|  |
| --- |
| Instituto tecnologíco de costa rica |
| Tarea Programada 4 |
| Ruby-Instagram-Twitter |
|  |
| ***Estefany Quesada Montero Esteban Aguilar Valverde Jennifer Barrantes Villalobos*** |
| **19/06/2013** |

|  |
| --- |
|  |

***Titulo: Aplicación web Ruby­Sinatra***

Tabla de contenido

[Descripción del problema 3](#_Toc359410591)

[Diseño del programa 4](#_Toc359410592)

[Librerías usadas 6](#_Toc359410593)

[Análisis de resultados 9](#_Toc359410594)

[ descripción de las tareas 9](#_Toc359410595)

[Manual de usuario 9](#_Toc359410596)

[ instrucciones de uso 9](#_Toc359410597)

[ instrucciones de ejecución 18](#_Toc359410598)

[Conclusión personal 19](#_Toc359410599)

Descripción del problema

La aplicación debe permitir un criterio de búsqueda, (hashtags), el cual es el usuario el que lo especifica, donde deberá usarlo tanto en Twitter e Instagram, para obtener los contenidos de los tweets y las fotos respectivamente.

El numero de resultados podrá ser configurable por medio de la aplicación o desde un archivo de configuración del servidor.

La aplicacion debe ser Orientada a Objetos, programada desde Ruby, y utilizando Sinatra como framework de la misma.

Diseño del programa (decisiones de diseño, algoritmos usados, explicación de la forma de obtener el contenido de los APIs, explicación de la conexión con dichos APIs, manejo de OAuth, etc.)

La aplicación se desarrollo en Ruby como lenguaje de programación, y Sinatra como framework para manejo de aplicaciones web, utilizando sistema operativo Linux para ejecutarse.

Para la aplicacion se utilizaron estas librerias

#Importar las bibliotecas y el archivo Instagram.rb

require 'rubygems'

require 'erb'

require 'sinatra';

require 'Instagram';

require 'Twitter';

Para el desarrollo de esta se crearon las siguientes variables

#Variables globales del sistema

$descrip #Variable que tiene el paramtro de busqueda

$numero #Variable que tiene el numero de imagenes a mostrar

#Parametros axiliares en caso de faltar un valor de busqueda

$param\_busqueda = "house"

$param\_numero = 10

Búsqueda de Instagram

# Clase que se encarga de buscar las imÃ¡genes en instagram

class BuscarInstagram

def initialize(tag,cant\_hash\_tags)

Instagram.api\_key = "b907fe707248d846c970de37fdf212e0"

Instagram.shared\_secret = "d6300bd479506d23"

end

@busqueda1=Instagram.tag\_recent\_media(tag,options={:count=>cant\_hash\_tags})

end

def retorno

return @busqueda1

end

end

Utiliza una llave para login, y realiza la busqueda

class Search

def initialize(hash,cant\_hash\_tags)

# Datos de conexion a twitter

TWITTER\_CONSUMER\_KEY = 'consumerKey'

TWITTER\_CONSUMER\_SECRET = 'consumerSecret'

TWITTER\_ACCESS\_TOKEN = 'accessToken'

TWITTER\_ACCESS\_SECRET = 'accessSecret'

# Configurar el cliente

Twitter.configure do |config|

config.consumer\_key = TWITTER\_CONSUMER\_KEY

config.consumer\_secret = TWITTER\_CONSUMER\_SECRET

config.oauth\_token = TWITTER\_ACCESS\_TOKEN

config.oauth\_token\_secret = TWITTER\_ACCESS\_SECRET

end

# Instanciar

client = Twitter::Client.new

printf("Utilizando el usuario: %s\n", username)

printf("Buscando la palabra clave: %s\n", keywordSelected)

# Encontrar tweets usando el REST API de twitter

tweets = Helpers.find\_tweets(keywordSelect,keywordSelect)

tweets.each do |tweet|

@busqueda = Twitter.search(hash, :count => cant\_hash\_tags).results.map do ("\nDel usuario: %s\nTweet: %s\n",tweet['from\_user'],tweet['text'])

client.update(tweet['from\_user'])

end

#Se devuelve las respuestas generadas

def retornoBusqueda()

return @busqueda

end

end

Accede a la cuenta de twitter

Librerías usadas (para manejo de páginas web, conexión a APIs, etc.)

