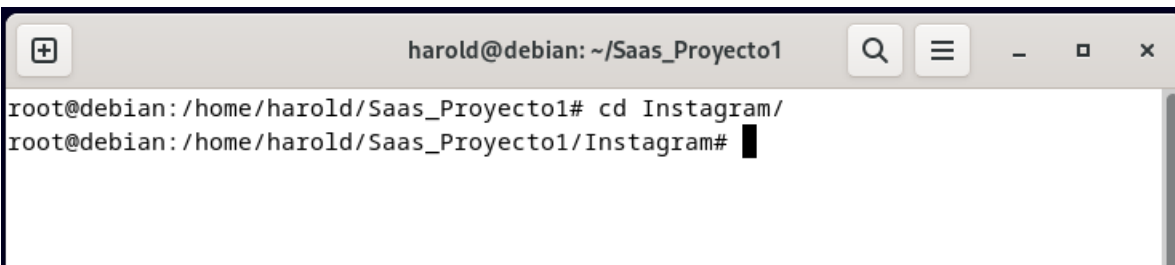


Trabajado por Harold Garcia y Alli Herrera.

1. Como primer paso e inicio del desarrollo de la aplicación se debe crear un proyecto en este caso con el nombre Instagram.

```
root@debian:/home/harold/Saas_Proyecto1# rails new Instagram
create
create  README.md
create  Rakefile
create  .ruby-version
create  config.ru
create  .gitignore
```

1.1. Creación del nuevo proyecto.

A terminal window titled 'harold@debian: ~/Saas_Proyecto1' with search, menu, and window control icons. The terminal shows the user navigating from the project root to the 'Instagram' subdirectory.

```
root@debian:/home/harold/Saas_Proyecto1# cd Instagram/
root@debian:/home/harold/Saas_Proyecto1/Instagram#
```

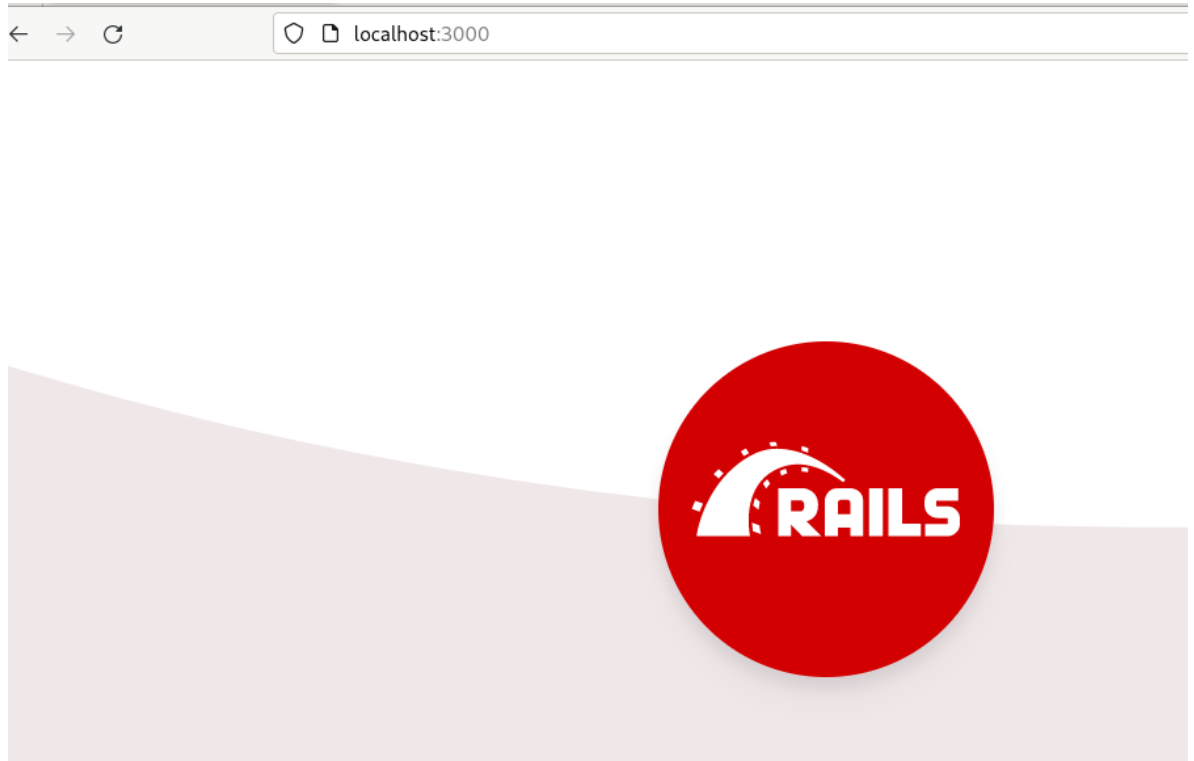
1.2. Una vez terminado el proceso de creación del proyecto, ingresar al directorio.

```
root@debian:/home/harold/Saas_Proyecto1# cd Instagram/
```

1.3. Para visualizar el resultado iniciar el servidor de Rails.

```
root@debian:/home/harold/Saas_Proyecto1/Instagram# rails s
=> Booting Puma
=> Rails 7.1.4 application starting in development
=> Run `bin/rails server --help` for more startup options
Puma starting in single mode...
* Puma version: 6.4.3 (ruby 3.1.2-p20) ("The Eagle of Durango")
* Min threads: 5
* Max threads: 5
* Environment: development
* PID: 3570
* Listening on http://127.0.0.1:3000
* Listening on http://[::]:3000
Use Ctrl-C to stop
```

- 1.4. En el navegador ir a la dirección localhost:3000, si todo sea realizado correctamente cargará el mensaje de bienvenida como el mostrado en la figura 114.



