recompensa por itens perdidos. A associação de tags ao item conta com um sistema de preenchimento automático, enquanto o usuário está digitando uma tag, requisições vão sendo mandadas para o sistema e são realizadas buscas por tags com esse texto. Se o usuário digitar uma tag não cadastrada no sistema essa tag é adicionada.

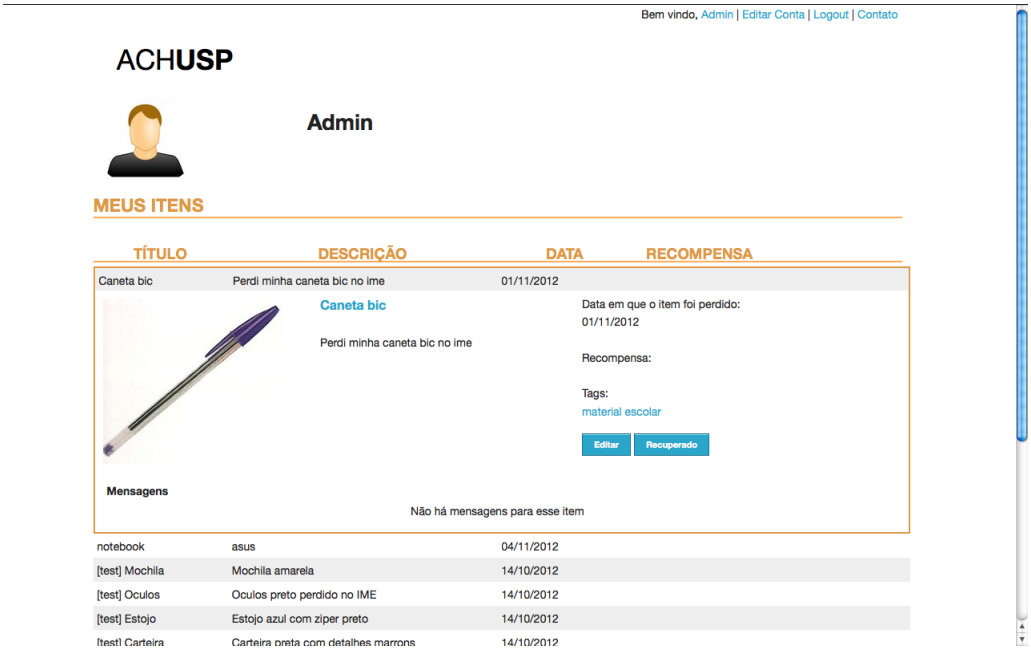


Figura 5: Tela do usuário

Essa tela mostra todos os itens que o usuário cadastrou, ou interagiu, mandando uma mensagem. Nela é possível clicar em qualquer item e abrir suas informações e ver as mensagens referentes ao item, e marcá-lo como recuperado.

8 Conclusões

Produzir um sistema Web é uma tarefa trabalhosa mas bastante recompensadora, e tive a oportunidade de aprender muitas coisas sobre desenvolvimento Web, área muito importante na computação atualmente. O uso de Rails foi muito bom, pois esse é um arcabouço muito bem pensado e que se encaixa perfeitamente com o desenvolvimento ágil, que é o modo como eu gosto de desenvolver. O desenvolvimento ágil também foi bem produtivo, o sistema no momento está com uma cobertura de testes de 85.8%, maior parte feita usando TDD, porém pretendo aumentar essa taxa.

A refatoração também foi bastante usada no projeto, pois como no inicio eu não conhecia muito bem Rails acabava não seguindo as melhores praticas, mas com o tempo fui conhecendo melhor a ferramenta e melhorando meu código, apesar de no fim do projeto, com prazos apertados, acabei deixando a refatoração um pouco de lado, e este é um problema muito comum em empresas também, onde os prazos apertados, acabam fazendo os desenvolvedores escreverem códigos com duplicação, confusos e difíceis de fazer manutenção.

Após o inicio da utilização do sistema, o feedback dos usuários tem sido positivo, em 25 dias do sistema no ar, foram cadastrados 15 usuários e 16 itens, sendo que a grande maioria dos cadastros aconteceu após a divulgação na lista de emails do IME. Espera-se que esse número aumente, pois quanto mais usuários o sistema obtiver melhor será o serviço.

8.1 Planos Futuros

O sistema desenvolvido ficou dentro do esperado, com todas as funcionalidades pensadas implementadas e o sistema rodando para que a comunidade USP possa usá-lo, porém há ainda algumas melhorias que o sistema pode receber, a maioria delas foram sugestões dos usuários.

- Cadastrar o projeto no CCSL - O próximo passo para a melhoria do sistema será a sua inclusão na página do CCSL[6] para que aumente a divulgação do site na USP e incentive outras pessoas a contribuírem com o projeto, seja melhorando o sistema ou implementando-o em outros locais. Além disso o CCSL também daria um novo domínio para hospedar o site, o que ajudaria ainda mais na divulgação.
- FAQ do sistema - Conforme os usuários forem utilizando o sistema de contato pretendo colocar as dúvidas mais frequentes em uma página para ajudar os usuários a utilizar todo o potencial do sistema

[6]Centro de Competência em Software Livre

9 Parte subjetiva

9.1 Desafios

Na produção desse trabalho pude aprender muito sobre o desenvolvimento de um sistema web, tentei utilizar as tecnologias mais atuais principalmente para que eu pudesse aprender mais sobre elas, mesmo sabendo que daqui a alguns anos elas provavelmente já estarão obsoletas. Lidar com tecnologias novas foi o meu primeiro desafio, pois tive que estudar bastante sobre Rails e as Gems que eu pretendia utilizar, HTML5, coffee script e SASS. Tive sorte pois todas as ferramentas utilizadas tinham documentações muito bem feitas e isso otimizou muito a aprendizagem.