* require 'rubygems'
* require 'json'
* require 'instagram'
* require 'rubygems'
* require 'erb'

***Cómo utilizar “erb” para un ambiente* web.**

Bien, primero que todo, que es “erb”.

“Erb” es una gema de Ruby la cual permite trabajar el lenguaje en sistemas empotrados o en ambientes web esta gema ya esta instalada en la configuración de Ruby por lo que no es necesaria su instalación.

**¿Cómo funciona?**

De forma simple erb permite llamar o visualizar archivos web que poseen la interfaz de usuario, lo realiza mediante una llamada al archivo “erb”.

Ejemplo:

erb :Prueba #Esto llama al archivo Prueba.erbDiferencias entre Ruby y ERB

No existe diferencia pues no son lenguajes diferentes en una forma directa, ya que un programa en Ruby o de formato “rb” contiene código de Ruby en este caso particular contendrá también elementos de codigo ofrecidos por Sinatra mientrea que un archivo formato “erb” contendrá únicamente codigo de HTML, ahora esto no quiere decir que un archivo “erb” no pueda tener código de Ruby ya que si

lo puede tener mediante un agregado a las etiquetas de HTMl.

***Muestra de etiquetas con código Ruby***.

<%\_\_\_\_\_\_\_\_\_\_%> # Código Ruby normal

<%=\_\_\_\_\_\_\_\_\_%> #Muestra de una expresión de Ruby (envía el resultado o valor que contiene la variable, es similar al puts de Ruby pero para TML)

<%#\_\_\_\_\_\_\_\_\_%> # Comentarios de Ruby

El siguiente es un ejemplo de código de Ruby dentro de un HTML en un

archivo “erb”. Nota (El fragmento se tomo de forma aleatoria).

<% for j in 0..2 do %>

<% case num

when 2, 8 %>

<td class="colcentro">

<% when 4, 6 %>

<td class="rowcentro">

<% when 5 %>

<td class="colcentro rowcentro">

<% else %>

<td>

<% end %>

***¿Un archivo “erb” se puede tratar como un HTML?***

Por supuesto ya que los “erb” son archivos HTML que están integrados para que puedan usar código de Ruby pero no por eso dejan de ser HTML que pueden ser abiertos por el navegador para revisar que mostraran al usuario.

Si la información del usuario la recive el “erb”. *¿Cómo la puede manejar Ruby?*

Bien en HTML usted no debe preocuparse tanto de como enviar la información ya que existe un botón solo enviara los datos cada vez que usted lo presione ahora Ruby siempre recibirá esta información como un string (si la información la requiere como otro tipo de dato usted deberá hacer la conversión al tipo deseado)mediante la siguiente notación:

variable\_de\_Ruby = params[:variable\_erb] #Esto por si solo no servirá ya que se requiere una forma de saber con que conector o titulo de Sinatra se esta interactuando por ello

en codigo HTML tenemos:

<form action="/calculos" method="post">

#

#Aquí adentro van los entry y el botón de envío

#

</form>

Bien ahora analizándolo, action contiene el titulo en Sinatra mientras que metod posee una acción esto se vera explicara mejor más adelante, ahora veamos el código en el archivo Ruby para entender la conexión.

post '/calculos' do

Como en post, aparece en el código comparándolo con Ruby podríamos decir que es como un equivalente a def y '/calculos' como vemos podemos decir que es el nombre del metodo pero este esta entre comillas simples por ser un titulo de sinatra.

