

Desenvolvimento de Aplicações Web com Ruby on Rails

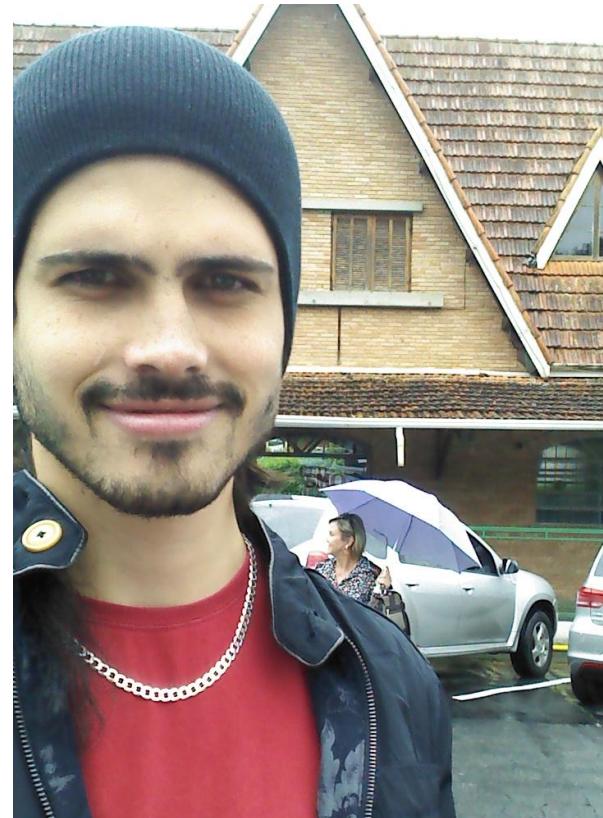
Arthur de Moura Del Esposte - esposte@ime.usp.br

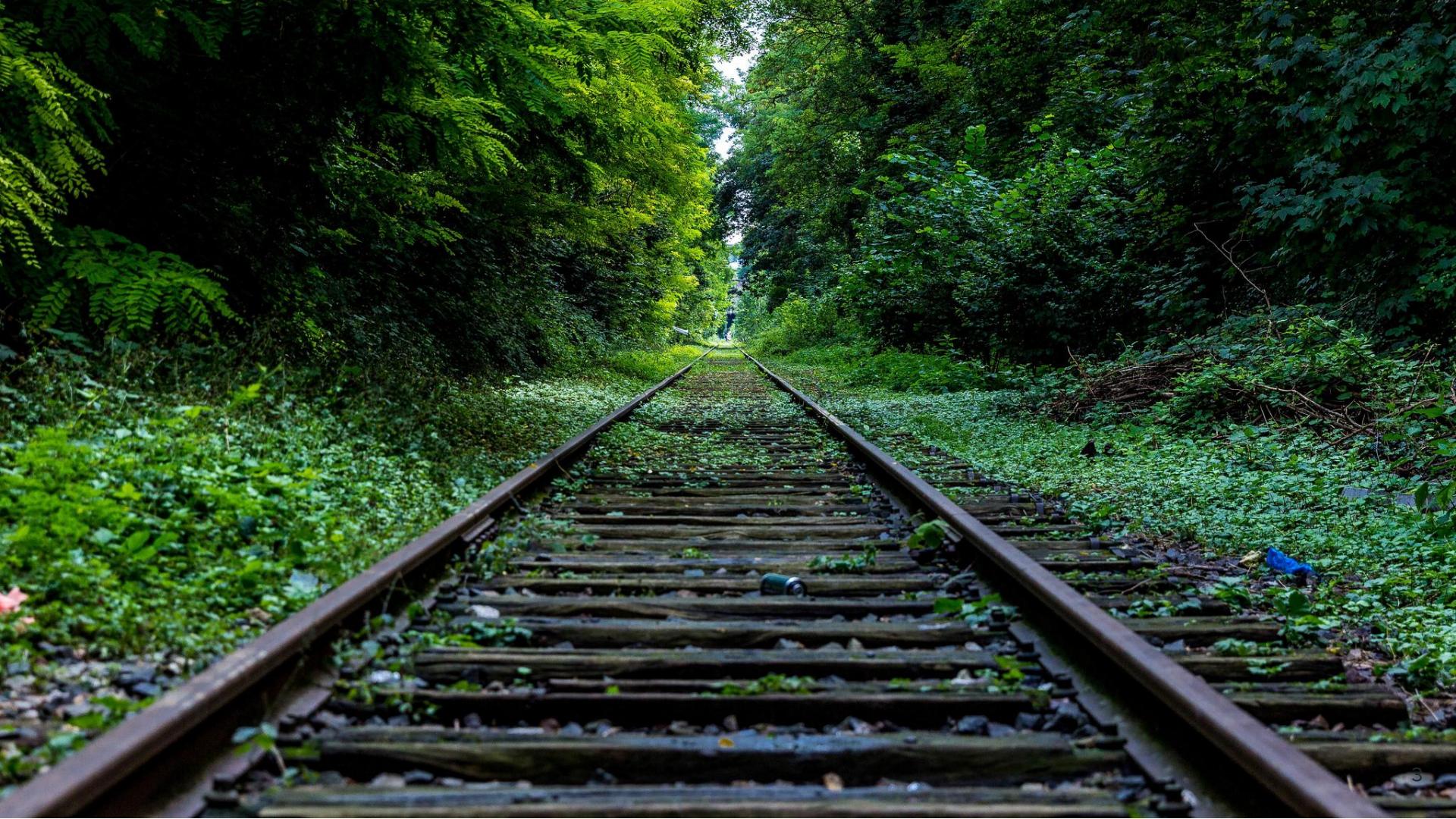


By Arthur Del Esposte licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0)

Apresentação

- Engenheiro de Software pela **Universidade de Brasília**
- Mestrando em Ciência da Computação pelo **IME - USP**
- Pesquisa sobre Arquiteturas Distribuídas em Plataformas de Cidades Inteligentes
- Desenvolvedor de **Software Livre**
- Outros interesses incluem: **Música, Colecionismo, Viagens, ect...**





Livre

Maturidade

Ferramentas
Modernas para
Web

Full-stack

Popular

Grande
Comunidade

Gems

Ágil





**KEEP CALM
AND
CODE IN
RUBY**

Rápido e Fácil?



Rápido e Fácil?



Exercite seu
conhecimento sempre!



Muito para aprender e exercitar!

- Ruby
- Arquitetura de aplicações Web
- Tecnologias Web - HTML, CSS e JavaScript
- Banco de Dados
- Testes automatizados
- Controle de Versão

Quem usa Rails?