2. Instalación de la gema devise. Devise es una gema que sirve para que los usuario de un sitio web creen usuario, inicien sesión, cierren su respectiva sesión así como la posibilidad de recuperar contraseñas, modificar sus perfiles etc.

2.1. Para incluir devise, agregar la gema en el archivo Gemfile.

```
GNU nano 7.2 Gemfile *
source "https://rubygems.org"

ruby "3.1.2"

# Bundle edge Rails instead: gem "rails", github: "rails/rails", branch: "main"
gem "rails", "~> 7.1.4"
gem 'devise'
```

2.2. Desde el terminal, ubicado en la ruta del proyecto instalar la gema.

```
root@debian:/home/harold/SaaS_Proyecto1/Instagram# bundle install
Don't run Bundler as root. Bundler can ask for sudo if it is needed, and
installing your bundle as root will break this application for all non-root
users on this machine.
```

3. Crear estructura de usuarios con devise. En los siguientes pasos se instalara la gema como parte del proyecto y luego se creará el modelo que la maquetación de cómo se crearán los usuarios.

3.1. Instalar devise en el Instagram.

```
root@debian:/home/harold/SaaS_Proyecto1/Instagram# rails generate devise:install
  create  config/initializers/devise.rb
  create  config/locales/devise.en.yml
=====

Depending on your application's configuration some manual setup may be required:

1. Ensure you have defined default url options in your environments files. Here
   is an example of default_url_options appropriate for a development environment
   in config/environments/development.rb:
```

3.2. Generar el modelo para los usuarios.

```
root@debian:/home/harold/SaaS_Proyecto1/Instagram# rails generate devise User
  invoke  active_record
  create  db/migrate/20241027015826_devise_create_users.rb
  create  app/models/user.rb
  invoke  test_unit
  create  test/models/user_test.rb
  create  test/fixtures/users.yml
  insert  app/models/user.rb
  route  devise_for :users
```

3.3. Realizar migración para la base de datos. Migrar es una característica de Active Record que permite tener un esquema de base de dato a través del tiempo, es decir cada vez que se genera un cambio se realiza una migración lo cual genera un nuevo esquema con el nuevo cambio por lo que será fácil regresar a un esquema anterior si algo no funciona o no era lo que se deseaba.

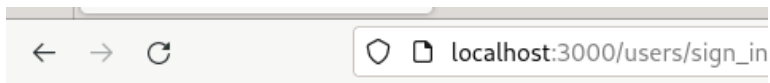
```
root@debian:/home/harold/Saas_Proyecto1/Instagram# rake db:migrate
== 20241027015826 DeviseCreateUsers: migrating =====
-- create_table(:users)
   -> 0.0025s
-- add_index(:users, :email, {:unique=>true})
   -> 0.0008s
-- add_index(:users, :reset_password_token, {:unique=>true})
   -> 0.0007s
== 20241027015826 DeviseCreateUsers: migrated (0.0046s) =====

root@debian:/home/harold/Saas_Proyecto1/Instagram# █
```

3.4. Correr el servidor para ver lo que se ha generado.

```
root@debian:/home/harold/Saas_Proyecto1/Instagram# rails s
=> Booting Puma
=> Rails 7.1.4 application starting in development
=> Run `bin/rails server --help` for more startup options
Puma starting in single mode...
* Puma version: 6.4.3 (ruby 3.1.2-p20) ("The Eagle of Durango")
* Min threads: 5
* Max threads: 5
* Environment: development
* PID: 3946
* Listening on http://127.0.0.1:3000
* Listening on http://[::]:3000
Use Ctrl-C to stop
```

3.5. En el navegador ir a la url localhost:3000/users/sign_in. En esta vista se muestra que se ha generado un formulario para iniciar sesión al igual que para las otras funcionalidades entre ellas la de crear usuarios.



Log in

Email

Password

☐ Remember me

Log in

[Sign up](#)

[Forgot your password?](#)

4. Poner estilo al Instagram a través la gema de bootstrap-sass.

4.1. Igual que la instalación de la gema devise, en el archivo Gemfile incluir la siguiente la gema bootstrap-sass' la cual permite poner estilo a la aplicación y usar todas bondades de bootstrap 3.

```
GNU nano 7.2                               Gemfile *
source "https://rubygems.org"

ruby "3.1.2"

# Bundle edge Rails instead: gem "rails", github: "rails/rails", branch: "main"
gem "rails", "~> 7.1.4"
gem 'devise'
gem 'bootstrap-sass', '~>3.3.5'
# The end of the file for Rails (https://github.com/rails/rails/blob/master/Gemfile)
```

4.2. Instalar las gemas desde el terminal.

Using web-console 4.2.1

Bundle complete! 17 Gemfile dependencies, 97 gems now installed.

