Rails Framework

Rails

- Instalação
- Ruby versão 1.8.7
  - Linux: ftp://ftp.ruby-lang.org/pub/ruby/1.8/ruby-1.8.7-p174.tar.gz
  - ou apt-get install ruby (Debian/Ubuntu)
  - Windows: http://rubyforge.org/frs/?group_id=167 (one click installer)
  - Ou ainda http://www.ruby-lang.org
- RubyGems (gerenciador de pacotes ruby)
  - http://rubyforge.org/frs/?group_id=126 (caso já não exista)
  - ruby setup.rb
- Rails (framework)
  - gem install rails

Rails

- Banco de dados
  - SQLite3
  - http://www.sqlite.org
  - Pré-compilados
  - sqlitedll-3.5.9.zip em C:\Windows\System32
  - gem install sqlite3-ruby

- IDE (ambiente de desenvolvimento)
  - Netbeans 6.7.1
  - http://www.netbeans.org (opção Ruby)
  - Java Runtime (necessário para rodar Netbeans)

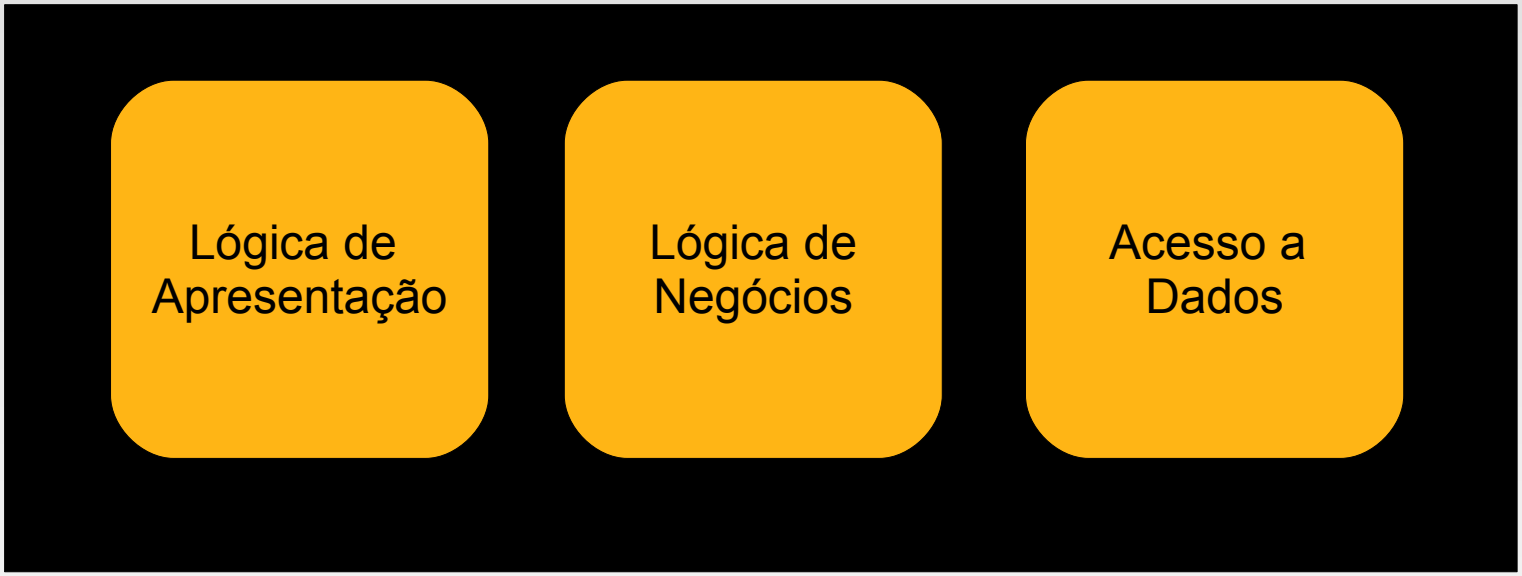
Rails

- Teste simples
  - rails path/to/your/new/application
  - cd path/to/your/new/application
  - ruby script/server

- Abra seu browser e digite a URL:
  - http://localhost:3000
  - e siga as instruções contidas na página

