PARTE I

1. Introdução

O craigslist.com é um site de classificados e fóruns divididos por cidades, moderado pela comunidade de usuários e gratuito (o único tipo de post pago é a oferta de empregos em algumas cidades, aluguel de imóveis em Nova Iorque e ofertas de serviços adultos). Nele é possível se anunciar qualquer coisa, desde itens em seções de achados & perdidos até imóveis. Os anúncios são feitos dentro de categorias como shows, anúncios pessoais, livros, móveis, etc.

Provavelmente a principal característica de web 2.0 nesse site é a simplicidade. Ele possui uma interface muito enxuta e pouquíssimas funcionalidades.

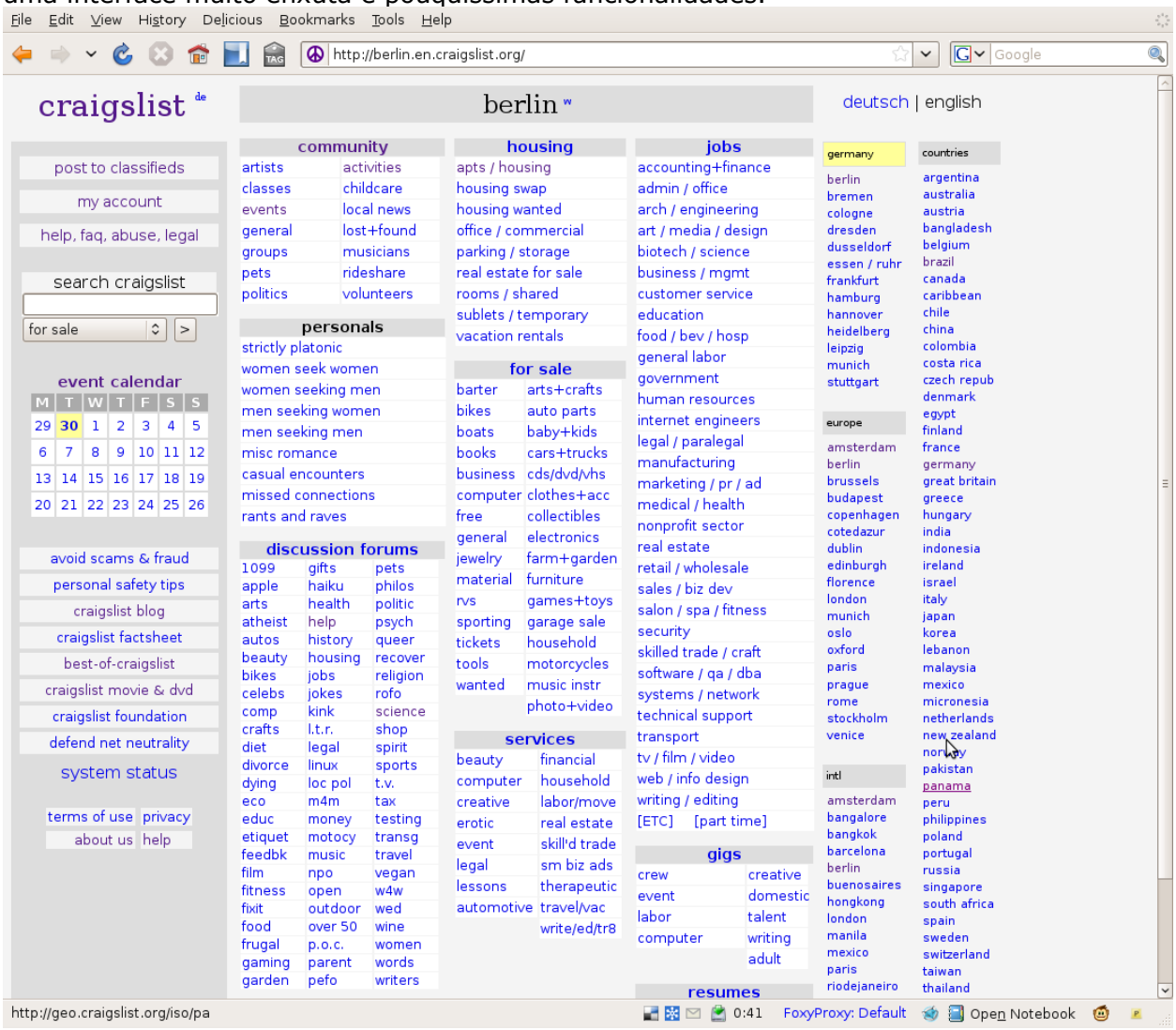

Figura 1 - Home

2. Características de web 2.0

O site é dividido basicamente em duas grandes partes:

2.1. Anúncios
2.1.1. Flags
Os usuários podem classificar um anúncio qualquer em quatro categorias:
* Categoria errada: categoria errada, discussão de outro anúncio
* Proibido: viola os termos de uso do site ou de outros posts
***** Spam / Excesso de posts: postado muito frequentemente, em múltiplas cidades ou categorias ou é muito comercial
* Melhores do craigslist: o anúncio aparece em uma lista dos melhores anúncios do site.

A FAQ informa que a comunidade não gosta de especulação sobre itens como ingressos, por exemplo, e costuma classificar esses anúncios como impróprios, o que acaba fazendo com que eles sejam retirados do site. Mas não há uma categoria específica para isso.

Casos como de material com quebra de copyright ou anúncios que denigrem a imagem de alguém não são administrados pela comunidade, mas sim pelo próprio site. Um usuário envia um email para os administradores que tomam as devidas ações - normalmente apenas removem o post.

Figura 2 - Anúncio e flags

2.1.2. Categorias
Todo anúncio deve ser criado em uma das seguintes categorias:

* Oferta de emprego
* Busca de emprego
* Oferta de casa
* Busca de casa
* Item a venda
* Item para comprar

* Oferta de pequenos serviços
* Busca de pequenos serviços
* Pessoais
* Comunidades
* Eventos

Além disso, as categorias possuem sub-categorias, como artistas, grupos e animais de estimações em comunidades, por exemplo.