Use `bundle info [gemname]` to see where a bundled gem is installed.

root@debian: /home/harold/SaaS_Proyecto1/Instagram#

4.3. En el archivo app/assets/javascripts/stylesheets/application.css agregar las siguientes líneas. Con lo cual se está configurando la aplicación para que exporte e interprete los estilos bootstrap y cambiar la extensión al archivo anterior a application.scss

```

application.scss X
Instagram > app > assets > stylesheets > application.scss
1  /*
2   * This is a manifest file that'll be compiled into application.css
3   * listed below.
4   * Any CSS (and SCSS, if configured) file within this directory,
5   * vendor/assets/stylesheets directory can be referenced here using a
6   * relative path.
7   * You're free to add application-wide styles to this file and they'll
8   * be compiled into this file. Styles in this file should be added
9   * after styles in vendor/assets/stylesheets files. Styles in this file should be added
10  * It is generally better to create a new file per style scope than
11  *
12  *= require_tree .
13  *= require_self
14  */
15  @import "bootstrap-sprockets";
16  @import "bootstrap";
17

```

4.4. En el app/assets/javascripts/application.js agregar lo siguiente. Con lo cual se está configurando la aplicación, para el uso de los JavaScript de bootstrap.

```

JS application.js X
Instagram > app > javascript > JS application.js
1  // Configure your import map in config/importmap.rb
2  import "@hotwired/turbo-rails"
3  import "controllers"
4  //require bootstrap-sprockets

```

5. Generando vistas, modelos y controladores.

5.1. Copiar el siguiente código en app/view/application.html.erb.

```

application.js  application.html.erb
gram > app > views > layouts > application.html.erb
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Instagram</title>
  <%= stylesheet_link_tag "application", media: "all", "data-turbolinks-track" => true %>
  <%= javascript_include_tag "application", "data-turbolinks-track" => true %>
  <%= csrf_meta_tags %>
</head>
<body>
  <!-- Navigation -->
  <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
    <div class="container">
      <!-- Brand and toggle get grouped for better mobile display -->
      <div class="navbar-header">
        <button type="button" class="navbar-toggle" data-toggle="collapse"
        target="#bs-example-navbar-collapse-1">
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
      </div>
      <!-- Collect the nav links, forms, and other content for toggling -->
      <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
        <ul class="nav navbar-nav">
          <li><a href="#">About</a></li>
          <li><a href="#">Services</a></li>
          <li><a href="#">Contact</a></li>
        </ul>
      </div>
    </div>
  </nav>
  <!-- /.navbar-collapse -->
</body>
</html>

```

5.2. Generar el modelo Post. El cual tendrá como único campo description que será de tipo text.

```

root@debian:/home/harold/SaaS_Proyecto1/Instagram# rails generate model Post description:text
  invoke  active_record
  create  db/migrate/20241027235203_create_posts.rb
  create  app/models/post.rb
  invoke  test_unit
  create  test/models/post_test.rb
  create  test/fixtures/posts.yml
root@debian:/home/harold/SaaS_Proyecto1/Instagram#

```

5.3. Una vez teniendo el modelo generar el controlador de nombre Posts con tres vistas las cuales serán new para crear nuevos post, index para el inicio y show para ver un post en particular.

```

root@debian:/home/harold/Saas_Proyecto1/Instagram# rails generate controller Posts new index show
   create  app/controllers/posts_controller.rb
   route   get 'posts/new'
          get 'posts/index'
          get 'posts/show'
   invoke  erb
   create  app/views/posts
   create  app/views/posts/new.html.erb
   create  app/views/posts/index.html.erb
   create  app/views/posts/show.html.erb
   invoke  test_unit
   create  test/controllers/posts_controller_test.rb
   invoke  helper
   create  app/helpers/posts_helper.rb
   invoke  test_unit
root@debian:/home/harold/Saas_Proyecto1/Instagram#

```

5.4. Cambiar las rutas de la aplicación en el archivo `app/config/routes.rb`, de tal manera que el index que se acaba de generar sea la primera vista al cargar la aplicación.

```

get "up" => "rails/health#show", as: :rails_healthcheck
Rails.application.routes.draw do
  root 'posts#index'
  resources :posts
  devise_for :users
end

```

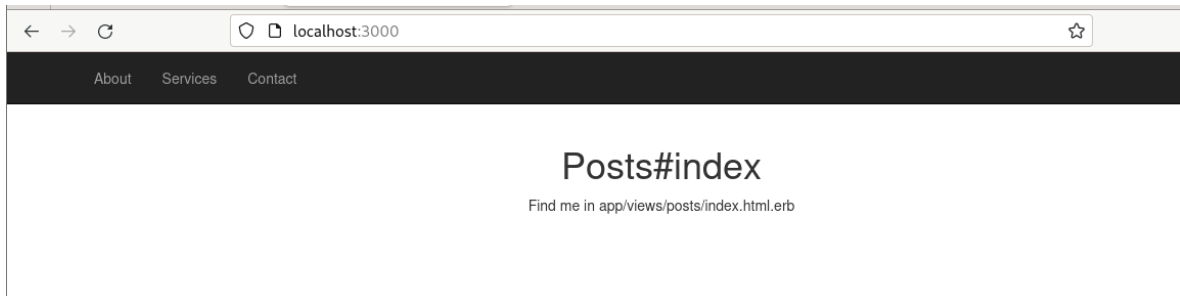
5.5. Iniciar el servidor.

```

root@debian:/home/harold/Saas_Proyecto1/Instagram# rails s
=> Booting Puma
=> Rails 7.1.4 application starting in development
=> Run `bin/rails server --help` for more startup options
Puma starting in single mode...
* Puma version: 6.4.3 (ruby 3.1.2-p20) ("The Eagle of Durango")
* Min threads: 5
* Max threads: 5
* Environment: development
* PID: 4079
* Listening on http://127.0.0.1:3000
* Listening on http://[::]:3000
Use Ctrl-C to stop