Desenvolvimento em camadas

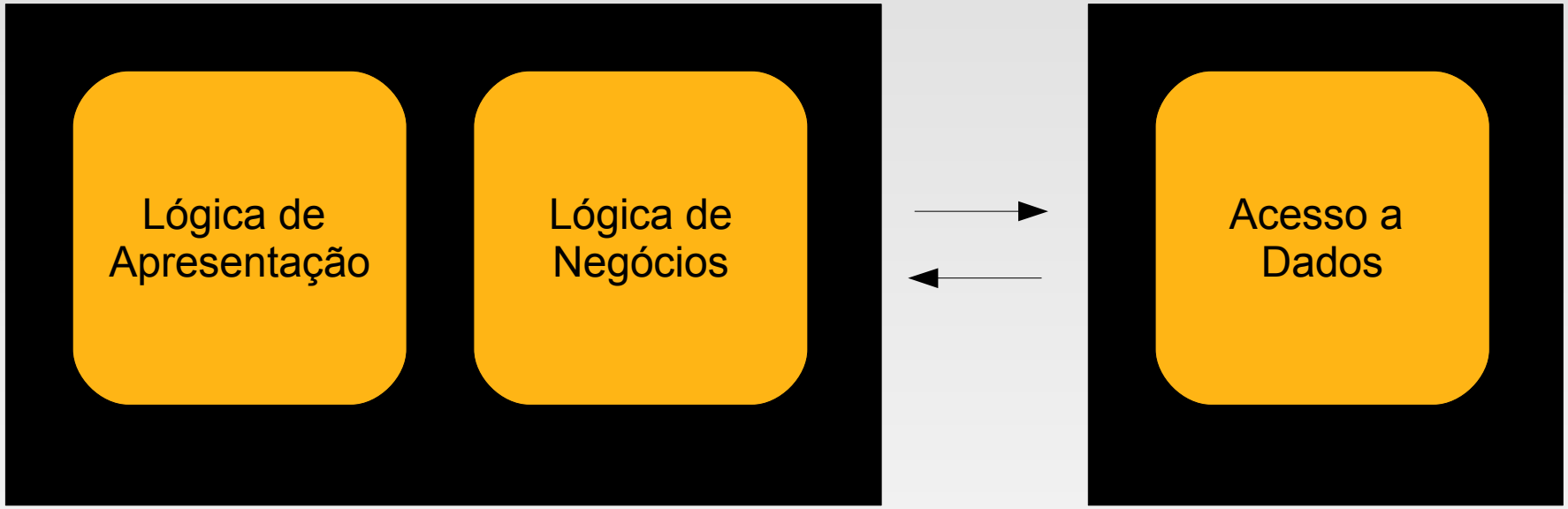
- Aplicações Monolíticas



- Aplicativos para apenas uma máquina
- Único módulo com todas as funcionalidades
- Dífícil manutenção

Desenvolvimento em camadas

- Aplicações em duas camadas



- Necessidade de compartilhar lógica de acesso a dados
- Base de dados colocada em máquina específica, separada das aplicações. Uso de estações clientes
- Cada alteração no aplicativos precisava ser atualizada no cliente

Desenvolvimento em camadas

- Aplicações em três camadas

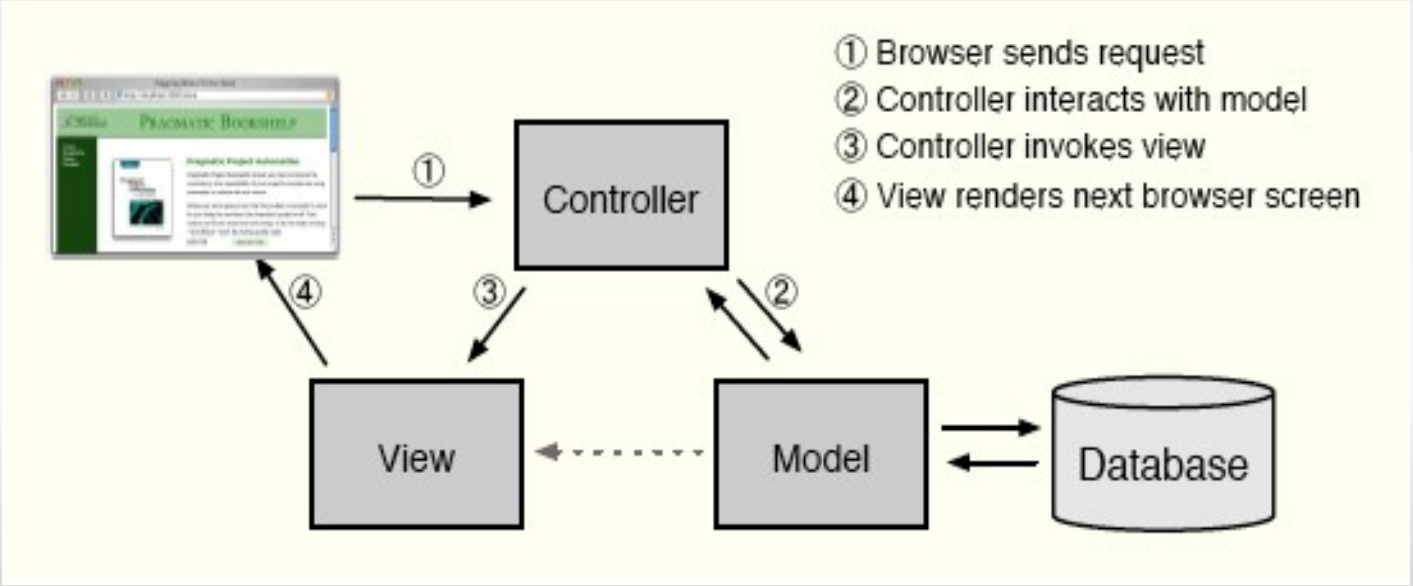
- Advento da internet: lógica de negócio separada da interface
- Acesso às aplicações sem instalar localmente
- Fácil manutenção e modularidade com orientação a objetos

