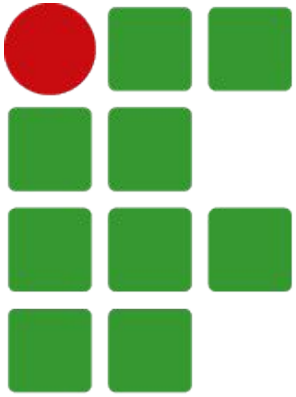


INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Ruby on Rails

Explorando Recursos

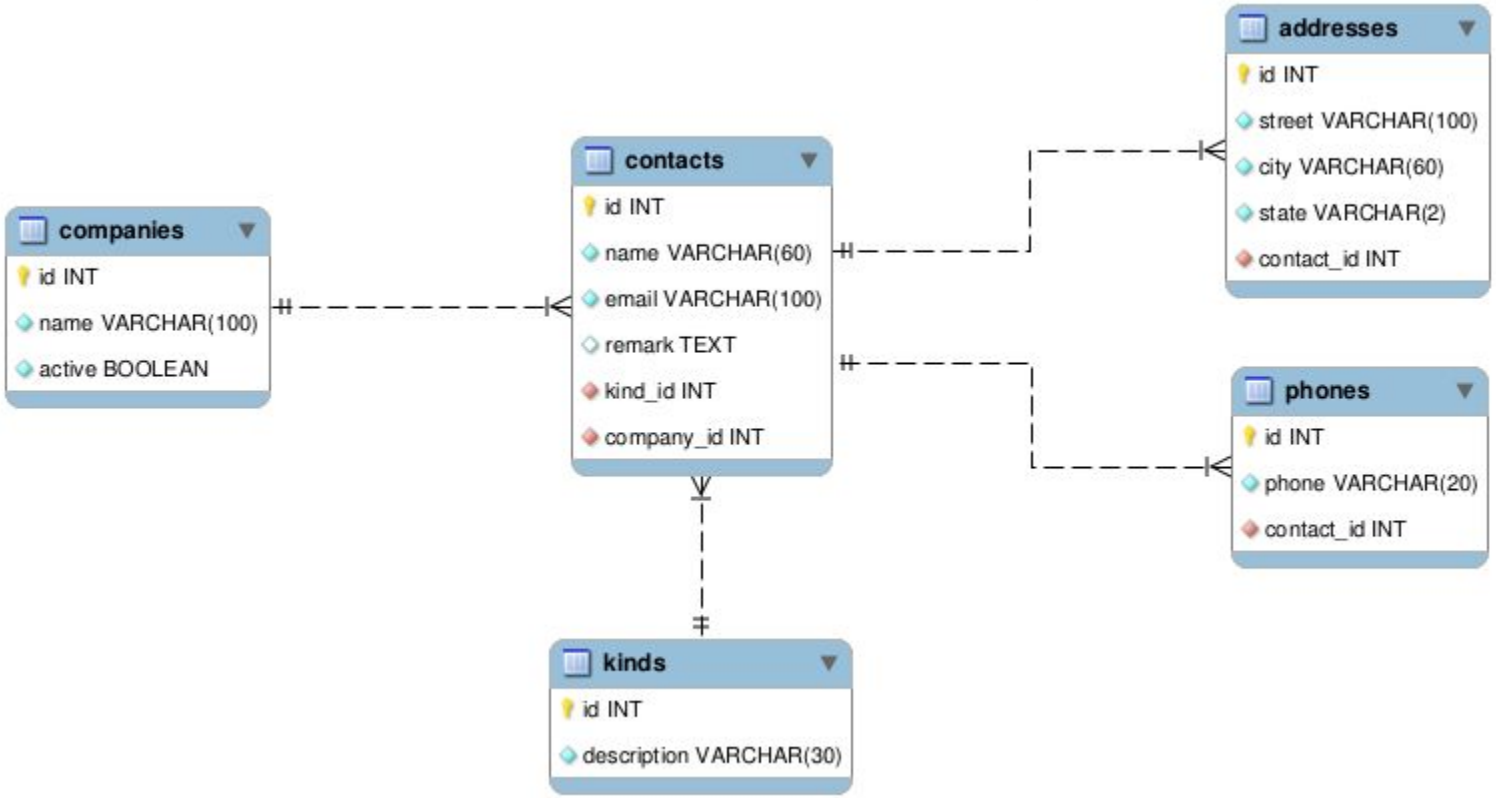
Desenvolvimento Web III

Prof. Diego Stiehl

INSTITUTO FEDERAL

Aumentando o Modelo

- Agora temos um cadastro de Empresas
  - Contatos são também de Empresas



INSTITUTO FEDERAL

CRUD

- Vamos fazer todo o CRUD de Empresas
  - Sem utilizar Scaffold
- Conhecer melhor os recursos aplicados
  - Models
  - Views
  - Controller
  - Rotas
  - Persistência
  - …

INSTITUTO FEDERAL

Model

- Gerar um model para a empresa
  - rails g model Company name
- Vai criar o model e a migration

```
class CreateCompanies <
ActiveRecord::Migration[5.2]
  def change
    create_table :companies do |t|
      t.string :name

      t.timestamps
    end
  end
end
```

INSTITUTO FEDERAL

Migration

- Ops! Esquecemos da propriedade “active”
  - Sem problemas. Ainda não migramos.
- Alterar migration:

```
class CreateCompanies <
ActiveRecord::Migration[5.2]
  def change
    create_table :companies do |t|
      t.string :name
      t.boolean :active, default: false

      t.timestamps
    end
  end
end
```

INSTITUTO FEDERAL

Alterar Modelo Existente

- Empresa e Contato ainda não estão ligados
  - Não posso alterar a migration de criação de Contato para adicionar o campo
    - Antiga (já executada)
- Como proceder?
  - Criar uma migration somente para adicionar este atributo em Contato

INSTITUTO FEDERAL

Nova Migration

- Criar somente migration (nome explicativo)
  - rails g migration AddCompanyToContacts company:references
- Arquivo criado
  - 20190918220142_add_company_to_contacts.rb

```
class AddCompanyToContacts < ActiveRecord::Migration[5.2]
  def change
    add_reference :contacts, :company, foreign_key: true
  end
end
```

INSTITUTO FEDERAL

