

LABORATORIO 8  
Ruby on Rails avanzado

Lara Avila Cesar Jesús  
Calagua Mallqui Jairo Andre

20210279F

Facultad de Ciencias, Universidad Nacional de Ingeniería

noviembre del 2023

## Vistas Parciales

La creación de una vista parcial implica dividir la vista en secciones más pequeñas y reutilizables. En el caso del código que se encuentra en la actividad, se está sugiriendo dividir la vista de películas en una vista parcial llamada `\_movie.html.erb`.

Primero, creamos un archivo llamado `\_movie.html.erb` en la carpeta de vistas (`app/views/movies/`). Copiamos el código HTML proporcionado en ese archivo.

```
_movie.html.erb X movie.rb
C: > Users > calag > Desktop > Desarrollo-software-2023-main > Semana7 > my
1 <div class="row">
2   <div class="col-8">
3     <%= link_to movie.title, movie_path(movie) %>
4   </div>
5   <div class="col-2">
6     <%= movie.rating %>
7   </div>
8   <div class="col-2">
9     <%= movie.release_date.strftime('%F') %>
10  </div>
11 </div>
```

## Validaciones de Modelos

Las validaciones en Rails se usan para asegurarse de que los datos ingresados cumplan ciertos criterios. En el modelo `Movie`, se están agregando validaciones para campos como `title`, `release\_date`, y `rating`.

Debemos agregar el código proporcionado en el modelo `Movie` para incluir estas validaciones.

```
_movie.html.erb X movie.rb X
C: > Users > calag > Desktop > Desarrollo-software-2023-main > Semana7 > myrottenpotatoes > app > models > mov
1 class Movie < ActiveRecord::Base
2   def self.all_ratings ; %w[G PG PG-13 R NC-17] ; end # shortcut: array of strings
3   validates :title, :presence => true
4   validates :release_date, :presence => true
5   validate :released_1930_or_later # uses custom validator below
6   validates :rating, :inclusion => {:in => Movie.all_ratings},
7     :unless => :grandfathered?
8   def released_1930_or_later
9     errors.add(:release_date, 'must be 1930 or later') if
10      release_date && release_date < Date.parse('1 Jan 1930')
11   end
12   @@grandfathered_date = Date.parse('1 Nov 1968')
13   def grandfathered?
14     release_date && release_date < @@grandfathered_date
15   end
16 end
17 end
18
19 class Movie < ActiveRecord::Base
20   before_save :capitalize_title
21   def capitalize_title
22     self.title = self.title.split(/\s+/).map(&:downcase).
23     map(&:capitalize).join(' ')
24   end
25 end
```

Ahora, debo proceder con la generación de una nueva migración usando “rails generate migration CreateMoviegoers”, así:

```
_movie.html.erb  movie.rb  20231115051140_create_moviegoers.rb
> Desktop > Desarrollo-software-2023-main > Semana7 > myrottenpotatoes > db > migrate
1  class CreateMoviegoers < ActiveRecord::Migration[7.0]
2    def change
3      create_table :moviegoers do |t|
4
5        t.timestamps
6      end
7    end
8  end
```

Ejecutamos la Migración nueva con rails db:migrate lo que debería de crear la tabla ‘moviegoers’ en mi base de datos.

```
PS C:\Users\calag\desktop\Desarrollo-software-2023-main\Semana7\myrottenpotatoes\db\migrate> rails db:migrate
== 20231115051140 CreateMoviegoers: migrating =====
====
-- create_table(:moviegoers)
   -> 0.0070s
== 20231115051140 CreateMoviegoers: migrated (0.0077s) =====
====
```

Y para aplicar los cambios, reiniciamos el servidor rails y lo que podemos ver es lo siguiente:

127.0.0.1:3000/movies

[Add new movie](#)