KICKSTARTER



[Mais histórias de sucesso em Ruby](#)

Bibliografia

- Armando Fox, David Patterson. Construindo Software como Serviço: Uma Abordagem Ágil Usando Computação em Nuvem
- Michael Hartl. Ruby on Rails Tutorial
- Caelum. Desenvolvimento Ágil para Web com Ruby on Rails 4

Recursos Online

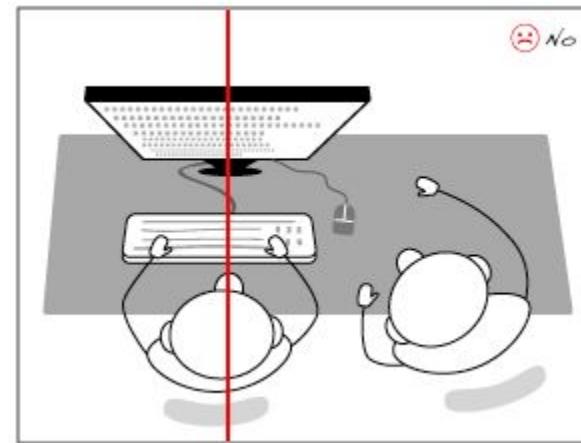
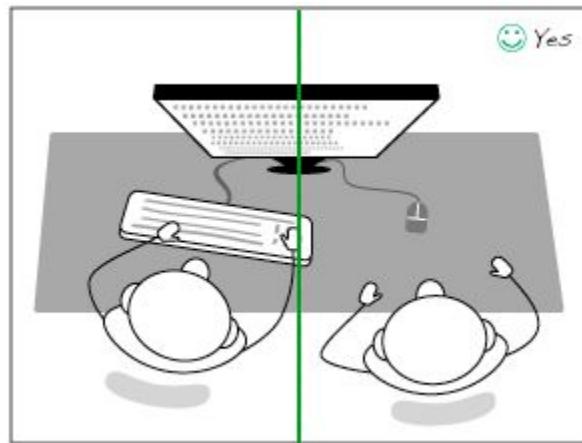
- Site Oficial: <http://rubyonrails.org/>
- Documentação: <http://guides.rubyonrails.org/>
- Ruby GEMs: <https://rubygems.org/>
- Screencasts: <http://railscasts.com/>
- Cursos Online:
 - <https://www.edx.org/course/agile-development-using-ruby-rails-uc-berkeleyx-cs169-1x>
 - <http://railsforzombies.org/>
 - <https://www.codecademy.com/pt>

Programação em Pares



- Produtividade
- Evita distrações
- Colaboração
- Comunicação constante
- Nivelamento no aprendizado

Programação em Pares



Contato



<https://gitlab.com/arthurmde>



<https://github.com/arthurmde>



<http://bit.ly/2jvND12>



<http://bit.ly/2j0ll09>

Centro de Competência em Software Livre - CCSL

esposte@ime.usp.br

Aula 01 - Aplicações Web e Ruby on Rails

Arthur de Moura Del Esposte - esposte@ime.usp.br



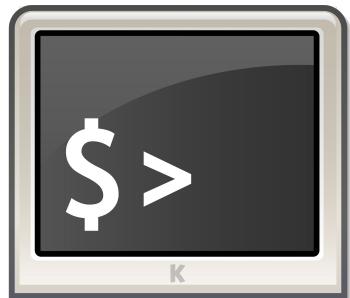
By Arthur Del Esposte licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0)

Agenda

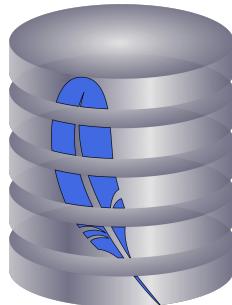
- Ambiente de desenvolvimento
- Arquitetura de aplicações Web
- Introdução a Rails
- Hello World!
- Executando a aplicação

Ambiente de Desenvolvimento

O que precisamos?



Terminal



Banco de Dados



Web Browser



Editor de Texto



Ruby Environment



Instalação do Ruby com Rails

- Para Windows: <http://rubyinstaller.org/>
- **RVM:** <https://rvm.io/rvm/install>
- Instalação dos pacotes necessários:
 - \$ sudo apt-get install git-core curl zlib1g-dev build-essential libssl-dev libreadline-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt1-dev libcurl4-openssl-dev python-software-properties libffi-dev nodejs





Ruby Version Manager

- Instalar e trabalhar com múltiplas versões
- Gerenciar conjunto de bibliotecas (GEMs)
- Instalação do RVM:

```
$ gpg --keyserver hkp://keys.gnupg.net --recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3  
$ \curl -sSL https://get.rvm.io | bash -s stable --rails
```

- Sempre que for usar o RVM:

```
$ /bin/bash --login
```



Ruby Version Manager

- Listar versões de Ruby disponíveis:
- Verificar instalação e versão do Ruby:
- Verificar local de instalação do Ruby:
- Verificar versão do Rails
- Verificar local de instalação do Rails
- Para instalar uma versão específica do Rails:

```
$ rvm list
```

```
$ ruby -v
```

```
$ which ruby
```

```
$ rails -v
```

```
$ which rails
```

```
$ gem install rails -v 5.0.1
```



GEM - Bundler

```
$ gem install bundler
```

- Bundler proporciona um ambiente para gerenciamento de dependências de RubyGems para projetos em Ruby
- Vamos usar o bundler para instalar as dependências necessárias dos nossos projetos

Outras ferramentas do ambiente

- Verificar instalação do git
- Verificar instalação do sqlite

```
$ git -v
```

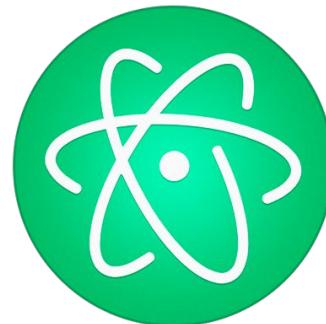
```
$ sqlite --help
```

IDE

- [Ruby Mine](#)
- [Cloud9](#)

Editores de Texto

- [VIM](#)
- [EMACS](#)
- [Gedit](#)
- [Atom](#)
- [Sublime](#)



Mais sobre escolhas de [IDE vs Editor de Textos](#)



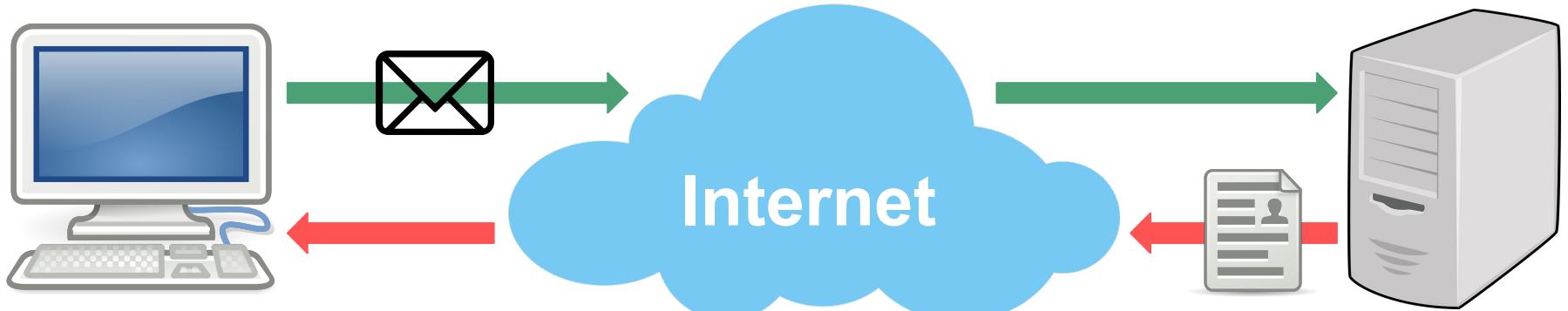
Github

- Github: <https://github.com/>
- Repositório de código-fonte online
- Usa o git como controle de versão
- Tutorial básico: <https://guides.github.com/activities/hello-world/>
- Não teremos aula sobre como utilizar o git e o Github, mas vocês aprenderão o básico a medida que vamos utilizando!
- Crie um conta no Github!

