Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Informática

Relatório de Preparação de Estágio

Ano lectivo 2007/2008

Título

Subtítulo

Ricardo Horta e Vale Otero dos Santos

Supervisão na UC: Prof. Carlos Vaz
Supervisão na Mentes Virtuais: Eng. Marcos Garcia

Julho, 2008

Resumo

O presente relatório descreve o trabalho efectuado durante o estágio de carácter profissional integrado no Mestrado em Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

O estágio foi dividido em duas fases, sendo a primeira correspondente a uma carga horária de 16 horas semanais, onde se fez um estudo da *framework* Ruby on Rails, bem como a sua experimentação, pondo em prática os conhecimentos adquiridos. Foi também feito um pequeno comparativo com outras alternativas ao desenvolvimento de aplicações web.

Na segunda fase, trabalhando a tempo inteiro, o estagiário desenvolveu uma aplicação de anúncios utilizando uma série de tecnologias e metodologias descritas neste documento. Paralelamente foram estudadas algumas alternativas a problemas concretos no desenvolvimento da aplicação, bem como as soluções mais adequadas, que são de extremo interesse para a empresa em que o estágio foi realizado, adquirindo-se um *know-how* no desenvolvimento em Ruby on Rails.

Os objectivos do estágio foram alcançados, cumprindo-se as tarefas inicialmente propostas. **TODO: mais**

Índice

Índice de figuras

Índice de tabelas

Glossário

1. Introdução

O presente documento tem como objectivo relatar o estágio realizado na Mentes Virtuais pelo aluno Ricardo Santos. Aqui se descreve todo o estudo e desenvolvimento dos resultados finais, assim como todo o contexto e actividades em que o estagiário esteve envolvido. Tenta-se assim manter o equilíbrio entre a metodologia utilizada, os processos envolvidos e os resultados obtidos, descrevendo os momentos que constituíram a experiência profissional em contexto empresarial, um importante componente na formação do aluno.

1.1. Contextualização

A arquitectura base das aplicações web não é muito variável, e como tal é possível construir um suporte comum. Surgem assim as *frameworks* de desenvolvimento de aplicações web, construídas para facilitar a modularidade e a não repetição de código, reduzindo o tempo de execução do trabalho, consequentemente os custos de produção, aumentando a rentabilidade da empresa.

Ruby on Rails é uma das *frameworks* com mais entusiastas actualmente, pela sua simplicidade e agilidade, tendo sido adoptada pela Mentes Virtuais nos últimos anos e havendo já resultados bastante satisfatórios. No entanto ainda há alguma falta de *know-how* acerca de algumas tecnologias envolvidas ou das melhores práticas no desenvolvimento de aplicações, dado que a migração é recente e a própria *framework* tem apenas quatro anos, sofrendo constantes alterações desde então.

1.2. Motivação e objectivos

PRIMEIRA FASE

O objectivo deste estágio é, não só estudar a viabilidade desta migração, mas também analisar objectivamente os motivos da escolha. Pretende-se ao mesmo tempo fazer um estudo sobre a framework em questão e suas potencialidades, bem como desenvolver pequenas aplicações, usando a *framework* Ruby on Rails, de modo a ganhar algum *know-how* para desenvolvimento de aplicações com esta ferramenta.

SEGUNDA FASE

Roomblick, aplicação proposta para desenvolvimento no contexto do presente estágio surge de propostas concretas **TODO: completar**. Esta aplicação surge da análise da necessidade de um sistema web de arrendamento de quartos em Portugal, com uma boa interface e um conjunto de funcionalidades chave, que até aqui não existia.

1.3. Instituição

A associação de empresas para uma rede de inovação em Aveiro – INOVA-RIA, constituída a 29 de Julho de 2003, tem por objectivo a consolidação de um *cluster* de telecomunicações que contribua para o desenvolvimento da região de Aveiro.

Do conjunto de empresas é possível destacar associados como a PT Inovação, a grande impulsionadora desta associação empresarial. Existem ainda outros associados não menos importantes dos quais importa realçar a Mentes Virtuais, empresa onde será efectuado o estágio curricular.

A Mentes Virtuais é uma empresa de consultoria informática e soluções multimédia que existe desde Janeiro de 2003. Possui um vasto conhecimento na concepção de aplicações web e serviços móveis sustentado pela experiência dos seus colaboradores.