### All Movies

Movie Title	Rating	Release Date	More Info
Star Wars	PG	1977-04-25 00:00:00 UTC	<a href="#">More about Star Wars</a>
Aladdin	G	1992-11-25 00:00:00 UTC	<a href="#">More about Aladdin</a>
When Harry Met Sally	R	1989-07-21 00:00:00 UTC	<a href="#">More about When Harry Met Sally</a>
The Help	PG-13	2011-08-10 00:00:00 UTC	<a href="#">More about The Help</a>
Raiders of the Lost Ark	PG	1981-06-12 00:00:00 UTC	<a href="#">More about Raiders of the Lost Ark</a>
Aladdin	G	1992-11-25 00:00:00 UTC	<a href="#">More about Aladdin</a>
When Harry Met Sally	R	1989-07-21 00:00:00 UTC	<a href="#">More about When Harry Met Sally</a>
The Help	PG-13	2011-08-10 00:00:00 UTC	<a href="#">More about The Help</a>
Raiders of the Lost Ark	PG	1981-06-12 00:00:00 UTC	<a href="#">More about Raiders of the Lost Ark</a>
Star Wars	PG	1977-04-25 00:00:00 UTC	<a href="#">More about Star Wars</a>
Requiem for a Dream	R	2000-10-27 00:00:00 UTC	<a href="#">More about Requiem for a Dream</a>
Aladdin	G	1992-11-25 00:00:00 UTC	<a href="#">More about Aladdin</a>
When Harry Met Sally	R	1989-07-21 00:00:00 UTC	<a href="#">More about When Harry Met Sally</a>
The Help	PG-13	2011-08-10 00:00:00 UTC	<a href="#">More about The Help</a>
Raiders of the Lost Ark	PG	1981-06-12 00:00:00 UTC	<a href="#">More about Raiders of the Lost Ark</a>
Aladdin	G	1992-11-25 00:00:00 UTC	<a href="#">More about Aladdin</a>
When Harry Met Sally	R	1989-07-21 00:00:00 UTC	<a href="#">More about When Harry Met Sally</a>
The Help	PG-13	2011-08-10 00:00:00 UTC	<a href="#">More about The Help</a>
Raiders of the Lost Ark	PG	1981-06-12 00:00:00 UTC	<a href="#">More about Raiders of the Lost Ark</a>
Aladdin	G	1992-11-25 00:00:00 UTC	<a href="#">More about Aladdin</a>
When Harry Met Sally	R	1989-07-21 00:00:00 UTC	<a href="#">More about When Harry Met Sally</a>
The Help	PG-13	2011-08-10 00:00:00 UTC	<a href="#">More about The Help</a>
Raiders of the Lost Ark	PG	1981-06-12 00:00:00 UTC	<a href="#">More about Raiders of the Lost Ark</a>
T	G	2023-10-31 00:00:00 UTC	<a href="#">More about T</a>
Ed	G	2023-10-31 00:00:00 UTC	<a href="#">More about Ed</a>
	G	2023-11-05 00:00:00 UTC	<a href="#">More about The Roundup</a>

## Comprobación de Resultados en la Consola

Abrimos la consola de Rails ejecutando `rails console` en la terminal para verificar los resultados. Podemos crear una nueva película y verificar si las validaciones funcionan como se esperaba, así:

```
PS C:\Users\calag\desktop\Desarrollo-software-2023-main\Semana7\myrottenpotatoes\db\migrate> rails console
Loading development environment (Rails 7.0.8)
irb(main):001> m = Movie.new(title: '', rating: 'RG', release_date: '1929-01-01')
=>
#<Movie:0x0000015f96d5cfe0
...
irb(main):002> m.valid?
=> false
irb(main):003> m.errors[:title]
=> ["can't be blank"]
irb(main):004> m.errors[:rating]
=> []
irb(main):005> m.errors[:release_date]
=> ["must be 1930 or later"]
irb(main):006> m.errors.full_messages
=> ["Title can't be blank", "Release date must be 1930 or later"]
irb(main):007>
```

## Controladores

El código del controlador `MoviesController` administra la creación, edición, actualización y eliminación de películas.

Abrimos el archivo `movies\_controller.rb` en el directorio `app/controllers`.