Arquitetura de Aplicações Web

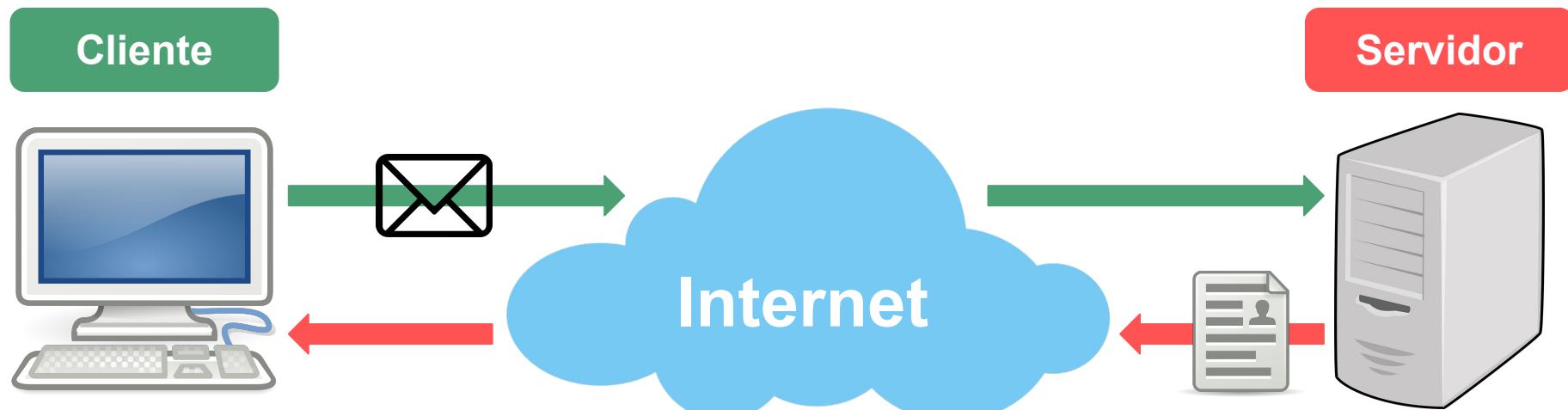
World Wide Web

- Sistema de documentos em hipermídia que são interligados e executados na Internet
- Infraestrutura Física (Redes)
- Protocolos (IP, TCP, HTTP)
- Apresentação (Browser, HTML, CSS)



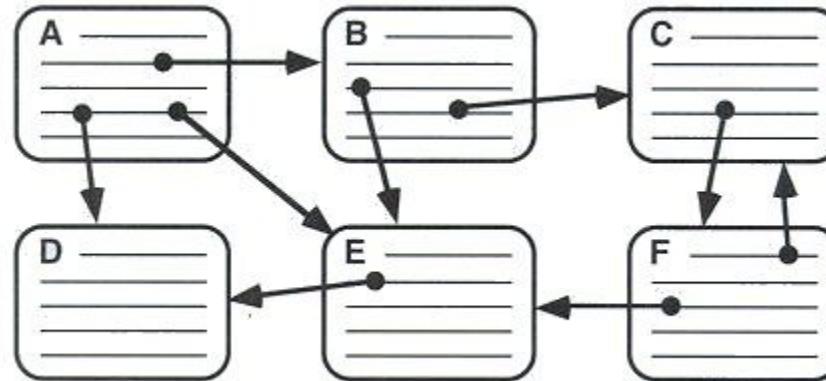
World Wide Web

- **Cliente:** envia requisições e processa respostas
- **Servidor:** recebe requisições, **processa** e envia respostas



Hipermídia

- É um meio não-linear de textos que inclui:
 - Imagens
 - Vídeo
 - Texto puro
 - **Hyperlinks**
- **HTML** - HyperText Markup Language



Web 1.0

- Páginas Estáticas
- Poucos usuários
- Pouca interatividade para o usuário
- Ainda assim, **revolucionário!**

World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#) , [Policy](#) , November's [W3 news](#) , [Frequently Asked Questions](#) .

What's out there?

Pointers to the world's online information, [subjects](#) , [W3 servers](#), etc.

Help

on the browser you are using

Software Products

A list of W3 project components and their current state. (e.g. [Line Mode](#) , [X11 Viola](#) , [NeXTStep](#) , [Servers](#) , [Tools](#) , [Mail robot](#) , [Library](#))

Technical

Details of protocols, formats, program internals etc

Bibliography

Paper documentation on W3 and references.

People

A list of some people involved in the project.

History

A summary of the history of the project.

How can I help?

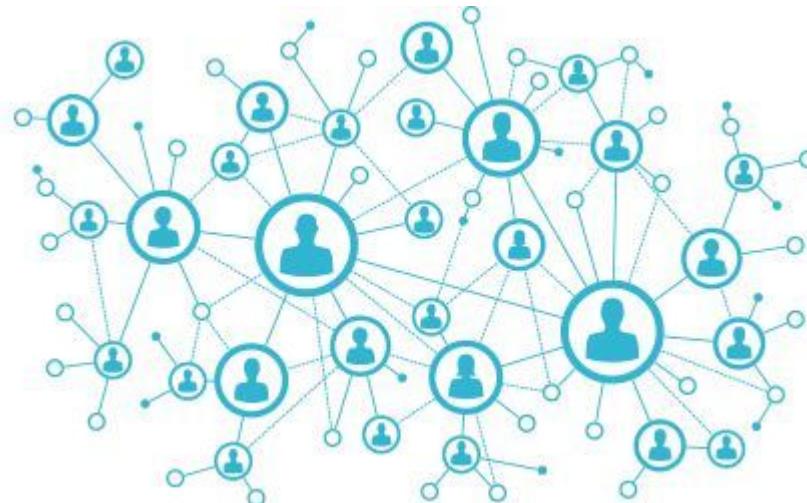
If you would like to support the web..

Getting code

Getting the code by [anonymous FTP](#) , etc.

Web 2.0

- Blogs, Chats, Mídias Sociais
- Conteúdo produzido colaborativamente
- Usabilidade
- Interatividade
- **Aplicações Web**



Aplicações Web

- Sistemas Cliente-Servidor projetados para serem utilizados através de Cliente Web (Navegador, MobileApps)
- Os clientes interagem com a aplicação enviando requisições ao servidor
- O servidor processa as requisições, cria páginas **dinamicamente** e responde ao cliente com páginas **estáticas**
- Servidores podem armazenar dados em Banco de Dados
- Desenvolvido por tecnologias padrões como HTML, CSS e JavaScript

URI - Uniform Resource Identifier

- Um endereço completo para acessar um recurso na Web é definido por sua **URI**
- Uma URI contém:
 - Nome do protocolo de comunicação
 - Nome do hospedeiro (servidor)
 - Número da porta (opcional)
 - Recurso que deve ser recuperado

Exemplo: **http://www.localhost:3000/movies/3**

Protocolos

- Para acessar uma aplicação Web, um usuário acessa um endereço do servidor na internet (ex: www.google.com.br) através de um Navegador (ex: Firefox)
- O Navegador e o Sistema Operacional lidam com toda parte de comunicação, que inclui:
 - a. Traduzir o endereço
 - b. Empacotar a requisição
 - c. Enviar a requisição para o servidor
 - d. Receber a resposta
 - e. Tratar a resposta
 - f. Apresentar a resposta