Migrar

- rails db:migrate

- Observação:
  - Antes do Rails 5, vários comandos do rails eram executados com rake
    - rake db:migrate

INSTITUTO FEDERAL

ORM

- Já temos o modelo no BD
  - Temos company_id em Contact
- Precisamos indicar que haverá um atributo company (objeto de Company)
- Responsável por isso: **belongs_to**
- No model Contact, adicionar:
  - belongs_to :company
    - Agora poderemos fazer: **contact.company**.name

INSTITUTO FEDERAL

Controller

- Criar controller de Empresas, já com a ação de index
  - rails g controller companies index
- Arquivos criados
  - app/controllers/companies_controller.rb
  - app/views/companies/index.html.erb
- Rota criada
  - get 'companies/index'

INSTITUTO FEDERAL

Rotas

- A rota criada nos obriga a acessar via URL:
  - http://xxxxxx/companies/index
    - Não ficou tão bonito
- Arrumar rota
  - get 'companies', to: **'companies#index'**
    - Agora podemos acessar assim:
      - http://xxxxxx/companies

View index

- Fazer index.html.erb listar todas Empresas
  - Foco na funcionalidade
    - Veremos estilização mais à frente

```
<h1>Empresas</h1>
<% @companies.each do |company| %>
    <div>
        <%= company.name %>
        |
        <%= company.active? ? "ATIVA" : "DESATIVADA" %>
    </div>
<% end %>
```

INSTITUTO FEDERAL

Rails Console

- Rails Console é um IRB vinculado à aplicação
- Abrir o Rails Console
  - rails console
    - Abreviação: rails c
- Usa o IRB clássico por padrão
  - Sabemos que você gosta mais do Pry

INSTITUTO FEDERAL

Pry

- Para adicionar o Pry ao Rails Console
  - Existe uma gem que faz a mágica
- https://github.com/rweng/pry-rails
- Adicionar ao Gemfile
  - gem 'pry-rails', group: :development
- Executar:
  - bundle install
    - Atalho: bundle ← install é a ação padrão
- Abrir o “rails c” novamente
  - Agora com Pry

