

# Desenvolvimento de Aplicações Web com Ruby on Rails

Arthur de Moura Del Esposte - [esposte@ime.usp.br](mailto:esposte@ime.usp.br)



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

# Agenda

- Página inicial, Layout e Menu
- Autenticação e gerenciamento de usuários
- Finalização do projeto

# Projeto da Disciplina: **MyMovies**





# MyMovies - Plano inicial

- Para exercitar os conceitos de Rails, ao longo da disciplina iremos desenvolver o projeto **MyMovies** onde usuários poderão gerenciar os filmes que assistiram. O aplicativo deve conter:
- Cadastro/Edição de filmes
- Lista de filmes
- Páginas individuais de filmes
- Páginas de Diretores listando os filmes cadastrados desse diretor
- Páginas dos Atores, listando os filmes em que eles atuaram
- Login de Usuário
- Usuário poderá ter uma lista de filmes assistidos
- Usuário poderá classificar os filmes assistidos



# MyMovies - O que falta...

- Para exercitar os conceitos de Rails, ao longo da disciplina iremos desenvolver o projeto **MyMovies** onde usuários poderão gerenciar os filmes que assistiram. O aplicativo deve conter:
- Cadastro/Edição de filmes
- Lista de filmes
- Páginas individuais de filmes
- Páginas de Diretores listando os filmes cadastrados desse diretor
- Páginas dos Atores, listando os filmes em que eles atuaram
- **Login de Usuário**
- **Usuário poderá ter uma lista de filmes assistidos**
- **Usuário poderá classificar os filmes assistidos**

# Página inicial e Layout

---



# Página inicial

- Até o momento, não criamos a página inicial do nosso projeto
- Nós poderíamos usar uma página já existente para isso ou criar uma nova, adicionando uma controladora e view correspondente a essa nova página
- É muito importante que consigamos chegar em todas as outras páginas através da página inicial
- A princípio, vamos usar a página que lista todos os filmes como página inicial
- Para isso, precisamos adicionar a seguinte rota:

```
root 'movies#index'
```