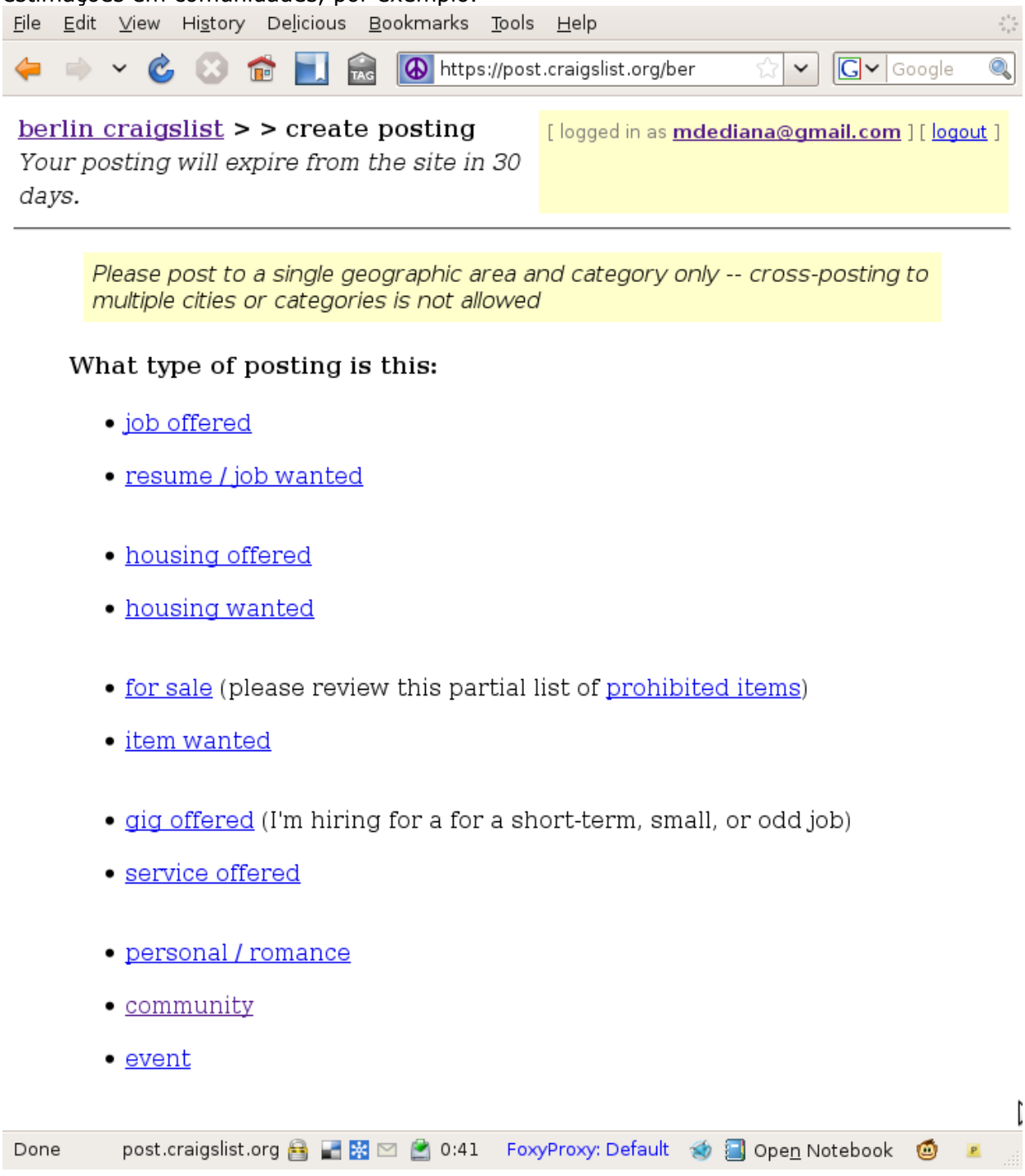

Figura 3 - Categorias de novo anúncio

2.1.3. Busca
Os usuários podem fazer busca por palavras-chave e categoria. Além disso, ao entrar em uma categoria específica, eles podem fazer buscas em sub-categorias.

2.2. Fóruns
2.2.1. Flags
Os usuários podem classificar um post qualquer em três categorias:
* conduta inapropriada
* categoria errada
* spam

