Relatório Final

MAC0416 – Tópicos Especiais em Desenvolvimento para Web

Compartilhamento de letras de música

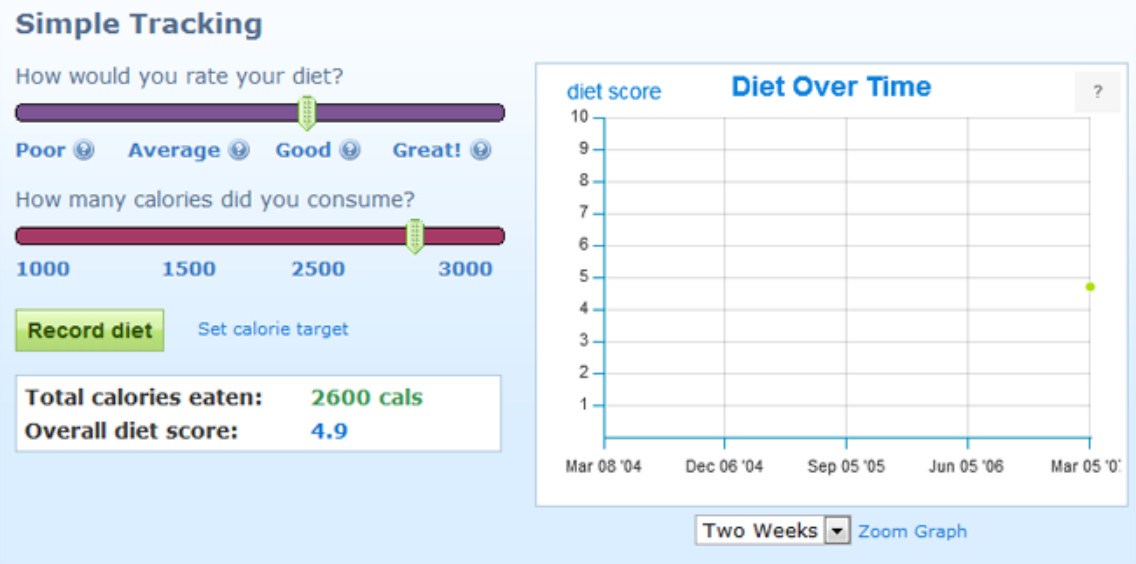
Nilo César Teixeira – Nº USP 2869389
Edson Wu Chen – Nº USP 3286203
01/07/2009

Parte I – Análise do site Traineo.com

Traineo.com é uma comunidade virtual de apoio à prática de exercícios físicos e redução de peso para qualquer tipo de pessoa. O site provê ferramentas, fóruns de discussão, e até mesmo motivadores para encorajar o internauta e auxiliá-lo a atingir suas metas pessoais. Todo usuário pode solicitar a motivação de qualquer um dos membros do site, e também pode se oferecer como "motivador".

Criado em meados de 2006, o site foi um dos pioneiros a implementar "features" da Web 2.0, principalmente em termos de interface de usuário, como "pop-overs", "slides" para entrada de dados, e gráficos indicadores do progresso do usuário.

Figura 1 – Slides para entrada de dados



Primeiramente o usuário configura suas metas: o peso que deseja alcançar e um prazo estipulado para isto. Então, passa a registrar diariamente suas atividades físicas no site, que vai coletando os dados e apresentando estatísticas enquanto o usuário progride.

