

Ruby On Rails

Com.pensar 2016



PUC Minas
Poços de Caldas

Luiz Alberto Ferreira Gomes

Curso de Ciência da Computação

28 de abril de 2016

Agenda

- 1 Apresentação do Curso
- 2 Aplicação Web
- 3 Arquiteturas de WebApps?
- 4 Por Dentro do Rails
- 5 Bibliografia Recomendada

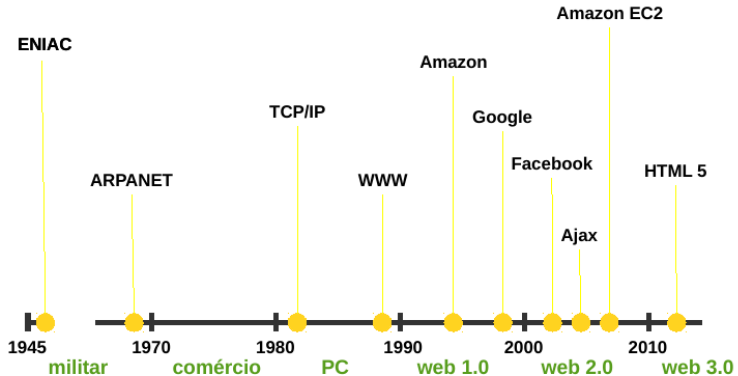
Apresentação do Curso

- Este curso tem como objetivo explorar o **desenvolvimento de aplicações web** considerando **padrões de projetos** fundamentais e filosofias associadas a **arquiteturas modernas** de aplicações web , juntamente com os seus principais componentes.
- Ao final deste curso, espera-se que o aluno seja capaz de:
 - projetar, desenvolver e publicar uma aplicação web;
 - entender os principais **componentes** da arquitetura web apps e como eles se interagem;
 - utilizar a plataforma Ruby on Rails;
 - compreender melhor as **práticas** modernas de engenharia de software.

Conteúdo Programático

Data	Módulo
18/09	Introdução e Conceituação
25/09	Ruby on Rails
02/10	Interação com Banco de Dados
09/10	A Linguagem de Programação Ruby
16/10	A Linguagem de Programação Ruby
23/10	Middleware
30/10	Interface com o Usuário

Histórico



Web 1.0, 2.0 e 3.0

- **Web 1.0** : páginas estáticas e primeiros modelos de negócios.
- **Web 2.0** : interactividade(Ajax), redes sociais e comércio eletrônico.
- **Web 3.0** : 'Web Inteligente', interpretação da informação auxiliada por máquina
 - exemplo: sistemas de recomendação.
- Base tecnológica da Web 2.0 e 3.0.
 - javascript, xml, json(ajax).
 - interoperabilidade via Web Services.
 - infraestrutura via modelos de **computação em nuvem** (IAAS, PAAS e SAAS)
 - aplicações móveis

Modelos de Computação em Nuvem (1)

- **IAAS (Infrastructure As A Service)** : fornece a infraestrutura computacional física ou máquinas virtuais e outros recursos discos, firewalls, endereços IP e etc.
 - exemplos: Amazon EC2, Windows Azure, Google Compute Engine.
- **PAAS (Platform as a Service)** : fornece plataformas computacionais que tipicamente incluem sistemas operacionais, ambientes para execução de programas, bancos de dados, servidores web e etc.
 - exemplos: AWS Elastic Beanstalk, Windows Azure, Heroku e Google App Engine
- **SAAS (Software as a Service)** : fornece acesso sob demanda às aplicações de software, sem que o usuário tem que se preocupar com sua instalação, configuração e execução.
 - exemplos: Google Apps e Microsoft 365.