Arquitetura Rails

- Model-View-Controller (MVC)

  Padrão que divide em três componentes distintos a forma de interagir com uma aplicação

- Model – armazena, manipula e gera os dados. É o coração da aplicação
- View – apenas recebe e exibe a informação (interface); pode conter HTML, JS, XML
- Controller - mapeia eventos, interagem com o modelo e devolvem uma saída



Arquitetura Rails

- Vantagens do MVC
  - Como o modelo MVC gerencia múltiplos visualizadores usando o mesmo modelo é fácil manter , testar e atualizar sistemas múltiplos
  - É muito simples incluir novos clientes apenas incluindo seus visualizadores e controles
  - Torna a aplicação escalável
  - É possível ter desenvolvimento em paralelo para o modelo , visualizador e controle pois são independentes.

Model

- *ActiveRecord*

  Padrão que abstrai as interações com o modelo de dados, efetuando o mapeamento entre classes e entidades

- zero linhas de código SQL
- suporta vários SGBDs (SQLIte, Oracle, SQL Server, MySQL, etc)
- cada modelo corresponde tipicamente a uma tabela ou relação entre tabelas
- lógica de negócio implementada no modelo e não na base de dados
- uma instância da classe é uma linha na tabela

Model

Projetos

Id INTEGER
cliente_id INTEGER
codigo VARCHAR
designacao VARCHAR
data_inicio DATE
gestor_id INTEGER

```
class Projetos < ActiveRecord::Base
	has_one :gestor
	has_one :cliente
	has_many :etapas

	validates_presence_of :codigo, :designacao, :cliente, :gestor
	validates_uniqueness_of :codigo
end
```

Model


```
#criar nova instância do modelo, com alguns atributos já definidos
p = projeto.new :codigo => ”PRJ-001”, :designacao => ”Projeto #1”

#atribuir os atributos restantes
p.gestor = Gestor.find_by_name ”Milton Moura”
p.cliente = Cliente.find_by_name ”Microsoft”
p.data_inicio = Time.now

#guardar na base de dados
p.save

#obter o registro recentemente introduzido
p = Projeto.find_by_codigo ”PRJ-001”
```

Controller

- *ActionController*

  o controlador é o centro lógico da aplicação

- coordena as interações entre o utilizador, o interface de utilização e o modelo de dados
- responsável pela identificação das acções a desempenhar
- gere os pedidos HTTP, as sessões, cookies, etc

Controller

```
class ProjetosController < ApplicationController
        def index
                @projetos = Projeto.find :all
        end
end
```

- Neste Exemplo:
  - o pedido HTTP de origem terá sido do tipo: http://localhost/projetos/
  - é criado uma variável de instância do controlador, com uma lista de todos os projectos registados na base de dados
  - uma vez executado o código do método, o controlo é passado à view

View

- ERb - ruby embebido em html
- as views são blocos de html com conteúdo dinâmico proveniente de código ruby

```
<% @projetos.each do | p | %>
      <div class = ”projeto”>
            <b><%= p.codigo%><b> - <%= p.designacao %>
      </div>
<%end %>
```

- o objeto @projectos é o array de instâncias da classe Projeto, devolvido pelo controller
- as tags <% %> delimitam código executável que não deve ser traduzido para HTML
- as tags <%= %> delimitam código que deve ser ser visualizável

Estrutura Rails

- sild_epl
  - app
    - controllers
      - account_controller.rb
      - application.rb
      - front_office_controller.rb
      - licencas_controller.rb
      - pesquisa_controller.rb
    - helpers
    - models
    - views
      - account
      - front_office
      - layouts
      - licencas
        - _sidebar.rhtml
        - index.rhtml
        - new_issue.rhtml
        - new_request.rhtml
        - view.rhtml
      - pesquisa
      - shared
  - components
  - config
  - db
  - doc
  - lib
  - log
  - public
  - Rakefile
  - README
  - script
  - test
  - tmp
  - vendor

Passo a Passo

- Vamos ao NetBeans!

Aplicação por linhas de comando

- rails app_name
- cd app_name
- rake db:create
- ruby script/generate scaffold names nome:string

  rake db:migrate
- ruby script/server
- http://localhost:3000/names