

# **Universidad Nacional Autónoma de Nicaragua Unan León**



**Facultad de Ciencias y Tecnología.**

**Componente: Software como un servicio**

**Tema: Practica5**

Grupo:1

**Docente: Erving Montes**

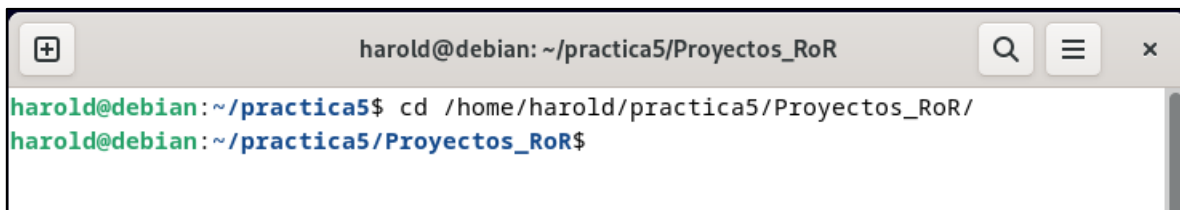
**Integrante:**

-Harold Steven Garcia Ramos.

**Fecha:** 4 de septiembre de 2024

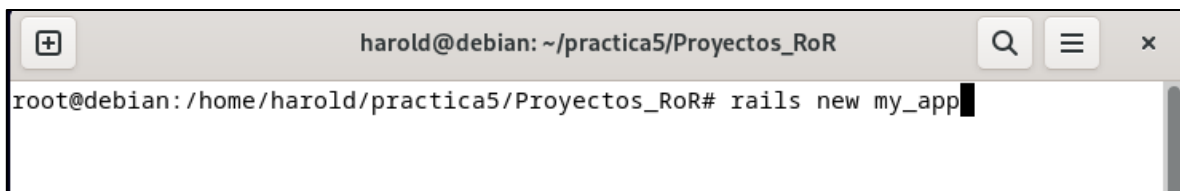
1. Creación de un proyecto nuevo.

1.1. Ubicarse en el directorio donde se va almacenar el proyecto.

A terminal window titled 'harold@debian: ~/practica5/Proyectos\_RoR' with search, menu, and close icons. The prompt is 'harold@debian:~/practica5\$'. The first command is 'cd /home/harold/practica5/Proyectos\_RoR/' and the second is 'cd /home/harold/practica5/Proyectos\_RoR/'.

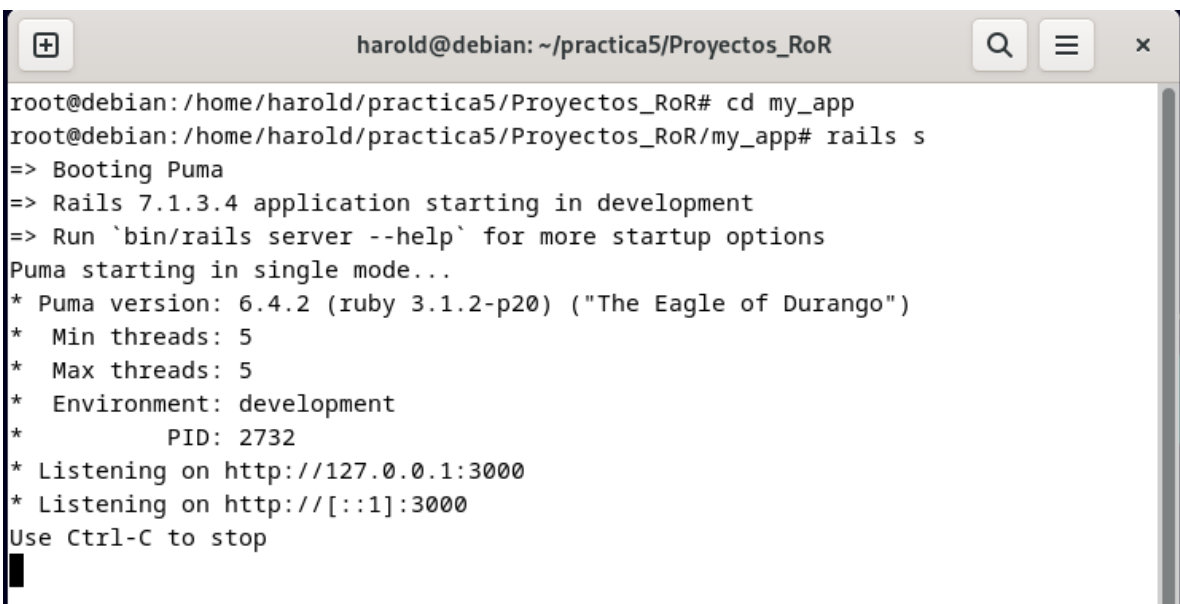
```
harold@debian:~/practica5$ cd /home/harold/practica5/Proyectos_RoR/
harold@debian:~/practica5/Proyectos_RoR$
```

