***UNIVERCIDAD NACIONAL AUTONOMA DE*** ***NICARAGUA UNAN-LEÒN***



***FACULTAD DE CIENCIAS Y TECNOLOGIA***

***Componente: Software como un Servicio***

***Docente: Ervin Montes***

***Grupo: #1***

***Integrantes:***

***Ingrid Valeria Ruiz Ulloa 21-00483-0***

***Lenuel Gastón Pereira Hernández 19-03199-0***

***Fecha 28-10-2024***

***A la Libertad por la universidad***

**Proyecto 1: Desarrollo de una aplicación al estilo instagram.**

# Introducción

Con la implementación y realización de esta guía, se pretende que el estudiante realice de manera guiada una pequeña aplicación web similar a Instagram, haciendo uso de gemas muy útiles en el desarrollo de aplicación en Rails como son: paperclip; para publicación de imágenes, devise; para la gestión de usuarios, con la finalidad que éste consolide las competencias que ha venido adquiriendo con las asignaciones anteriores.

# Objetivos

* Consolidar los conocimientos sobre Ruby on Rails adquiridos en las asignaciones previas.
* Desarrollar una aplicación en Rails desde 0. • Conocer acerca de la gema devise.

**Tiempo**

* Tres sesiones de clases

# Requerimientos

|  |  |
| --- | --- |
| **Software** | **Hardware** |
| Sistema debian 9 virtualizado en Virtual box con:   * Ruby versión 2.4.1 * Rails 5.4.1 * Nodejs * Sublimetext | Computadora con características:   * Memoria ram mínimo 2GB * Procesador mínimo 2.1 GHz |

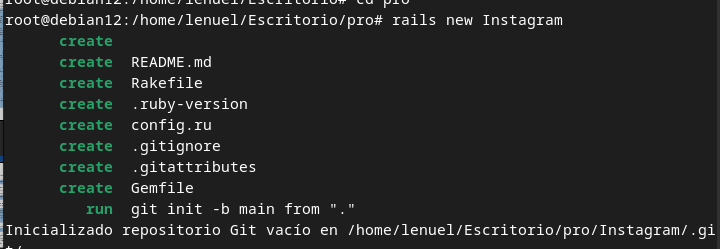
# Referencia

* Thoughtbot. Paperclip. github.com/thoughtbot/paperclip
* Beginner Programmers. Make an instagram like site in under 30 minutes. [www.youtube.com/watch?v=MpFO4Zr0EPE](http://www.youtube.com/watch?v=MpFO4Zr0EPE)
* Plataformatec. Devise. github.com/plataformatec/devise

# Desarrollo

En este proyecto, es en su mayoría guiado, todos los pasos descritos a continuación les llevará a desarrollar una aplicación web con la funcionalidad de crear usuarios y que los usuarios publiquen imágenes con su respectiva descripción.

1. Como primer paso e inicio del desarrollo de la aplicación se debe crear un proyecto en este caso con el nombre Instagram. 1.1. Creación del nuevo proyecto.

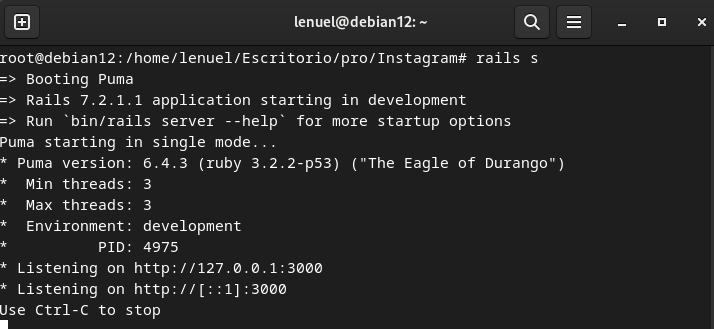


* 1. Una vez terminado el proceso de creación del proyecto, ingresar al directorio.