Sustituimos el contenido actual del archivo por el código que se nos proporciona, así:

Contenido actual:

```
movies_controller.rb  _movie.html.erb  movie.rb  20231115051140_create_movie

C:\Users\calag\Desktop\Desarrollo-software-2023-main\Semana7\myrottenpotatoes> app > contro

1 class MoviesController < ApplicationController
2   def index
3     @movies = Movie.all
4   end
5   def show
6     id = params[:id] # retrieve movie ID from URI route
7     @movie = Movie.find(id) # look up movie by unique ID
8     # will render render app/views/movies/show.html.haml by default
9   end
10  def new
11    @movie = Movie.new
12  end
13  def create
14    if (@movie = Movie.create(movie_params))
15      redirect_to movies_path, :notice => "#{@movie.title} created."
16    else
17      flash[:alert] = "Movie #{@movie.title} could not be created: " +
18        @movie.errors.full_messages.join(",")
19      render 'new'
20    end
21  end
22  def edit
23    @movie = Movie.find params[:id]
24  end
25  def update
26    @movie = Movie.find params[:id]
27    if (@movie.update_attributes(movie_params))
28      redirect_to movie_path(@movie), :notice => "#{@movie.title} updated."
29    else
30      flash[:alert] = "#{@movie.title} could not be updated: " +
31        @movie.errors.full_messages.join(",")
32      render 'edit'
33    end
34  end
35  def destroy
36    @movie = Movie.find(params[:id])
37    @movie.destroy
38  end
39 end
```

Nuevo contenido:

```
movies_controller.rb X _movie.html.erb movie.rb 20231115051140_create_mov
C: > Users > calag > Desktop > Desarrollo-software-2023-main > Semana7 > myrottenpotatoes > app > cont
1  class MoviesController < ApplicationController
2    def new
3      @movie = Movie.new
4    end
5
6    def create
7      @movie = Movie.create(movie_params)
8      if @movie.save
9        redirect_to movies_path, notice: "#{@movie.title} created."
10     else
11       flash[:alert] = "Movie #{@movie.title} could not be created: " +
12         @movie.errors.full_messages.join(",")
13       render 'new'
14     end
15   end
16
17   def edit
18     @movie = Movie.find(params[:id])
19   end
20
21   def update
22     @movie = Movie.find(params[:id])
23     if @movie.update_attributes(movie_params)
24       redirect_to movie_path(@movie), notice: "#{@movie.title} updated."
25     else
26       flash[:alert] = "#{@movie.title} could not be updated: " +
27         @movie.errors.full_messages.join(",")
28       render 'edit'
29     end
30   end
31
32   def destroy
33     @movie = Movie.find(params[:id])
34     @movie.destroy
35     redirect_to movies_path, notice: "#{@movie.title} deleted."
36   end
end
```

Este código define acciones para las operaciones CRUD relacionadas con las películas. Estas acciones son utilizadas en las vistas 'new', 'edit' y 'show'. Además, el método 'movie\_params' es una medida de seguridad que filtra los parámetros permitidos antes de asignarlos a un modelo, lo cual es esencial para proteger nuestra aplicación contra ataques de asignación masiva.

No está de más el asegurarnos de tener las rutas correspondientes en nuestro archivo 'config/routes.rb':

```
movies_controller.rb  routes.rb  _movie.html.erb
C: > Users > calag > Desktop > Desarrollo-software-2023-main > Sem
1  Myrottenpotatoes::Application.routes.draw do
2    resources :movies
3    root :to => redirect('/movies')
4  end
```

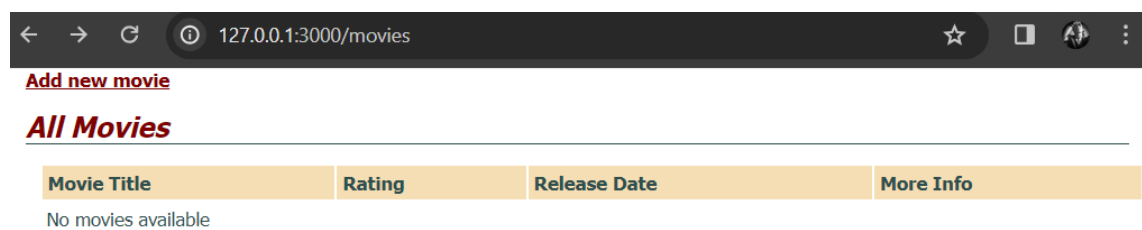
Esto creará rutas RESTful para las aplicaciones CRUD en nuestro controlador.

Finalmente, reiniciamos nuestro servidor para aplicar los cambios.