2. Ruby on Rails

Ruby on Rails (Rails) é uma framework desenvolvida em Ruby – uma linguagem de programação não compilada, orientada a objectos, com uma sintaxe limpa – com vista à implementação de aplicações web com recurso a uma base de dados. Foi inventada por David Heinemeier Hansson (1979 – Copenhaga), programador dinamarquês, tendo sido utilizada numa aplicação que estava a desenvolver – Basecamp (www.basecamphq.com) – e disponível ao público em Julho de 2004.

Dando prioridade ao modo simples de fazer as coisas e tirando partido da linguagem Ruby permite uma alta produtividade no desenvolvimento de aplicações web. Meses após o lançamento oficial esta framework passou de desconhecida a uma das frameworks de escolha para implementação de uma variedade das chamadas aplicações Web 2.0, não sendo apenas uma moda junto dos mais entusiastas mas também adoptada por multinacionais para desenvolvimento de aplicações web robustas. **TODO: devo dar exemplos?**

Uma aplicação em Rails usa convenções simples de programação que permitem um mínimo de configuração e segue as seguintes filosofias chave:

- **Don't repeat yourself (DRY)**: reduzir duplicações, particularmente quando se trata de programação. A informação está localizada num único sítio não ambíguo.
- **Convention over configuration (CoC)**: significa que apenas temos de especificar aspectos não convencionais da nossa aplicação.

3. Outras *frameworks*

Existem dezenas de outras *frameworks*, algumas há bastante mais tempo que Ruby on Rails. De seguida apresentam-se algumas das mais usadas e conhecidas no mundo do desenvolvimento de aplicações web.

3.1. Django

Django é uma *framework Open Source* desenvolvida em Python, uma linguagem bastante semelhante a Perl ou Ruby. A linguagem Python é usada de modo transversal, da mesma forma que Ruby é usado na framework Ruby on Rails.

O objectivo principal é facilitar a criação de aplicações complexas e com muitos acessos a uma base de dados de forma rápida e simples. Foca-se na reutilização e ligação entre os diversos componentes bem como no desenvolvimento ágil e o princípio DRY.

Com uma interface de administração gerada automaticamente e com autenticação integrada, os projectos em Django estão mais virados para CMS sendo menos flexível do que Rails.

3.2. J2EE

Apesar de Ruby on Rails ser uma framework nova e excitante para muitos, o core da arquitectura segue os padrões presentes em J2EE. A filosofia de desenvolvimento de aplicações web é o que diferencia estas duas frameworks. Rails dá primazia a código explícito ao invés de ficheiros de configuração e a natureza dinâmica do Ruby gera muito código no runtime. Grande parte da framework Ruby on Rails foi criada como um projecto único e o desenvolvimento de uma aplicação beneficia de um conjunto de componentes homogéneos. Em contraste, uma stack típica de J2EE tende em ser construída de componentes diferentes (um conjunto de bons componentes entre um grande leque) que são geralmente desenvolvidos independentemente e são extensivamente usados ficheiros XML para os ligar e tratar da configuração.

Seria, no entanto, um erro dispensar-se completamente J2EE em favor de Rails. J2EE é um standard muito bem estabelecido com inúmeras implementações sólidas. É uma tecnologia com provas dadas e robustez comprovada.

No entanto trabalhar com uma framework em Java implica conhecer bem muitas das tecnologias envolvidas (Struts, Hibernate, Spring, Axis, etc.) sendo o seu número muito elevado e por vezes bastante complexo e imbricado, tornando a escolha das tecnologias adequadas a um projecto uma tarefa difícil. Para se atingir um nível de produtividade semelhante ao de outras frameworks exige-se assim muito mais prática e habituação. Além de ser necessária muito mais configurações numa aplicação em J2EE, existe o overhead do processo de compilação e muitas vezes os erros cometidos no

código não são detectados na compilação mas apenas no runtime e consequentemente será difícil reagir rapidamente a mudanças, característica fundamental numa interface web.

Apesar de tudo os programadores Java têm ao dispor um vasto leque de escolhas. Além de haver um grande número de frameworks J2EE é relativamente fácil fazer integração com a infinidade de aplicações Java já desenvolvidos para os mais variados propósitos.

4. Estudo comparativo de *frameworks*

Neste capítulo descreve-se um estudo comparativo, feito pelo estagiário, das *frameworks* apresentadas anteriormente. Este estudo tem como principal objectivo experimentar e avaliar cada uma das *frameworks* não tanto a nível de *performance* mas de facilidade de uso e rapidez no desenvolvimento de uma mesma aplicação web.