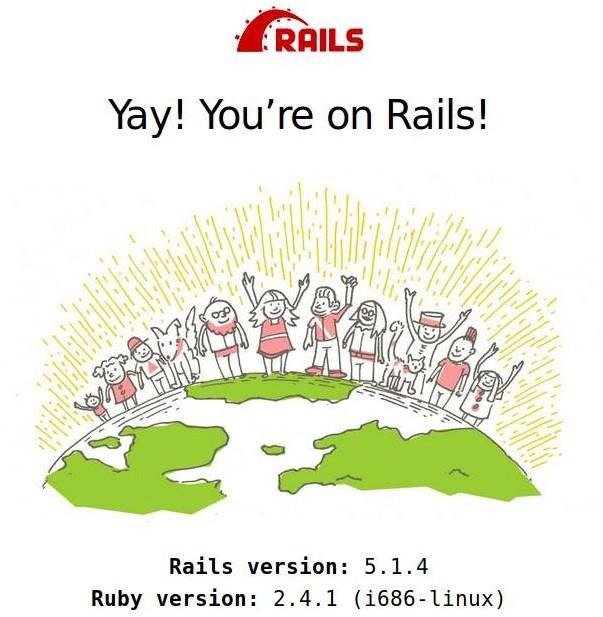
Texto

Descripción generada automáticamente

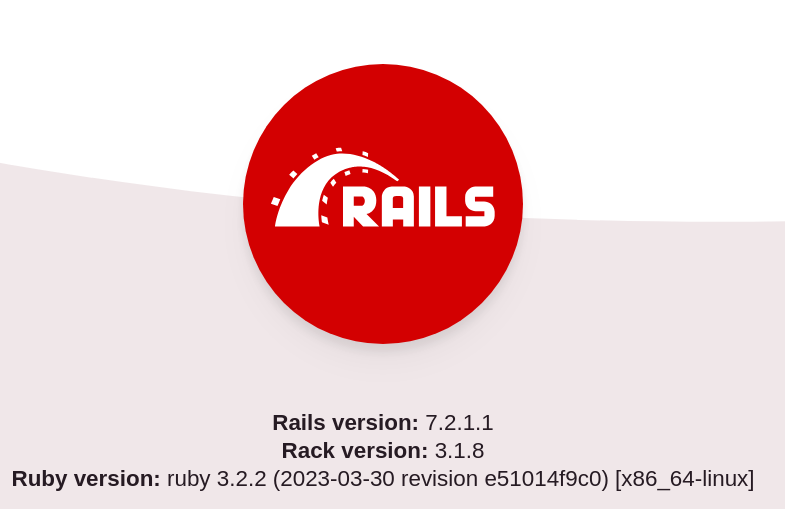
* 1. Para visualizar el resultado iniciar el servidor de Rails.



* 1. 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.**



## Figura 114 Página de inicio por defecto



1. 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.

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

Interfaz de usuario gráfica, Texto, Aplicación

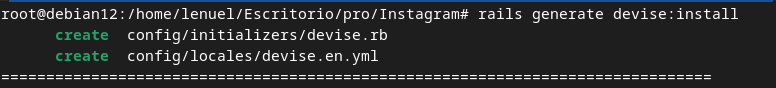
Descripción generada automáticamente

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

Texto

Descripción generada automáticamente

1. 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.
   1. Instalar devise en el Instagram.



* 1. Generar el modelo para los usuarios.

Texto

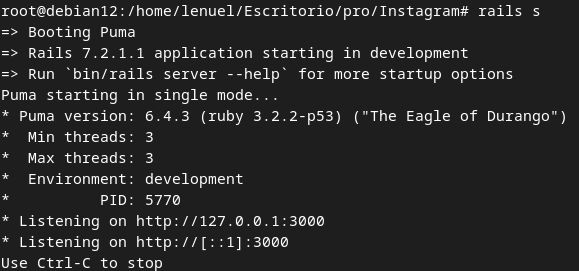
Descripción generada automáticamente

* 1. 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.

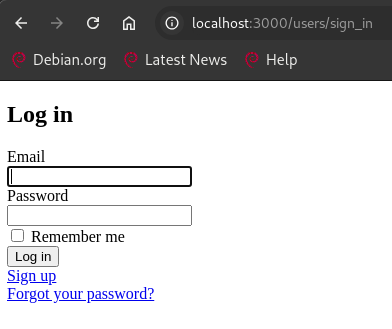
Texto

Descripción generada automáticamente

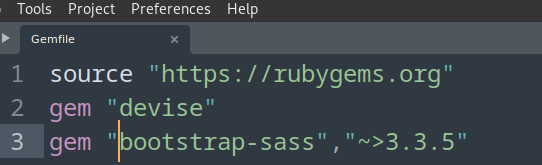
* 1. Correr el servidor para ver lo que se ha generado.



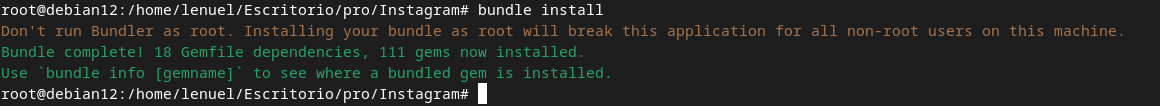
* 1. En el navegador ir a la url **localhost:3000/users/sign\_in**. En esta vista se muestra que se ha generado un formulario para iniciar sesión al igual que para las otras funcionalidades entre ellas la de crear usuarios.



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



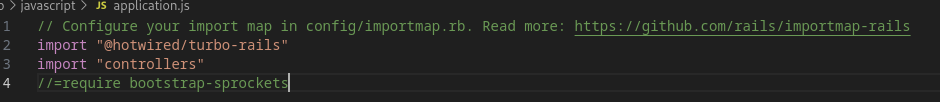
* 1. Instalar las gemas desde el terminal.



* 1. **Interfaz de usuario gráfica, Aplicación

     Descripción generada automáticamente**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**

* 1. 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.



1. Generando vistas, modelos y controladores**.** 
   1. Copiar el siguiente código en **app/view/application.html.erb**.

Texto

Descripción generada automáticamente

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

Texto

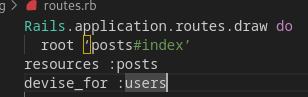
Descripción generada automáticamente

* 1. 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.

Texto

Descripción generada automáticamente

* 1. 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.



* 1. Iniciar el servidor.