Figura 2 – BMI (Índice de massa corporal) e gráfico de progresso

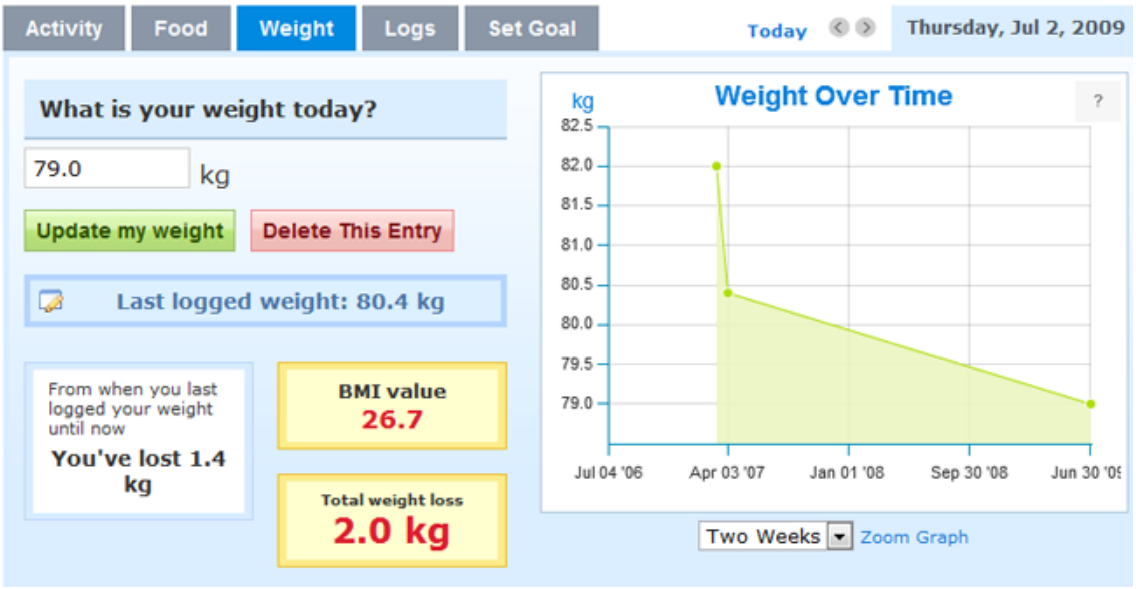
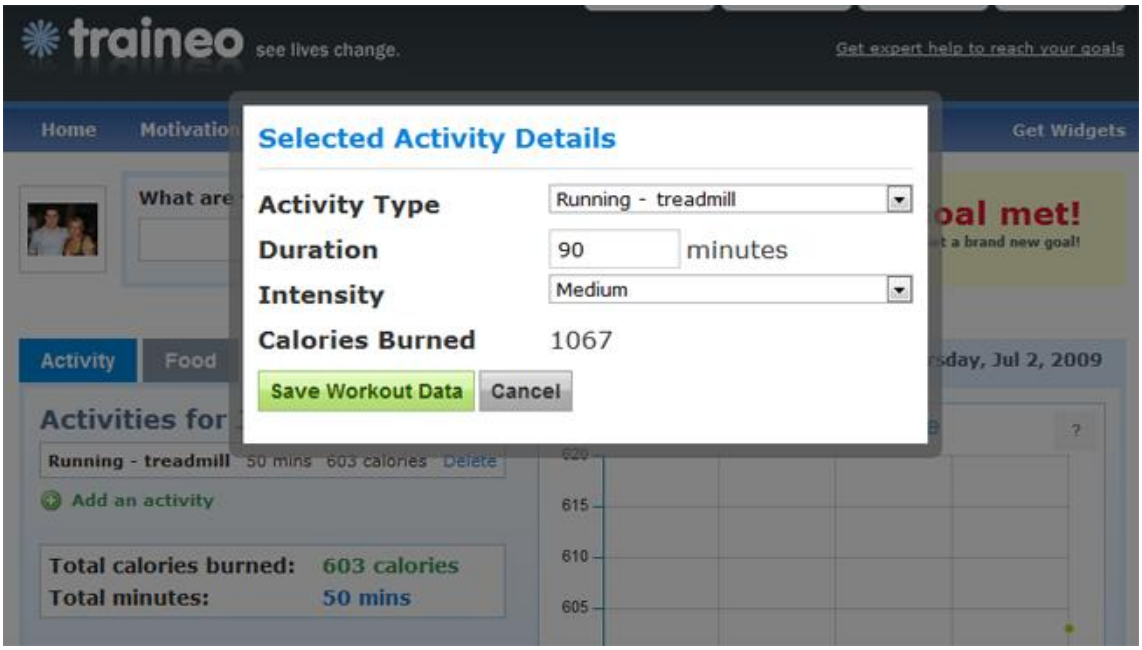


Figura 3 – Pop-overs para informar atividades físicas



O usuário pode também fazer uso dos grupos e fóruns de discussão, apesar do site concentrar sua base de usuários nos Estados Unidos. Além disso, o site apresenta problemas com "encoding" (ao se utilizar acentos nos "posts" dos fóruns, por exemplo) e há poucos brasileiros. Mesmo assim, vale a pena avaliar a ferramenta pelo propósito a que se dispõe, e decidir sair do sedentarismo com a ajuda do site e/ou um de seus concorrentes.