Django

- automatic admin interface bastante avançado
- generic views
- auth/auth
- il8n
- mod_python

php

- manutenção

- nao escrever SQL

php -> mto verbose, dificil de manter

5. Processo de desenvolvimento

5.1. Ambiente de desenvolvimento

O ambiente de desenvolvimento usado neste projecto é constituído por várias ferramentas e, fundamentalmente, a *framework* Ruby on Rails. Todas as tecnologias envolvidas têm como característica comum serem *Open Source*, fomentando a sustentabilidade e independência do projecto em relação a tecnologias proprietárias.

Ruby on Rails

Subversion?

Nginx?

Mais?

5.2. Método de trabalho na empresa

O desenvolvimento de um produto na Mentes Virtuais, principalmente na área de aplicações web, não implica o desenvolvimento de documentação extensiva, uma vez que as equipas de trabalho são pequenas (duas a quatro pessoas). Existe um grande diálogo entre os elementos de um projecto, sendo encorajada a troca de ideias e conhecimento entre eles.

No entanto, todos os projectos têm uma documentação básica (requisitos e alguns protótipos quando necessário) e têm um controlo de features e bugs num sistema de gestão de projectos – Redmine (www.redmine.org) instalado num servidor interno onde se pode seguir o desenvolvimento de um projecto, bem como adicionar ou editar documentação já realizada, ficando acessível imediatamente, de modo simples a todos os elementos, podendo haver colaboração na sua especificação (Figura 1).

Este sistema permite ainda alertar por email os participantes de um projecto de alterações efectuadas, criar um diagrama de Gantt e consultar um calendário de metas.

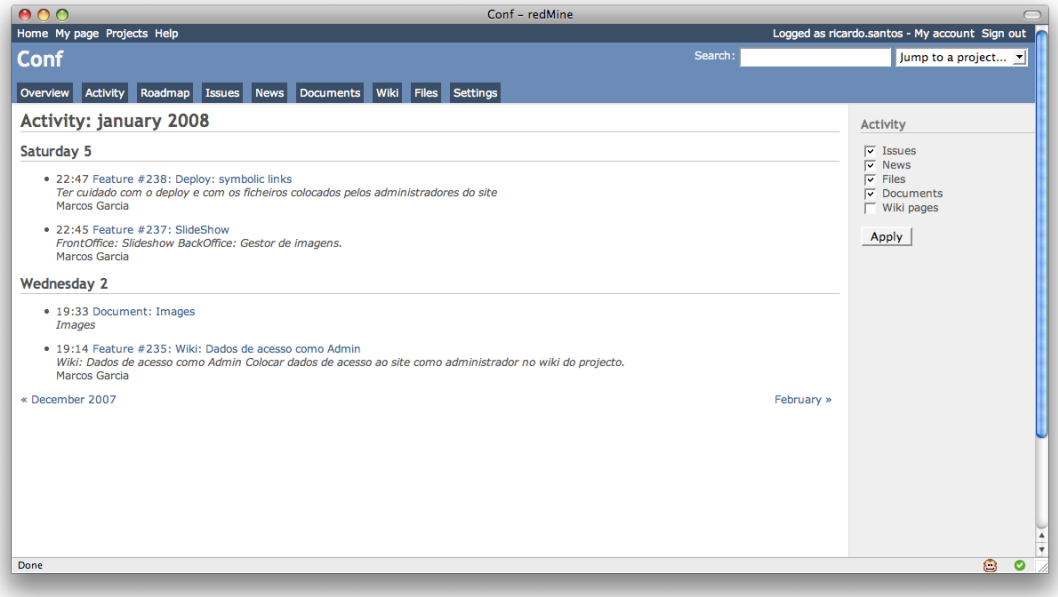

Figura 1 – Exemplo de uma página de um projecto usando o gestor de projectos redmine.

Além disso todos os projectos têm um repositório de controlo de versões, sendo usado um servidor SVN para o efeito. Desta forma anulam-se os

problemas adjacentes ao desenvolvimento em paralelo por diferentes pessoas numa equipa.

A primeira versão funcional da aplicação é sempre posta em produção o mais rapidamente possível para que o cliente possa dar o seu parecer e normalmente surgem novos requisitos. À medida que as novas funcionalidades são implementadas a aplicação em produção é actualizada, obedecendo-se ao ciclo presente na Figura 2.