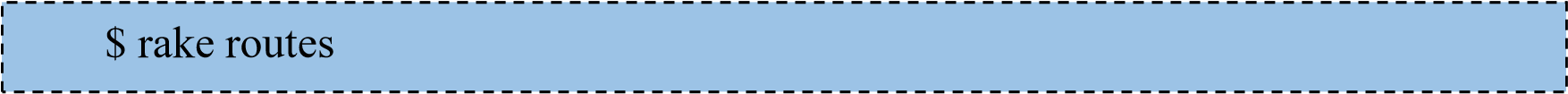
* 1. En el navegador dirigirse a la dirección localhost:3000. Cargará la vista index con la apariencia que se ejemplifica en la figura 3 ese diseño que parece se está cargando desde la vista **app/view/application.html.erb**. que se agregó en el paso 5.1 y carga el contenido de index debido a **<%= yield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

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## Figura 116 Vista index como página de inicio

1. Configuración de link\_to en la vista.
   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



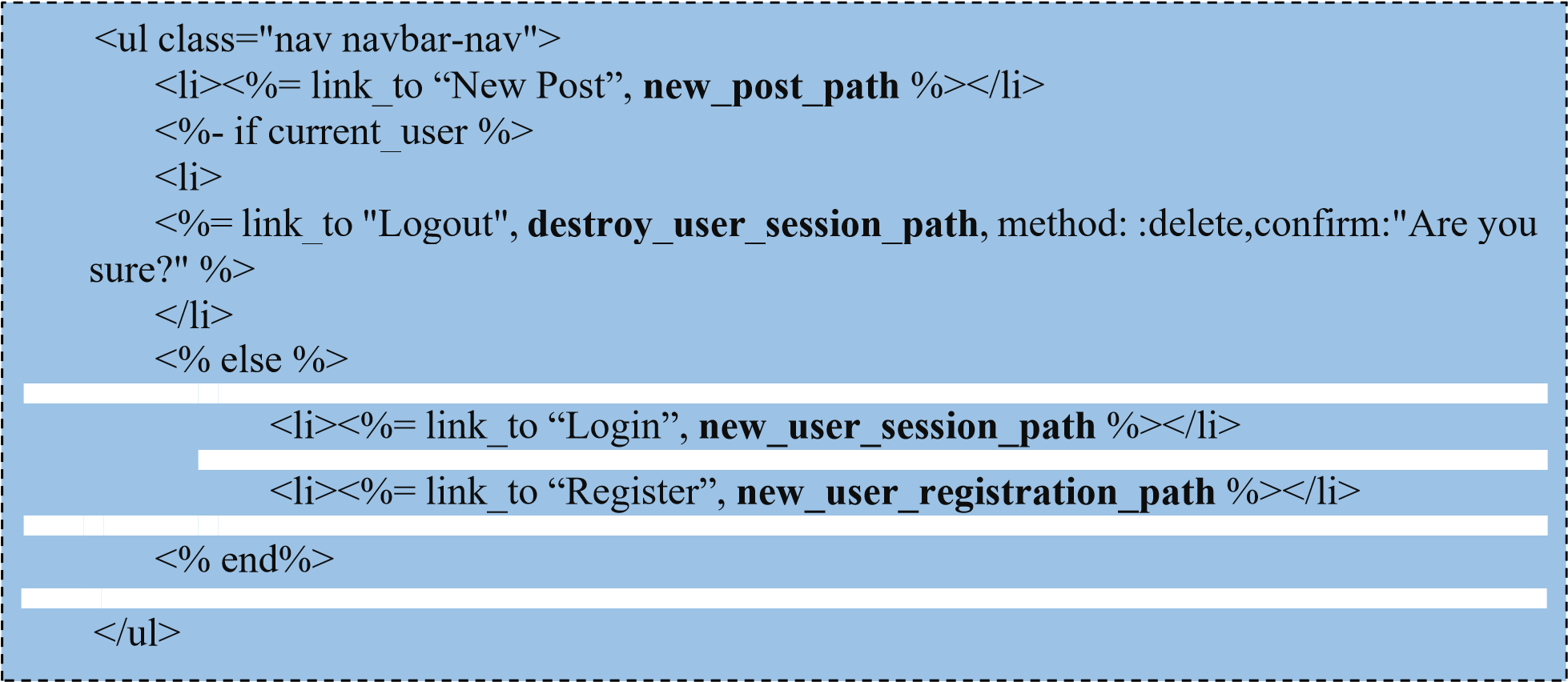
* 1. Se debe obtener una lista similar a la siguiente figura.

Texto

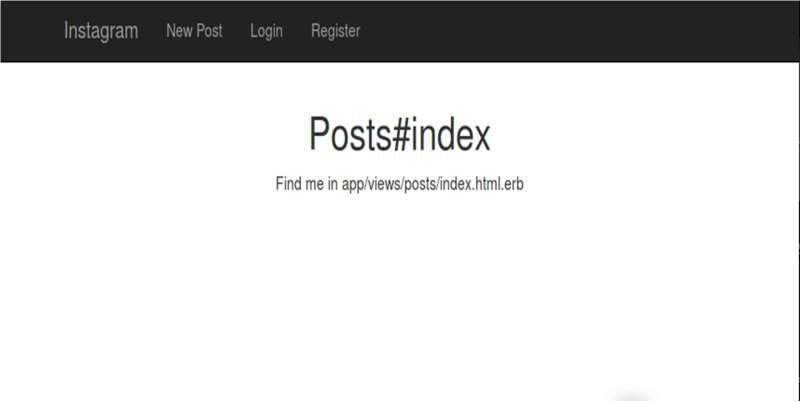
Descripción generada automáticamente

## Figura 117 Lista de las rutas de la aplicación

1. Modificará el archivo **app/view/application.html.erb** y modificar el código de **<ul class="nav navbar-nav">** 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.