- Salve, execute a aplicação e entre em <http://localhost:3000>

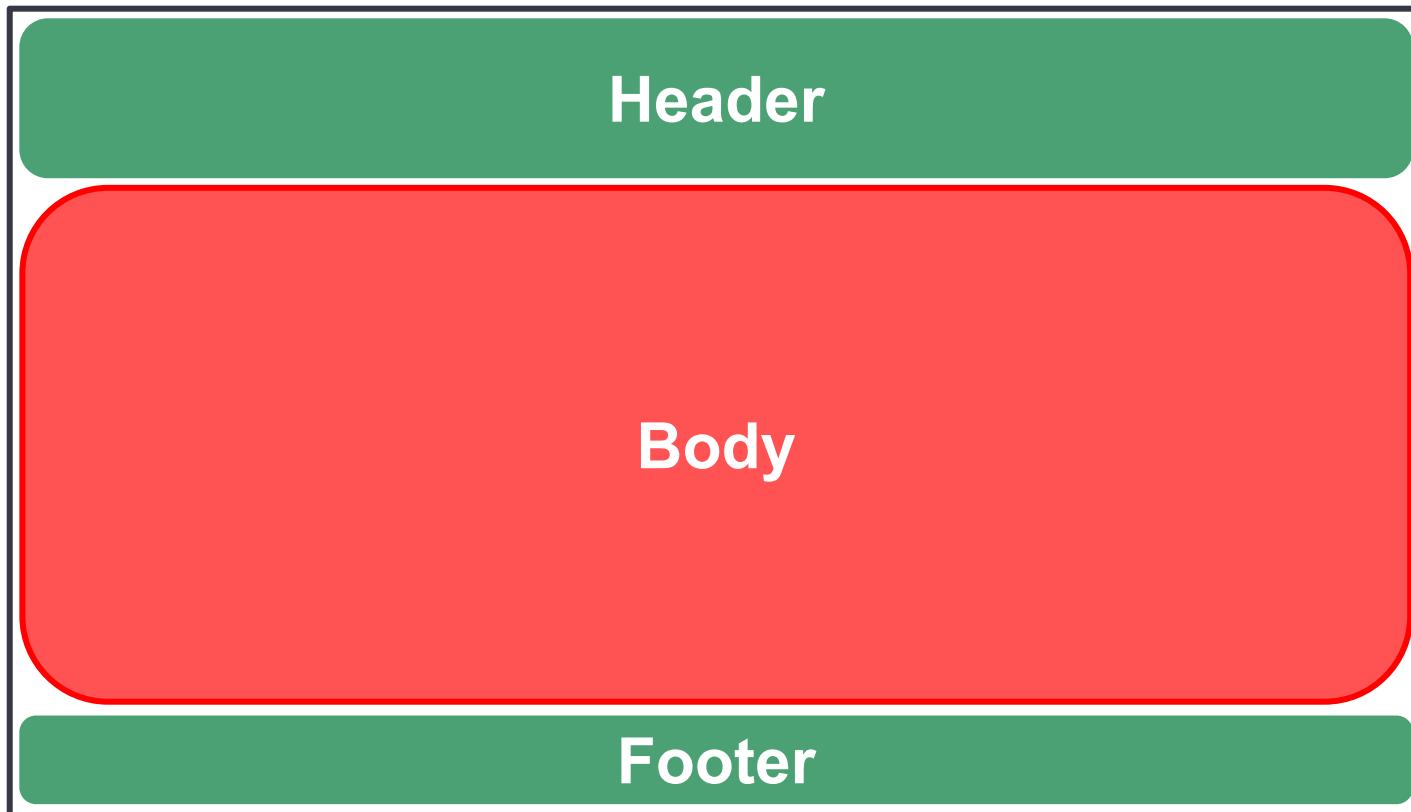


# Layout

- Observe que nossa página inicial não possui links para outras páginas
- Geralmente, a maior parte das aplicações web possuem um menu fixo que se repete ao longo de todas as outras páginas para que seja possível navegar por todos os recursos
- O esquema de uma página é chamado de **layout**
- Um **layout** define a estrutura de páginas web, incluindo posicionamento dos principais elementos, como **cabeçalho**, **corpo** e **rodapé**
- Em nosso projeto queremos uma layout onde o cabeçalho e rodapé das páginas sejam iguais, porém variando o corpo principal



# Layout - Exemplo





# Layout

- Os layouts disponíveis em nossa aplicação ficam na pasta **app/views/layouts**
- Quando criamos um projeto novo, o Rails já cria o layout **app/views/layouts/application.html.erb**
- Visualize esse arquivo para entender seu funcionamento
- O **yield** dentro do layout indica a parte onde será renderizado o corpo da página que está sendo acessada
- Ou seja, as visões que criamos até o momento compõem **parte** da página que é renderiza, uma vez que o Rails aplica o layout padrão em cada uma dessas páginas através do **yield**
- Inspecione o elemento da suas páginas e veja que elas possuem a mesma estrutura básica



# Layout

- Modifique o arquivo de layout, adicionando um **h1** com o título da aplicação e uma **div** nova para encapsular o conteúdo principal:

```
<body>
  <div class="header">
    <h1 class="app-title">
      <%= link_to("MyMovies", "/") %>
    </h1>
  </div>
  <div class="main">
    <%= yield %>
  </div>
</body>
```



# Menu

- Vamos criar um Menu no topo da página com os links para as páginas mais importantes do nosso projeto
- Esse Menu deve aparecer em todas as páginas, portanto vamos adicioná-lo em nosso layout principal
- Copie o [código deste link](#) e use-o para substituir o conteúdo anterior do nosso layout
- Além disso, [copie o código deste outro link](#) para substituir o estilo da nossa página contido em **app/assets/stylesheets/application.css**
- Salve e veja as modificações em seu projeto



# Outras opções

- Nós temos a possibilidade de ter mais de um layout sendo utilizado em nosso projeto
- Poderíamos ter layouts específicos para serem aplicados nas páginas de erros, como HTTP 404 (**Not Found**) ou HTTP 403 (**Forbidden**)
- Além disso, podemos ter mais partes que variam em nossas páginas (ex: menu dinâmico baseado na página atual)
- Poderíamos aplicar layouts com temas definidos, como por exemplo usando Bootstrap ou uma Gem que te ajuda com layouts: [Rails Layout Gem](#)
- Para mais informações e opções de renderização e layout, veja [essa documentação no site do Rails](#)



# MyMovies - Commitando Modificações

- Registre uma nova versão com as modificações feitas até agora:

```
$ git add .
```

```
$ git commit -m "Initial commit"
```

- Atualize suas modificações no Github empurrando seus commits para lá:

```
$ git push origin master
```

# Gerenciamento de Usuário

---



# Usuários

- A maior parte das **aplicações Web** necessitam que seus usuários estejam cadastrados e autenticados para permitir que eles façam determinadas ações dentro do sistema
- No caso da nossa aplicação **MyMovies**, precisamos de usuários autenticados no sistema para que eles possam:
  - Marcar os filmes que já viram
  - Avaliar os filmes
  - Cadastrar informações sobre os filmes, atores, diretores e premiações em nosso BD
- Por outro lado, podem haver várias páginas públicas que não necessitam de autenticação:
  - Lista de Filmes, Atores e Diretores
  - Páginas de Atores
  - Páginas de Filmes



# Usuários - Login e Autenticação

- Existem várias funcionalidades que devem ser implementadas para suportar login e autenticação de usuários em uma aplicação Web:
  - Registro de usuário com encriptação de senha
  - Edição e remoção da conta do usuário
  - Envio de emails para confirmação e verificação se a conta já está confirmada antes do login
  - Suporte para troca de senha ou para quem esquece a senha
  - Gerenciamento e expiração de sessão do usuário para mantê-lo
  - Integrar a autenticação com serviços externos (i.e. Facebook e Google)
- Um sistema pode ter seu próprio sistema de gerenciamento de usuários ou pode confiar em um provedor de autenticação para autenticação única, por exemplo, com Facebook, Github, Google, etc.
- Autenticação por **terceiros** pode trazer funcionalidades novas também, como permitir o compartilhamento e curtida de um conteúdo (Facebook)