Parte II – Endeavour: parte objetiva

Durante a disciplina foi concebido um sistema de cadastro e compartilhamento de letras de música, denominado Endeavour, que em sua totalidade visa ser uma comunidade (ou rede social) agregada em torno de um objetivo comum: compartilhar letras de música em um ambiente online, visando maximizar sua corretude. Visávamos para o projeto:

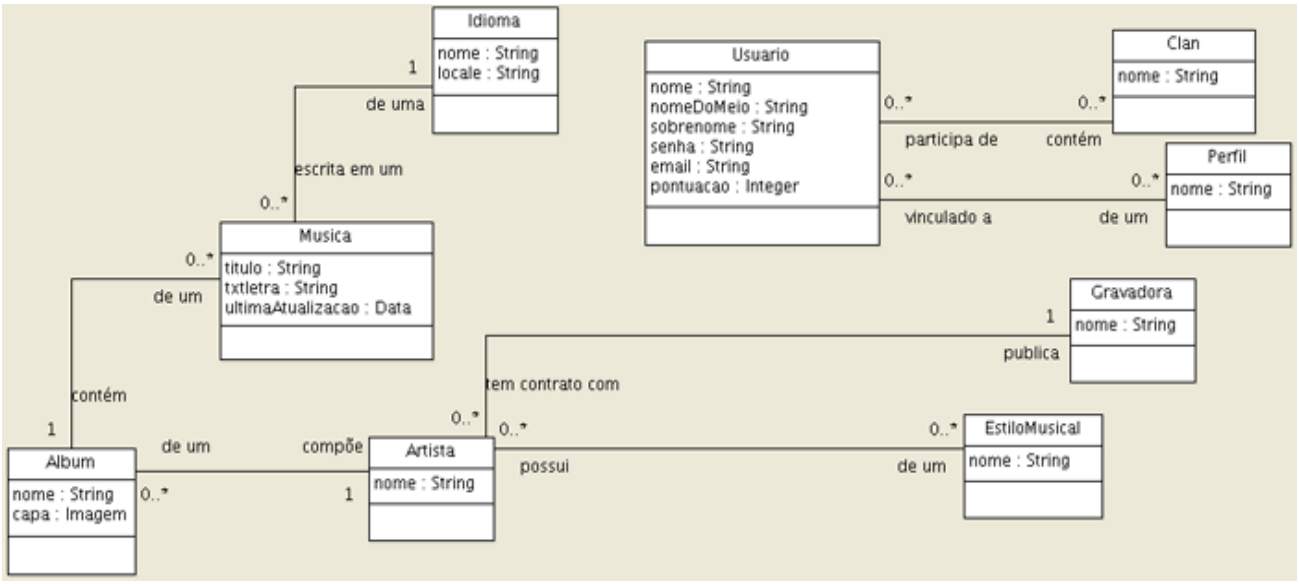
Objetivos gerais:

- Ser um agregador de letras de músicas;
- Reforçar a corretude das letras;
- Unir pessoas com afinidade por escutar música acompanhando letras.

Objetivos específicos:

- Cadastro de letras com metadados (faixa, álbum, artista, estilo, gravadora);
- Usuários recebem classificação por quantidade de conteúdo postado;
- Usuários têm sua classificação revista quando correções ao seu conteúdo são feitas;
- Cada usuário pode se associar a amigos e grupos de trabalho, compartilhando classificação.

Assim usuários se registram e informam as letras num ambiente que fomente a corretude e a participação. Posto isto, no começo do semestre as entidades levantadas foram as seguintes:



Tais entidades persistiram no projeto até o fim do semestre, tanto em termos de classes no projeto como de tabelas no banco de dados.

A pergunta que nos fazíamos é se seria possível implementar todo o escopo em um prazo tão curto.

Com o avanço da disciplina, após analisarmos todas as tecnologias de Web 2.0 e Inteligência Coletiva que tínhamos que implementar, e o tempo disponível, resolvemos focar na funcionalidade do item principal do produto: **edição de letras**. É sobre a evolução deste "feature" do sistema que iremos discorrer.

Tecnologias implementadas no sistema

Inicialmente tínhamos que nos familiarizar com o desenvolvimento Web utilizando Java. Para isto, de grande valia foi a utilização da apostila FJ-21 da Caelum (www.caelum.com.br), que cobriu os tópicos principais do que precisávamos no começo.

Assim, na etapa I do trabalho, implementamos o sistema utilizando Servlets e HSQLDB (este enquanto não tínhamos um ambiente de "deploy" configurado, o que foi providenciado ao longo do semestre pelo professor e por um dos alunos da disciplina), sem nenhuma das tecnologias Web 2.0. Apesar disto, já utilizávamos MVC, embora com um "controller" não-padronizado.

Esta nossa versão tinha funcionalidade bem limitada e serviu para que pudéssemos colocar em prática os conceitos Web aprendidos no começo do curso (utilizamos XHTML 1.1 conforme solicitado pelo prof.).