* require 'sinatra';

Ahora el API de Sinatra de forma inmediata empezara la conexión entre nuestro código en Ruby con la pagina de Sinatra despues de conectado el sistema enviara al usuario un puerto local

(generalmente es el puerto 4567).

Sinatra permite realizar varias funciones o trabajos en el código Ruby en esta guia solo veremos tres que creo son las más importantes.

Get – Este permite visualizar la pagina (puede contener o llamar a métodos funcionales).

Post – Este es usado para realizar acciones como llamar métodos o cálculos, no muestra los resultados.

Redirect – Llama a otro titulo de Sinatra en el código de Ruby, los redirect son llamados dentro de otros métodos o un Post.

Ejemplo:

Get:

get '/' do

erb :Prueba

end

Post

post '/calculos' do

$numero1

$numero2

= params[:campo1].to\_i

= params[:campo2].to\_i

end

Redirect:

post '/calculos' do

#

#

#

redirect '/resultado'

end

* require 'Twitter';
* require 'open-uri'
* require 'hpricot'
* require 'twitter\_oauth'

Análisis de resultados:

* descripción de las tareas Se logro completar todo, pero algunas pruebas fallan

Se debe a la conexión entre Ruby y Instagram

Manual de usuario:

* instrucciones de uso (cómo levantar el servidor para el webapp, etc.),

***¿Cómo sabe Sinatra que mostrar al inicio?***

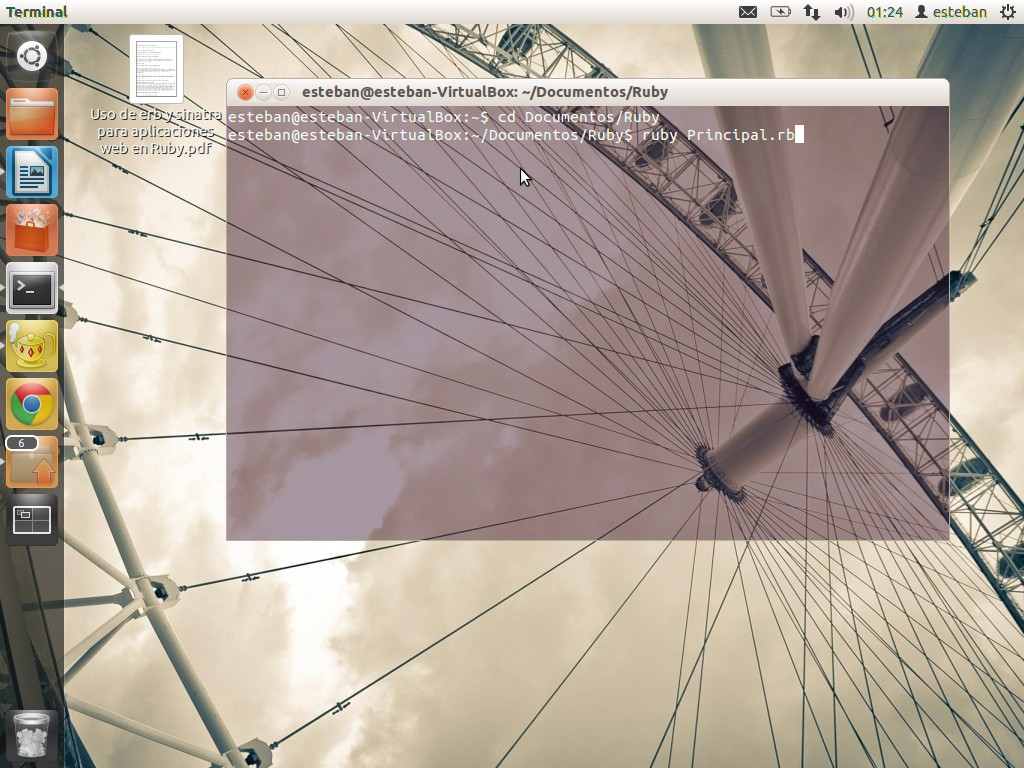
Es simple, Sinatra siempre buscara el get con el titulo '/' ya que lo tomara como la raíz de todo el programa, y realizara lo que este contenga.