# Autenticação baseada em Token

- **Cliente** - O usuário se registra na aplicação, enviando um HTTP POST informando ***username*** e ***senha***
- **Servidor** - O servidor recebe a requisição e gera um **Hash** a partir da senha antes de armazená-la no banco de dados. Se alguém ganhar acesso ao seu banco de dados, ele não poderá obter a senha dos usuários:
  - \$2a\$10\$6aOmc.wbz4yt/czCGvFrIu6UYrngif1B1rwVmzJxV0YzylxMi6qA.
  - \$2a\$10\$i0GOBtbnozTmH2efWka2OezCsxlgaoeiEjnTYsfn6zRLTp1XyLzy
- **Cliente** - o usuário envia uma requisição para fazer login no sistema, provendo o ***username*** e ***senha*** para o servidor
- **Servidor** - o servidor procura o *username* no banco de dados, aplica o mesmo algoritmo de **Hash** sobre a senha provida pelo usuário e compara com a **Hash** armazenada para esse usuário, respondendo se está certo ou não

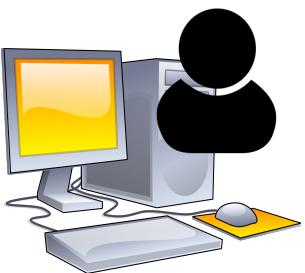


# Autenticação baseada em Token

- **Servidor** - se as credenciais estiverem corretas, o sistema gera um **token** de acesso para identificar unicamente a sessão do usuário:
  - O sistema armazena o token gerado no banco de dados associado ao usuário
  - Envia o token de sessão através de um cookie HTTP. Assim, a cada requisição feita pelo cliente, o browser irá anexar o cookie com o token de autenticação
- **Cliente** - quando o usuário logado fizer requisições de novas páginas que requerem autorização, o servidor obtém o token de sessão para recuperar a sessão do usuário. Quando o usuário deslogar, o token é destruído no servidor.