1.2. Generar un nuevo proyecto.

A terminal window titled 'harold@debian: ~/practica5/Proyectos\_RoR' with search, menu, and close icons. The prompt is 'root@debian: /home/harold/practica5/Proyectos\_RoR#'. The command 'rails new my\_app' is entered.

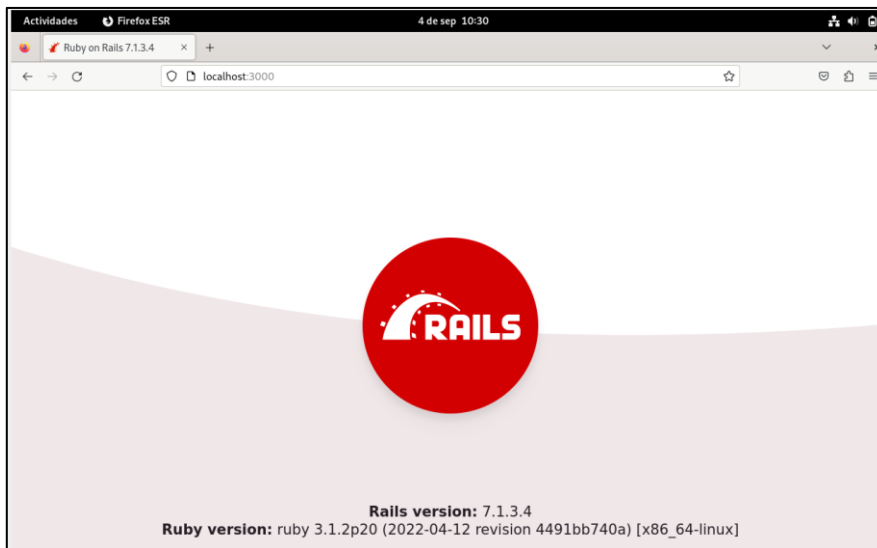
```
root@debian: /home/harold/practica5/Proyectos_RoR# rails new my_app
```

1.3. Ubicarse dentro del proyecto e iniciar el servidor para comprobar que funciona sin ningún problema.

A terminal window titled 'harold@debian: ~/practica5/Proyectos\_RoR' with search, menu, and close icons. The prompt is 'root@debian: /home/harold/practica5/Proyectos\_RoR#'. The first command is 'cd my\_app' and the second is 'rails s'. The output shows Puma starting in single mode and listening on http://127.0.0.1:3000 and http://[::]:3000.

```
root@debian: /home/harold/practica5/Proyectos_RoR# cd my_app
root@debian: /home/harold/practica5/Proyectos_RoR/my_app# rails s
=> Booting Puma
=> Rails 7.1.3.4 application starting in development
=> Run `bin/rails server --help` for more startup options
Puma starting in single mode...
* Puma version: 6.4.2 (ruby 3.1.2-p20) ("The Eagle of Durango")
* Min threads: 5
* Max threads: 5
* Environment: development
* PID: 2732
* Listening on http://127.0.0.1:3000
* Listening on http://[::]:3000
Use Ctrl-C to stop
```