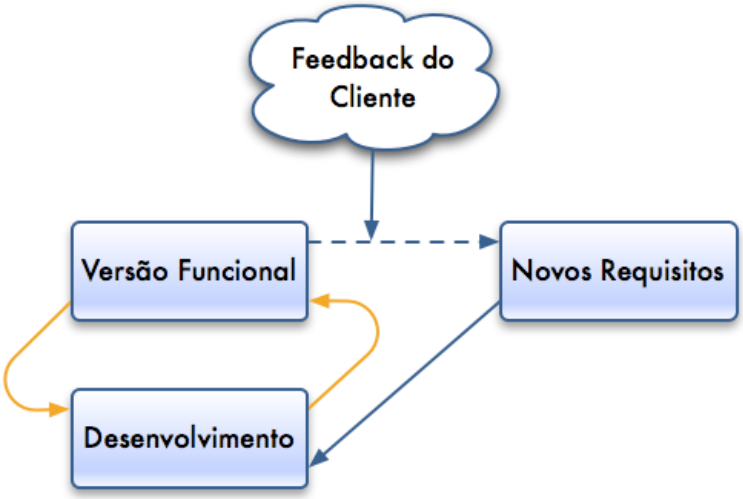

Figura 2 – Ciclo de desenvolvimento de um projecto. A amarelo está o ciclo de desenvolvimento sem recurso ao cliente. O feedback do cliente pode ou não levar ao aparecimento de novos requisitos.

A equipa de desenvolvimento pode também, sempre que se proporcione, propor novas funcionalidades, havendo assim uma comunicação bidireccional mais ou menos regular, entre ambas as partes.

5.3. Test Driven Development

Normalmente, quanto mais rapidamente é necessário desenvolver código menos testes são feitos para acelerar o processo de desenvolvimento. No entanto, a escassez de testes tende a gerar menos produtividade e a tornar o código menos robusto. Menor produtividade implica mais pressão, criando-se assim um ciclo vicioso.

Test Driven Development (TDD) é uma característica da *Extreme Programming* e permite atribuir à mesma pessoa funções de desenvolvimento e teste em simultâneo. Seguindo uma metodologia TDD é necessário escrever um conjunto exaustivo de testes, associados a pequenos detalhes funcionais da aplicação em questão.

Segundo alguns autores **TODO: referência (link)?** é necessário testar todas as potenciais fontes de problema. Ou seja, não é necessário testar nada que não possa gerar problemas. Isto implica alguma subjectividade quando se avaliam

as potenciais fontes, sendo obrigatório haver um entendimento muito grande se se tartar de uma equipa e não de um único indivíduo.

Com TDD este risco é eliminado, uma vez que antes da implementação de código vem a escrita dos testes para todas as novas funcionalidades a ser implementadas. Estes testes vão falhar inicialmente, havendo depois uma escrita de código com vista a que a aplicação passe nos testes.

Após a escrita dos testes para determinadas funcionalidades e de desenvolvido o código para passar nesses testes volta-se ao início. Assim, a metodologa TDD segue um ciclo como o apresentado na Figura 3.

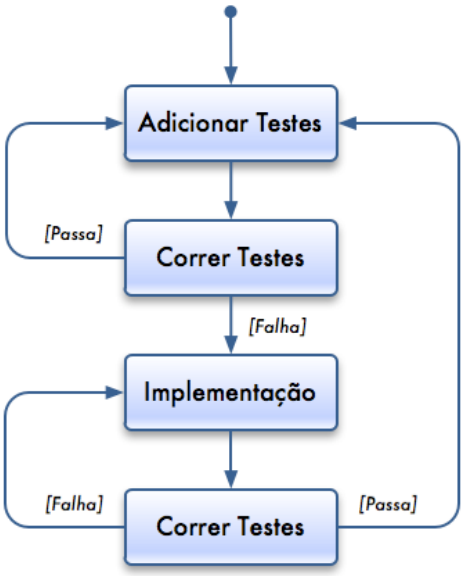

Figura 3 – Ciclo de escrita de testes seguindo Test Driven Development.

Todo o trabalho desenvolvido é assim mantido numa só equipa, com controlo absoluto de todo o código desenvolvido em paralelo aos testes da aplicação em pequenos ciclos.

A *framework* Ruby on Rails está bem preparada para este tipo de testes, sendo criada uma directoria "test" para cada projecto, onde são gerados ficheiros de teste, automaticamente, sempre que é criado um novo modelo. Os testes são divididos em três tipos:

- **Testes unitários:** sobre os modelos da aplicação.
- **Testes funcionais:** aplicados aos controladores e à interacção entre modelos.
- **Testes de integração:** mais gerais, guiados por 'histórias', que verificam as interacções entre várias acções suportadas pela aplicação, envolvendo diferentes controladores.