```

5.6. En el navegador dirigirse a la dirección localhost:3000. Cargará la vista index con la apariencia que se ejemplifica en la figura 3 ese diseño que parece se está cargando desde la vista app/view/application.html.erb. que se agregó en el paso 5.1 y carga el contenido de index debido a que redendiriza las otras vistas bajo ésta



6. Configuración de link_to en la vista.

6.1. Consultar las rutas. Para ello usar el comando routes que permite visualizar todas las urls y el acceso a las diferentes acciones de los controladores

```
root@debian:/home/harold/Saas_Proyecto1/Instagram# rails routes
```

6.2. Se debe obtener una lista similar a la siguiente figura.

```
root@debian:/home/harold/Saas_Proyecto1/Instagram# rails routes
           Prefix Verb   URI Pattern
           Controller#Action
root      GET    /
posts     GET    /posts(.:format)
           posts#index
           POST   /posts(.:format)
           posts#create
new_post  GET    /posts/new(.:format)
           posts#new
edit_post GET    /posts/:id/edit(.:format)
           posts#edit
post      GET    /posts/:id(.:format)
           posts#show
           PATCH  /posts/:id(.:format)
           posts#update
           PUT    /posts/:id(.:format)
           posts#update
           DELETE /posts/:id(.:format)
           posts#destroy
new_user_session GET    /users/sign_in(.:format)
           devise/sessions#new
user_session POST   /users/sign_in(.:format)
```

7. Modificará el archivo `app/view/application.html.erb` y modificar el código de por el siguiente. Lo que se logra es configurar los `link_to` de acuerdo a las acción que va a realizar cada link para ello se está usando las url obtenidas en el paso anterior.

```
views > layouts > <> application.html.erb
<!DOCTYPE html>
<head>
  <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
    <div class="container">
      <div class="collapse navbar-collapse">
        <!-- Collect the nav links, forms, and other content for toggling -->
        <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
          <ul class="nav navbar-nav">
            <li><%= link_to "New Post", new_post_path %></li>
            <%- if current_user %>
            <li>
              <%= link_to "Logout", destroy_user_session_path, method: :delete %>
            </li>
            <% else %>
            <li><%= link_to "Login", new_user_session_path %></li>
            <li><%= link_to "Register", new_user_registration_path %></li>
            <% end%>
          </ul>
        </div>
      </div>
    </div>
  </nav>
</head>
<body>
  <!-- /.navbar-collapse -->
</body>
</html>
```

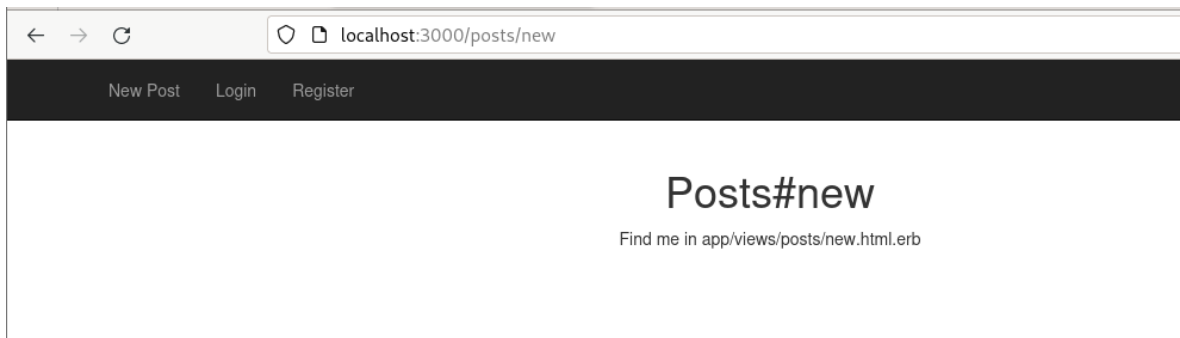
7.1. Verificación de las vistas, ver cada una de las vistas para comprobar que se estén cargando correctamente las acciones configuradas. En la vista `index` es donde se mostrarán todos los post publicados. Para acceder solo ir en el navegador a `localhost:3000`.