Como ya se nos había explicado, nos aparecerán errores que debemos corregir, es por eso que tuve que modificar el archivo 'index.html.haml' de la siguiente manera:

```
14  %tbody
15  - if @movies
16  - @movies.each do |movie|
17    %tr
18      %td= movie.title
19      %td= movie.rating
20      %td= movie.release_date
21      %td= link_to "More about #{movie.title}", movie_path(movie)
22  - else
23    %tr
24      %td(colspan="4") No movies available
```

Lo que hace que en mi navegador ya aparezca lo siguiente:



← → ↻ ⓘ 127.0.0.1:3000/movies ☆ □ 🌐 ⋮

[Add new movie](#)

## All Movies

Movie Title	Rating	Release Date	More Info
No movies available			

## Antes de Guardar un Modelo

Se está sugiriendo agregar un método en el modelo `Movie` que capitalice el título antes de que se guarde en la base de datos. Esto se hace utilizando el callback `before\_save`.

Abrimos el archivo Movie y agregamos el código que se nos proporciona al modelo para definir el callback `before\_save`.

Agrega el método `capitalize\_title` al modelo `Movie` para que cada título se estandarice antes de guardarse en la base de datos.

```
18 class Movie < ActiveRecord::Base
19   before_save :capitalize_title
20   def capitalize_title
21     self.title = self.title.split(/\s+/).map(&:downcase).
22     map(&:capitalize).join(' ')
23   end
24 end
```

Guardamos el archivo y realizamos una prueba en la consola:

```
irb(main):008> m = Movie.create!(title: 'STAR wars', release_date: '27-5-1977', rating: 'PG')
TRANSACTION (0.1ms) begin transaction
Movie Create (1.1ms) INSERT INTO "movies" ("title", "rating", "description", "release_date", "created_at", "updated_at") VALUES (?, ?, ?, ?, ?, ?) [{"title", "Star Wars"}, [{"rating", "PG"}, [{"description", nil}, [{"release_date", "1977-05-27 00:00:00"}, [{"created_at", "2023-11-15 06:03:58.100454"}, [{"updated_at", "2023-11-15 06:03:58.100454"}]]
TRANSACTION (2.8ms) commit transaction
=>
#<Movie:0x0000015f97535e60
...
irb(main):009> m.title
=> "Star Wars"
irb(main):010>
```

Notamos que nos devuelve "Star Wars" en lugar de "STAR wars".

## Filtros en el Controlador

Los filtros del controlador, en este caso, se utilizan para verificar si un usuario está autenticado antes de realizar ciertas acciones en la aplicación. Este filtro se está aplicando en el `ApplicationController`.

Nos piden comprobar el resultado del siguiente código:

```
6 class ApplicationController < ActionController::Base
7   before_action :set_current_user # change before_filter
8   protected # prevents method from being invoked by a route
9   def set_current_user
10    # we exploit the fact that the below query may return nil
11    @current_user ||= Moviegoer.where(:id => session[:user_id])
12    redirect_to login_path and return unless @current_user
13  end
14 end
```

Ya que no tengo un controlador y vistas para manejar las sesiones, entonces lo creo de la siguiente manera:

```
C: > Users > calag > Desktop > Desarrollo-software-2023-main > Semana7 > myrottenpotatoes > app > controllers > session
1 class SessionsController < ApplicationController
2   def new
3     render 'new'
4   end
5
6   def create
7     user = User.find_by(email: params[:email])
8
9     if user && user.authenticate(params[:password])
10      # Las credenciales son válidas, establece la sesión
11      session[:user_id] = user.id
12      redirect_to root_path, notice: 'Inicio de sesión exitoso'
13    else
14      # Las credenciales son inválidas, muestra un mensaje de error y vuelve al form
15      flash.now[:alert] = 'Credenciales inválidas'
16      render 'new'
17    end
18  end
19
20  def destroy
21    session[:user_id] = nil
22    redirect_to root_path, notice: 'Cierre de sesión exitoso'
23  end
24 end
```

Agrego rutas para el controlador de sesiones en 'config/routes.rb':

```
1 Myrottenpotatoes::Application.routes.draw do
2   resources :movies
3   resources :sessions, only: [:new, :create, :destroy]
4   root :to => redirect('/movies')
5 end
```



Ahora, este filtro se ejecutará antes de cada acción en cualquier controlador que herede de 'ApplicationController'.