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**.



## Figura 118 Vista index

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

Texto

Descripción generada automáticamente

## Figura 119 Vista new

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.

Interfaz de usuario gráfica

Descripción generada automáticamente

## Figura 120 Vista Login

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.

Interfaz de usuario gráfica

Descripción generada automáticamente

1. Configuración de la vista New Post.
   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.

|  |
| --- |
|  |

* 1. Configurar el controlador de los Post, en el archivo **posts\_controller.rb,** del directorio **app/controllers.rb/**, agregar el método new al controlador con el siguiente código.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* 1. 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.

Texto

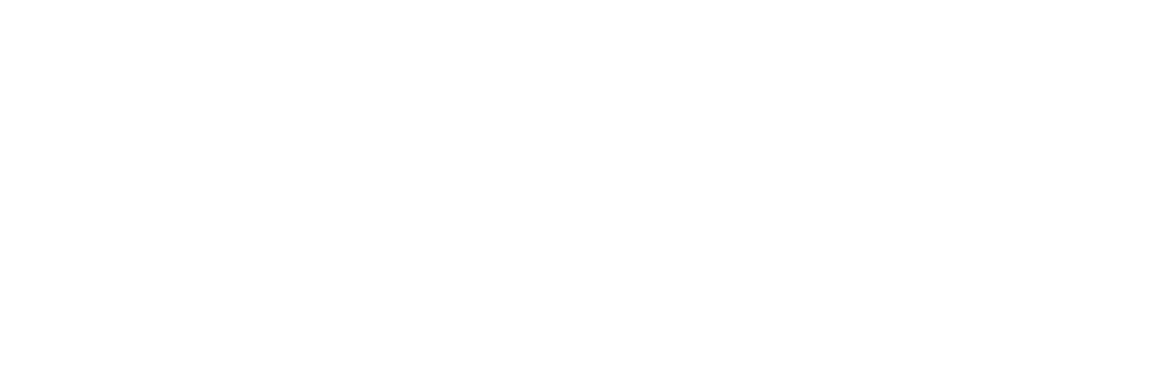
Descripción generada automáticamente

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



**Figura 123 Notificación de un usuario nuevo registrado**

1. 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="row">

<

%

flash.each

do |a,b|

%>

<

dir

class="alert

alert

-

success"><%=

b%>

<

/dir

>

<

%end%

>

<

div

class="col

-

lg

-

12

text

-

center">

<

%

=

yield

%>

<

/div

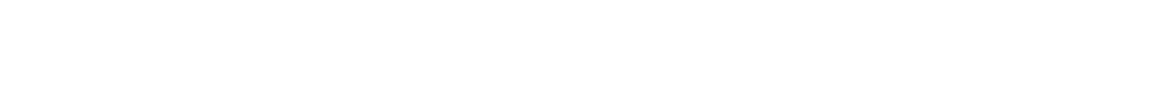
>

<

/div

>

1. Instalación de las gemas paperclip y aws-sdk.
   1. Agregar en la gema al proyecto en el archivo **Gemfile.**



gem "paperclip","~>5.0.0"

gem

'aws

-

sdk',

'<

2.0

'

* 1. Instalar las gemas.

$ bundle install

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



1. 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:
   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.

|  |
| --- |
| class AddAttachmentImageToPosts < ActiveRecord::Migration[5.1] def self.up  change\_table :posts do |t| t.attachment :image end  end def self.down  remove\_attachment :posts, :image  end end |

* 1. Migrar los datos.