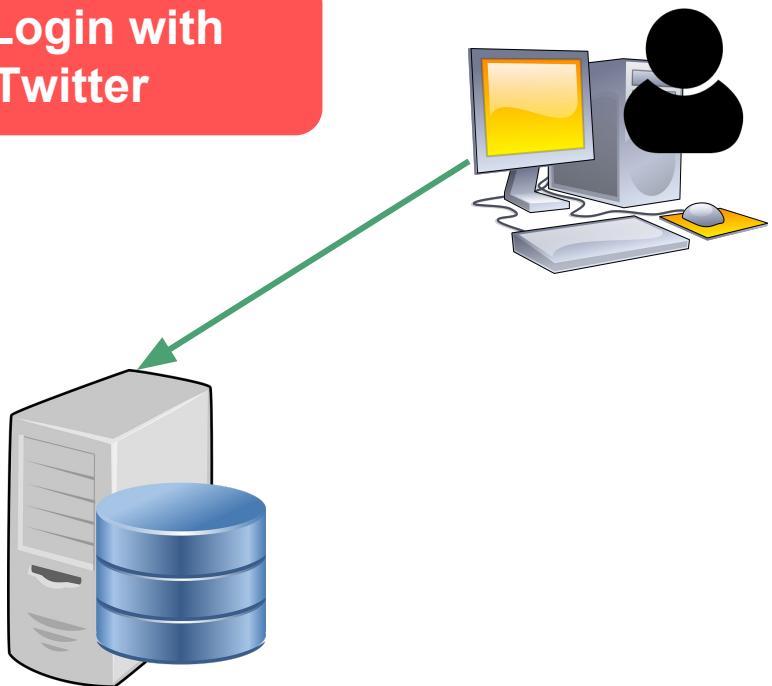
# Autenticação por Terceiros





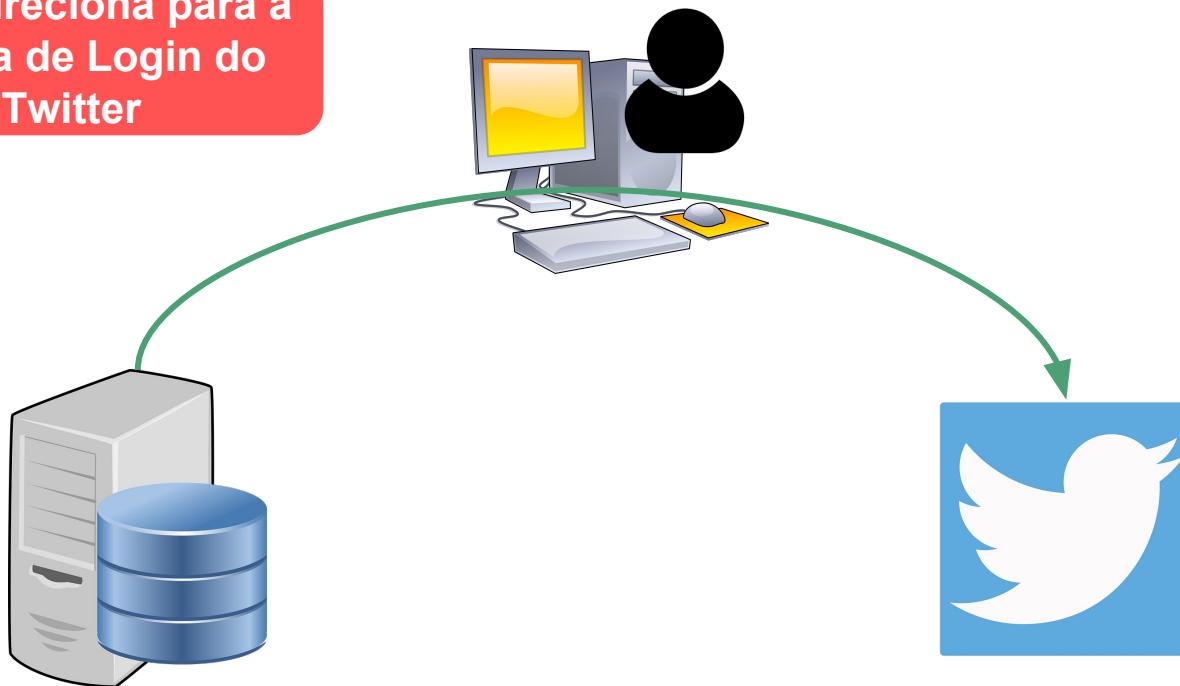
# Autenticação por Terceiros

1 - Login with Twitter



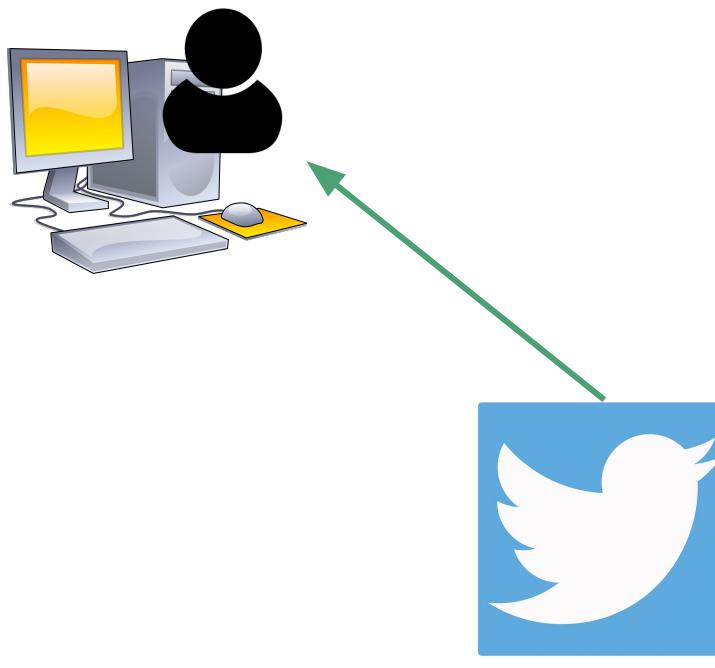
# Autenticação por Terceiros

2 - Redireciona para a  
página de Login do  
Twitter



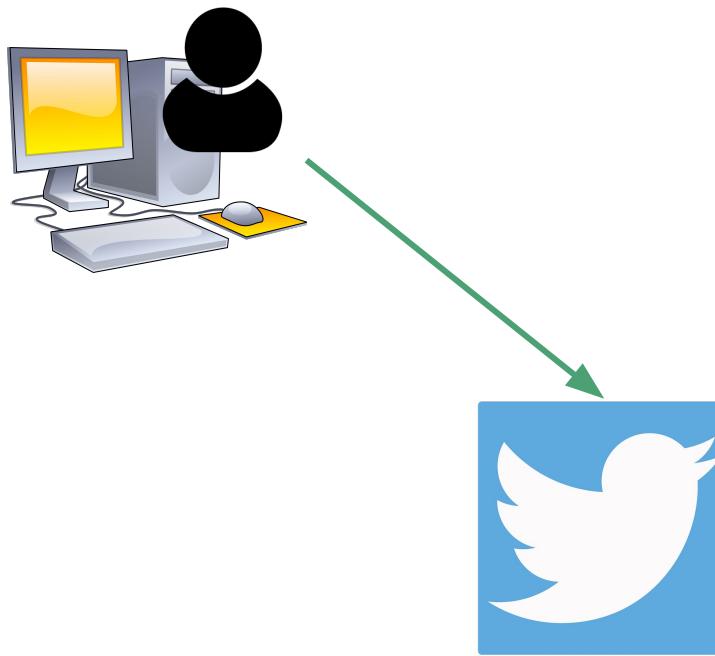
# Autenticação por Terceiros

**3 - Pergunta se pode autenticar com essa aplicação**



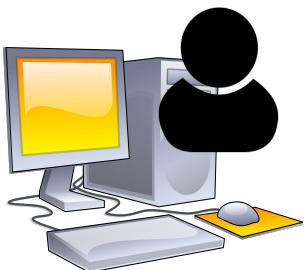
# Autenticação por Terceiros

4 - O usuário confirma



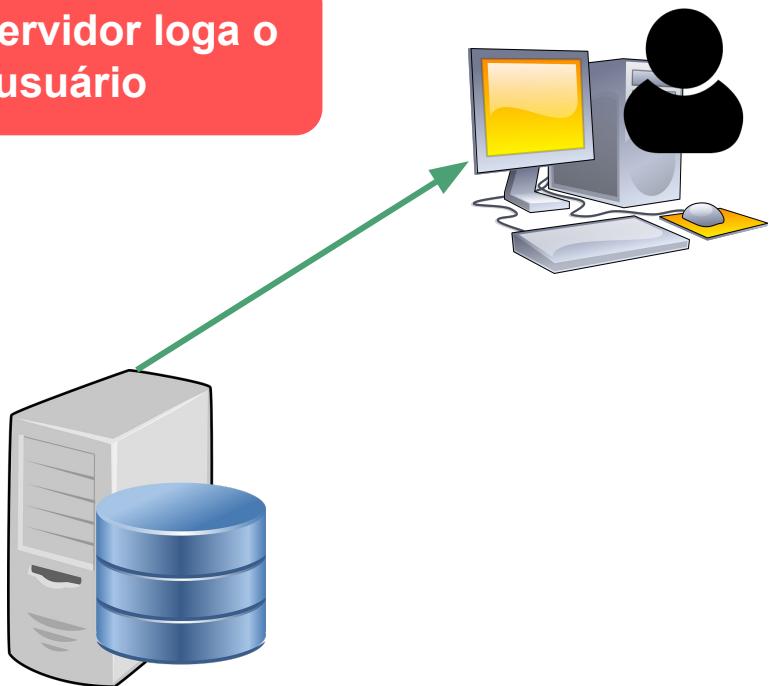
# Autenticação por Terceiros

5 - O Twitter faz um  
*callback* com o token  
de autenticação



# Autenticação por Terceiros

6 - O Servidor loga o usuário



# Aplicando o princípio DRY para Autenticação

Foto: [Christopher Michel](#)



# Devise

- No mundo do Rails, existe uma Gem que fornece vários serviços relacionados a autenticação e gerenciamento de usuários que podemos reutilizar em nossa aplicação
- A Gem Devise é uma solução de autenticação flexível que podemos integrar em nossa aplicação Rails que fornece uma infraestrutura MVC
- Adicione essa Gem no Gemfile do seu projeto MyMovies e instale usando o bundler

```
gem 'devise'
```

```
$ bundle install
```



# Devise

- Execute o seguinte gerador para criar as configurações básicas:

```
$ rails generate devise:install
```

- Abra os arquivos gerados dessa maneira
- Como o Devise envia emails de confirmação para usuários, precisamos configurar as informações relacionados ao envio de email
- Essa configuração pode variar de acordo com ambiente
- No arquivo **config/environments/development.rb** adicione a seguinte configuração:

```
config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }
```



# Devise - Model

- Gere a model User com o **devise**

```
$ rails generate devise User
```

- Veja o arquivo **app/models/user.rb** gerado desta maneira
- O devise tem várias opções (módulos) que podem ser usadas na autenticação
- Verifique também a migração gerada
- Aplique a migração:

```
$ rake db:migrate
```