Ao iniciar o trabalho, quis fazê-lo com BDD e esse foi o segundo desafio, pois era necessário pensar no comportamento do sistema, fazer os testes e produzir código que fizesse esses testes passarem, e apesar de eu já ter tido alguma experiência com TDD, achei o inicio desse processo bem mais demorado do que deveria, porém com o tempo fui pegando o jeito e as coisas foram melhorando. O processo de BDD foi muito bom ao meu ver, pois tive um número muito pequeno de bugs.

A maior dificuldade em todo o sistema foi a implementação da interface do usuário, pois essa foi uma parte que eu sabia muito pouco e tive que ir quebrando a cabeça durante o processo. Nunca pensei que fosse tão complexo lidar com diferentes browsers e resoluções de tela. Porém gostei muito do resultado final, e acredito que os usuários do sistema também gostarão pois a interface é bonita e intuitiva.

Um desafio que não foi superado foi o uso de ontologias no sistema. As ontologias seriam usadas como um serviço onde o sistema pudesse buscar informações de classes de objetos para a busca pudesse devolver mais resultados relevantes para os usuários. Porém não consegui encontrar uma ontologia que cobrisse o domínio da aplicação, por isso acabamos descartando as ontologias e partindo para outros modos de melhorar as buscas.

9.2 Paralelos com as disciplinas do BCC

Acredito que a grande maioria das disciplinas de computação que eu cursei ajudaram de algum modo na produção desse trabalho. Porém algumas se destacam:

- **MAC0122 Princípios de Desenvolvimento de Algoritmos** - Na minha opinião a disciplina mais importante do curso de BCC, pois é nela que temos toda a base de computação que usaremos no resto do curso, além de ser onde temos nossos primeiros reais desafios na programação. Com certeza a lógica de programação que eu adquiri em 122 me ajudou muito a produzir alguns algoritmos desse trabalho.
- **MAC0211 Laboratório de Programação I** - Nessa disciplina tivemos nossa primeira experiência com um sistema mais complexo, desenvolvido em etapas e com muito mais código do que os EPs, nela também aprendemos a base para fazer testes e depuração de programas. Em 211 também aprendemos a utilizar diversas ferramentas do UNIX (como awk, grep, tail etc.), que de um modo indireto foram muito importantes para o desenvolvimento do trabalho.
- **MAC0242 Laboratório de Programação II** - Disciplina onde tivemos nossa primeira experiência com uma linguagem orientada a objetos, aprendemos os principais conceitos de orientação a objetos e desenvolvemos um projeto grande seguindo padrões de orientação a objetos e com uma interface gráfica.

- **MAC0426 Sistemas de Bancos de Dados** - A parte mais importante do sistema são os dados nele cadastrados, sejam de itens ou de usuários, por isso foi muito importante conhecer bancos de dados relacionais e saber usá-los.
- **MAC0332 Engenharia de Software** - Apesar de não ter utilizado as técnicas aprendidas em engenharia de software nesse trabalho, essa disciplina foi importante pois foi onde tive minha primeira experiência com um sistema web no curso de BCC. Nela descobri que sistemas web são muito diferentes de desktop e que o uso de arcabouços para auxiliar o desenvolvimento é muito importante, pois agiliza os processos e não obriga o desenvolvedor a "reinventar a roda ".
- **MAC0342 Laboratório de Programação Extrema** - Disciplina mais importante para o desenvolvimento desse trabalho, pois nela aprendi os princípios de métodos ágeis e tive a minha primeira experiência em um grande projeto de software, que diferente dos EPs presentes no curso, seria realmente utilizado por um grupo de usuários. além disso também tive minha primeira experiência com Rails nessa matéria, e isso me incentivou bastante a escolher essa tecnologia no desenvolvimento desse trabalho.

10 Referências

[1] "http://www.stm.sp.gov.br/index.php?option=com_content&view=article&id=3543:central-de-achados-e-perdidos-da-cptm-aqui-voce-pode-encontrar-o-que-perdeu
904:servicos-aos-usuarios&Itemid=18.

[2] http://agilemanifesto.org/iso/ptbr.

[3] "https://github.com/plataformatec/devise.

[4] https://github.com/ryanb/cancan.

[5] "https://github.com/thoughtbot/paperclip.

[6] "https://github.com/cucumber/cucumber/wiki.

[7] http://jquery.com/.

[8] http://www.businessinsider.com/biggest-websites-in-the-united-states-2013-2
op=1.

[9] "http://http://www.extremeprogramming.org/rules/userstories.html".

[10] Satam Alang. *Collective Inteligence in Action*. Manning, 2009.

[11] David Flanagan. *A linguagem de programação Ruby*. O'Reilly, 2009.

[12] Ralph Johnson, John Vlissides, Richard Helm, and Erich Gamma. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 2004.

[13] Dave Thomas and James David Heinemeier Hanson. *Desenvolvimento Web ágil com Rails*. bookman, 2008.