Agenda

- 1 Apresentação do Curso
- 2 Aplicação Web**
- 3 Arquiteturas de WebApps?
- 4 Por Dentro do Rails
- 5 Bibliografia Recomendada

Aplicação Web (1)

Definição(Aplicação Web)

Uma **aplicação web** é aquela que acessada pelos usuários por meio de uma **rede de computadores**, utiliza um **navegador** (em inglês: *browser*); e consiste de uma coleção de **scripts** no cliente e no servidor, páginas **HTML** e outros recursos que podem estar espalhados por vários servidores. Ele é acessada pelos usuários via **um endereço** que faz referência a um servidor web (por exemplo: www.inf.pucpcaldadas.br).

- Exemplos: webmail, lojas virtuais, homebanking, wikis, blogs e etc.

Aplicação Web (2)

- Há um pouco mais do que isso:
 - Rede de Computadores:
 - a **Internet**, um sistema global de redes de computadores interconectadas.
 - utiliza o conjunto de protocolos TCP/IP.
 - Web (World Wide Web):
 - um sistema de documentos (em inglês: *web pages*) **vinculados** que são acessados através da Internet via protocolo HTTP.
 - Web pages contêm documentos **hypermedia**: textos, gráficos, imagens, vídeos e outros recursos multimídia, juntamente com *hiperlinks* para outras páginas
 - **Hiperlinks** formam a **estrutura básica** da Web.
 - A estrutura da Web é a que a torna **útil** e de **valor**.
- Vantagens:
 - **Conveniência** pela utilização um web browser como cliente.
 - **Compatibilidade** inerente entre plataformas.

Aplicação Web (3)

- Habilidade de **atualizar** e **manter** as aplicações web sem instalação e distribuição de software em vários clientes em potencial.
- **Redução** dos custos de TI.

■ **Desvantagens:**

- Interfaces com usuário ainda **não são tão boas** quanto as das aplicações tradicionais.
- Maior risco de **comprometimento** da **privacidade** e **segurança dos dados**.
- Mais **difícil** de **desenvolver** e **depurar** do que uma aplicação tradicional, pois existem mais partes a se considerar.

Agenda

- 1 Apresentação do Curso
- 2 Aplicação Web
- 3 Arquiteturas de WebApps?**
- 4 Por Dentro do Rails
- 5 Bibliografia Recomendada

Arquiteturas de Aplicações Web (1)

- As aplicações **web modernas** envolvem uma quantidade significativa de **complexidade**.
 - especialmente no lado do servidor.
- Uma típica aplicação web envolve **inúmeros protocolos, linguagens de programação e tecnologias** que compõem a pilha de tecnologia web.
- Desenvolver, manter e ampliar uma aplicação web complexa é **difícil**.
 - mas, construindo-o usando uma **base de princípios de sólidos de projeto** pode-se simplificar cada uma dessas tarefas.
- Engenheiros de software usam **abstrações** para lidar com este tipo de complexidade.
 - *Design patterns* fornecem abstrações úteis para sistemas orientados a objetos.

Design Patterns (1)

Definição (Design Patterns)

Um padrão de projeto é uma descrição da **colaboração de objetos** que interagem para resolver um problema de software em geral dentro de um contexto particular.

- Um design pattern é um **modelo abstrato** que pode ser aplicado recorrentemente.
- A idéia é aplicar padrões de projeto, a fim de **resolver problemas específicos** que ocorrem durante a construção de sistemas reais.
- Os padrões de projeto fornecem uma maneira de **comunicar** as soluções em um projeto, ou seja, é a terminologia que engenheiros de software usam para falar sobre projetos.

Modelo Cliente-Servidor (1)

- A arquitetura **cliente-servidor** é a arquitetura mais básica para descrever a cooperação entre os componentes de uma aplicação web.
- A arquitetura cliente-servidor pode ser subdividida em:
 - **servidor** que "escuta" por requisições e fornece os serviços ou recursos de acordo com cada uma.
 - **cliente** que estabelece a conexão com o servidor para requisitar serviços ou recursos.
- Existe um protocolo **request/response** associado com qualquer arquitetura cliente-servidor.