1.4. Abrir el navegador y escribir la siguiente dirección.



1. Generar un nuevo controlador al proyecto.

1.1. Verificar cuál es el controlador existente dentro del proyecto. De la siguiente forma se obtendrá el nombre de los controladores que existen hasta el momento.

```
harold@debian: ~/practica5/Proyectos_RoR
root@debian: /home/harold/practica5/Proyectos_RoR/my_app# ls app/controllers/*_controller.rb
app/controllers/application_controller.rb
root@debian: /home/harold/practica5/Proyectos_RoR/my_app#
```

1.2. Generar el nuevo controlador y una acción correspondiente a ese controlador.

```
harold@debian: ~/practica5/Proyectos_RoR
root@debian: /home/harold/practica5/Proyectos_RoR/my_app# rails generate controller welcome index
  create  app/controllers/welcome_controller.rb
  route  get 'welcome/index'
  invoke  erb
  create  app/views/welcome
  create  app/views/welcome/index.html.erb
  invoke  test_unit
  create  test/controllers/welcome_controller_test.rb
  invoke  helper
  create  app/helpers/welcome_helper.rb
  invoke  test_unit
root@debian: /home/harold/practica5/Proyectos_RoR/my_app#
```

1.3. Verificar nuevamente los controladores del proyecto. Podrá observar que ahora ya existe uno nuevo, y que se corresponde con el creado anteriormente.

```
harold@debian: ~/practica5/Proyectos_RoR
root@debian: /home/harold/practica5/Proyectos_RoR/my_app# ls app/controllers/*_controller.rb
app/controllers/application_controller.rb  app/controllers/welcome_controller.rb
root@debian: /home/harold/practica5/Proyectos_RoR/my_app#
```

1.4. Abrir el archivo `welcome_controller.rb` que se encuentra en el directorio (`app/controllers/`) y verificar que existe el método `index`, que fue creado al momento de generar el controlador, este es el método (acción) que tiene relación con la vista `index.html.erb`.

```
harold@debian: ~/practica5/Proyectos_RoR
GNU nano 7.2 app/controllers/welcome_controller.rb
class WelcomeController < ApplicationController
  def index
  end
end
```

## 2. Trabajar con la vista

Para comenzar a trabajar con las vistas se debe ubicar el directorio (`app/view/`), que es donde se almacenan las vistas del proyecto, ahí se observa un archivo llamado `index.html.erb`, este es por medio del cual los usuarios finales interactúan con la aplicación

2.1. Editar el archivo `index.html.erb` del nuevo controlador y escribir el siguiente código html.

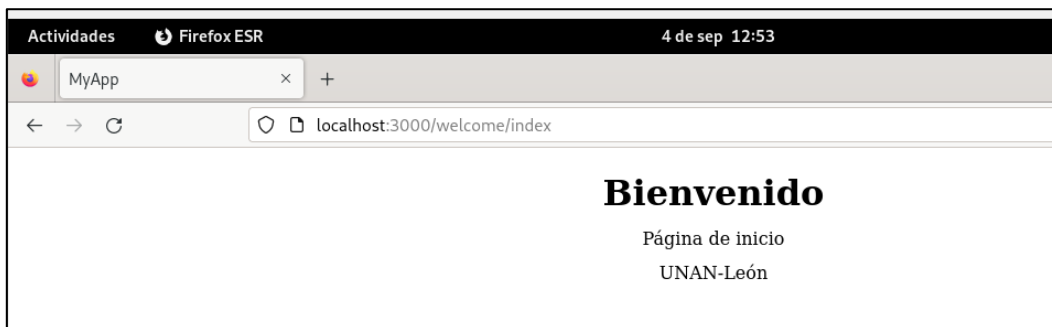
```
harold@debian: ~/practica5/Proyectos_RoR
GNU nano 7.2 index.html.erb *
<h1>Welcome#index</h1>
<p>Find me in app/views/welcome/index.html.erb</p>
<h1>Bienvenido</h1>
<p>Página de inicio</p>
<p>UNAN-León</p>
```

2.2. Editar el archivo `welcome.scss`, que se encuentra en el directorio (`app/assets/stylesheets/`), en este directorio es donde se almacenan todas las hojas de estilos que contiene el proyecto. Todos los archivos de estilos en rails terminan en `“.scss”` debido a que el framework permite utilizar el lenguaje de estilos llamado Sass (Syntactically Awesome StyleSheets), es una extensión de css que agrega potencia y elegancia al lenguaje básico. Para darle un poco de formato a las vistas, agregar el siguiente código.