INSTITUTO FEDERAL

Criar Empresas

- Vamos criar três empresas exemplo
- No Rails Console

```
Company.create!(name: 'IFPR', active: true)
Company.create! name: 'UTFPR'
Company.create! name: 'UFPR', active: true
```

INSTITUTO FEDERAL

Show

- Criar action no controller
  - def show
- Criar rota
  - get 'companies/**:id**', to: 'companies#show', as: :company
- Criar view
  - app/views/companies/show.html.erb

INSTITUTO FEDERAL

Action/View Show

- Definir no método (action):

```
def show
  @company = Company.find params[:id]
end
```

- Definir na view:

```
<h1><%= @company.name %></h1>
<p><%= @company.active? ? "ATIVA" : "DESATIVADA" %></p>
```

INSTITUTO FEDERAL

Helper link_to

- Criar links para
  - index ⟶ show (Abrir)
    - <%= link_to "Abrir", **company** %>
  - show ⟶ index (Voltar)
    - <%= link_to "Voltar", **companies_path** %>
- Dica:
  - Usando o “back” do browser via JavaScript
    - <%= link_to "Voltar", :back %>

INSTITUTO FEDERAL

Comentários no ERB

- Para comentar algo nos arquivos .html.erb
  - Usar a seguinte sintaxe:
    - <%#= %>
  - Exemplo:
    - <%#= link_to "Voltar", :back %>

INSTITUTO FEDERAL

New

- Criar action no controller
  - def new
- Criar rota
  - get 'companies/new', to: 'companies#new', as: :new_company
- Criar view
  - app/views/companies/new.html.erb

INSTITUTO FEDERAL

Ordem das Rotas

- A ordem das rotas é importante
  - Em caso de conflitos:
    - O que aparece antes, ganha
- A rota new deve aparecer antes de show
  - Senão vai chamar companies com id “new”
  - Ordem correta:
    - get 'companies/new', to: 'companies#new', as: :new_company
    - get 'companies/:id', to: 'companies#show', as: :company

INSTITUTO FEDERAL

Action New

- Definir no método (action):

```
def new
  @company = Company.new
end
```

- @company novo (vazio) como base de formulário

INSTITUTO FEDERAL

View New

```
<h1>Nova Empresa</h1>

<%= form_with model: @company, local: true do |form| %>
    <div>
        <%= form.label :name %>
        <%= form.text_field :name, id: :company_name %>
    </div>
    <div>
        <%= form.label :active %>
        <%= form.check_box :active, id: :company_active %>
    </div>
    <div>
        <%= form.submit 'Salvar' %>
    </div>
<% end %>

<div>
    <%= link_to "Voltar", companies_path %>
</div>
```

INSTITUTO FEDERAL

Form Gerado

- No browser, ver:
  - Form e Inputs gerados
  - action
  - method
  - Atributos name

INSTITUTO FEDERAL

Submissão

- Um “form_with model” já trabalha no padrão REST de requisições
- Irá submeter o formulário gerado para:
  - /companies ← Método **POST**
    - Deve cair na action <u>create</u>

INSTITUTO FEDERAL

Create

- Criar action no controller
  - def create
- Criar rota
  - **post** 'companies', to: 'companies#create'
- Create não tem View
  - Criar registro e direcionar/renderizar

INSTITUTO FEDERAL

Action Create

- No método, verificar o que foi recebido

```
def create
  render plain: params.inspect
end
```

- Resultado

```
{
    "utf8"=>"✓",
    "authenticity_token"=>"TOKEN GIGANTE",
    "company"=>{
        "name"=>"Unioeste",
        "active"=>"0"
    },
    "commit"=>"Salvar",
    "controller"=>"companies",
    "action"=>"create"
}
```

INSTITUTO FEDERAL

Action Create

- Implementar

```
  def create
    permitted_params = params.
      require(:company).permit(:name, :active)

####@company = Company.new(params) # ERRO
    @company = Company.new(permitted_params)
    if @company.save
      redirect_to @company
    else
      render :new
    end
  end
```

INSTITUTO FEDERAL

Alterar View New

- Adicionar (no começo)

```
<% if @company.errors.any? %>
    <ul>
        <% @company.errors.full_messages.each do |message| %>
            <li><%= message %></li>
        <% end %>
    </ul>
<% end %>
```