Domain Name System - DNS

- O endereço que digitamos em um Navegador é apenas um nome fácil de se lembrar
- Esse endereço deve ser traduzido para o endereço real de um servidor na Internet - o endereço IP (ex: 192.168.123.23)
- O Navegador faz essa tradução automaticamente utilizando um serviço de nomes DNS
- DNS consiste em um protocolo para realização do serviço de tradução de hostnames (www.google.com.br) para o endereço IP (172.217.28.35)
- O nome localhost serve para referenciar o próprio computador (127.0.0.1)

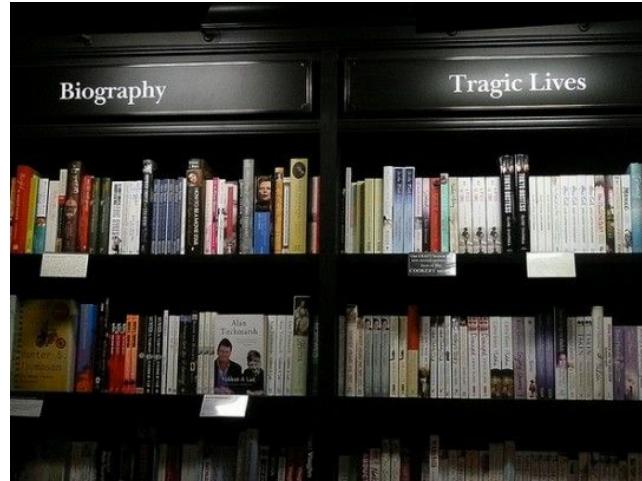
```
$ nslookup www.google.com.br
```

Porta

- Múltiplos protocolos e aplicações podem estar sendo executados em um mesmo computador (ex: DNS, email, SSH)
- Além do endereço IP, é necessário identificar a porta na qual a Aplicação Web está sendo executada - porta 80 por padrão (HTTP) e 443 (HTTPS)
- Portanto, para acessar uma Aplicação Web sempre usamos seu endereço e porta. Para uma aplicação rodando localmente, usamos:
 - a. localhost:3000
 - b. 127.0.0.1:3000

Caminho do recurso

- Além do endereço e porta, sempre são passados o caminho ou identificação do recurso que desejamos obter de uma Aplicação Web
- Toda informação que vem após o endereço é utilizado para identificar o recurso solicitado ao servidor. Se nada for informado, receberemos a página inicial
 - www.w3.org/Protocols/
 - rvm.io/rvm/install
 - bookstore:80/ruby_on_rails.pdf



HTTP - HyperText Transfer Protocol

- Navegadores e servidores se comunicam usando o HTTP, onde o navegador realiza **requisições** que devem ser **respondidas** pelo servidor, geralmente transferindo algum conteúdo
- O HTTP não mantém estados, portanto uma requisição é independente da outra
- Para manter informações de usuários (ex: Você está logado? Qual o seu usuário? Em que página você está?) as aplicações usam mecanismos de **Cookies HTTP e Sessões** que serão explicados em mais detalhes no futuro

HTTP - Requisições

- Existem vários tipos de requisições que um cliente pode fazer para o servidor
- Cada forma é definida como um **método de solicitação**, onde os principais são:
 - **GET**: requisita a representação de um recurso sem produzir efeitos colaterais
 - **POST**: envia dados para serem processados relacionados ao recurso especificado. Os dados são incluídos no corpo do comando e geralmente criam novos registros no servidor
 - **PUT**: semelhante ao **POST**, mas destinado para editar um recurso existente
 - **DELETE**: exclui um recurso no servidor
- Uma requisição é definida por um método e uma URI:

```
GET http://srch.com:80/main/search?q=nuvem&lang=pt_BR#top
```

HTTP - Respostas

- Ao processar uma requisição, o servidor retorna sempre um código numérico indicando o estado da resposta para que o cliente reaja adequadamente
 - **1xx** (Informação) - utilizada para avisar ao cliente que a requisição foi recebida e está sendo processada
 - **2xx** (Sucesso) - indica que a requisição do cliente foi bem sucedida
 - **3xx** (Redirecionamento) - informa ações adicionais que devem ser tomadas para completar uma requisição
 - **4xx** (Erro no Cliente) - avisa que o cliente fez uma requisição que não pode ser atendida
 - **5xx** (Erro no Servidor) - ocorreu um erro no servidor ao cumprir uma requisição válida
- Muitas respostas podem conter conteúdo dentro de seu corpo relacionado ao recurso (ex: HTML)

HTTP - Respostas mais comuns

200 - OK	Requisição completa com Sucesso
204 - No Response	Servidor recebeu a requisição mas não há informações para serem enviadas de volta
401 - Unauthorized	Quando é possível autenticar, mas ainda não foi autenticado
403 - Forbidden	A requisição não foi processada pelo servidor devido a questões de autorização
404 - Not Found	O recurso requisitado não foi encontrado
500 - Internal Error	O servidor encontrou uma condição não esperada

HTTP - Respostas

- Ao processar uma requisição, o servidor retorna sempre um código numérico indicando o estado da resposta para que o cliente reaja adequadamente
 - **1xx** (Informação) - utilizada para avisar ao cliente que a requisição foi recebida e está sendo processada
 - **2xx** (Sucesso) - indica que a requisição do cliente foi bem sucedida
 - **3xx** (Redirecionamento) - informa ações adicionais que devem ser tomadas para completar uma requisição
 - **4xx** (Erro no Cliente) - avisa que o cliente fez uma requisição que não pode ser atendida
 - **5xx** (Erro no Servidor) - ocorreu um erro no servidor ao cumprir uma requisição válida
- Muitas respostas podem conter conteúdo dentro de seu corpo relacionado ao recurso (ex: HTML)



Representação - HTML

- A linguagem universal que todos navegadores entendem é o **HTML** (HyperText Markup Language)
- HTML é uma linguagem que combina texto e marcações (anotações sobre o texto) de forma hierárquica para compor estruturas de páginas
- Cada marcador possui um tipo, atributos, início e fim. Tudo entre esse início e fim está hierarquicamente abaixo desse marcador ou dentro dele
- Os atributos **id** e **class** dos marcadores HTML são utilizados para conectar a estrutura HTML com a sua aparência visual
- Esses dois atributos são essenciais para a manipulação dinâmica da página também
- Inspecione o elemento de uma página com seu navegador para visualizar a estrutura do HTML (ex: <http://www.usp.br/>)

Representação - CSS

- A linguagem para formatação visual do HTML é o CSS (Cascading Style Sheets)
- CSS permite associar instruções de **estilo** aos elementos HTML utilizando os **ids** e **classes** desses elementos
 - Cores
 - Posicionamento
 - Fonte
 - Dimensões
- Inspecione o elemento de uma página com seu navegador para visualizar a estrutura do HTML (ex: <http://www.usp.br/>)

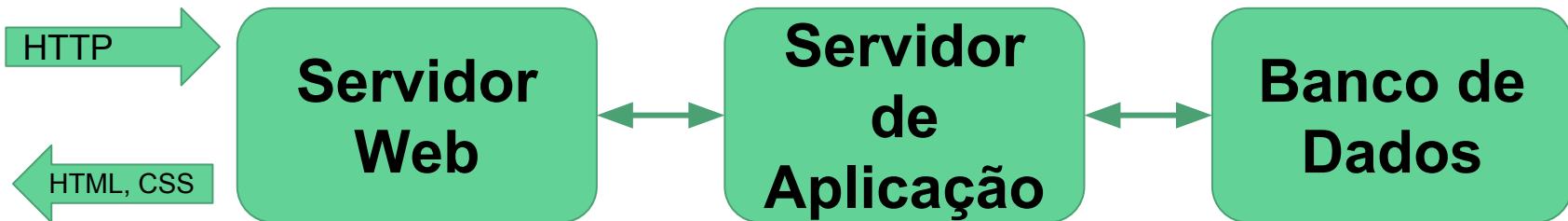
Representação - CSS

- Para aplicar as propriedades nos elementos do HTML, o CSS usa seletores que são baseados em regras para seleção de elementos