Modelo Cliente-Servidor (2)

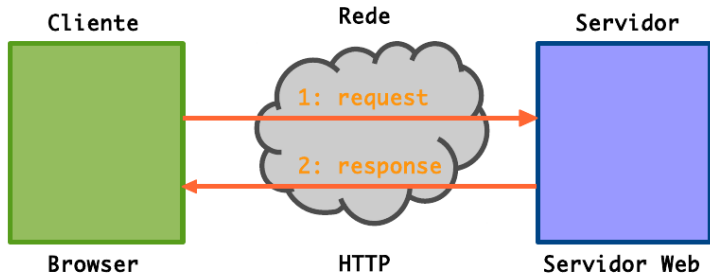


Figura: Arquitetura cliente servidor.

Modelo Cliente Servidor (1)

- É sem dúvida é o padrão de projeto de arquitetura mais conhecido
- O ponto chave de uma arquitetura cliente-servidor é **distribuir** os componentes de uma aplicação entre o cliente o servidor de alguma forma.
 - o servidor realiza as tarefas, consultas e transações
 - o cliente fica com uma responsabilidade menor: a de receber informações
- A fim de construir aplicações web complexas, vários design patterns ajudam a **organizar** como peças são dispostas dentro da arquitetura cliente-servidor.

Arquitetura N-Tier (1)

Definição (Arquitetura N-Tier)

A arquitetura n-tier é um *design pattern* muito útil que estrutura o modelo cliente-servidor.

- Este padrão de projeto é baseado no conceito de **quebrar** um sistema em partes diferentes ou camadas que podem ser separados fisicamente:
 - cada camada é responsável por fornecer uma **funcionalidade específica** ou coesa.
 - uma camada apenas interage com as **camadas adjacentes** a ela por meio de uma **estrutura** bem definida por meio de **interfaces**.

Arquitetura N-Tier (2)

Exemplos (Arquitetura 2-Tier)

- Servidores de impressão
- Aplicações web antigas:
 - Interface com o usuário (navegador) residia no cliente (thin).
 - Servidor fornecia as páginas estáticas (HTML).
 - Interface entre os dois via *Hypertext Transfer Protocol* (HTTP).
- Camadas **adicionais** aparecem quando a **funcionalidade** do aplicativo é ainda **mais dividida**.
- Quais são as vantagens de um tal projeto?
 - A abstração fornece um meio para **gerenciar** a complexidade.
 - Camadas podem ser atualizados ou substituídos de forma **independente** a medida que os requisitos ou tecnologia.

Arquitetura N-Tier (3)

- a nova só precisa usar as **mesmas interfaces** que a antiga utilizada.
- Ele fornece um **equilíbrio** entre inovação e padronização.
- Sistemas tendem a ser muito mais **fáceis** de construir, manter e atualizar.

Arquitetura 3-Tiers (1)

- Uma das mais comuns é a arquitetura em 3 camadas:
 - Apresentação
 - a interface com o usuário.
 - Aplicação (lógica)
 - recupera modifica e/ou exclui dados na camada de dados, e envia os resultados do processamento para a camada de apresentação.
 - Camada de dados
 - a fonte dos dados associados ao aplicativo.
- As aplicações web modernas frequentemente são construídas **utilizando** uma arquitetura em 3 camadas:
 - Apresentação
 - o navegador web do usuário.
 - Aplicação (lógica)

Arquitetura 3-Tiers (2)

- o servidor web e lógica associada com **geração** de conteúdo web dinâmico.
- por exemplo, a coleta e formatação do resultados de uma pesquisa.
- Camada de dados
 - um banco de dados.

Arquitetura 6-Tiers para Aplicações Web (1)

- A camada de aplicação é frequentemente subdividida em dois níveis:
 - Camada de lógica de negócios
 - modelos os **objetos de negócios** associados ao aplicativo, por exemplo, contas, estoques, etc.
 - captura as regras de negócio associadas a esses objetos.
 - Camada de acesso a dados
 - responsável por acessar os dados e passá-los para a camada de lógica de negócios.
 - por exemplo, saldos de contas, transações, etc.
- A camada de apresentação é muitas vezes subdividida em dois níveis:
 - Camada de clientes
 - os componentes da interface do usuário do lado do cliente.