# Devise - Views

- Crie as Views necessárias para o login do usuário:

```
$ rails generate devise:views
```

- As views criadas facilitam a integração da autenticação em nossa aplicação
- Podemos customizar essas páginas para nossa aplicação
- Veja as rotas disponíveis para o usuário:

```
$ rake routes
```

- Acesse [http://localhost:3000/users/sign\\_up](http://localhost:3000/users/sign_up) e cadastre um usuário



# Devise - Controladoras, filtros e helpers

- O Devise já possui as controladoras adequadas para tratar o login de usuário
- Mas caso necessário, podemos estendê-las para nossa aplicação
- Além disso, o Devise possui vários helpers que nos ajudam a controlar o acesso da aplicação que podem ser usadas dentro das nossas **controladoras** e **Views**
- Para fazer uma controladora requerer autenticação basta adicionar o seguinte filtro:

```
before_action :authenticate_user!
```



# Devise - Controladoras, filtros e helpers

- Vamos adicionar esse filtro para as nossas ações
- Lembre-se que todas as ações que apenas apresentam dados de música não necessitam que o usuário esteja autenticado!
- Adicione o seguinte filtro em **MoviesController**, **ActorsController** e **DirectorsController**

```
before_action :authenticate_user!, except: [:index, :show]
```

- Para testar, abra uma janela anônima do seu navegador e tente acessar a página <http://localhost:3000/movies/new>



# Devise - Controladoras, filtros e helpers

- O Devise também nos dá três helpers muito úteis que podem ser utilizados nas controladoras e views:
  - **user\_signed\_in?** - retorna se o usuário atual está logado na aplicação
  - **current\_user** - retorna um objeto correspondente ao usuário atual
  - **user\_session** - retorna o objeto da sessão do usuário
- Podemos usar esses helpers para:
  - Adicionar links relacionados ao login no layout da aplicação
  - Colocar mensagens customizadas relacionadas ao usuário (ex: Olá, Arthur)
  - Apresentar determinados botões na tela somente se um usuário estiver logado
  - Criar links customizados a partir do usuário atual