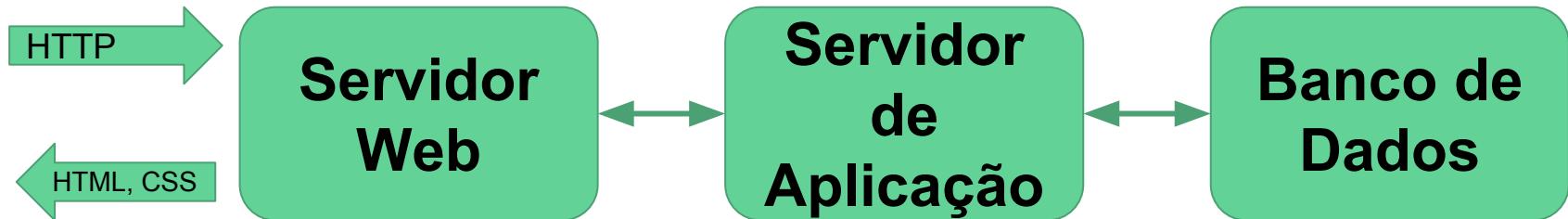
Seletor	O que é selecionado
h1	Qualquer elemento h1
div#mensagem	O div cujo ID é mensagem
.red	Qualquer elemento com a classe red
div.red, h1	O div com classe red ou qualquer h1
div#mensagem h1	O elemento h1 filho de (dentro de) div#mensagem

Arquitetura de Aplicações Web

- Uma Aplicação Web em um servidor recebe requisições HTTP e devem processá-las através de códigos e scripts em alguma linguagem de programação (ex: Ruby, Java, PHP)
- Porém deve responder sempre com conteúdos estáticos que serão interpretados e apresentados por diversos Navegadores (HTML + CSS + JavaScript)

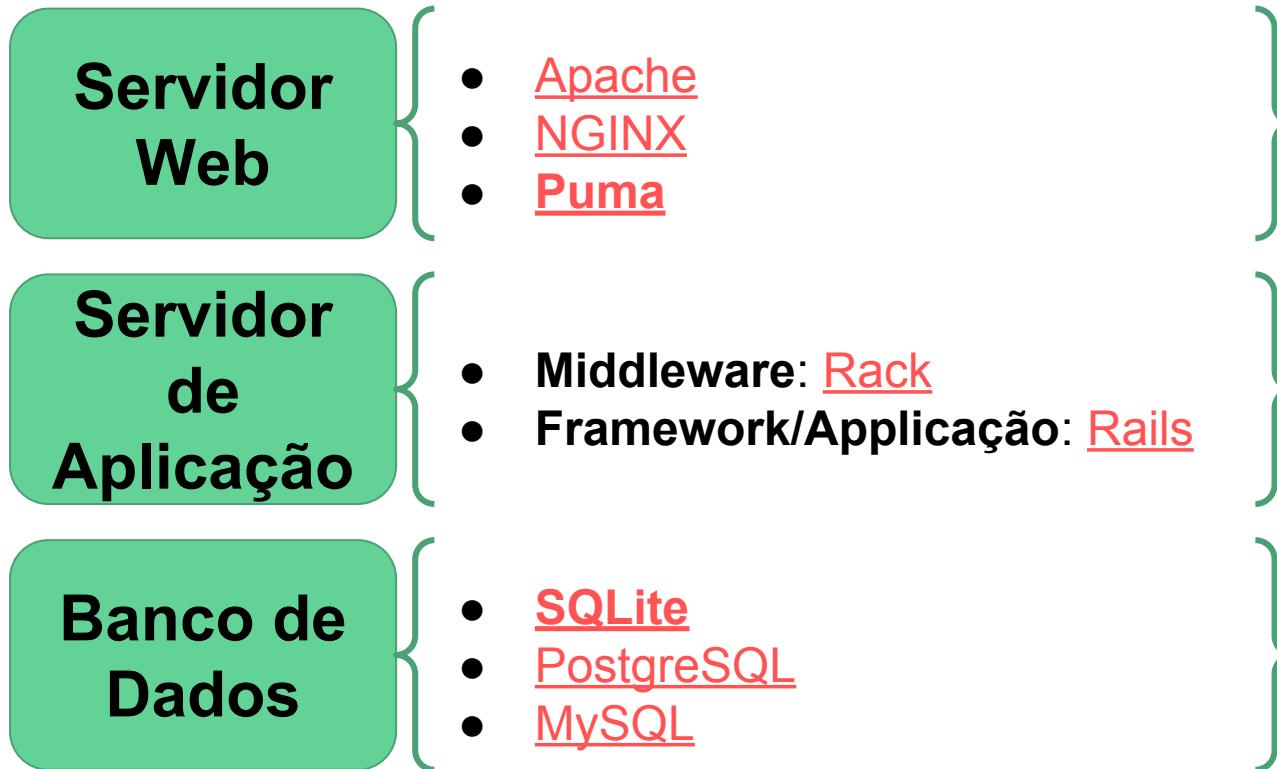


Arquitetura de Aplicações Web



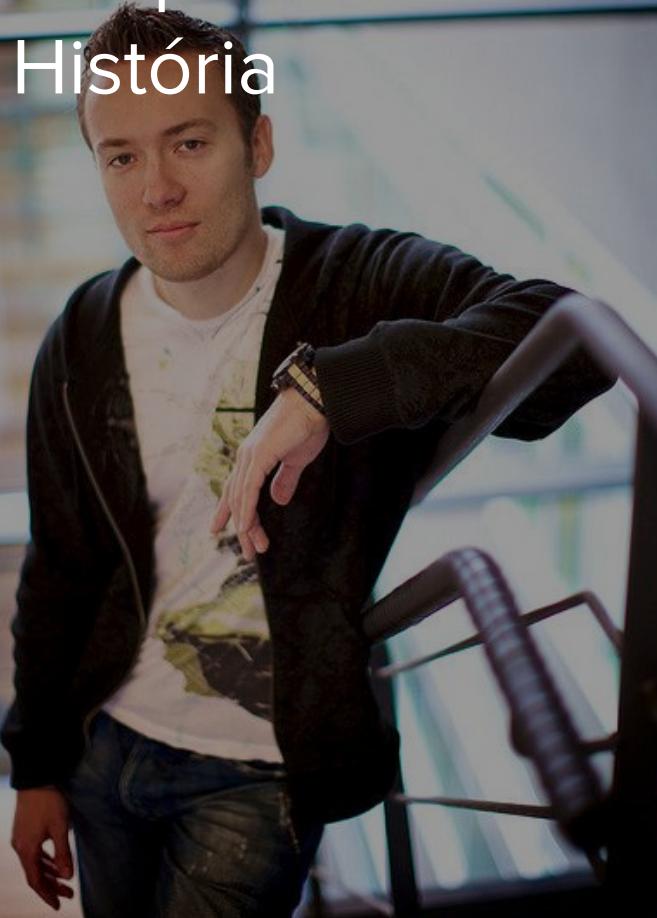
- **Servidor Web:** Recebe requisições HTTP vindas do exterior, repassa para a camada lógica (Servidor de Aplicação) e devolve itens estáticos (HTML + CSS)
- **Servidor de Aplicação:**
 - **Middleware:** Recebe as requisições HTTP e encaminham para o trecho de código apropriado para tratá-la
 - **Framework ou Aplicação:** Código-fonte que processa as informações e lógica para geração de conteúdo dinâmico
- **Banco de Dados:** Armazenam e recuperam os dados da aplicação

