


Laboratorio de Computación IV

Clase 21

Andrés Fortier

Repaso - Listados



- Tres problemas recurrentes
 - Restringir los resultados (búsquedas o filtros)
 - Un campo de búsqueda por propiedad a filtrar.
 - Un único campo de búsqueda para todas las propiedades con coincidencias parciales.
 - Ordenar.
 - Pagar.

Antes de comenzar

- Rails admin usa kaminari para paginar y genera conflictos con will_paginate
 - Crear archivo `/config/initializers/kaminari.rb`

```
/config/initializers/kaminari.rb
```

```
Kaminari.configure do |config|  
  config.page_method_name = :per_page_kaminari  
end
```

Perfil de usuario



- Queremos agregar datos de perfil al usuario
 - Nombre.
 - Apellido.
 - Fecha de nacimiento.
- Debe ser opcional
 - Registración sólo e-mail y password.
 - Edición de perfil completa.

Perfil de usuario



- Opciones de implementación
 - Agregar campos a la clase `User`.
 - Crear un modelo `Profile`
 - Editarlo como parte de `User` (modelo anidado).
 - Editarlo como un objeto separado.

Perfil de usuario

```
$ rails g model Profile first_name:string last_name:string  
date_of_birth:date user:references
```

```
/app/models/profile.rb
```

```
class Profile < ActiveRecord::Base  
  belongs_to :user  
end
```

Perfil de usuario

```
/app/models/user.rb
```

```
class User < ActiveRecord::Base
  ...
  has_many :articles, foreign_key: "author_id"

  has_one :profile, :dependent => :destroy

  after_create :create_profile

  def create_profile
    Profile.create(user: self)
  end
  ...
end
```

Perfil de usuario

```
/db/migrate/XYZ_create_profiles.rb
```

```
class CreateProfiles < ActiveRecord::Migration
  def change
    create_table :profiles do |t|
      t.string :first_name
      t.string :last_name
      t.date :date_of_birth
      t.references :user, index: true

      t.timestamps
    end
  end
end
```


Perfil de usuario

```
/db/migrate/XYZ_create_profiles.rb
```

```
class CreateProfiles < ActiveRecord::Migration
  def up
    create_table :profiles do |t|
      t.string :first_name
      t.string :last_name
      t.date :date_of_birth
      t.references :user, index: true

      t.timestamps
    end

    User.find_each do |user|
      Profile.create!(user: user)
    end
  end
end
...
```

Perfil de usuario

```
/db/migrate/XYZ_create_profiles.rb
```

```
...  
def down  
  drop_table :profiles  
end  
  
end
```

Perfil de usuario



- Hagan un commit y armen un nuevo branch.
- Pasos
 - Modificar el formulario de edición de usuario de devise.
 - Permitir actualizar el perfil a través del usuario.

Modelos anidados

- Indicarle a AR que se puede actualizar un modelo a través de otro.

```
/app/models/user.rb
```

```
class User < ActiveRecord::Base
  ...
  has_many :articles, foreign_key: "author_id"

  has_one :profile, :dependent => :destroy
  accepts_nested_attributes_for :profile
  after_create :create_profile
  ...
end
```

Modelos anidados

- Agreguemos un link para editar el perfil

```
/app/views/layouts/_navigation_links.html.erb
```

```
<% if user_signed_in? %>
  <li>
    <%= link_to "My Profile", edit_user_registration_path %>
  </li>
  <li>
    <%= link_to('Logout', destroy_user_session_path, :method
=> :delete) %>
  </li>
<% else %>
  ...
end
```

Modelos anidados

- Modificamos el formulario generado por `rails_layout`

```
/app/views/devise/registrations/edit.html.erb
```


```
<%= form_for(resource ...) do |f| %>  
  ...  
<% end %>
```

Modelos anidados

```
/app/views/devise/registrations/edit.html.erb
```

```
...
<%= f.fields_for :profile do |ff| %>
  <div class="form-group">
    <%= ff.label :first_name %>
    <%= ff.text_field :first_name, class: 'form-control' %>
  </div>
  <div class="form-group">
    <%= ff.label :last_name %>
    <%= ff.text_field :last_name, class: 'form-control' %>
  </div>
  <div class="form-group">
    <%= ff.label :date_of_birth %>
    <%= ff.date_field :date_of_birth, class: 'form-control'
%>
  </div>
<% end %>
...
```

Modelos anidados



- Finalmente debemos indicarle al controller que esos parámetros son permitidos.
- Para esto debemos ajustar Devise
 - Crear un nuevo controller.
 - Indicarle a device que use ese controller para registrar.