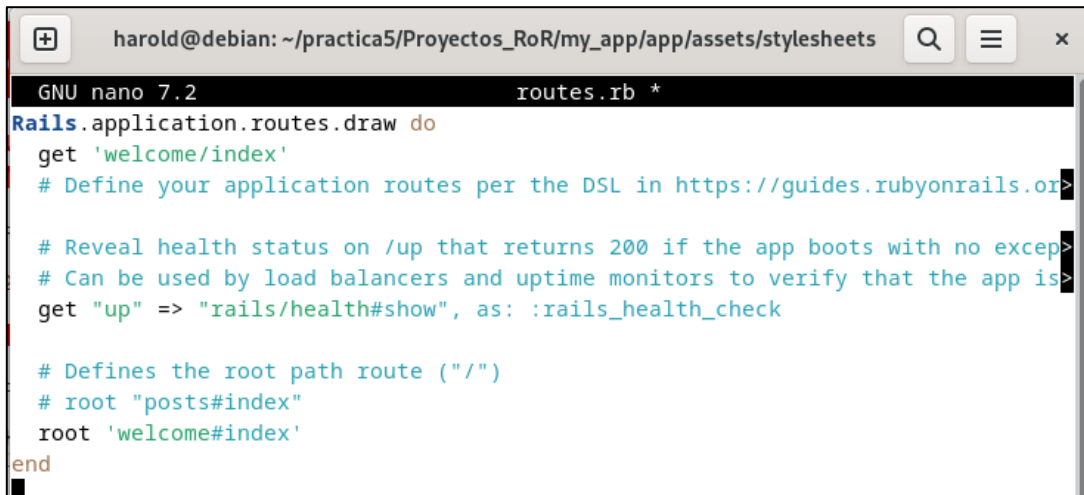


```
harold@debian: ~/practica5/Proyectos_RoR/my_app/app/assets/stylesheets
GNU nano 7.2 application.css
*
* You're free to add application-wide styles to this file and they'll appear a>
* compiled file so the styles you add here take precedence over styles defined>
* files in this directory. Styles in this file should be added after the last >
* It is generally better to create a new file per style scope.
*
*= require_tree .
*= require_self
*/
h1
{
  color: black;
  height: 30px;
  text-align: center;
}
p
{
  height: 15px;
  text-align: center;
}
[ 26 líneas escritas ]
^G Ayuda  ^O Guardar  ^W Buscar  ^K Cortar  ^T Ejecutar  ^C Ubicación
^X Salir   ^R Leer fich. ^\ Reemplazar ^U Pegar    ^J Justificar ^_ Ir a línea
```

2.3. Ir a la dirección (<http://localhost:3000/welcome/index>).



3. Si navega directamente en <http://localhost:3000/> no se tendrá acceso a la vista antes modificada, para esto, se debe modificar el archivo routes.rb, de manera que inicie (root) con la vista de nombre index.html.erb.