Arquitetura de Aplicações Web



Ruby on Rails

Um pouco de História



- **1995:** Disponibilização pública do Ruby por Yukihiro “Matz” Matsumoto
- **1999:** Fundançam da empresa **37Signals**
- **2003:** Rails é criado por David Heinemeier Hansson - DHH
- **2005:** Rails é publicado como um Software Livre \o/
- **2015:** Rails 5.0 é lançado
- **Atualmente:** Rails é o framework mais usado pelas Startups

Yukihiro "Matz"
Matsumoto

*“Rails is
the killer
app for
Ruby”*





Rails

- **Rails** é um framework livre para criação de **aplicações Web**
- Combina **Ruby** (back-end) com HTML, CSS e JavaScript (front-end)
- Distribuído como uma **Gem**
- Ele define uma estrutura específica para organizar o código que visa encorajar as “melhores decisões” de projeto:
 - Encoraja algumas práticas e métodos
 - Às vezes, desencoraja práticas consideradas ruins
- **Full-stack**
- Grande comunidade!



Rails

- Orientado a Objetos com Ruby
- Modular
- Geração de HTML a partir de código Ruby
- Vários **Helpers** reutilizáveis
- Processamento de submissão de formulários
- Acesso ao banco de dados com interfaces padronizadas
- Testes automatizados
- Envio de emails
- Muitas Gem



DRY - Don't Repeat Yourself

Foto: [Christopher Michel](#)

Convention Over Configuration

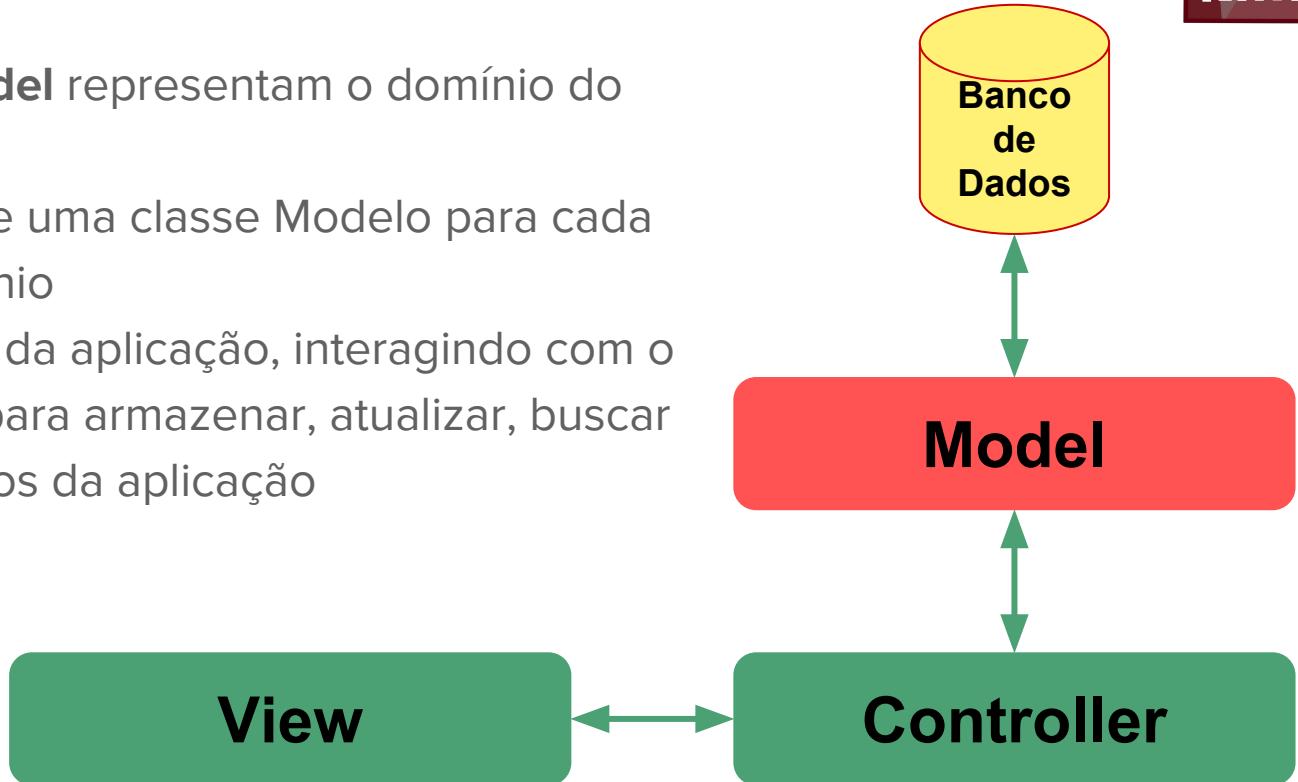


Arquitetura MVC

- **MVC** é a sigla para Model-View-Controller
- Padrão arquitetural que divide a aplicação em três partes interconectadas para separação da **Interface de Usuário** da lógica da aplicação
- **Model** - contém todas as classes relacionadas ao domínio do problema. Gerencia diretamente a lógica, dados e regras da aplicação
- **View** - camada que apresenta os dados e recebe interações do usuário. Idealmente, podemos ter várias interfaces diferentes (Ex: web e mobile)
- **Controller** - Recebe requisições e entradas, manipula as classes da **Model** e gera os resultados que devem ser apresentados pela **View**