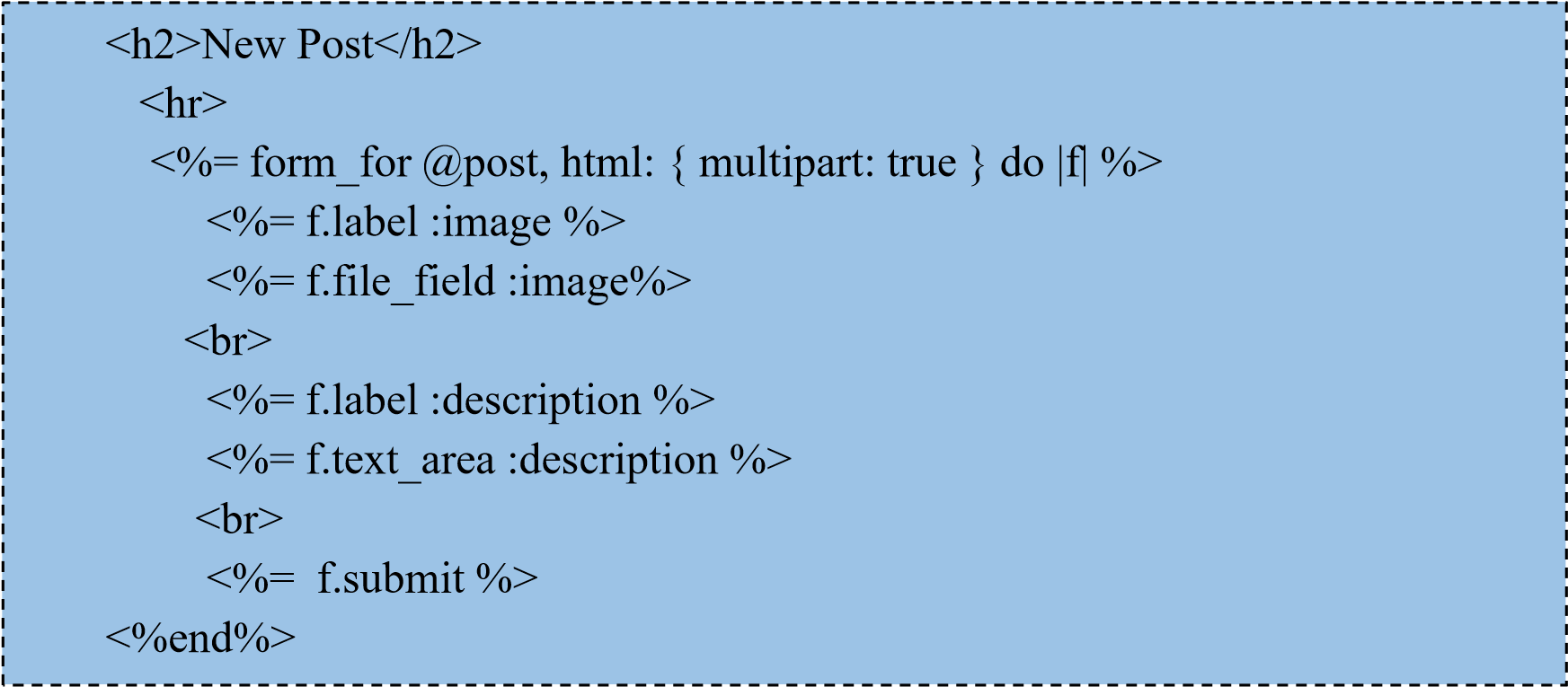
$ rake db:migrate

1. Configurar modelo, vista y controlador paperclip.
   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.

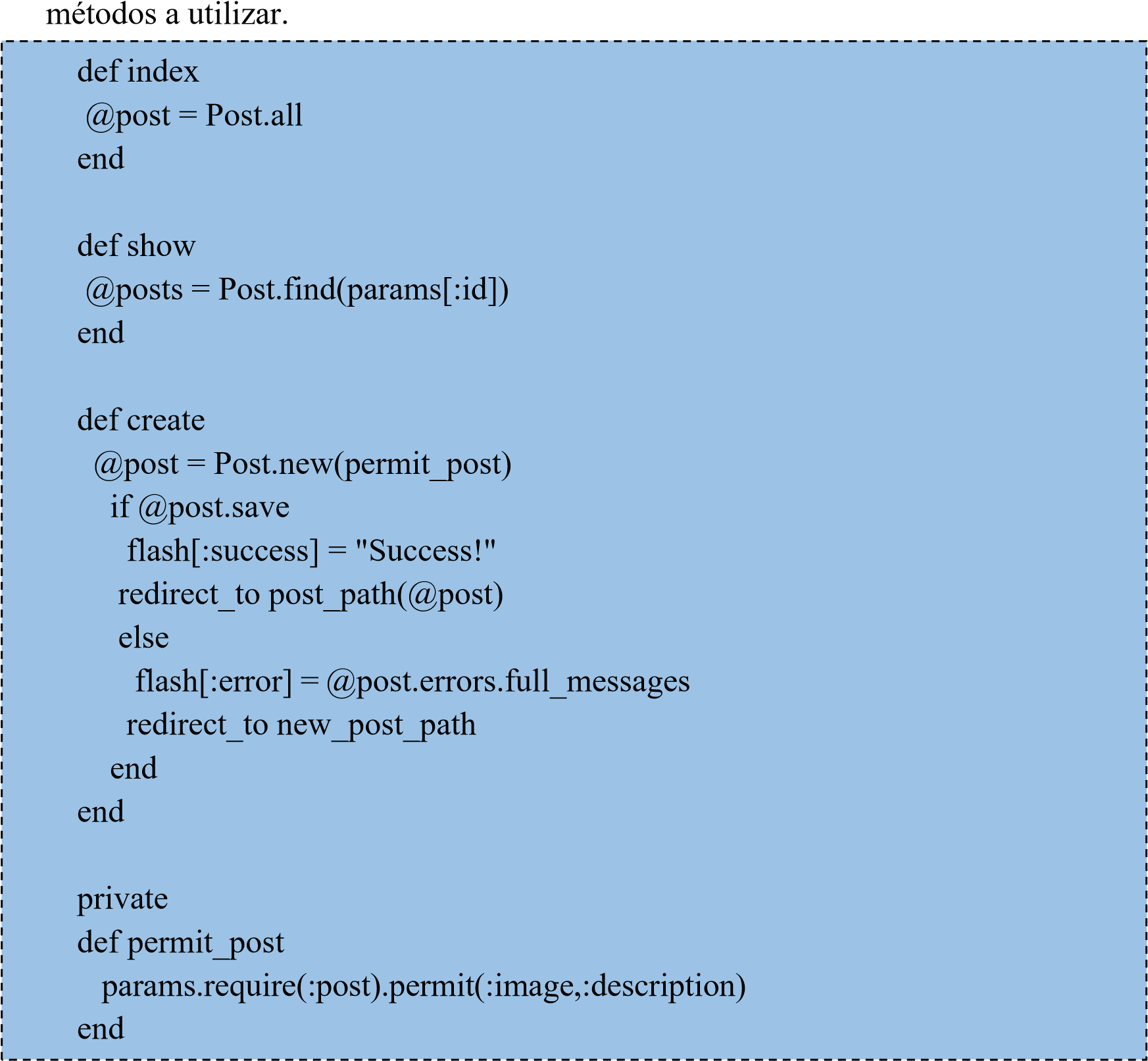
has\_attached\_file :image, styles: {large: "600x600>", medium: "300x300>",