Por ejemplo, en el controlador de películas agregaré la siguiente línea de código:

```
routes.rb  movies_controller.rb X  sessions_controller.rb  movie.rb  application_co
C: > Users > calag > Desktop > Desarrollo-software-2023-main > Semana7 > myrottenpotatoes > app > controllers >
1  class MoviesController < ApplicationController
2    before_action :set_current_user, only: [:new, :create, :edit, :update, :destroy]
```

Esto aplicará el filtro solo a las acciones especificadas en la lista 'only'

SSO y autenticación a través de terceros:

Ejecuto el siguiente comando en la terminal "rails generate model Moviegoer name:string provider:string uis:string", y se me generará el archivo moviegoer.rb:

```
moviegoer.rb X  routes.rb  movies_control
C: > Users > calag > Desktop > Desarrollo-software-2023-main
1  class Moviegoer < ActiveRecord::Base
2    def self.create_with_omniauth(auth)
3      Moviegoer.create!(
4        :provider => auth["provider"],
5        :uid => auth["uid"],
6        :name => auth["info"]["name"])
7    end
8  end
```

Autenticación a través de un tercero con OmniAuth

Editaré el archivo 'config/routes.rb' de la siguiente manera:

```
moviegoer.rb  routes.rb X  movies_controller.rb  sessi
C: > Users > calag > Desktop > Desarrollo-software-2023-main > Semana7 > myro
1  Myrottenpotatoes::Application.routes.draw do
2    resources :movies
3    resources :sessions, only: [:new, :create, :destroy]
4
5    get 'auth/:twitter/callback' => 'sessions#create'
6    get 'auth/failure' => 'sessions#failure'
7    get 'auth/twitter', :as => 'login'
8    post 'logout' => 'sessions#destroy'
9
10   root :to => redirect('/movies')
11 end
```

Ahora también debo de editar el archivo sessions\_controller.rb, así:

```
iegoer.rb  sessions_controller.rb X  routes.rb  movies_controller.rb  movie.rb  applica
ers > calag > Desktop > Desarrollo-software-2023-main > Semana7 > myrottenpotatoes > app > controllers > sessions_cont

class SessionsController < ApplicationController
  skip_before_action :set_current_user

  def new
    render 'new'
  end

  def create
    if auth_hash.present?
      # Autenticación a través de terceros
      user = User.find_or_create_from_auth_hash(auth_hash)
    else
      # Autenticación tradicional
      user = User.find_by(email: params[:email])

      if user && user.authenticate(params[:password])
        # Las credenciales son válidas, establece la sesión
        session[:user_id] = user.id
        redirect_to root_path, notice: 'Inicio de sesión exitoso'
        return
      else
        # Las credenciales son inválidas, muestra un mensaje de error y vuelve al formulario
        flash.now[:alert] = 'Credenciales inválidas'
        render 'new'
        return
      end
    end

    # Autenticación exitosa a través de terceros, establece la sesión
    session[:user_id] = user.id
    redirect_to root_path, notice: 'Inicio de sesión exitoso a través de terceros'
  end
end
```

Edito, además, el archivo omniauth.rb:

```
moviegoer.rb  sessions_controller.rb  omniauth.rb X  routes.rb  n
C: > Users > calag > Desktop > Desarrollo-software-2023-main > Semana7 > myrottenpotatoes > confi
1  # Replace API_KEY and API_SECRET with the values you got from Twitter
2  Rails.application.config.middleware.use OmniAuth::Builder do
3    provider :twitter, "API_KEY", "API_SECRET"
4  end
```

Debo reemplazar las API con las que obtenga de Twitter, pero lamentablemente la página no me dejó crear una cuenta por error de ellos:

**Paso 4 de 5**

## Agregar un número de teléfono

Introduce el número de teléfono que quieres asociar a tu cuenta de X. Se te enviará un código de verificación a ese número.

Código de país

+51 Perú

✓

Tu número de teléfono

958927550

Permite que las personas que tienen tu número de teléfono te encuentren y se conecten contigo en X. [Más información](#)

✓

Permite que X use tu número de teléfono para personalizar nuestros servicios, incluidos los anuncios (si lo permiten tus preferencias de anuncios). Si no

✓

Siguiente

Alcanzaste el límite de códigos SMS. Inténtalo de nuevo dentro de 24 horas.

(El código SMS nunca llegó a mi celular).