MVC - Modelos

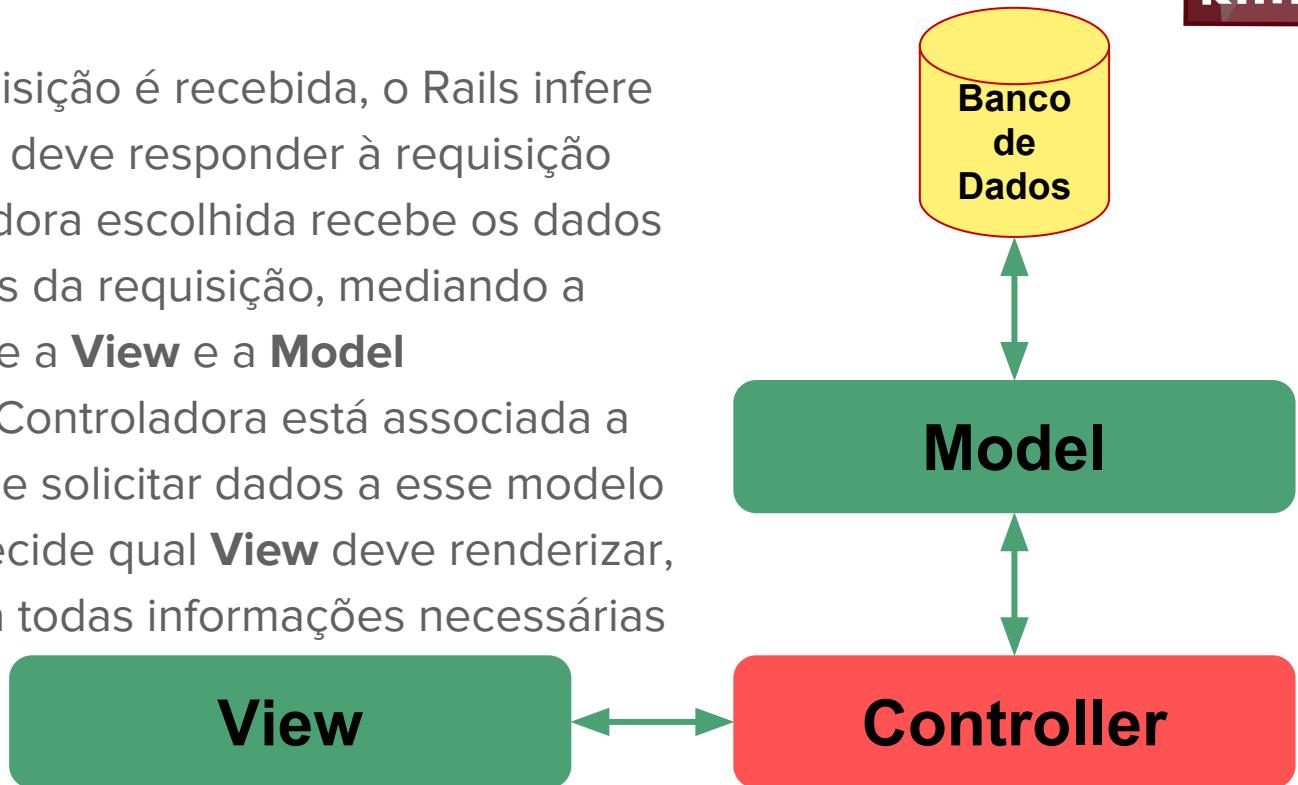
- As classes de **Model** representam o domínio do problema
- Geralmente, existe uma classe Modelo para cada entidade do domínio
- Manipulam dados da aplicação, interagindo com o **banco de dados** para armazenar, atualizar, buscar e remover os dados da aplicação





MVC - Controladoras

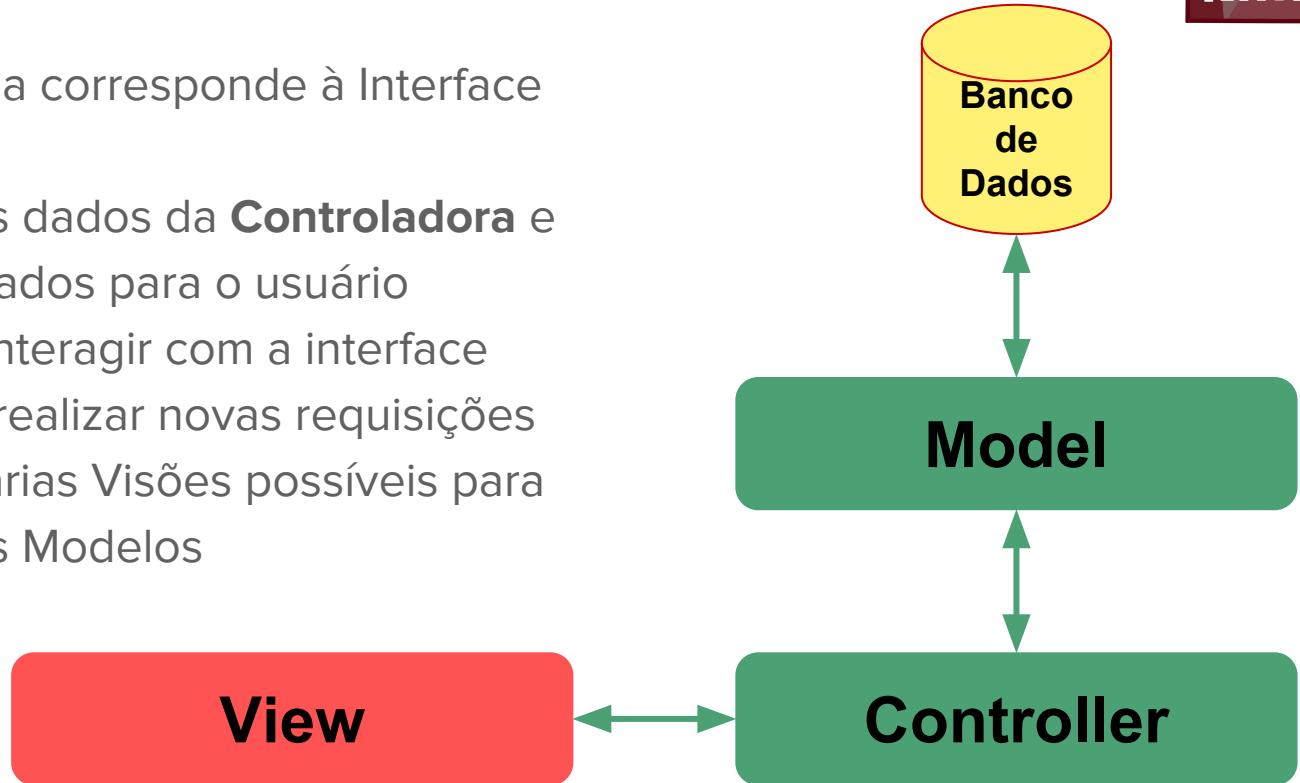
- Quando uma requisição é recebida, o Rails infere qual Controladora deve responder à requisição
- A classe Controladora escolhida recebe os dados importantes vindos da requisição, mediando a comunicação entre a **View** e a **Model**
- Geralmente cada Controladora está associada a um Modelo, e pode solicitar dados a esse modelo
- A Controladora decide qual **View** deve renderizar, passando para ela todas informações necessárias





MVC - View

- A **Visão** do sistema corresponde à Interface Gráfica
- A **Visão** recebe os dados da **Controladora** e renderiza esses dados para o usuário
- Usuários podem interagir com a interface renderizada para realizar novas requisições
- Geralmente, há várias Visões possíveis para acessar as classes Modelos



Hello World



Rails CLI

- **Rails** possui uma interface de linha de comando para manipulação dos seus projetos
- Essa interface executa scripts muito úteis para geração de código e estruturas básicas
- **Crie um novo projeto Rails:**

```
$ mkdir workspace
```

```
$ cd workspace
```

```
$ rails new hello_world
```



Pastas Criadas

- O diretório do projeto Rails contém vários arquivos e pastas auto-gerados que compõem a estrutura de uma aplicação Rails
 - **app** - A maioria dos arquivos específicos da nossa aplicação ficam nessa pasta. Contém as *controllers*, *views*, *models*, *helpers*, *mailers* e *assets* para sua aplicação
 - **bin** - Executáveis do rails e das gems instaladas
 - **config** - configuração da sua aplicação relacionado a rotas, banco de dados e mais
 - **db** - contém o esquema atual do DB e as migrações que registram as mudanças no DB
 - **lib** - módulos estendido para sua aplicação e bibliotecas auxiliares
 - **log** - arquivos com log da aplicação
 - **public** - contém arquivos estáticos e *assets* compilados
 - **test** - testes unitários e outros utilitários de testes automatizados
 - **tmp** - arquivos temporários
 - **vendor** - bibliotecas e aplicações de terceiros