Na etapa II, começamos a realmente implementar algo interessante. Segue abaixo o cronograma conforme a tecnologia utilizada:

Ajax/Comet/Json

A idéia implementada na interface é nunca precisar de um recarregamento total da página a cada ação executada. Assim, **todas** as operações geram requisições AJAX para o servidor, e a resposta Json é tratada pelo Javascript.

Inicialmente **criamos uma view específica para retornar objetos Json.** Isto permaneceu até o fim do projeto. Assim, toda vez que precisávamos gerar uma resposta Json para o cliente, chamamos esta view e passamos para ela uma string Json para renderização no cliente.

A primeira chamada AJAX implementada foi popular o "combobox" de álbuns sempre que escolhido um artista no menu superior.

Figura 4 – Chamada AJAX para popular álbuns

Posteriormente, ao escolher um álbum, implementamos também chamadas AJAX para determinar as músicas do álbum e montá-las em uma interface que possibilitasse sua edição. Cada letra é carregada assim que se escolhe o título da música, e o corpo do campo de edição é preenchido com a letra, que então pode ser salva através de comando da interface.

jQuery

Dizemos que o jQuery foi o que definiu a cara da nossa aplicação. Com ele pudemos fazer animações que enriqueceram a interface gráfica, tais como: "slide up and down" ao escolher um álbum, "highlight" de títulos ao se passar o mouse sobre eles, "fade in and out" do campo de edição de letras no processo de carregamento de uma letra.

Também fizemos com ele **todas** as chamadas assíncronas (AJAX) e parseamento de objetos Json (respostas).

JSF

Utilizamos Java Server Faces para implementar o "feature" de login. Contudo, achamos que o framework de login não deveria ser uma preocupação para o protótipo que desejávamos implementar, e este feature ficou fora da versão final (entregue) do projeto.

Struts

Utilizando Struts 1.3, achamos pertinente refatorar o MVC e implementar a internacionalização do aplicativo.

Cada instância de controller passou a utilizar-se dos "forwards", conferiu-se a extensão "**.do**" para cada instância de controller, e também usamos os **MessageResources**, colocando todos os strings apresentados na interface nestes arquivos "resource", tornando o aplicativo legível em Português e Inglês.

Hibernate

Esta sem dúvida foi a mais trabalhosa das implementações. Utilizando "annotations", repassamos o código de **cada** classe Model (ou seja, cada entidade), e fizemos com que estas entidades pudessem ser persistidas usando o framework (anteriormente todo o código SQL estava manual e encapsulado em "DAOs" – Data Access Objects).

A ferramenta se mostrou mais poderosa ainda depois que decidimos alternar o banco de dados para Mysql: **todo** o código DDL de manutenção da estrutura do banco de dados passou a ser gerenciado pelo Hibernate, então sempre que precisávamos mudar de banco de dados para efeito de desenvolvimento bastava alterar **uma linha** no código de configuração e deixar o framework recriar e popular toda a estrutura.

jUnit/Selenium

Após a implementação do Hibernate, estávamos incertos quanto à corretude do código que havíamos feito. Assim, criamos dezenas de testes de aceitação em jUnit que cobriram 100% do código de acesso ao banco de dados, para nos certificarmos de que tudo estava funcionando.