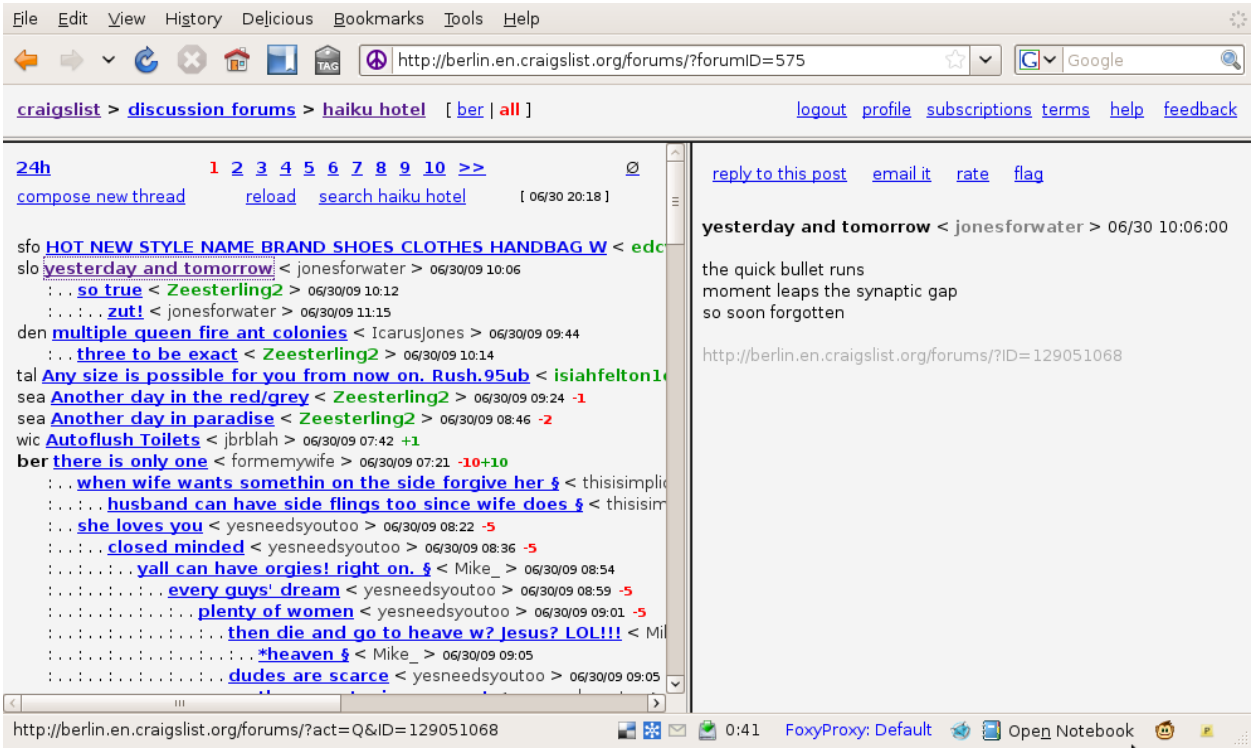

Figura 4 - Fórum e post

Figura 5 - Flags em posts

2.2.2. Notas

Os usuários possuem uma cota de notas que podem ser distribuídas pelos posts com uma cota diária (ou seja, há uma quantidade limitada de notas a ser dada), mas nas vezes em que visitamos o site ele apresentava problemas e pedia para voltar mais tarde.

PARTE II

A arquitetura utilizada no blog foi muito parecida com arquiteturas de aplicações web java, onde fez-se uso de frameworks conhecidos como Struts2, Spring e JPA/Hibernate. A escolha do Struts na versão 2.x foi devido à sua simplicidade e leveza, diferente da versão 1.x, onde o código era complexo e muito acoplado. Além disso, a enorme comunidade que suporta o framework, corrigindo defeitos e implementando novos plugins aumentou nossa velocidade de desenvolvimento. Já a utilização de Spring, apesar de ter uma certa complexidade em torno de sua configuração, facilitou no processo de injeção de dependência e gerenciamento de transações. Isso nos ajudou a simplificar o código, retirando do código das classes de domínio, os códigos responsáveis por problemas ortogonais (como certos tratamentos de exceções e transações). Para persistência de dados, escolhemos utilizar JPA como abstração e Hibernate como implementação concreta. Existe uma pequena curva de aprendizado quanto à essas ferramentas (como processo de mapeamento de entidades através das anotações, e escritas das consultas utilizando JPQL ou HQL), mas após isso, a velocidade de desenvolvimento aumenta, já que tira do programador a responsabilidade de escrever consultas SQL diretamente.

Foram utilizados também alguns recursos provenientes de contâineres web java, como filtros e listeners, utilizados para tratar permissão de acesso somente para usuários logados na área de administração e para a implementação do padrão de projeto "Open Session In View", que faz com que a conexão seja aberto no começo da requisição HTTP e fechada somente do fim da requisição, possibilitando assim o uso de lazy loading dentro de qualquer camada em que o objeto estiver. Repare que essa implementação já existe dentro do Spring, e o trabalho foi apenas configurá-lo.

Outra tentativa do grupo foi a utilização de alguns conceitos de DDD (Domain-Driven Design), onde criamos nossa linguagem ubíqua (ou onipresente) e a utilizamos por todo o projeto, desde a especificação até o código. Para isso, utilizamos padrões como *Repository Pattern* e conceitos como agregações, objetos de valor, entidades e serviços. Também tivemos em mente alguns conceitos *SOLID* durante o desenvolvimento, com o objetivo de escrever um código simples e de fácil leitura. Outro ítem importante que levamos em conta visando esse objetivo foi a refatoração de código, feita constantemente durante todas as fases do projeto. Trechos complexos de código e variáveis com nome pouco expressivos foram alvo constante desse processo. Acreditamos que hoje nosso código tenha uma alta legibilidade dispensando qualquer necessidade de comentário em trechos de código.

Observando cada pacote do projeto percebe-se a separação de responsabilidades. Tentamos ao máximo manter essa separação de forma clara, utilizando idéias do SRP *(Single Responsibility Principle)*. Uma breve explicação de cada um deles:

- O pacote br.usp.ime.blog.actions contém todas as Struts 2 Actions do blog. Eles representam as ações do front-end do blog, executando ações como visualizar post, enviar comentário, avaliar, e etc. Repare que todos eles herdam de BaseBlogAC, que contém certas ações padrões para todos os Actions, como exibição do lado direito (que contém informações como ranking, etc).
- O pacote br.usp.ime.blog.actions.admin contém as ações relacionadas à manutenção do blog, como novo post ou nova página.
- O pacote br.usp.ime.blog.exceptions contém um aspecto (utilizando Spring AOP) que trata lançamento de exceções no repositório, convertendo a exceção lançada para RepositorioException. Isso simplifica o código do repositório, evitando o tratamento da exceção em cada um dos métodos.
- O pacote br.usp.ime.blog.dominio contém as classes de domínio e todas as regras de negócio deles. É o principal pacote do projeto. Repare que todas as classes contém anotações JPA.