Arquitetura 6-Tiers para Aplicações Web (2)

- Apresentação camada de lógica
 - scripts do lado do servidor para a geração de páginas web.
- Finalmente, o servidor web é muitas vezes separados em sua própria camada da Web e o servidor de banco de dados na sua camada de dados.

Arquitetura 6-Tiers para Aplicações Web (3)

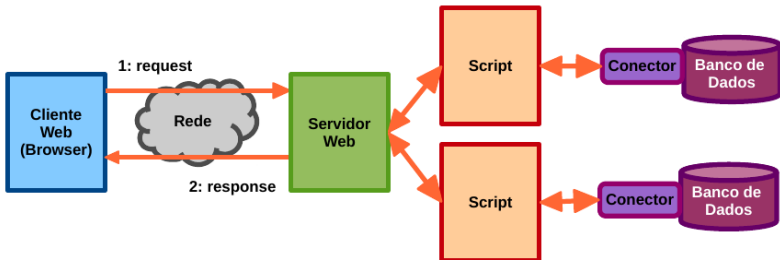


Figura: Arquitetura em 6 Camadas.

Para Saber Mais

- <https://www.ruby-lang.org/en/>
 - referência oficial da linguagem Ruby onde a toda a sua documentação está disponível para ser consultada.
- <http://rubyonrails.org/>
 - referência oficial do framework Rails onde a toda a sua documentação está disponível para ser consultada.
- <http://www.codecademy.com/pt/tracks/ruby>
 - interessante curso iterativo em português sobre a linguagem Ruby.

Agenda

- 1 Apresentação do Curso
- 2 Aplicação Web
- 3 Arquiteturas de WebApps?
- 4 Por Dentro do Rails**
- 5 Bibliografia Recomendada

Por Dentro do Rails (1)

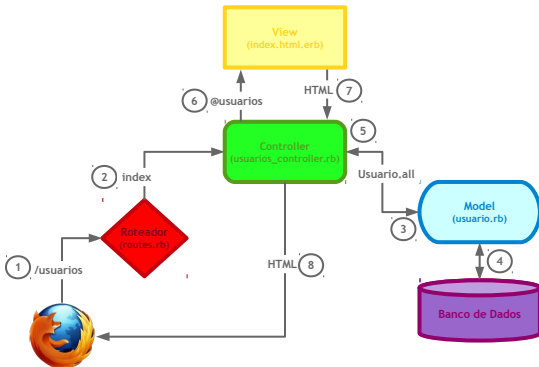
- Ruby on Rails (Rails) é framework para o desenvolvimento de aplicações web
- Construído utilizando a linguagem Ruby
- 100% open-source - MIT license
- Fornece a pilha completa para Web apps
- Lançado em 2004 e continua evoluindo rapidamente
- Algumas empresas que utilizam Rails: Twitter, Hulu, GitHub, Yellow Pages e etc

Por Dentro do Rails (2)

- Rails é uma **gem** Ruby (gem é um pacote Ruby)
- Rails fornece uma extenso conjunto de geradores de código e scripts de automação de testes
- Um conjunto de ferramentas adicionais são fornecidos como parte do ecossistema Rails:
 - **Rake** - utilitário similar ao **make do Unix** para criar e migrar bancos de dados, limpar sessões de uma Web app
 - **WEBrick** - servidor web de desenvolvimento para execução de aplicações Rails
 - **SQLite** - um servidor de banco de dados simples pré-instalado como o Rails
 - **Rack Middleware** - interface padronizado para interação entre um servidor web e uma Web App

Model-View-Controller

- O framework Rails é contruído em cima do Design Pattern Model View Controller(MVC):



Agenda

- 1 Apresentação do Curso
- 2 Aplicação Web
- 3 Arquiteturas de WebApps?
- 4 Por Dentro do Rails
- 5 Bibliografia Recomendada**

Bibliografia Recomendada