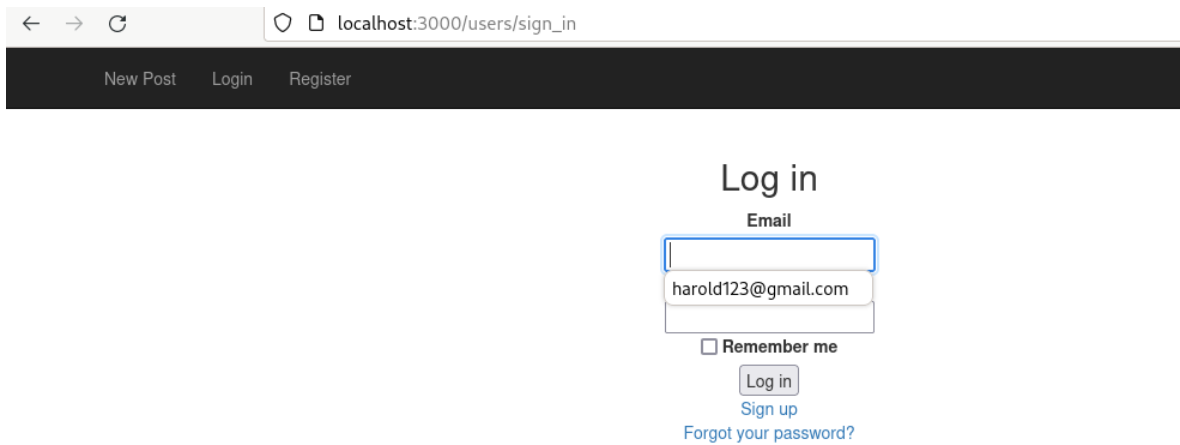
Posts#index

Find me in app/views/posts/index.html.erb

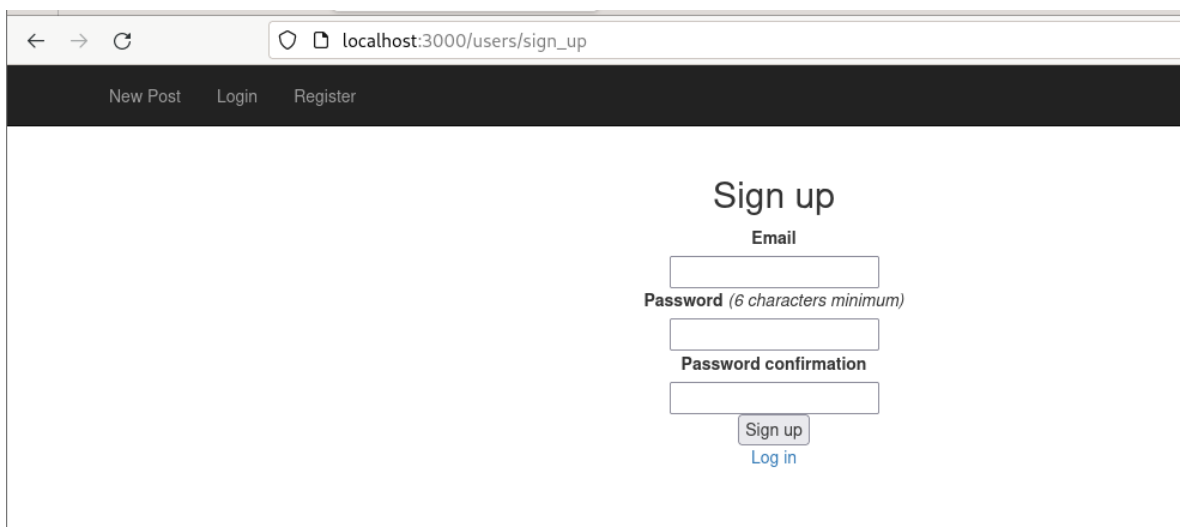
7.2. Ir a la dirección `localhost:300/posts/new`, en la vista `New`, es donde se crearán los post para ser publicados.



7.3. Ir a la dirección `localhost:3000/users/sign_in`, esta vista Login es la que permitirá a los usuarios registrados iniciar sesión en la aplicación.



7.4. Ir a la dirección `localhost:3000/users/sign_up`, en la vista Register es donde se dan de alta a los nuevos usuarios.



8. Configuración de la vista New Post.

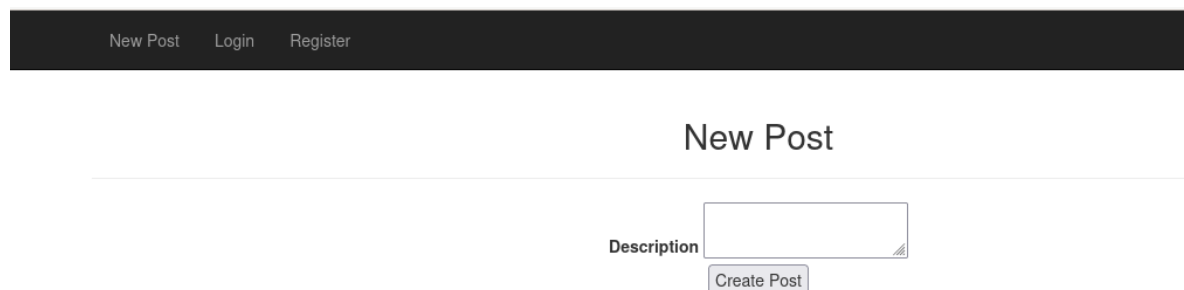
8.1. Editar el archivo `app/views/posts/new.html.erb`. En el siguiente código se crea el formulario a este punto solo con el campo descripción.

```
app > views > posts > <> new.html.erb
1  <h2>New Post</h2>
2  <hr>
3  <%= form_for @post do |f| %>
4    <%= f.label :description %>
5    <%= f.text_area :description %>
6    <br>
7    <%= f.submit %>
8  <%end%>
```

8.2. Configurar el controlador de los Post, en el archivo `posts_controller.rb`, del directorio `app/controllers/`, agregar el método `new` al controlador con el siguiente código.