***¿Cómo terminar la conexión?***

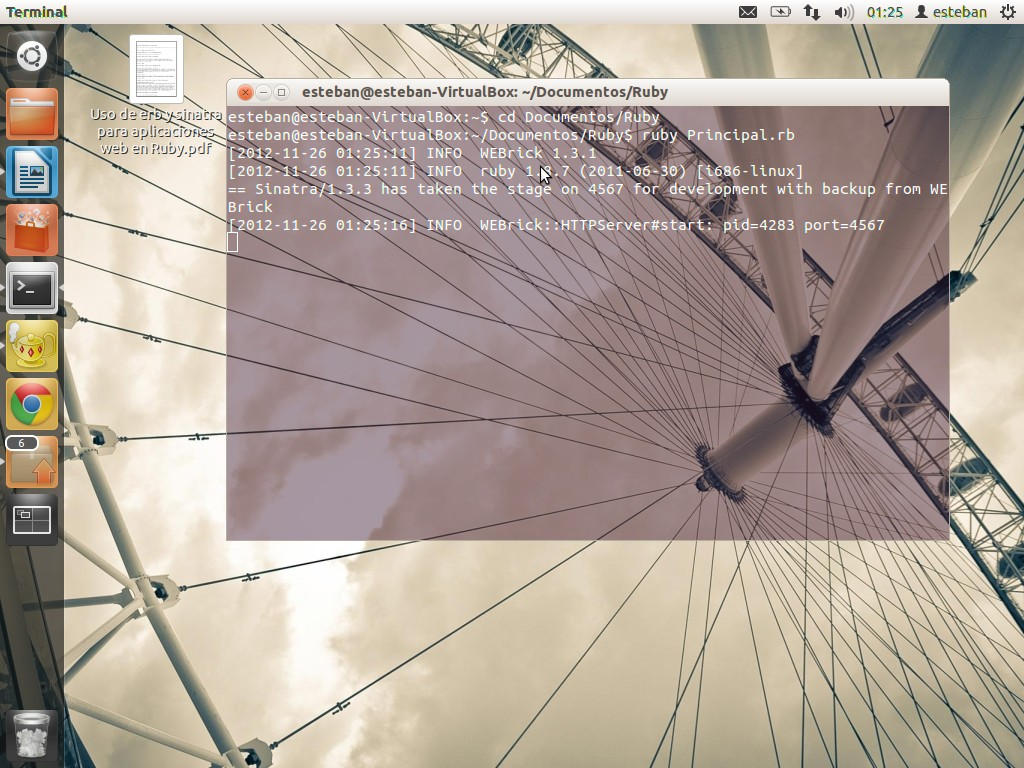
Control C y cerrar la pagina del navegador que contenía el programa