thumb: "150x150>" }, default\_url: "/images/:style/missing.png" validates\_attachment\_content\_type :image, content\_type: /\Aimage\/.\*\z/

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

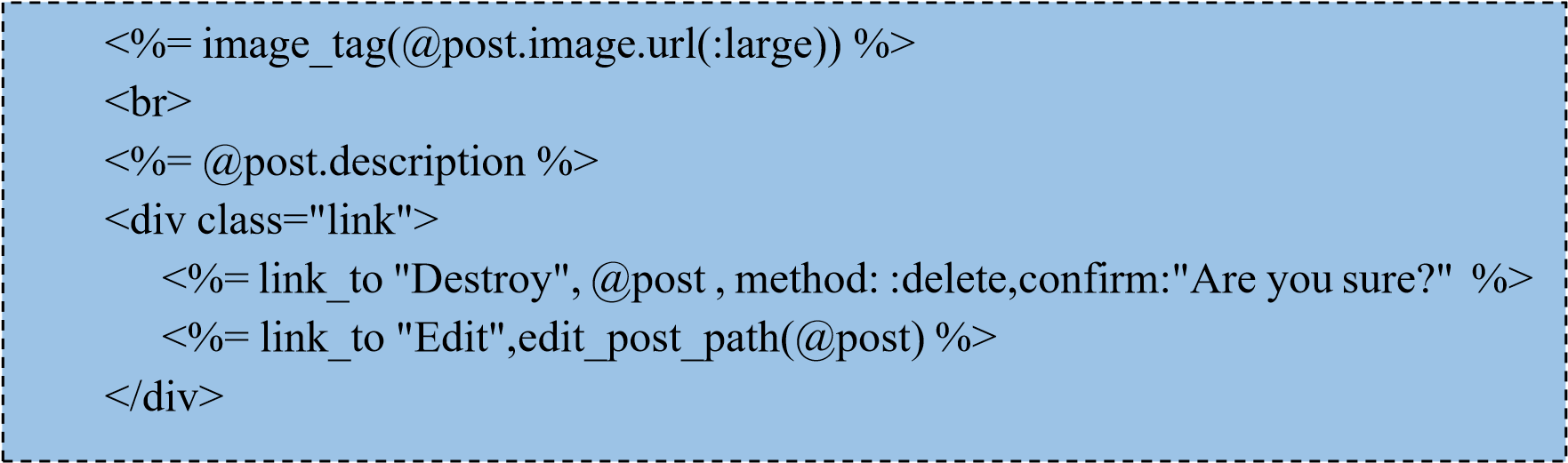


* 1. Editar el controlador en app/controllers.rb/**posts\_controller.rb** y agregar los