Arquivos Gerados

- Alguns arquivos importantes:
 - **config.ru** - configurações utilizada pelo Middleware **Rack** para iniciar a aplicação
 - **Gemfile** e **Gemfile.lock** - permite que você especifique quais versões de dependências Gems são necessárias para sua aplicação. Eles são usados pela Gem **Bundler**
 - **Rakefile** - localiza e carrega Tasks que podem ser executadas no terminal para automatizar algumas tarefas. Esse arquivo é usado pela Gem **Rake** que é uma espécie de Makefile escrito em Ruby. As Tasks específicas do seu app devem ficar em **lib/tasks**
 - **README.md** - contém uma breve explicação textual sobre o projeto e como configurar seu ambiente



Executando o nosso projeto

- Para executar nosso projeto, basta chamar bin/**rails server**
- **Abra o navegador para ver sua aplicação sendo executada!**
- Para interromper, aperte **Ctrl + C** no terminal

```
$ cd hello_world
```

```
$ rails server
```



Gemfile

- Verifique o arquivo Gemfile
- Adicione a seguinte linha no final do arquivo:

```
group :development, :test do
  gem 'rspec-rails', '~> 3.5'
end
```

- Instale a nova gem:

```
$ bundle install
$ rails generate rspec:install
$ rspec
```



Rota

- Execute novamente a aplicação e acesse a página: <http://localhost:3000/hello>
- Temos um **Routing Error**
- Verifique o log da aplicação para visualizar o erro
- Isso ocorre pois devemos especificar rotas em nossa aplicativo indicando recursos acessíveis
- Cada rota deve possuir uma Controladora e uma ação correspondente que deverá tratar suas requisições
- Verifique as rotas disponíveis:

```
$ rake routes
```



Rota

- Para criarmos a página **hello**, devemos adicionar uma nova rota
- Abra o arquivo **config/routes.rb** e adicione dentro da definição de rotas:

```
get 'hello' => 'hello#index'
```

- Verifique suas rotas novamente:

```
$ rake routes
```

- Execute novamente a aplicação e acesse a página: <http://localhost:3000/hello>
- Continuamos ter o mesmo erro?



Criando a Controladora

- Antes de criar uma controladora que irá tratar as requisições de hello, veja o arquivo **app/controllers/application_controller.rb**
- Esse arquivo mostra a Controladora principal do nosso projeto: **ApplicationController**
- Ela herda da classe definida pelo Rails **ActionController::Base**
- No fundo, toda Controladora é uma classe normal cujos métodos tratam requisições da aplicação
- Todas as nossas Controladores devem herdar de **ApplicationController**
- Logo, quando precisarmos definir comportamentos que valem para todo o sistema, poderemos implementá-los na classe **ApplicationController**



Criando a Controladora

- Baseado na ApplicationController, crie o arquivo `app/controllers/hello_controller.rb` e defina a classe **HelloController** como subclasse da ApplicationController
- Não é necessário fazer **require**. O Rails já adiciona os caminhos dos arquivos principais para todas as classes!
- As seguintes **Convenções** se aplicam:
 - Cada controladora deve ser definida em um arquivo diferente
 - O nome do arquivo deve ser terminar com `_controller.rb`
 - O nome da classe deve seguir o padrão `UsedNameController`
- Execute novamente a aplicação e acesse a página: <http://localhost:3000/hello>



Criando a ação **index**

- Os métodos de uma controladora que tratam requisições são chamados de **ações**
- Para tratar o erro anterior, basta criarmos um método em HelloController chamado **index**, conforme definimos na rota
- Crie o método **index** em branco e veja o que acontece ao acessar a página
- O Rails espera que para cada ação renderize alguma **página HTML (View)**



Criando a View HelloController#index

- Note que podemos ter nomes de ações iguais, desde que estejam em controladoras diferentes
- Para criar o template que irá ser renderizado para a ação **HelloController#index**, crie o diretório correspondente à controladora na pasta **app/views**:

```
$ cd app/views
```

```
$ mkdir hello
```



Criando a View HelloController#index

- Agora, crie o arquivo `app/views/hello/index.html` e insira o seguinte código:

```
1 <h1>
2   Hello World!
3 </h1>
```

- Execute novamente a aplicação e acesse a página: <http://localhost:3000/hello>
- **Pergunta:** Por quais partes do **MVC** já passamos?



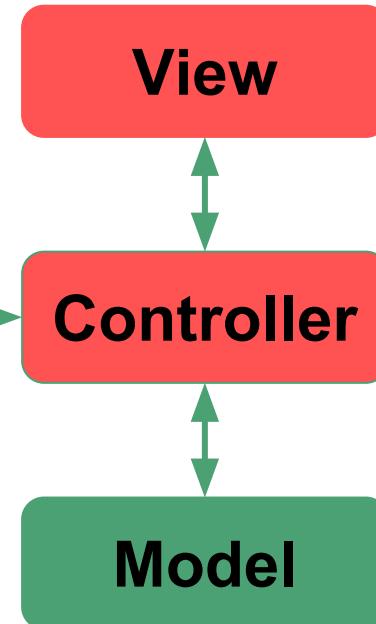
MVC

Requisição
HTTP



Rails Routes

Identifica qual controladora e ação devem tratar a requisição baseado na Rota



`app/views/hello/index.html`

`app/controllers/hello_controller.rb`

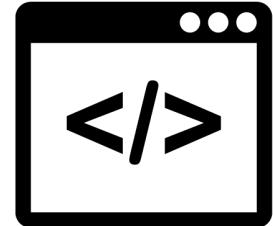
`app/models/`

Verifique as rotas criadas até o momento e responda as seguintes

```
$ rake routes
```



1. Por que o verbo utilizado é o GET?
2. Faria sentido trocar o método para POST, PUT ou DELETE?
3. Qual é o código HTTP esperado quando um requisição na página hello é bem sucedida?



Exercício

Crie a página **/goodbye** na sua aplicação que escreve na tela “Goodbye!”

- Obs: A sua página deve ser acessada em <http://localhost:3000/goodbye>

Revisão!



O que já vimos!

- Ambiente de desenvolvimento
- Arquitetura de aplicações Web
- Introdução a Rails
- Hello World!
- Executando a aplicação

Atividades Sugeridas!

Resolver os seguintes desafios

1. Crie as seguintes novas ações dentro de HelloController com suas páginas correspondentes:
 - o nice: Renderiza “You look nice!”
 - o angry: Renderiza “I hate you!”
 - o love: Renderiza “I love you!”
 - o options: Renderiza uma lista com opções de páginas já criadas (hello, goodbye, nice, angry, love) com links HTML para essas páginas
2. Leia os comentários existentes no arquivo config/routes.rb e descubra como definir a página HelloController#index como inicial. Posteriormente, defina essa página com a página inicial!
3. Crie um projeto chamado movies cuja página inicial seja uma lista dos seus cinco filmes favoritos.

Estudar

- Estudar o básico do **HTTP**
- Compreender melhor a arquitetura **MVC**
- Estudar e criar páginas **HTML** estáticas
- Estudar e criar estilos para suas páginas HTML com **CSS**
- Estudar o *workflow* básico de **git**

Obrigado!

Contato



<https://gitlab.com/arthurmde>



<https://github.com/arthurmde>



<http://bit.ly/2jvND12>



<http://bit.ly/2j0ll09>

Centro de Competência em Software Livre - CCSL

esposte@ime.usp.br