***¿Cómo usar el programa?***

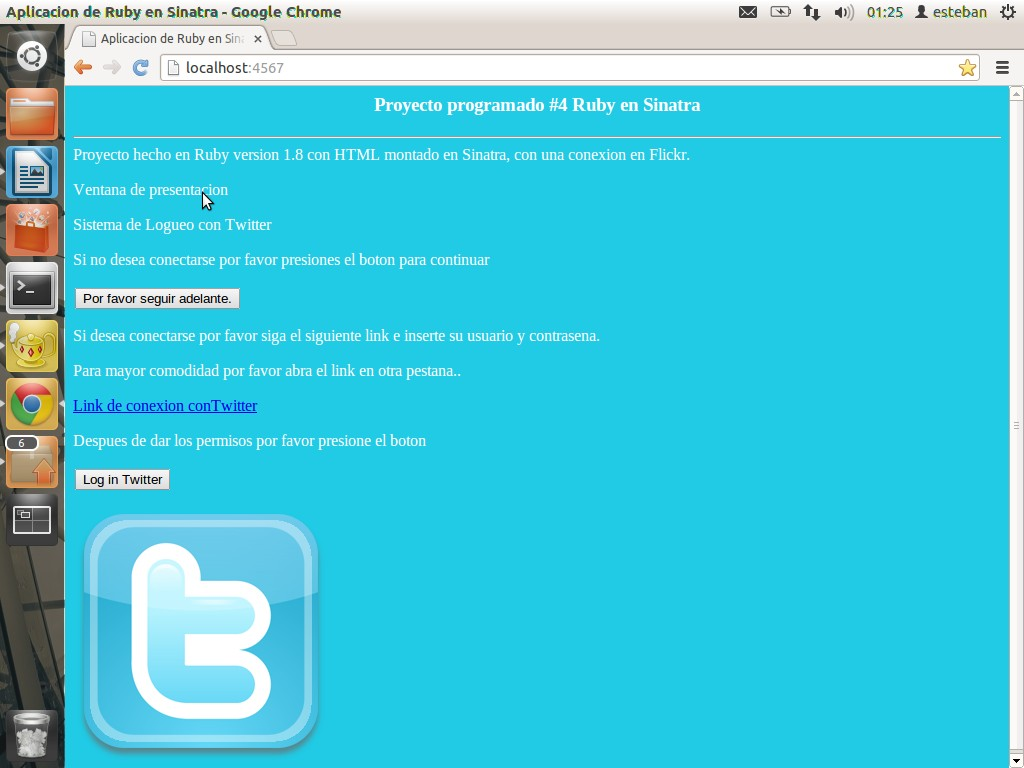
Primero se debe abrir la terminal, ir a la dirección del programa y cargarlo



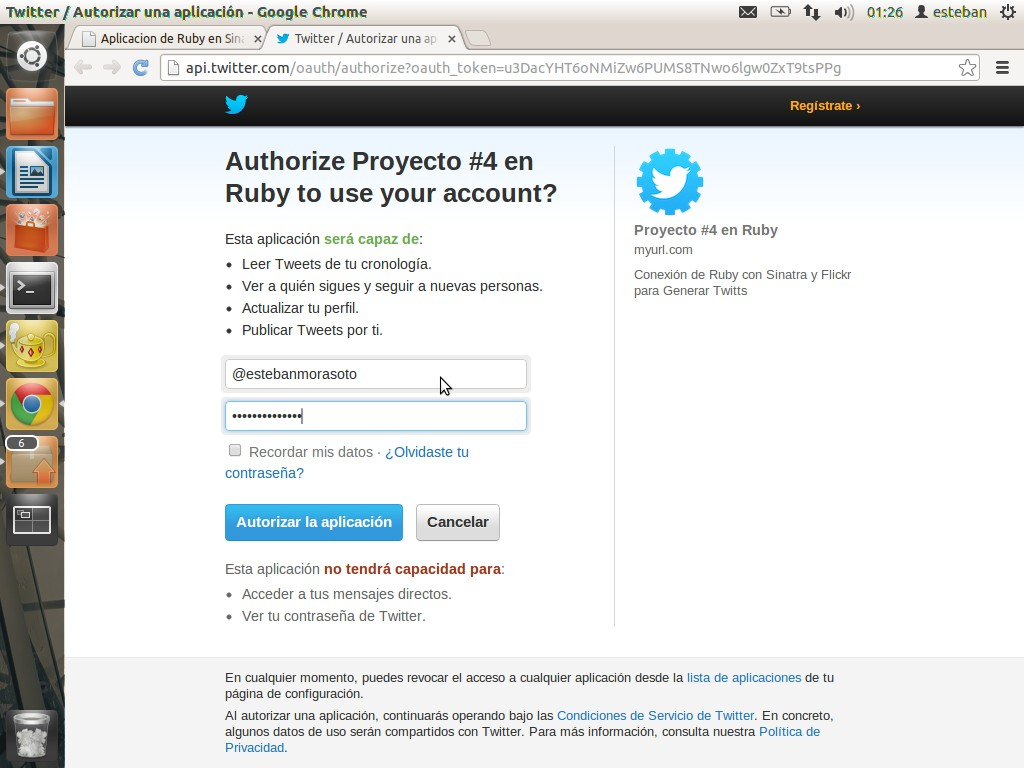
Después aparecerá la conexión con Sinatra la cual indica cual puerto usar para visualizar el programa.