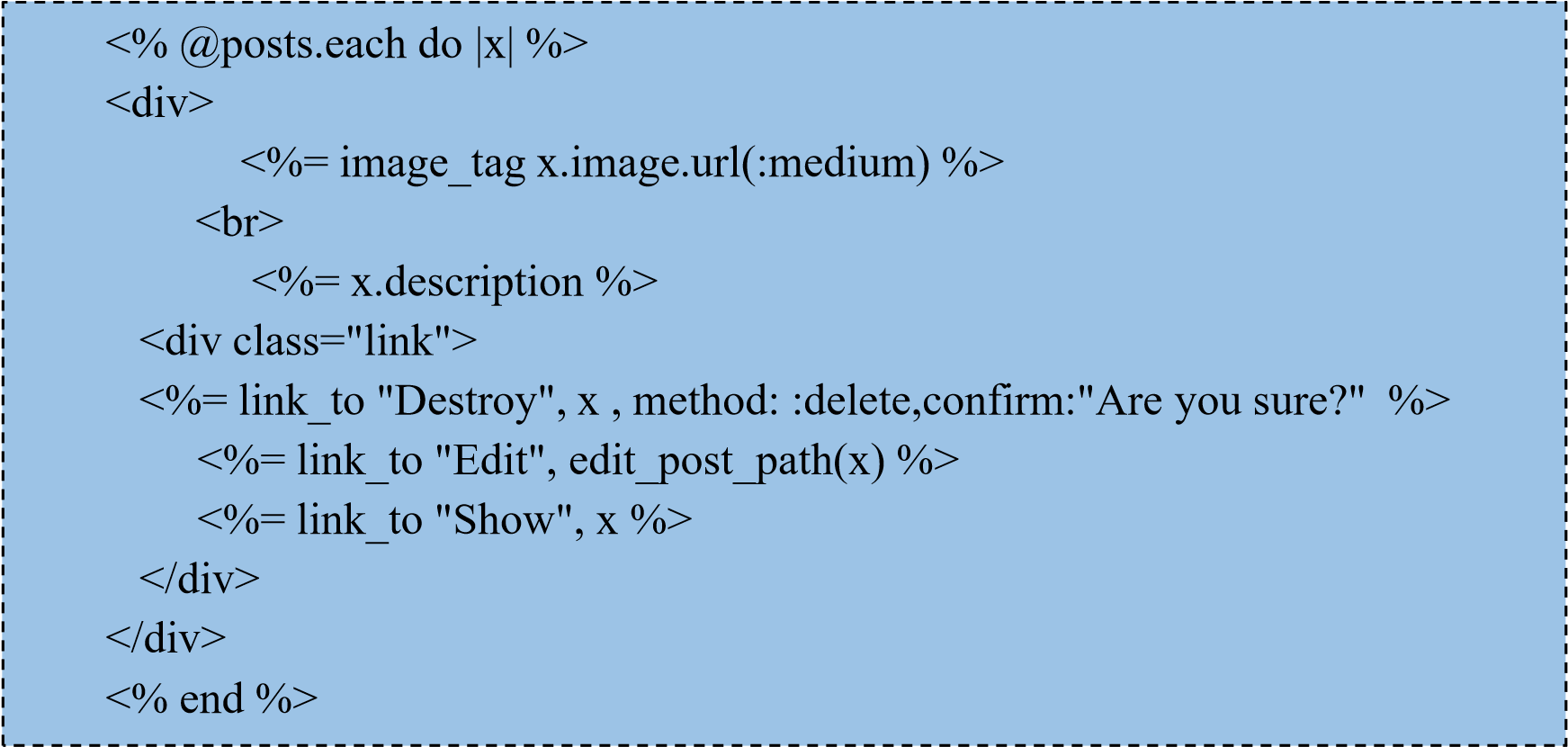


El método **index** será invocado por la página de inicio y lo que hace es listar todos los post que han sido creados, el método show muestra un post en particular para identificarlo recibe como parámetro el id de ese post. El método créate es el encargado generar el nuevo post

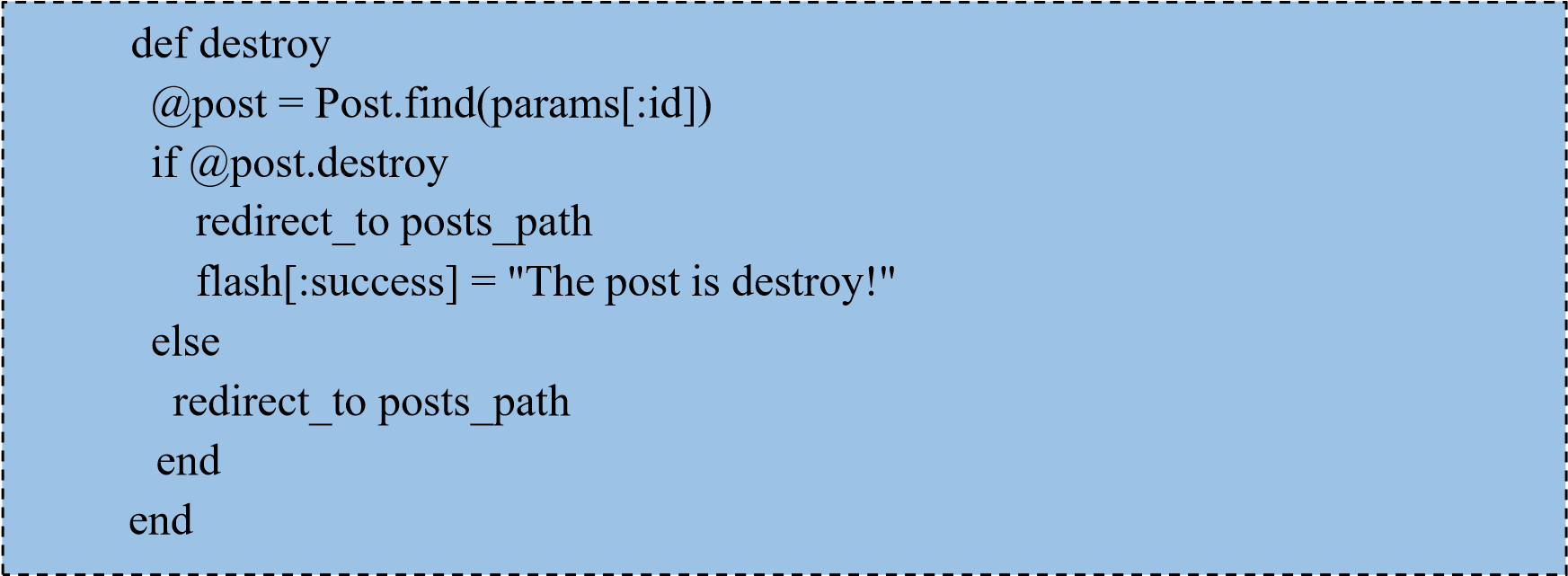
1. Configuración de la vista.
   1. Agregar el siguiente código en app/views/posts/**show.html.erb**. El cual permite que se visualice un pos en particular.