- O pacote br.usp.ime.blog.filtros contém o filtro que gerencia o processo de acesso às ações de administração. Somente usuários logados podem ter acesso, e esse filtro gerencia esse processo.
- O pacote br.usp.ime.blog.repositorios contém as interfaces dos repositórios. Em seguida temos duas implementações diferentes: uma usando JDBC puro e outra usando Hibernate. Obviamente a implementação em Hibernate é muito mais simples, devido ao auxílio do framework.
- Os pacotes br.usp.ime.blog.servicos.* cuidam de serviços epeciais que são feitos pela aplicação. O pacote ic por exemplo, é responsável pelos algoritmos de inteligência coletiva, assim como o pacote twitter é responsável pela integração com o Twitter e o pacote ranking é responsável pelos algoritmos de ranqueamento de posts do blog.
- O pacote br.usp.ime.blog.webservices contém a implementação dos webservices, tanto em SOAP quanto em REST.

Além disso, outras observações se fazem necessárias, como:

- Testes em JUnit e em Selenium foram efetuados, visando a execução de testes unitários e de aceitação.
- A utilização do Spring foi parte vital do sistema, já que o mesmo toma conta do processo de injeção de dependência (diminuindo o acoplamento entre as classes), cuida do aspecto (exceção dos blogs) e gerenciamento de transação (que foi feita utilizando-se AOP e inserida direto nas Actions).
- O framework Struts2 é responsável pela camada de apresentação. Ele torna o código simples e leve, já que diversos plugins facilitam o trabalho, como plugin para integração com Spring e para resultado em JSON.
- JQuery foi utilizado para facilitar a codificação de código no cliente (JavaScript) e chamadas assíncronas (AJAX).
- Utiliza-se o banco de dados MySQL, que é open-source e possui ótima performance.

Um ponto negativo da arquitetura é a falta de algum framework para gerenciar as dependências do projeto como o Maven, já que muitos frameworks são utilizados e por isso a necessidade de diversos arquivos .jar no projeto.

Quanto a participação dos integrantes do grupo, deve-se ressaltar a alta qualidade de comunicação que foi mantida durante todo o projeto. A primeira parte da implementação do projeto foi feita com programação pareada, ajudando muito o sincronismo de idéias, já que os próprios participantes eram os clientes. Devido a isso, todas as características emergiram de debates e nada foi implementado sem o consenso de ambos os integrantes. Na segunda parte, específica de cada tecnologia, na maioria das vezes uma pessoa era responsável por implementar a tecnologia no projeto e passar informações como dificuldades, vantagens e desvantagens que encontrou durante o processo. Em caso de frameworks com mesmo objetivo (como é o caso de Struts e JSF), os integrantes debatiam sobre qual seria a melhor opção para o problema e escolhiam qual seria utilizado na versão final. Já na parte de inteligência coletiva, optamos novamente por desenvolver juntos os algoritmos, já que eles possuíam uma certa dificuldade de implementação. Resumindo, ambos integrantes se dedicaram muito ao projeto e o trabalho dos dois foi fundamental para a conclusão do mesmo.

PARTE III

Achamos a disciplina em geral bastante proveitosa, em especial no que diz respeito a parte de inteligência coletiva. Algo que nos surpreendeu foi a forma como diversas técnicas vindas de áreas tão distintas quanto inteligência artificial e estatística se juntam para a criação da

inteligência coletiva. Além disso, também foi surpreendente ver como o uso dessas técnicas é relativamente simples, ainda mais com ajuda de projetos F/OSS já maduros.

A área que menos aproveitamos foi a parte de tecnologias, por já termos alguma experiência com Java. Mesmo assim, foi possível desenvolver alguns novos conceitos. A ressalva é o fato de a disciplina ter sido bastante trabalhosa, mesmo usando conceitos já conhecidos gastávamos uma média de 4 homem-horas/semana nos trabalhos.

Talvez valha a pena aumentar um pouco a carga sobre linguagens dinâmicas, que costumam ser as mais usadas nos grandes sites da web 2.0. Ao mesmo tempo, tecnologias como Spring, apesar de muito valiosas para o programador, talvez possam ficar de fora por não possuirem conceitos específicos de web.