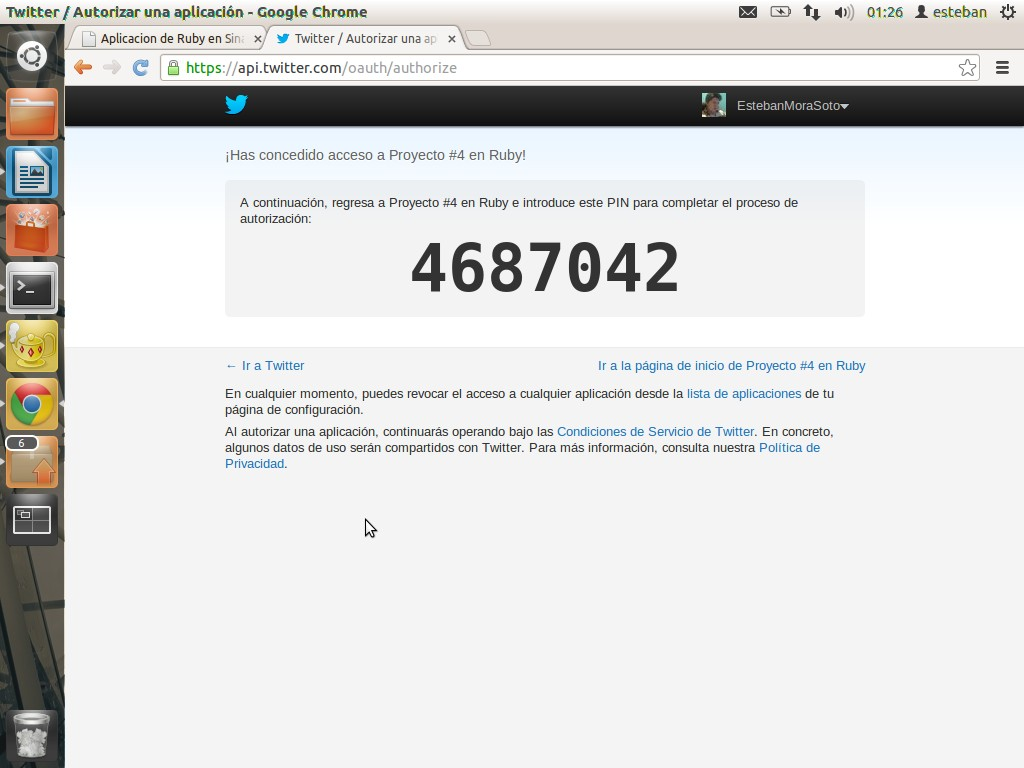


Ahora por favor abrir su navegador y colocar http://localhost:4567/ para que aparezca la aplicación deberá aparecer la pantalla que en nuestro caso es esta:



En esta imagen puede verse un link, por favor ábralo para acceder al sistema de permisos de Twitter, en caso de no querer conectarse solo debe dar click al botón de omisión.