1. 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.



1. Crear el método **destroy** dentro del archivo **posts\_controller.rb** del directorio **/app/controllers/,** agregar el siguiente código debajo del método **create**.



1. 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 posteada.

**Figura**

**124**

**Creación**

**de**

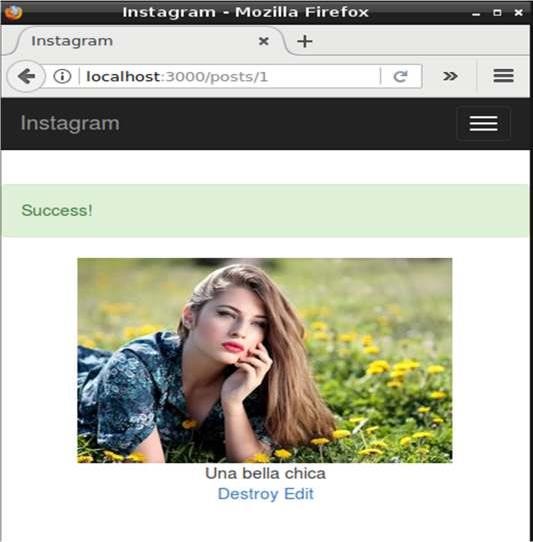
**un**

**nuevo**

**Post**



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



## Figura 125 Post creado correctamente

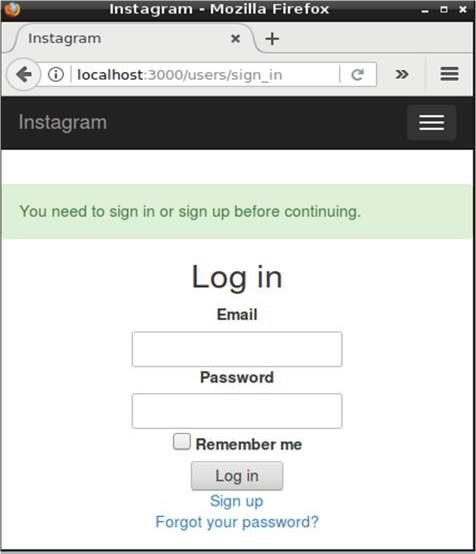
1. Validaciones.

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.

* 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).



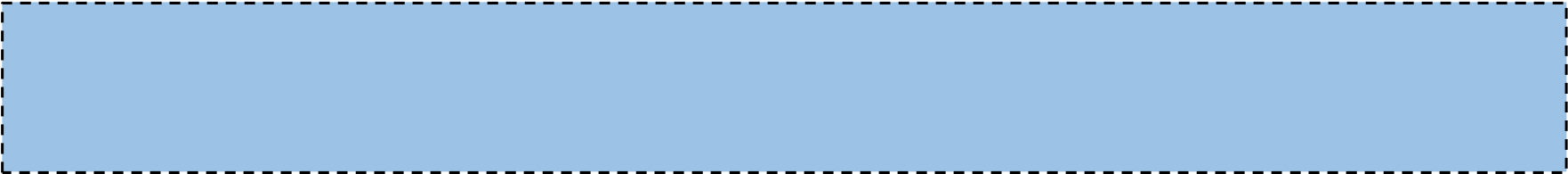
* 1. Resultado de la primera validación se puede observar en la siguiente figura.



**Figura 126 No se permite crear Post sino está logueado**

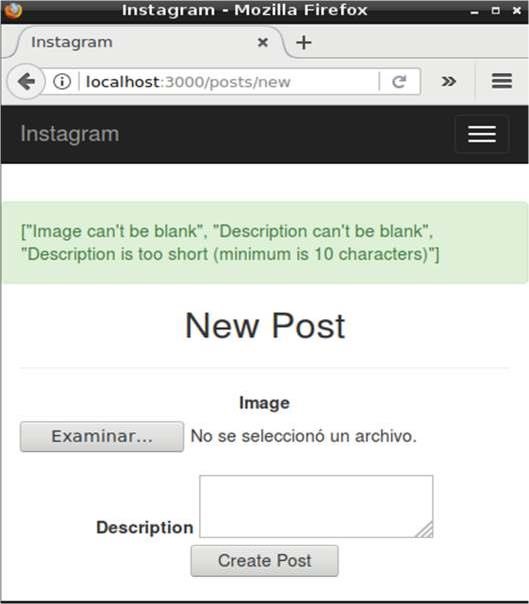
* 1. No permitir publicar post vacíos, ni con descripción menor a diez caracteres. Para ello editar **/app/models/post.rb** y agregar el

validates :image, presence: true

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

siguiente código.

En la siguiente figura se muestra el resultado de la segunda validación.



**Figura 127 Validaciones al momento de crear un nuevo Post**

# Ejercicios propuestos para ser entregados al docente

1. Realizar y analizar cada uno de los enunciados de la guía.
2. Aplicar **css** a las vistas según su criterio.
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.