Os testes são escritos em Ruby, havendo um conjunto de funções simples já definidas na própria linguagem, tendo sido extendidas com funções expecíficas e Rails (para testes directamente ligados à *framework* como rotas, redireccionamentos, etc.).

Para corer os testes basta, através de uma linha de comandos, correr um commando, cujo *output* será semelhante ao da **TODO: figura**.

5.4. Riscos Iniciais

Um projecto de desenvolvimento de um produto com as características do *Roomblick* acarreta um conjunto de riscos, associados à utilização de tecnologias em constante desenvolvimento, bem como à introdução de conceitos que se revelaram inovadores. Para este caso em particular, a utilização de tecnologias e ferramentas para as quais não existia *know-how* interno até então foi considerado um risco de algum peso, que não poderia ser descartado no planeamento. A ocorrência no projecto de um evento relacionado com este risco teria efeitos negativos no seu desenvolvimento, com consequências que poderiam envolver o não cumprimento de todos os objectivos propostos. No decorrer da fase de desenvolvimento, as funcionalidades disponibilizadas pelas *frameworks* em uso revelaram, no entanto, potencial na sua aplicabilidade transversal ao projecto.

5.5. Planeamento

O plano inicialmente proposto para o desenvolvimento do projecto, disponível no Anexo #, sofreu alterações ao longo do tempo, com o intuito de o fazer corresponder à realidade e adaptá-lo às alterações.

Breve descrição do processo de planeamento e do planeamento em si…

6. Roomblick

6.1. Outras soluções

O mercado de arrendamento ou partilha de quartos *on-line* não um mercado inexplorado. Assim, existem algumas soluções que foram alvo de estudo preliminary no sentido de apurar funcionalidades que poderiam alargar o leque de requisitos do *Roomblick*.

As soluções existentes deveriam ser direccionadas para a população portuguesa, dado que, numa primeira fase, a aplicação será condicionada a Portugal.

6.1.1. Bquarto

http://www.bquarto.pt

Esta é uma aplicação desenvolvida por uma empresa nacional, e que contempla quartos apenas em Portugal.

Permite procurar por alguém que procura quarto para arrendar ou alguém que queira arrendar um quarto.

A interface é confusa, com HTML de fraca qualidade (tabelas em todo o site).

Erros de HTML -> Failed validation, 48 Errors

Erros de javascript frequentes

Permite pesquisa sem registo

Sistema de contacto com "interessa-me" bastante prático

Utilização:

"Tenho Quartos": 272 em Lisboa, 16 em Aveiro

"Procuro Quartos": 544 Lisboa, 32 Aveiro

Não dá para ver quão antigos são os anúncios, havendo apenas uma "data de último login" do responsável (e são todos recentes, com máximo de 1 mês)

Dá para ver quantos utilizadores estão on-line e varia geralmente entre 10~15 durante o dia.

Figura 4 – Aspecto da página de entrada de BQuarto.pt.

Google maps para visualização das cidades apenas, sem qualquer ligação aos quartos.

6.1.2. Easyquarto

www.easyquarto.com.pt

Aplicação internacional, sendo este domínio apenas para Portugal.

Suporta várias línguas.

Com imensos problemas de português, como se pod ever na Figura 5, entre eles:

- más traduções com português do Brasil em muitas situações (contato, se torne um associado, etc.);
- erros ortográficos;
- frases ou títulos em inglês;
- erros gramaticais.

Figura 5 – Exemplos de erros de português na aplicação "easyquarto"

Não permite pesquisa sem registo

Formulários muito extensos

Interface confusa

Erros de HTML -> Failed validation, 583 Errors

Google Maps com a localização do quarto

Localização mal inserida quando se fornece a morada!

Só tem uma fotografia pequena por cada quarto

Muitas opções quer na publicação de novo anúncio quer na pesquisa

Duas componentes: procura de arrendamento e disponibilização de arrendamento

Muitas vezes não se percebe em qual das duas componentes estamos

Pesquisas de "preciso de um quarto" e "tenho um quarto" estão trocadas!!

Utilização:

“Tenho Quartos”: 378 em Lisboa, 15 em Aveiro

“Preciso Quartos”: 1884 Lisboa, 100 Aveiro

Anúncios mais datados que no bquarto (máximo 3 meses)

O bquarto tem muito mais detalhes acerca do quarto que o easyroom, e os formulários estão de forma mais organizada.

6.2. Especificação

7. Referências

1. **http://oodt.jpl.nasa.gov/better-web-app.mov**

Vídeo comparativo de frameworks de desenvolvimento de aplicações web.