# Ruby on Rails aplicado práticas de Banco de Dados Ágeis

Reinaldo Saraiva
Orientador: Ricardo Ramos

IFPI

Agosto / 2009

# Roteiro

# Introdução

- A indústria de software vem buscando reduzir os riscos dos projetos constantemente ao passar dos anos
- Surgiu em meados da década de 90 uma corrente filosófica com proposta mais focada nos aspectos humanos dos projetos de software
- Em 2001,17 profissionais se reuniram em Utah (EUA) para redigir o Manifesto pelo Desenvolvimento Ágil de Software
- O manifesto estabelece um conjunto de valores que são adotados nos projetos ágeis:
  - Indivíduos e interações ao invés de processos e ferramentas;
  - Software funcionando ao invés de documentação abrangente;
  - Colaboração com o cliente ao invés de negociação de contratos e
  - Responder a mudanças ao invés de seguir um plano.

# Introdução

- Crescimento dos modelos evolutivos, como: Rational Unified Process (RUP), Extreme Programming (XP), Agile Unified Process (AUP) e Scrum.
- Adoção de práticas que permitam simplicidade e facilidades para profissional de dados
- Diante deste cenário foi criado a framework Ruby on Rails, que além de ser uma excelente ferramenta para desenvolvimento de aplicações web, ainda permite aplicar práticas de banco de dados agéis

# Filosofia Rails

- DRY (“Don’t Repeat Yourself”) - sugere que escrever o mesmo código várias vezes é uma coisa ruim.
- Convenção ao invés de Configuração - significa que o Rails faz suposições sobre o que você quer fazer e como você estará fazendo isto, em vez de deixá-lo mudar cada minúscula coisa através de intermináveis arquivos de configuração.
- REST é um modelo para as aplicações web serem organizadas através de recursos e verbos HTTP padrão é o modo mais rápido para proceder.

# Arquitetura Rails

O Rails é organizado usando a arquietura Modelo, Visão e Controle, bastante conhecido como MVC.
Os benefícios desta arquitetura são:

- Isolação entre a lógica de negócios e a interface de usuário
- Facilidade de manter o código DRY
- Manter claro onde tipos de código diferentes pertencem para facilitar a manutenção

# Colaboração entre DBAs e Desenvolvedores

COMUNICAÇÃO ! ! !

Esta colaboração pode ser facilmente obtida com Rails, principalmente por sua arquitetura e princípios, que facilitam o aprendizado do indivíduo e da organização - essenciais para aumentar a eficiência em relação ao tempo. Sem o aprendizado os erros são constantes, o que representa desperdício de recursos.

# Todo mundo recebe a sua própria base de dados exemplo

Rails e rodar em três ambientes de execução diferentes:

- O ambiente development (desenvolvimento) é usado em seu próprio computador enquanto você interage manualmente com a aplicação;
- O ambiente test (teste) é usado para rodar testes automatizados ;
- O ambiente production (produção) é usado quando a aplicação está pronta para executar em ambiente de produção;

# Integração Contínua com desenvolvedores

- O Rails tem suporte a integração contínua do esquema do banco de dados através das migrations
- ActiveRecord faz o trabalho de ORM(Object-Relational Mapping), além de controle das mudanças ocorridas no banco de dados
- Migrations é a forma conveniente de alterarmos o banco de dados de uma maneira organizada e estruturada
- O ActiveRecord marca as migrations que foram executadas, precisando apenas atualizar o código

# Testando banco de dados

- As bases de dados são testadas principalmente para ajudar a estabilizar o desenvolvimento de uma aplicação
- Os testes também permitem verificar as migrações tanto das base de dados de testes como das legadas ou produção

# Fixtures

- Fixtures são uma forma de organizar os dados de teste
- As fixtures são a forma que o Rails encontrou para ter amostra de dados
- São independentes de banco de dados e assume um formato: YAML.

Abaixo um exemplo de arquivo YAML:

Listing 1: Arquivo de fixture YAML

```
1 reinaldo:
2         nome: Reinaldo Saraiva do Carmo
3         data_nascimento: 1978-05-07
4         profissao: Gerente de TI
5 katiana:
6         name: Katiana Ferreira Oliveira Saraiva
7         data_nascimento: 1976-11-22
8         profissao: administradora
```

# Refatoração em banco de dados

- É quando uma simples mudança no esquema de uma base de dados melhora a sua concepção(projeto), embora mantendo simultaneamente a sua semântica
- Para automatizar muitas das refatorações é essencial para o banco de dados que as alterações de esquema e a migração de dados sejam realizadas automaticamente por intermédio de alguma ferramenta
- Em Rails, é pratícavel usufruir de refatorações automáticas, visto que é possível lidar com entidades e tabelas, implementadas a um mapeamento mais direto. Isso leva para um dos pilares do Rails que é a Convenção, ao invés de Código e Não repetição de código.

# Conclusões

- Permite efetivamente uma maior colaboração entre DBA'S e desenvolvedores - Humanização do processo
- Estreitamento da comunicação agilizando as atividades concebidas no projeto de banco de dados
- Permite fácil manutenção
- Elevada eficiência evitando desperdício de recurso

# Vamos conversar um pouco mais?

# Referências


- Akita, F. (2006). Repensando a web com Rails. Brasport.
- Ambler, S. (2003). Agile Database Techniques: Efective Strategies for the Agile Software
- Developer. Wiley Application Development.
- Fowler, M. (2003). Evolutionary database design. http://www.martinfowler.com/articles/evodb.html, 1:15.
- Sadage P. J., S. A. (2006). Refactoring Databases: Evolutionary Database Design. Hardcover.
- Thomas, D. (2005). Agile Web Development with Rails: A Pragmatic Guide. Pragmatic Bookshelf.