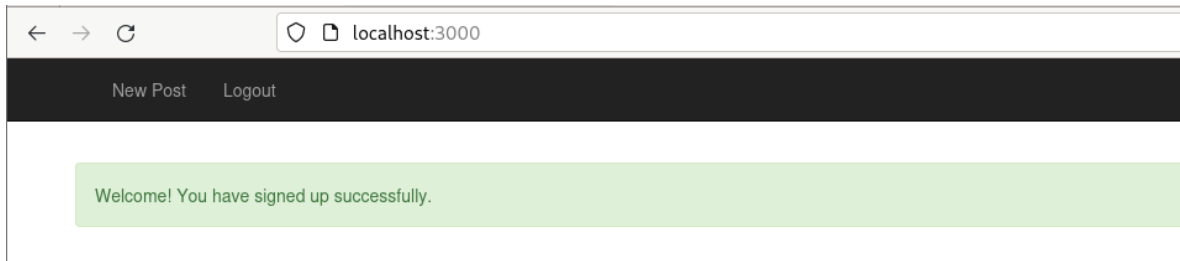
```
app > controllers > posts_controller.rb
1  class PostsController < ApplicationController
2    def new
3      @post = Post.new
4    end
5  end
```

8.3. Ir a la dirección `localhost:3000/posts/new` y ver que ya se ha creado un formulario para agregar un nuevo post como se muestra en la siguiente figura.



The screenshot shows a web browser window with a dark header bar containing links for "New Post", "Login", and "Register". The main content area has a heading "New Post" followed by a horizontal line. Below the line, there is a form with a label "Description" next to a text area input field. At the bottom right of the form is a button labeled "Create Post".

9. Mostrar notificación de actividades cuando un usuario se registra.



10. En el archivo `app/view/application.html.erb`, en el `div` que tiene la clase “row” agregar el siguiente código. Que lo que hace es mostrar alertas tipo notificaciones como el de la Figura 123.

```
<div class="container">
  <div class="row">
    <% flash.each do |a,b| %>
      <div class="alert alert-success"><%= b%> </div>
    <%end%>
    <div class="col-lg-12 text-center">
      <%= yield %>
    </div>
  </div>
<!-- /.row -->
</div>
```

11. Instalación de las gemas `paperclip` y `aws-sdk`.

11.1. Agregar en la gema al proyecto en el archivo `Gemfile`.

```
root@debian:/home/harold/1_Proyecto/Instagram# rails active_storage:install
Copied migration 20241031013831_create_active_storage_tables.active_storage.rb f
rom active_storage
root@debian:/home/harold/1_Proyecto/Instagram#
```

11.2. Instalar las gemas.

```

GNU nano 7.2                                app/models/post.rb *
class Post < ApplicationRecord
  has_one_attached :image
  validates :image, presence: true
  validates :description, presence: true, length: { minimum: 10}
end

```

12. Crear la estructura de paperclip que permitirá manejar todo lo referente a la publicación de imágenes

```

<h2>New Post</h2>
<hr>
<%= form_for @post, html: { multipart: true } do |f| %>
  <%= f.label :image %>
  <%= f.file_field :image %>
  <br>
  <%= f.label :description %>
  <%= f.text_area :description %>
  <br>
  <%= f.submit %>
<% end %>

```

13. Al momento de desarrollar esta guía, se ha encontrado una incompatibilidad de paperclip con Rails 5.1 al intentar hacer migración, para solucionarlo hay que especificar con que versión de Rails se está trabajando, para ello:

13.1. En el directorio /db/migrate/, buscar el último fichero de migración creado, agregar al final la versión, como se muestra a continuación la versión en color rojo.

```

GNU nano 7.2                                app/views/posts/show.html.erb *
<h1>Posts#show</h1>
<p>Find me in app/views/posts/show.html.erb</p>
<%= image_tag(@post.image) if @post.image.attached? %>
<br>
<%= @post.description %>

```

13.2. Migrar los datos.

14. Configurar modelo, vista y controlador paperclip.

14.1. Configurar el modelo `app/models/post.rb`, en este archivo copiar el código a continuación, en el cual se establecen las dimensiones de las imágenes, una ruta por omisión y valida que el contenido a subir sea de tipo imagen.

```
app > models > post.rb
1  class Post < ApplicationRecord
2    has_one_attached :image
3    validates :image, presence: true
4    validates :description, presence: true, length: { minimum: 10}
5  end
6
```

15. Modificar el archivo `app/views/posts/new.html.erb`. Crear el formulario que se mostrará, para publicar los post.

```
app > views > posts > new.html.erb
1  <h2>New Post</h2>
2    <hr>
3    <%= form_for @post, html: { multipart: true } do |f| %>
4      <%= f.label :image %>
5      <%= f.file_field :image %>
6      <br>
7      <%= f.label :description %>
8      <%= f.text_area :description %>
9      <br>
10     <%= f.submit %>
11   <% end %>
12
```