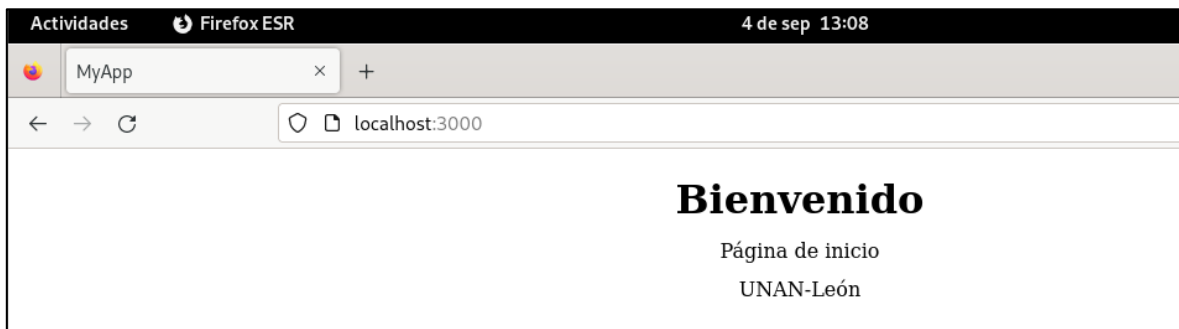


```
harold@debian: ~/practica5/Proyectos_RoR/my_app/app/assets/stylesheets
GNU nano 7.2 routes.rb *
Rails.application.routes.draw do
  get 'welcome/index'
  # Define your application routes per the DSL in https://guides.rubyonrails.org

  # Reveal health status on /up that returns 200 if the app boots with no excep>
  # Can be used by load balancers and uptime monitors to verify that the app is>
  get "up" => "rails/health#show", as: :rails_health_check

  # Defines the root path route ("/")
  # root "posts#index"
  root 'welcome#index'
end
```

4. Actualizar el navegador en la dirección <http://localhost:3000/>



5. Uso del lenguaje Ruby en el framework Rails.

5.1. Para ver el funcionamiento ir al archivo welcome\_controller.rb y crear un array global con los días de la semana dentro del método index.



```
harold@debian: ~/practica5/Proyectos_RoR/my_app/app/assets/stylesheets
GNU nano 7.2 app/controllers/welcome_controller.rb *
class WelcomeController < ApplicationController
  def index
    @dias_semana= ["Lunes", "Martes", "Miercoles", "Jueves", "Viernes", "Sabado", "D>
  end
end
```

5.2. Para poder hacer uso del array dentro de la vista, se debe utilizar lenguaje ERB en el archivo de la vista index.html.erb; editar el archivo y escribir el siguiente.



```
harold@debian: ~/practica5/Proyectos_RoR/my_app/app/assets/stylesheets
GNU nano 7.2 index.html.erb *
<h1>Bienvenido</h1>
<p>Página de inicio</p>
<p>UNAN-León</p>
<h3>Imprimiendo un array Ruby dentro de una vista</h3>
  <% @dias_semana.each do |dia| %>
    <%=dia %><br>
  <% end %>
```

5.3. De igual forma se puede hacer uso de los distintos métodos propios de los array, en el caso anterior se utilizó el método each para recorrer el array. Escribir el siguiente código en el archivo index.html.erb.



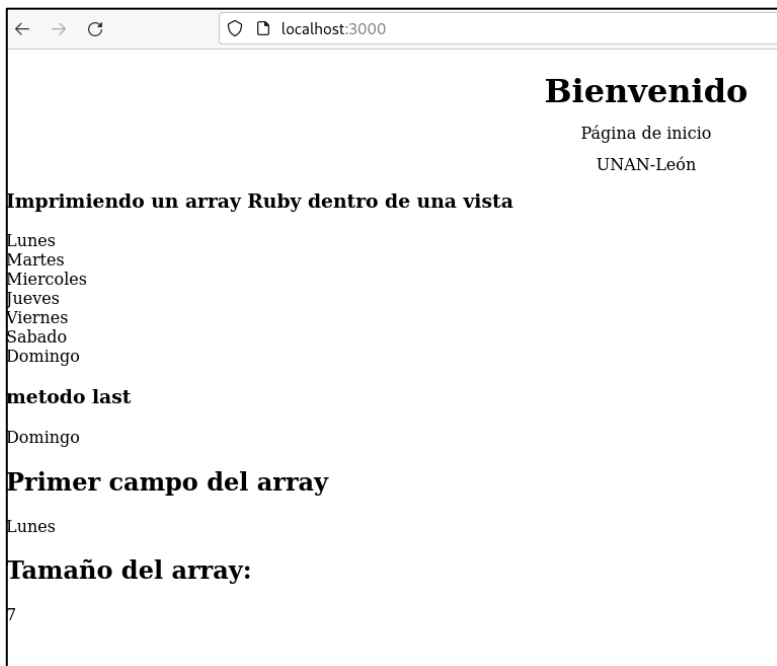
```
harold@debian: ~/practica5/Proyectos_RoR/my_app/app/assets/stylesheets
GNU nano 7.2 index.html.erb *
<h1>Bienvenido</h1>
<p>Página de inicio</p>
<p>UNAN-León</p>
<h3>Imprimiendo un array Ruby dentro de una vista</h3>
  <% @dias_semana.each do |dia| %>
    <%=dia %><br>
  <% end %>
<h3>metodo last</h3>
<%= @dias_semana.last %>
```