Modelos anidados

```
/app/controllers/registrations_controller.rb
```

```
class RegistrationsController <
  Devise::RegistrationsController

  private

  def account_update_params
    params
      .require(:user)
      .permit(:email, :password, :password_confirmation,
              :current_password,
              profile_attributes: [
                :id,
                :first_name,
                :last_name,
                :date_of_birth
              ])
  end
end
```

Modelos anidados

/config/routes.rb

```
...  
devise_for :users,  
  :controllers => { registrations: 'registrations' }  
...
```

Modelo independiente

- No necesitamos un CRUD completo
 - Read en las vistas que lo necesiten.
 - Update.

Modelo independiente

- Modifiquemos el navbar

```
/app/views/layouts/_navigation_links.html.erb
```

```
<% if user_signed_in? %>
  <li> <p class="navbar-text"> Logged in as
    <% if current_user.profile.first_name.present? %>
      <%= current_user.profile.first_name %>
    <% else %>
      <%= current_user.email %>
    <% end %>
  </p> <li>
    <%= link_to('Logout', destroy_user_session_path, :method
=> :delete) %>
  </li>
<% else %>
  ...
<% end %>
```

Modelo independiente

- Programemos el edit/update

```
/config/routes.rb
```

```
Rails.application.routes.draw do
```

```
  ...
```

```
    get '/profile/edit', to: 'profiles#edit',  
      as: 'edit_profile'
```

```
    patch '/profile', to: 'profiles#update'
```

```
end
```

Modelo independiente

- Agreguemos un link para editar en el navbar

```
/config/routes.rb
```

```
<%# add navigation links to this file %>
<% if user_signed_in? %>
  <li> <p class="navbar-text"> Logged in as
    <% if current_user.profile.first_name.present? %>
      <%= current_user.profile.first_name %>
    <% else %>
      <%= current_user.email %>
    <% end %>
  </p> </li>
  <li>
    <%= link_to "My Profile", edit_profile_path %>
  </li>
  ...
end %>
```

Modelo independiente

- Creemos el controller (sin manejo de errores)

```
/app/controllers/profiles_controller.rb  
class ProfilesController < ApplicationController  
  after_action :verify_authorized  
  
  def edit  
    @profile = current_user.profile  
    authorize @profile  
  end  
  
  ...  
end
```

Modelo independiente

```
/app/controllers/profiles_controller.rb
```

```
...  
def update  
  @profile = current_user.profile  
  authorize @profile  
  @profile.update!(profile_params)  
  flash[:notice] = "Your profile has been updated."  
  render 'edit'  
end  
  
private  
  
  def profile_params  
    params.require(:profile).permit(:first_name,  
:last_name, :date_of_birth)  
  end  
end
```


Modelo independiente

- Creemos una política

```
/app/policies/profile_policy.rb
```

```
class ProfilePolicy < ApplicationPolicy

  def update?
    user.present? &&
      ((record.user == user) || (user.has_role? :admin))
  end

end
```

Modelo independiente

- Y finalmente la vista



Modelo independiente

```
/app/views/profiles/edit.html.erb
```

```
<%= form_for @profile do |f| %>
  <div class="form-group">
    <%= f.label :first_name %>
    <%= f.text_field :first_name, class: 'form-control' %>
  </div>
  <div class="form-group">
    <%= f.label :last_name %>
    <%= f.text_field :last_name, class: 'form-control' %>
  </div>
  <div class="form-group">
    <%= f.label :date_of_birth %>
    <%= f.date_field :date_of_birth, class: 'form-control'
%>
  </div>
  <p>
    <%= f.submit 'Update', :class => 'btn btn-primary' %>
  </p>
<% end %>
```

Modelo independiente

- Con esto completamos la edición.