15.1. Editar el controlador en `app/controllers/posts_controller.rb` y agregar los métodos a utilizar.

```

def new
  @post= Post.new
end

def index
  @post= Post.all
end

def show
  @post= Post.find(params[:id])
end

def create
  @post= Post.new(post_params)
  if @post.save
    flash[:success]= "Post creado con exito"
    redirect_to post_path(@post)
  else
    flash[:error]= @post.errors.full_messages
    render :new
  end
end

private
def post_params
  params.require(:post).permit(:image, :description)
end
end

```

16. Configuración de la vista.

16.1. Agregar el siguiente código en app/views/posts/show.html.erb. El cual permite que se visualice un pos en particular.

```

app > views > posts > <> show.html.erb
1   <h1>Posts#show</h1>
2   <% if @post.image.attached? %>
3   |   <%= image_tag @post.image %>
4   <% end %>
5   <br>
6   <%= @post.description %>
7

```


17. Para visualizar todos los post, editar el archivo `app/views/posts/index.html.erb` y agregar el siguiente código ERB. Con lo cual se mostrarán todos los post generados.

```
app > views > posts > <> index.html.erb
1  <h1>Posts#index</h1>
2  <% @Posts.each do |post| %>
3    <div>
4      <% if post.image.attached? %>
5        <%= image_tag post.image.variant(resize_to_limit: [300, 300]) %>
6      <% end %>
7      <p><%=post.description %></p>
8    </div>
9  <% end %>
10
```

18. Crear el método `destroy` dentro del archivo `posts_controller.rb` del directorio `/app/controllers/`, agregar el siguiente código debajo del método `create`.

```
end

def destroy
  @post= Post.find(params[:id])
  if @post.destroy
    redirect_to post_path
    flash[:success]= "The post is destroy!"
  else
    redirect_to post_path
  end
end
```

19. Funcionamiento.

19.1.Crear un nuevo post. Al momento de crear un nuevo Post, el formulario permite elegir una imagen almacenada y una descripción de ésta, para ser posteadada.

New Post Login Register

New Post

Examinar... yor.jpg

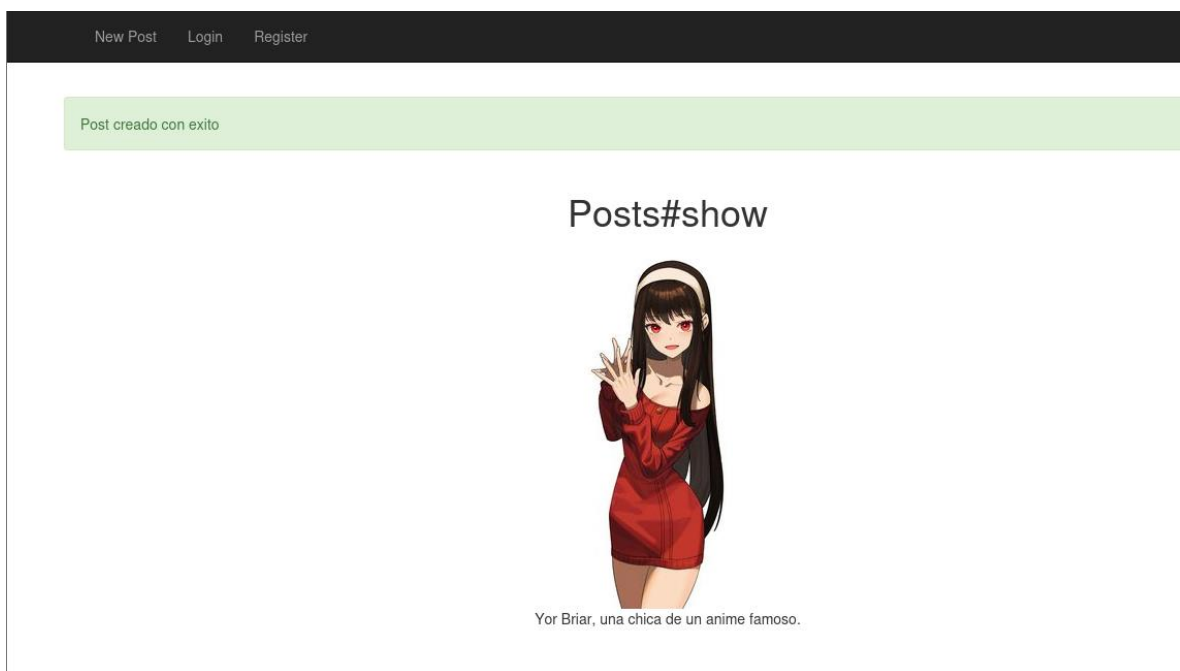
Image

Description

Yor Briar, una chica de un anime famoso.

Create Post

19.2.Un post ya publicado, en la vista show.



20. Validaciones. A este punto la aplicación es muy funcional, pero hay acciones que no se han tomado en cuenta, como es no permitir la creación de post sin contenido, que los usuarios estén logueados para publicar un post, poner un mínimo de caracteres en la descripción, entre otras, para ello Ruby on Rails permite realizar un sin número de validaciones para los modelos que permiten o deniegan dichas acciones.