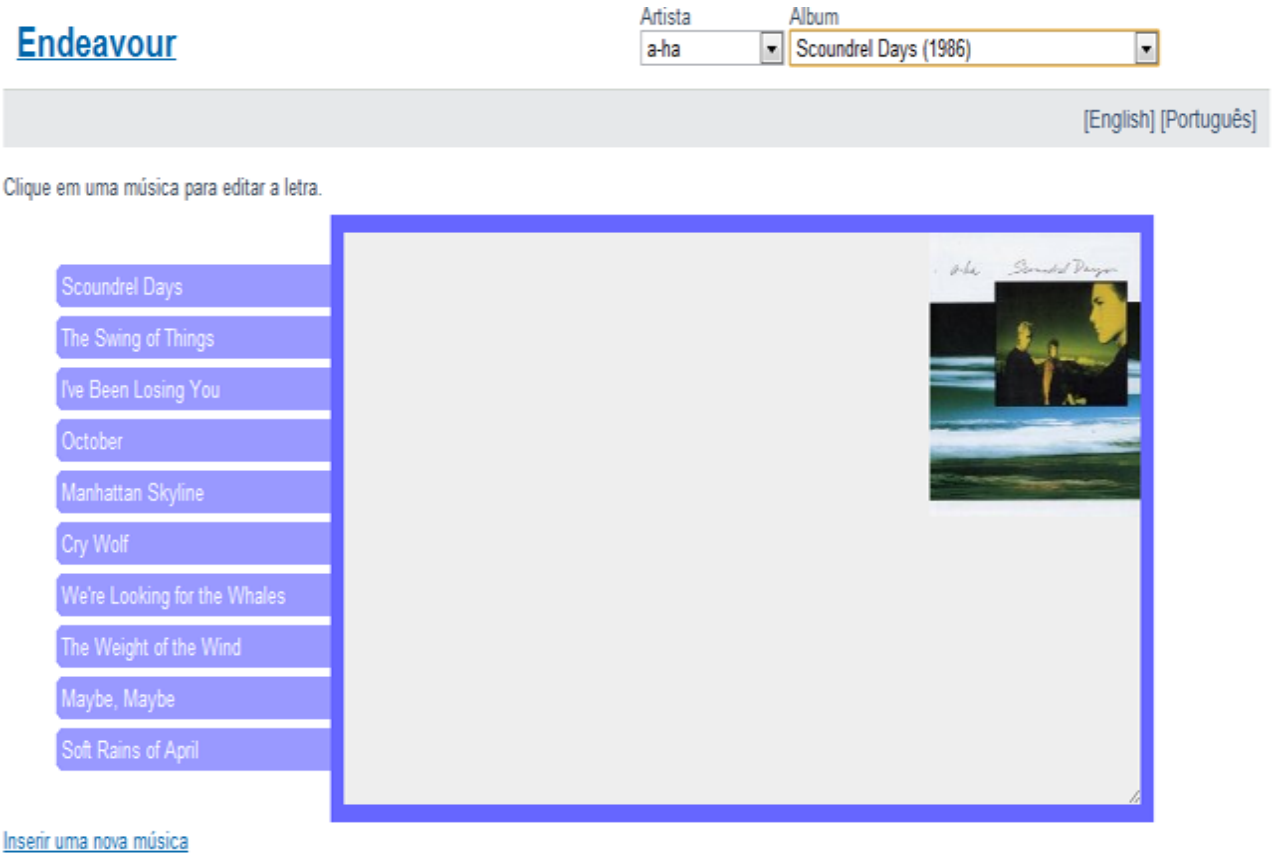
Além disso, criamos um teste de interface com Selenium bem simples, mas poderoso, que rodou junto com os testes jUnit.

Webservices

Tínhamos uma idéia fixa desde o começo do projeto: fazer com que o sistema fosse capaz de buscar capas de álbuns na Internet.

Com alguma pesquisa, conseguimos encontrar um provedor destas imagens (Amazon), e fizemos com que cada chamada para carregar as letras de um álbum também fosse capaz de carregar uma imagem pequena da capa no próprio editor. Tal imagem não ficou obstrutiva, pois ocupou um espaço que quase nunca é sobrescrito por um verso ("top-right").

Figura 5 – Carregamento da capa do álbum



Inteligência coletiva

Tivemos pouco tempo para implementar esta tecnologia, então escolhemos um caminho coerente: já que todos os algoritmos apresentados na disciplina agem sobre texto, **resolvemos cadastrar todas as letras dos álbuns existentes no sistema, "na mão"**!

Após isto, implementamos um "feature" curioso: achar a letra que **mais se parece** com a letra atual. Isto concluiu nosso trabalho entregue.

Parte III – Sobre a discliplina

Finalmente, nossas considerações sobre a disciplina.

Acho que foi senso comum dizer que a disciplina foi trabalhosa. Acredito que ela tenha sido algo além disso. Na minha opinião (Nilo), ela foi informativa da realidade do desenvolvimento estado-da-arte para web, e foi uma grande oportunidade de praticar o que o mercado valoriza em termos de desenvolvimento Java.

Tivemos muitas desistências (inclusive a minha dupla foi uma delas), que acharam que os prazos eram surreais e não pudessem ser cumpridos.

Acredito que isso foi relevado ao longo da disciplina, para desespero dos que desistiram no começo, e isto foi necessário: sem o alongamento de prazos, acho que realmente não seria factível implementar o cronograma desta matéria.

Gostaríamos de ter mais tempo e realmente absorver todo o conteúdo, mas tivemos que ser seletivos visto que esta foi uma entre todas as matérias do semestre.

Também achei criativos os problemas “estudo de caso” dados nas duas provas, e recomendo esta disciplina para os futuros alunos.

Obrigado.