Ejercicios propuestos.

2. Muestre en la vista index.html.erb el primer dato almacenado y el tamaño completo del array, haga uso de métodos similares a los vistos anteriormente.

```
harold@debian: ~/practica5/Proyectos_RoR/my_app/app/assets/sty
GNU nano 7.2 index.html.erb
<h1>Bienvenido</h1>
<p>Página de inicio</p>
<p>UNAN-León</p>
<h3>Imprimiendo un array Ruby dentro de una vista</h3>
  <% @dias_semana.each do |dia| %>
    <%= dia %><br>
  <% end %>
<h3>metodo last</h3>
<%= @dias_semana.last %>
<h2>Primer campo del array</h2>
<%= @dias_semana.first %>
<h2>Tamaño del array:</h2>
<%= @dias_semana.size %>
```





3. Generar un nuevo controlador de nombre “hash” y una acción que es la que tendrá relación con la vista llamada “mostrar”, similar al realizado al inicio de esta guía.

```
harold@debian: ~/practica5/Proyectos_RoR/my_app/app/assets/stylesheets
root@debian: /home/harold/practica5/Proyectos_RoR/my_app# rails generate controller Hash mostrar
create  app/controllers/hash_controller.rb
route  get 'hash/mostrar'
invoke erb
create  app/views/hash
create  app/views/hash/mostrar.html.erb
invoke test_unit
create  test/controllers/hash_controller_test.rb
invoke helper
create  app/helpers/hash_helper.rb
invoke test_unit
root@debian: /home/harold/practica5/Proyectos_RoR/my_app#
```

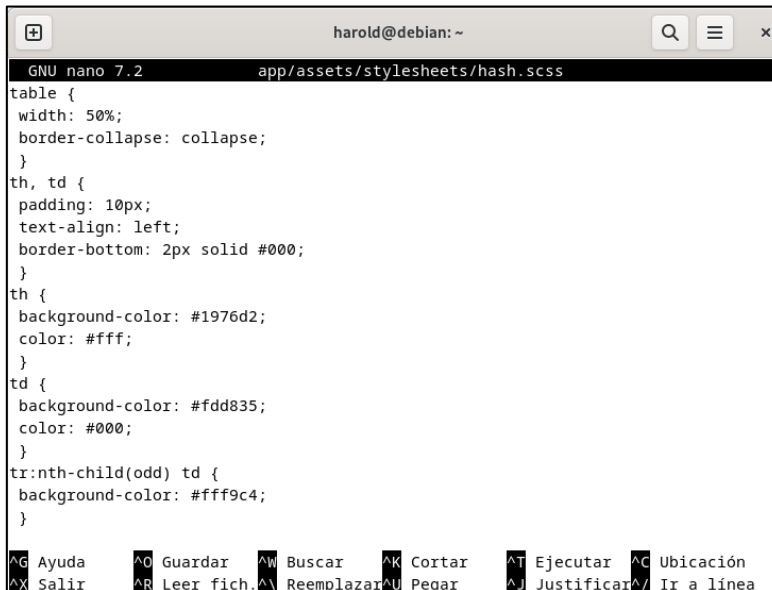
4. Dentro de la acción mostrar que se creó junto al nuevo controlador del proyecto, crear un hash con los datos de una persona: nombre, apellido, teléfono, correo.

```
harold@debian: ~/practica5/Proyectos_RoR/my_app/app/assets/stylesheets
GNU nano 7.2 app/controllers/hash_controller.rb *
class HashController < ApplicationController
  def mostrar
    @persona= {
      nombre: "Harold",
      apellido: "Garcia",
      telefono: "45976315",
      correo: "hola123@gmail.com"
    }
  end
end
```