20.1. Post solo visibles cuando el usuario haya iniciado sesión. Editar /app/controllers/posts_controller.rb (Poner justo después de la declaración de la clase).

```

class PostsController < ApplicationController
  before_action :authenticate_user!
  def new

```

20.2. Resultado de la primera validación se puede observar en la siguiente figura.

You need to sign in or sign up before continuing.

Log in

Email

harold123@gmail.com

☐ Remember me

[Sign up](#)

[Forgot your password?](#)

20.3. No permitir publicar post vacíos, ni con descripción menor a diez caracteres. Para ello editar /app/models/post.rb y agregar el

```

validates :image, presence: true
validates :description, presence: true, length: { minimum: 10}
end

```

Ejercicios propuestos para ser entregados al docente

2. Aplicar css a las vistas según su criterio.

New Post

Logout



Yor Briar, una chica de un anime famoso.

Eliminar

Actualizar



Yor Briar, una chica de un anime famoso.

Eliminar

Actualizar



un tipo del anime tokyo ghoul!!!

Eliminar

Actualizar

3. Realizar validación, que solo el usuario dueño del post pueda borrarlo.
4. Realizar la funcionalidad Edit para ello se debe crear la vista y los controladores necesarios como se realizó con las otras funcionalidades.

Examinar... No se ha seleccionado ningún archivo.

Image

Description

ghoul, este post fue editado!!

Actualizar Post



Yor Briar, una chica de un anime famoso.

Eliminar

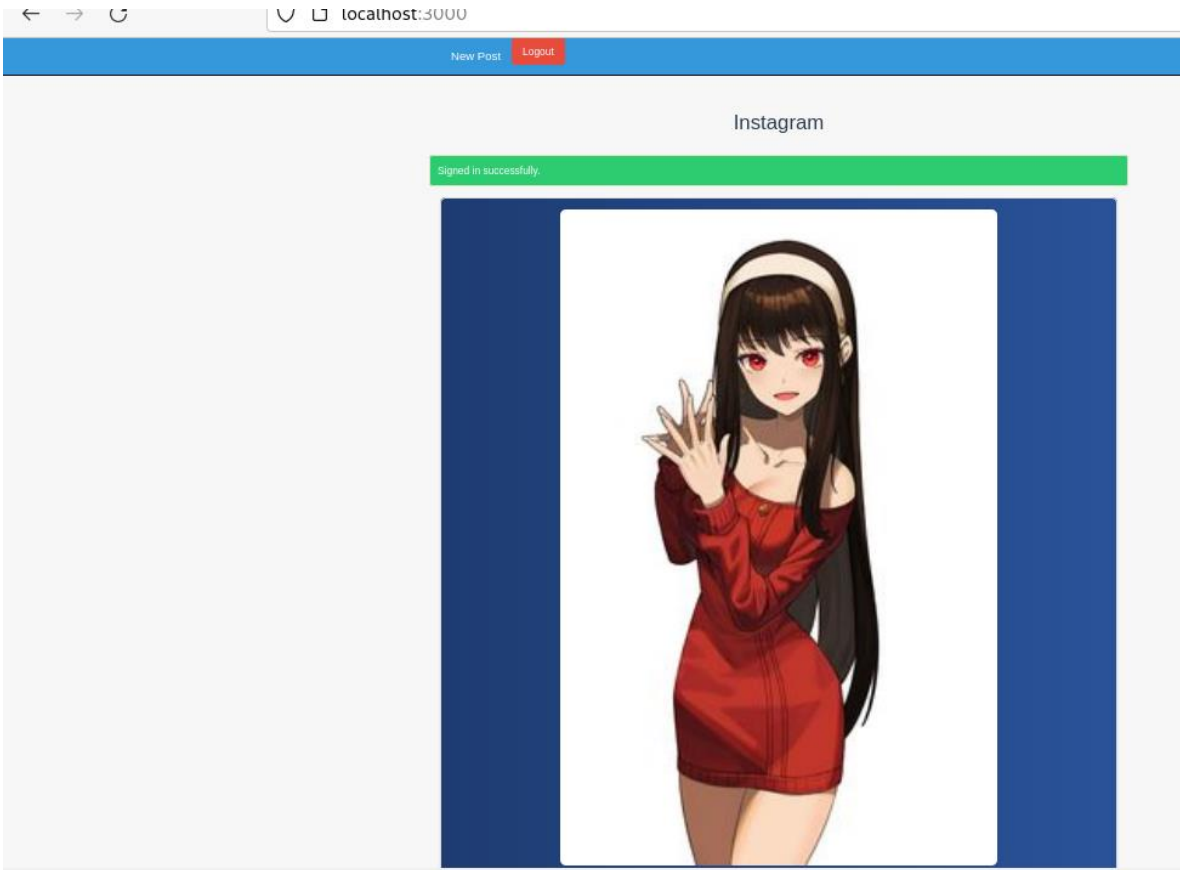
Actualizar



un tipo del anime tokyo ghoul, este post fue editado!!

Eliminar

Actualizar





Yor Briar es una chica de anime muy famosa!!

Eliminar

Actualizar