# Devise - Menu

- Use esses helpers para adicionar links relacionados ao login no layout da aplicação. O código completo do layout está [disponível neste link](#)

```
<% if user_signed_in? %>
  <li class="menu-item sign-out">
    <%= link_to("Sign Out", "/users/sign_out", method: :delete) %>
  </li>
<% else %>
  <li class="menu-item sign-in">
    <%= link_to("Sign In", "/users/sign_in") %>
  </li>
  <li class="menu-item sign-up">
    <%= link_to("Sign Up", "/users/sign_up") %>
  </li>
```



# Devise - Botões para usuários logados

- Use esses helpers para apresentar os botões de editar e excluir somente para usuários logado no sistema. Veja o exemplo da página **Actors#index**

```
<td><%= link_to 'Show', actor %></td>
<% if user_signed_in? %>
  <td><%= link_to 'Edit', edit_actor_path(actor) %></td>
  <td><%= link_to 'Destroy', actor, method: :delete, data: { confirm: 'Are you sure?' } %></td>
<% end %>
```



# Devise - Botões para usuários logados

- Use esses helpers para apresentar os botões de editar e excluir somente para usuários logado no sistema. Veja o exemplo da página **Actors#index**

```
<td><%= link_to 'Show', actor %></td>
<% if user_signed_in? %>
  <td><%= link_to 'Edit', edit_actor_path(actor) %></td>
  <td><%= link_to 'Destroy', actor, method: :delete, data: { confirm: 'Are you sure?' } %></td>
<% end %>
```



# MyMovies - Commitando Modificações

- Registre uma nova versão com as modificações feitas até agora:

```
$ git add .
```

```
$ git commit -m "Initial commit"
```

- Atualize suas modificações no Github empurrando seus commits para lá:

```
$ git push origin master
```

# Avaliação de Filmes

---



# MyMovies - O que falta...

- Para exercitar os conceitos de Rails, ao longo da disciplina iremos desenvolver o projeto **MyMovies** onde usuários poderão gerenciar os filmes que assistiram. O aplicativo deve conter:
- Cadastro/Edição de filmes
- Lista de filmes
- Páginas individuais de filmes
- Páginas de Diretores listando os filmes cadastrados desse diretor
- Páginas dos Atores, listando os filmes em que eles atuaram
- Login de Usuário
- **Usuário poderá ter uma lista de filmes assistidos**
- **Usuário poderá classificar os filmes assistidos**



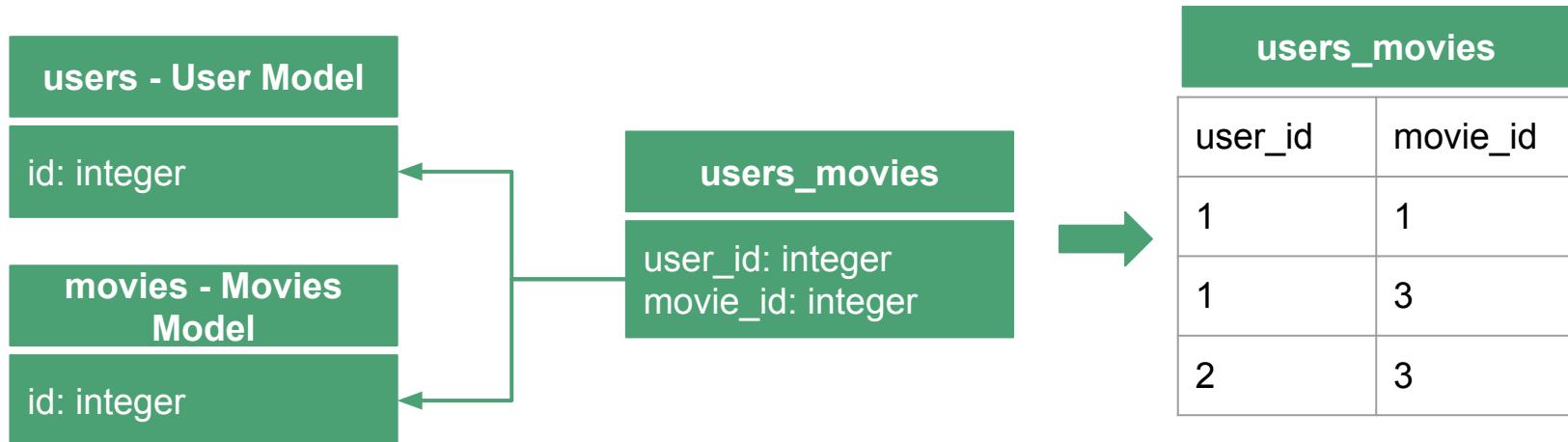
# Avaliação de Filmes

- A Model **User** funciona como outra qualquer em nossa aplicação
- Vamos criar relações com essa model para que usuários possam avaliar filmes
- A relação entre User e Movie será de N x N (muitos para muitos), uma vez que:
  - Um mesmo usuário pode assistir e avaliar vários filmes
  - Um mesmo filme pode ser assistido por vários usuários
- A princípio poderíamos utilizar a relação **has\_and\_belongs\_to\_many** para mapear os filmes assistidos pelos usuários
- Porém, para existe um atributo que só existe quando um usuário assiste um determinado filme: a classificação deste usuário
- Logo, precisamos que a tabela intermediária não guarde apenas as chaves estrangeiras para User e Movie, mas também a avaliação do usuário.