Asociaciones y claves foráneas:

Ejecuto el siguiente comando en la terminal: “rails generate migration create\_review”, así:

```
PS C:\Users\calag\Desktop\Desarrollo-software-2023-main\Semana7\myrottenpotatoes\db> rails generate migration create_reviews
C:/Users/calag/Desktop/Desarrollo-software-2023-main/Semana7/myrottenpotatoes/config/initializers/omniauth.rb:2:in '<main>': u
ninitialized constant OmniAuth (NameError) from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/engine.
rb:667:in 'load'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/engine.rb:667:in 'block in load_config_initialize
r'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/activesupport-7.0.8/lib/active_support/notifications.rb:208:in 'instrument
'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/engine.rb:666:in 'load_config_initializer'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/engine.rb:620:in 'block (2 levels) in <class:Engi
ne>'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/engine.rb:619:in 'each'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/engine.rb:619:in 'block in <class:Engine>'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/initializable.rb:32:in 'instance_exec'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/initializable.rb:32:in 'run'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/initializable.rb:61:in 'block in run_initializers
'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:228:in 'block in tsort_each'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:350:in 'block (2 levels) in each_strongly_connected_component'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:422:in 'block (2 levels) in each_strongly_connected_component_from'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:431:in 'each_strongly_connected_component_from'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:421:in 'block in each_strongly_connected_component_from'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/initializable.rb:50:in 'each'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/initializable.rb:50:in 'tsort_each_child'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:415:in 'call'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:415:in 'each_strongly_connected_component_from'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:349:in 'block in each_strongly_connected_component'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:347:in 'each'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:347:in 'call'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:347:in 'each_strongly_connected_component'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:226:in 'tsort_each'
    from C:/Ruby30-x64/lib/ruby/3.0.0/tsort.rb:205:in 'tsort_each'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/initializable.rb:60:in 'run_initializers'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/railties-7.0.8/lib/rails/application.rb:372:in 'initialize!'
    from C:/Users/calag/Desktop/Desarrollo-software-2023-main/Semana7/myrottenpotatoes/config/environment.rb:5:in '<main>'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/bootsnap-1.16.0/lib/bootsnap/load_path_cache/core_ext/kernel_require.rb:32
:in 'require'
    from C:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/bootsnap-1.16.0/lib/bootsnap/load_path_cache/core_ext/kernel_require.rb:32
```

Como era de esperarse, no se puede realizar la creación satisfactoriamente debido a la falta de las API.

Lo que haré entonces es mostrar los pasos que debería de seguir de haber funcionado correctamente esta última parte.

Editar el archivo ‘db/migrate/\*\_create\_reviews.rb’ de la siguiente manera:

# CreateReviews migration

class CreateReviews < ActiveRecord::Migration

  def change

    create\_table 'reviews' do |t|

      t.integer 'potatoes'

      t.text 'comments'

      t.references 'moviegoer'

      t.references 'movie'

    end

  end

end

Luego, crear el modelo 'Review' en 'app/models/review.rb', así:

```
# review.rb

class Review < ActiveRecord::Base

  belongs_to :movie

  belongs_to :moviegoer

end
```

Finalmente, debo agregar la siguiente línea dentro de las clases 'Movie' y 'Moviegoer' en sus archivos respectivos ('movie.rb' y 'moviegoer.rb').

```
# Inside movie.rb and moviegoer.rb

has_many :reviews
```