INSTITUTO FEDERAL

Validação Simples

- Para gerarmos um erro na inserção
  - Precisamos de uma validação
- Validação: name obrigatório
  - Adicionar ao model de Company:
    - validates_presence_of :name
- Método **.save** só retornará true se a condição for satisfeita

INSTITUTO FEDERAL

Edit

- Criar o arquivo de view:
  - app/views/companies/edit.html.erb
    - Copiar o conteúdo (duplicar) a view new.html.erb
    - Alterar somente primeira linha:
      - <h1>Editando <%= @company.name %></h1>
- Criar a action
  - def edit
- Criar a rota
  - get 'companies/:id/edit', to: 'companies#edit', as: :edit_company

INSTITUTO FEDERAL

Ação Edit

- Definir no método (action):

```
def edit
  @company = Company.find params[:id]
end
```

- Na view de index, link para editar
  - <%= link_to "Editar", edit_company_path(company) %>
- Na view show, também
  - <%= link_to "Editar", edit_company_path(@company) %>

INSTITUTO FEDERAL

Partials

- Edit e New apresentam essencialmente o mesmo código
- Podemos terceirizar o trecho repetido
  - Quem conterá o código será uma Partial View
- Partials tem a seguinte sintaxe de nome
  - _nome.html.erb
- Vamos criar a partial:
  - /app/views/companies/**_form.html.erb**

INSTITUTO FEDERAL

_form.html.erb

```
<% if @company.errors.any? %>
    <ul>
        <% @company.errors.full_messages.each do |message| %>
            <li><%= message %></li>
        <% end %>
    </ul>
<% end %>
<%= form_with model: @company, local: true do |form| %>
    <div>
        <%= form.label :name %>
        <%= form.text_field :name, id: :company_name %>
    </div>
    <div>
        <%= form.label :active %>
        <%= form.check_box :active, id: :company_active %>
    </div>
    <div>
        <%= form.submit 'Salvar' %>
    </div>
<% end %>
<div>
    <%= link_to "Voltar", companies_path %>
</div>
```

INSTITUTO FEDERAL

View New/Edit

- Novo new.html.erb

```
<h1>Nova Empresa</h1>

<%= render 'form' %>
```

- Novo edit.html.erb

```
<h1>Editando <%= @company.name %></h1>

<%= render 'form' %>
```

INSTITUTO FEDERAL

Submissão (em edit)

- Um “form_with model” já trabalha no padrão REST de requisições
- Se model tem um **id**
  - Está em edição
- Irá submeter o formulário gerado para:
  - /companies/:id ← Método **PATCH**
    - Deve cair na action update

INSTITUTO FEDERAL

Update

- Criar action no controller
  - def update
- Criar rota
  - **patch** 'companies/:id', to: 'companies#update'
- Update não tem View
  - Fazer o serviço e direcionar/renderizar
    - Parecido com o create

INSTITUTO FEDERAL

Action Update

- Implementar

```
def update
  permitted_params = params.
    require(:company).permit(:name, :active)

  if @company.update(permitted_params)
    redirect_to @company
  else
    render :edit
  end
end
```

INSTITUTO FEDERAL

Código Repetitivo

- Parâmetros permitidos com código repetitivo
- Criar método privado

```
private

  def permitted_params
    params.require(:company).permit(:name, :active)
  end
```

Chamada já tem que funcionar (sem variável):

```
@company = Company.new permitted_params
####################
if @company.update permitted_params
```

INSTITUTO FEDERAL

Código Repetitivo

- A montagem de @company com base no id dos parâmetros aparece quatro vezes
- Fazer método privado

```
def set_company
  @company = Company.find params[:id]
end
```

- Executar **antes** de diversos métodos
  - Adicionar no começo do controller:

```
before_action :set_company, only:
      [:show, :edit, :update, :destroy]
```

INSTITUTO FEDERAL

Destroy

- Criar action no controller
  - def destroy
- Criar rota
  - **delete** 'companies/:id', to: 'companies#destroy'
- Destroy não tem View
  - Fazer o serviço e direcionar

INSTITUTO FEDERAL

Action Destroy e Link

- Implementar

```
def destroy
  @company.destroy
  redirect_to companies_path
end
```

- Na View index, adicionar link:
  - <%= link_to "Remover", company, method: :delete, data: {confirm: 'Certeza?'} %>