# Associações - N para N

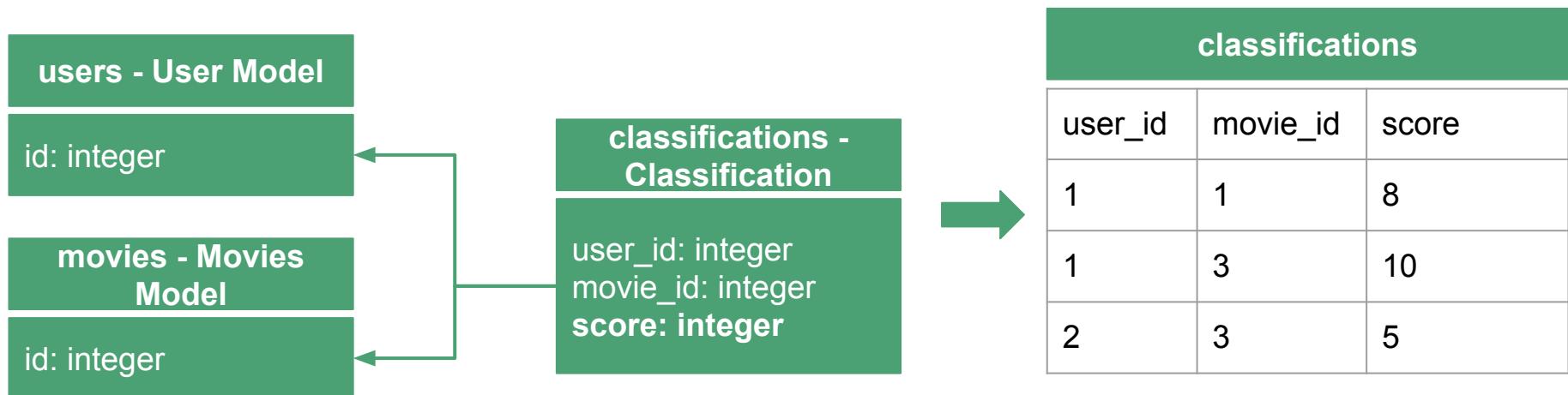
- N para N com `has_and_belongs_to_many`





# Associações - N para N - has\_many :through

- Precisaremos de uma classe intermediária: **Classification**





# Associações - N para N - has\_many :through

- N para N com **has\_many :through**

```
class Movie < ApplicationRecord
  has_many :classifications
  has_many :users, through: :classifications
end

class Classification < ApplicationRecord
  belongs_to :user
  belongs_to :movie
end

class User < ApplicationRecord
  has_many :classifications
  has_many :movies, through: :classifications
end
```



# Associações - N para N - has\_many :through

- Crie a migração para a tabela **classifications**

```
$ rails g migration create_classifications
```

```
class CreateClassifications < ActiveRecord::Migration[5.0]
  def change
    create_table :classifications do |t|
      t.belongs_to :movie, index: true
      t.belongs_to :user, index: true
      t.integer :score
      t.timestamps
    end
  end
end
```



# Rails Console - has\_many :through

```
> user = User.last
> movie = Movie.last
> user.classifications << Classification.new(movie_id: movie.id, score: 8)
> user.reload
> user.movies
> movie.reload
> movie.classifications
> movie.users
```



# Validação em Classification

- Para limitar que um usuário tenha somente uma avaliação por filme, devemos acrescentar a seguinte validação:

```
class Classification < ApplicationRecord
  belongs_to :user
  belongs_to :movie

  validates :user_id, uniqueness: { scope: :movie_id }
  validates :score, numericality: {
    only_integer: true,
    greater_than_or_equal_to: 0,
    less_than_or_equal_to: 10
  }
end
```



# Classificação de usuário

- A página de um filme deve apresentar a classificação do usuário, caso ele já o tenha classificado, ou apresentar um campo para que ele possa inserir uma nota de 0 a 10
- Para isso, vamos adicionar uma nova ação em MoviesController, assim como uma nova rota:

```
post 'movies/:id/classifications' => "movies#create_classification"
```



# Classificação de usuário - View e Controller

- Altere o arquivo **app/views/movies/show.html.erb** para o [código desse link](#)
- Esse arquivo agora têm um pequeno formulário que permite ao usuário logado classificar em filme
- Porém, ainda não criamos a ação na controller para tratar essa nova requisição
- Altere o arquivo **app/controllers/movies\_controller.rb** para o [código desse link](#)
- Repare que adicionamos a nova ação no filtro existente
- Veja o que a ação **create\_classification** faz