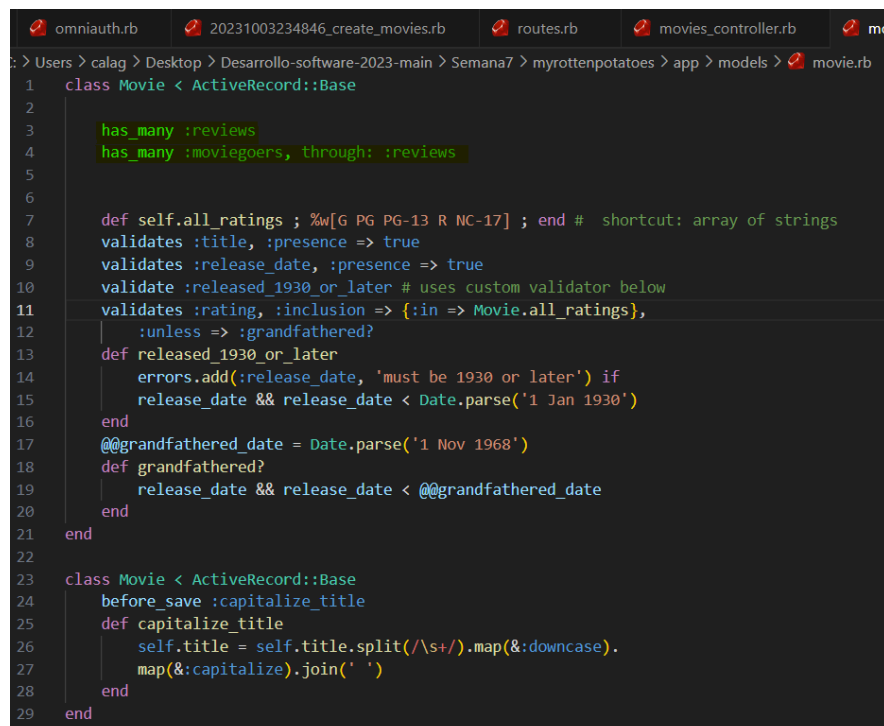
Luego, ejecutar 'rake db:migrate' para aplicar la migración.

Con estos cambios ya estaría configurado la autenticación a través de terceros, y también hemos creado asociaciones y claves foráneas para las críticas (reviews) de películas.

### Asociaciones indirectas

#### Editar el modelo Movie

En el modelo 'Movie' debemos agregar la siguiente línea de texto para establecer la asociación indirecta a través de 'reviews', así:



```
omniauth.rb 20231003234846_create_movies.rb routes.rb movies_controller.rb movie.rb
> Users > calag > Desktop > Desarrollo-software-2023-main > Semana7 > myrottenpotatoes > app > models > movie.rb
1  class Movie < ActiveRecord::Base
2
3    has_many :reviews
4    has_many :moviegoers, through: :reviews
5
6
7    def self.all_ratings ; %w[G PG PG-13 R NC-17] ; end # shortcut: array of strings
8    validates :title, :presence => true
9    validates :release_date, :presence => true
10   validate :released_1930_or_later # uses custom validator below
11   validates :rating, :inclusion => {:in => Movie.all_ratings},
12     :unless => :grandfathered?
13   def released_1930_or_later
14     errors.add(:release_date, 'must be 1930 or later') if
15       release_date && release_date < Date.parse('1 Jan 1930')
16   end
17   @@grandfathered_date = Date.parse('1 Nov 1968')
18   def grandfathered?
19     release_date && release_date < @@grandfathered_date
20   end
21 end
22
23 class Movie < ActiveRecord::Base
24   before_save :capitalize_title
25   def capitalize_title
26     self.title = self.title.split(/\s+/).map(&:downcase).
27       map(&:capitalize).join(' ')
28   end
29 end
```

Consulta SQL:

Se nos proporciona la siguiente consulta SQL:

```
SELECT movies.*  
FROM movies  
JOIN reviews ON movies.id = reviews.movie_id  
JOIN moviegoers ON moviegoers.id = reviews.moviegoer_id  
WHERE moviegoers.id = 1;
```

Esta se traduce como: “Selecciona todas las columnas de la tabla ‘movies’ donde hay una relación entre ‘movies’, ‘reviews’, y ‘moviegoers’ y donde el ‘id’ de ‘moviegoers’ es igual a 1”.

Ejemplo de Uso de Asociaciones Indirectas

En el código proporcionado:

```
inception = Movie.find_by(title: 'Inception')  
alice, bob = Moviegoer.find(alice_id, bob_id)  
  
alice_review = Review.new(potatoes: 5)  
bob_review = Review.new(potatoes: 3)  
  
inception.reviews = [alice_review, bob_review]  
alice.reviews << alice_review  
bob.reviews << bob_review  
  
reviewers_names = inception.reviews.map { |r| r.moviegoer.name }
```

Se están creando instancias de películas, críticas y espectadores, y se están estableciendo las asociaciones a través de las críticas. Al final, se obtienen los nombres de los críticos que han revisado la película “Inception”.

Finalmente, no debemos olvidar ejecutar las migraciones después de realizar todos los cambios en la estructura de la base de datos utilizando “rake db:migrate”