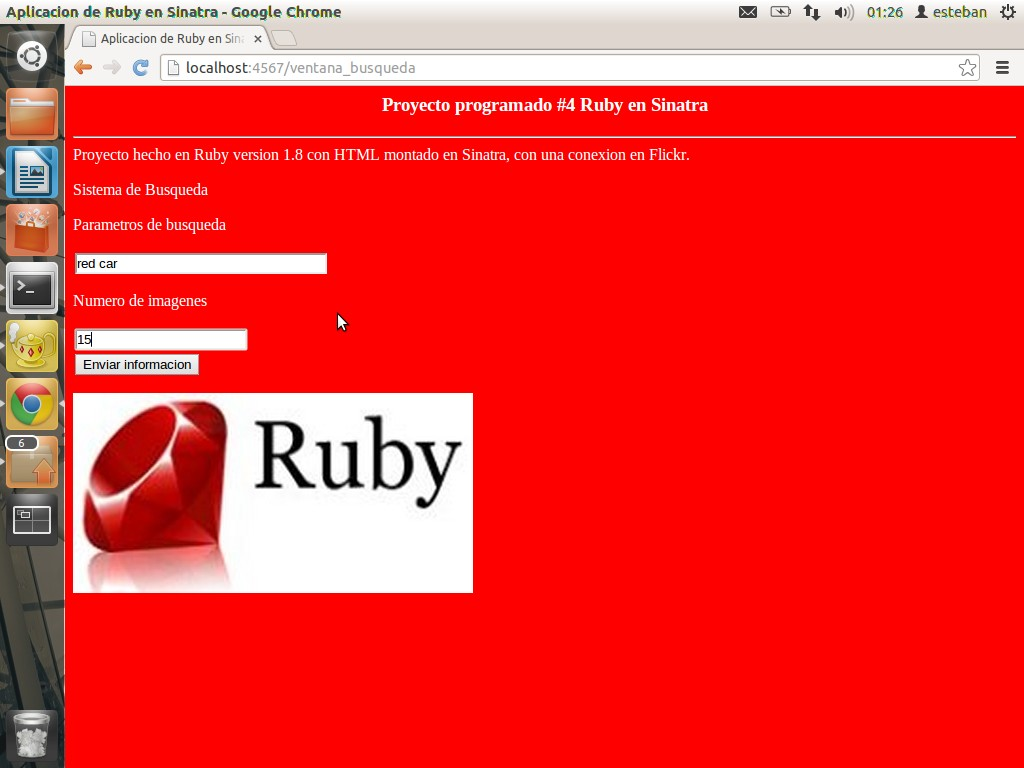


Para esta página el usuario debe colocar su nombre de usuario y su contraseña, después de eso debe dar la autorización sobre la aplicación, finalmente se cargara una página como la que se mostrara a continuación, esta pagina puede cerrarse.

Después de eso puede volver a la aplicación y acceder con una conexión a Twitter que lo llevara de inmediato el sistema de búsqueda.

Al cargar el sistema de búsqueda usted podrá elegir que buscar y en que cantidad de fotos que desee siempre que estén dentro de los parámetros permitidos, después puede dar click en el botón para

iniciar la búsqueda.



* instrucciones de ejecución

***Instalación de Ruby (versión 1.8)***

sudo apt­get install ruby1.8

***Instalación de ruby gems***

sudo apt­get install rubygems

***Instalación de conexión con Sinatra***

sudo gem install sinatra

***Instalación de “erb” para uso de HTML como interfaz gráfica web***

De acuerdo al la documentación oficial “ebr” ya esta integrado en la instalación de Ruby. Forma de llamarlo (require 'erb').

***Integración con Sinatra.***

Tal ves este punto ya muchos lo han investigado pero en lo personal esta guía estaría incompleta sin mencionar como funciona lo visto anteriormente en sinatra.

***Instalación de conexión con Sinatra***

sudo gem install sinatra

***Cargar Sinatra en el código de Ruby.***

require 'sinatra'

# Conclusión personal

Gracias al desarrollo de este proyecto se logro ampliar el conocimiento sobre entornos web, esto con la aplicación que se realizo en Sinatra, además de esto se logro ampliar la manipulación de API en redes sociales.

Logrando de esta manera el objetivo de la misma que era ampliar y entender los lenguajes Orientados a Objetos, utilizando Ruby como base.