# Classificação de usuário - Testando

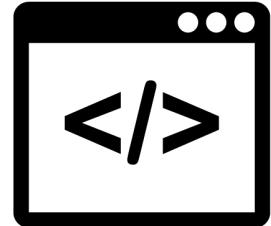
- Execute a aplicação como usuário **logado**:
  - Entre na página de um filme e classifique-o
  - Tente classificar com um valor inválido
  - Entre em uma página de um filme que você já classificou
- Execute a aplicação como usuário **não logado**:
  - Entre na página de um filme e verifique se é apresentada a classificação



# Classificação Geral

- É importante apresentar também a média de classificações de todos os usuários para um filme
- Além disso, podemos apresentar essa informação na tabela geral de filmes e ordená-los de acordo com a melhor classificação
- Para isso, implemente o seguinte método na Model **Movie**:

```
def average_score
  classifications.average(:score).to_f
end
```



## Exercício

- Acrescente a pontuação média dos filmes nas páginas:
  - Movies#index
  - Movies#show
- Crie uma página para o usuário, onde é listado os filmes que já assistiu com sua classificação. Cada item da lista deve ser um link para a página pública daquele filme



# MyMovies - Commitando Modificações

- Registre uma nova versão com as modificações feitas até agora:

```
$ git add .
```

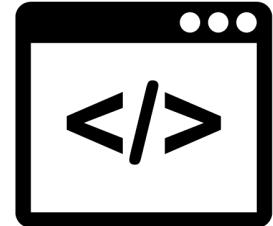
```
$ git commit -m "Initial commit"
```

- Atualize suas modificações no Github empurrando seus commits para lá:

```
$ git push origin master
```

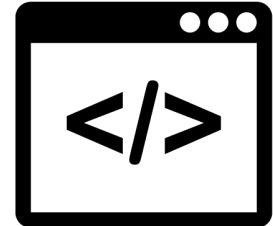
# Atividades Sugeridas!

---



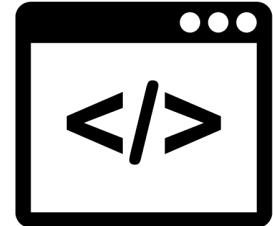
## Exercício

- Em vários momentos é importante usarmos imagens dentro de nossa página
- Use a [Gem Carrierwave](#) para adicionar imagens aos modelos Movie, Actor e Director



## Exercício

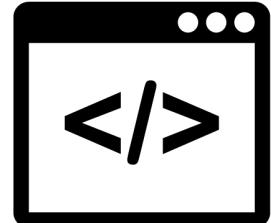
- Termine de ocultar os botões que levam para páginas que requerem autenticação para usuários **não logados**
- Existe um banco de dados online com informações sobre filmes que podemos estar utilizando em nosso projeto chamado TheMovieDB
- Crie scripts Ruby que consuma essa API para popular nosso banco de dados. Para isso você terá que ler a documentação da API
- Para facilitar o trabalho, pode usar essa GEM que encapsula vários detalhes relacionados ao acesso dessa API



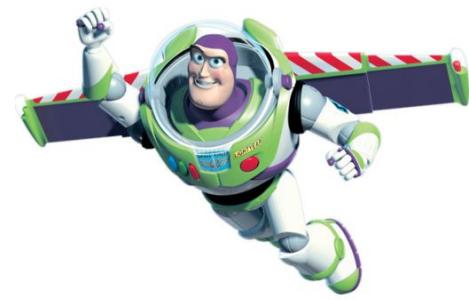
# Exercício

- Existem vários temas de Front-end (HTML + CSS + JavaScript) que podem estar sendo utilizados para deixar nossas páginas com melhor apresentação
- O Bootstrap é o Framework de front-end mais popular devido sua facilidade de uso, responsividade e estilização.
- Estude como usar o Bootstrap e aplique seus elementos no estilo das nossas páginas para deixá-las mais profissionais
- Você pode usar essa GEM para facilitar o trabalho

# Exercício



- Heroku é oferece uma Infraestrutura como Serviço para podermos implantar nossos sistemas
- Nós podemos utilizá-la para fazer implantações de testes gratuitamente
- Siga esse tutorial e tente implantar a sua aplicação **MyMovies** na infraestrutura no Heroku



# Para o Futuro - Estudos Avançados

- Ruby Gems voltada para Rails
- Rails avançado
- MVC
- Testes automatizados
- Front-end: HTML, CSS e JavaScript
- Ambiente de produção e implantação
- Sessão de usuários
- Padrões de Projetos de Aplicações Web e Rails
- Consultas, **Scopes** e Manipulação avançada de Modelos
- REST
- Git



# Para o Futuro

- Continue a estudar Ruby on Rails:
  - Leia livros
  - Faça exercícios
  - **Crie seu próprio programa**
  - Contribua para algum **software livre** existente
- **A melhor maneira de aprender algo em programação é exercitando e repetindo cada vez mais!**



**NEW WEB APP**

**IS COMING**

[memegenerator.net](http://memegenerator.net)



**KEEP CALM  
AND  
CODE IN  
RUBY**

# 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](mailto:esposte@ime.usp.br)

# Obrigado!