5. Modificar la vista del controlador creado anteriormente y mostrar los datos del hash en una tabla html.

```
harold@debian: ~
GNU nano 7.2 app/views/hash/mostrar.html.erb
<table>
<tr>
  <th>Nombre</th>
  <td><%= @persona[:nombre] %></td>
</tr>
<tr>
  <th>Apellido</th>
  <td><%= @persona[:apellido] %></td>
</tr>
<tr>
  <th>Telefono</th>
  <td><%= @persona[:telefono] %></td>
</tr>
<tr>
  <th>Correo</th>
  <td><%= @persona[:correo] %></td>
</tr>
</table>
```

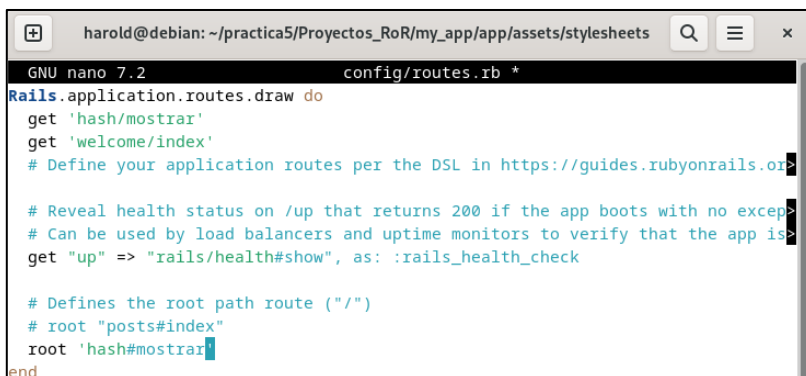
6. Agregar código css a la tabla, se mostrará en la vista como en la figura que se puede ver que la tabla contiene formato, colores, bordes.



A terminal window titled 'harold@debian: ~' showing the GNU nano 7.2 editor editing 'app/assets/stylesheets/hash.scss'. The code defines styles for a table with a width of 50%, collapsed borders, and specific background colors for header, body, and alternating rows. A bottom toolbar contains icons and shortcuts for various editor actions.

```
GNU nano 7.2 app/assets/stylesheets/hash.scss
table {
  width: 50%;
  border-collapse: collapse;
}
th, td {
  padding: 10px;
  text-align: left;
  border-bottom: 2px solid #000;
}
th {
  background-color: #1976d2;
  color: #fff;
}
td {
  background-color: #fdd835;
  color: #000;
}
tr:nth-child(odd) td {
  background-color: #fff9c4;
}
```

7. Configurar el archivo routes.rb para cambiar la página de inicio del proyecto y ahora hacerlo apuntar al nuevo controlador junto con su respectiva vista, la cual deberá quedar como se observa en la figura



A terminal window titled 'harold@debian: ~/practica5/Proyectos\_RoR/my\_app/app/assets/stylesheets' showing the GNU nano 7.2 editor editing 'config/routes.rb'. The code configures Rails routes, including 'hash/mostrar', 'welcome/index', a health check route, and the root path pointing to 'hash/mostrar'.

```
GNU nano 7.2 config/routes.rb *
Rails.application.routes.draw do
  get 'hash/mostrar'
  get 'welcome/index'
  # Define your application routes per the DSL in https://guides.rubyonrails.org/

  # Reveal health status on /up that returns 200 if the app boots with no excep>
  # Can be used by load balancers and uptime monitors to verify that the app is>
  get "up" => "rails/health#show", as: :rails_health_check

  # Defines the root path route ("/")
  # root "posts#index"
  root 'hash#mostrar'
end
```

Resultado

