

Tarea Programada IV

Lenguajes de Programación

15/06/2012

Tecnológico de Costa Rica

Ricardo P. Gago y Jose David Chaverri

Tabla de contenido

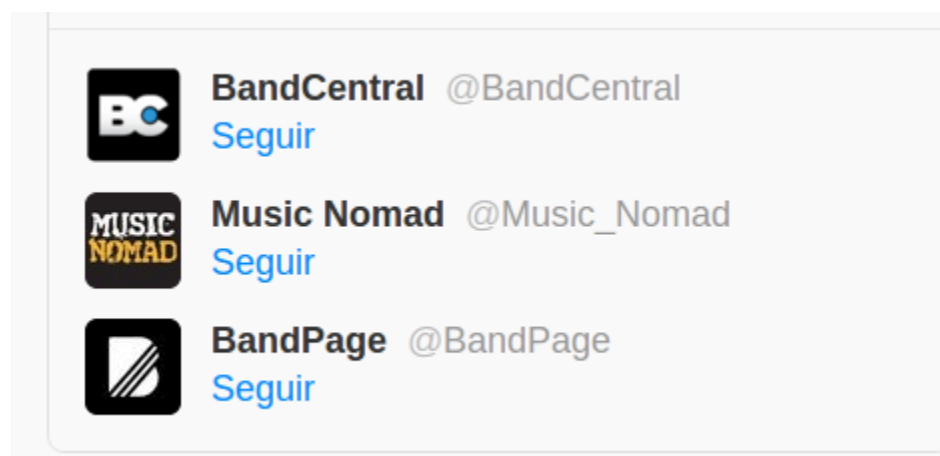
Descripción del problema	3
Diseño del programa.....	6
Conexión con Twitter	7
Extracción de los datos	10
Extracción de los URL's con respecto a la consulta	10
Análisis de resultados	14
Manual de usuario	15
Etapas de instalación	15
Ejecución del programa	16
Conclusiones	19
Bibliografía	20

Descripción del problema

En el campo de la música, las bandas independientes usualmente tienen dificultades a la hora de divulgar su música, ya que no tienen los presupuestos de mercadeo que tienen grupos más conocidos, ya que por lo general son bandas de jóvenes aficionados que no tiene acceso a los recursos con los que cuentan las bandas mas reconocidas las cuales gozan de un patrocinio de grande compañías de entretenimientos, pero esto no significa que la calidad de la música de estas pequeñas bandas sea menor, por el contrario son bandas con mucha creatividad y vigor, con letras que transmiten la cultura y sentir de la localidad donde surgen estas pequeñas bandas.

Existen ciertos “websites” que han ayudado a disminuir esa brecha, entre los cuales se encuentra Bandcamp (www.bandcamp.com). Este es básicamente un sitio en el que las bandas pueden subir y vender su música, para que la misma sea escuchada y/o descargada por los usuarios, lo cual facilita la divulgación de su trabajo. La cantidad de música en este tipo de sitios es enorme y de gran calidad, letra de cultura nacional, crítica al sistema, en fin los centenares de contenidos hacen de este sitio un lugar donde puedes navegar y encontrar a tu próximo grupo favorito.

Con páginas como esta y el crecimiento experimentado por las redes sociales y las redes de información como Twitter se ha creado un tipo de simbiosis informáticas entre las bandas que se alojan en el sitio Bandcamp y los usuarios de este tipo de redes, lo cual genera una forma nueva, rápida e innovadora de dar publicidad a los nuevos talentos, logrando con esto que los mismos lleguen cada vez a más público, con lo que pueden incluso llegar a convertirse en una banda muy reconocidas y darle fuerza al sitio o proyectos con objetivos similares tales como:



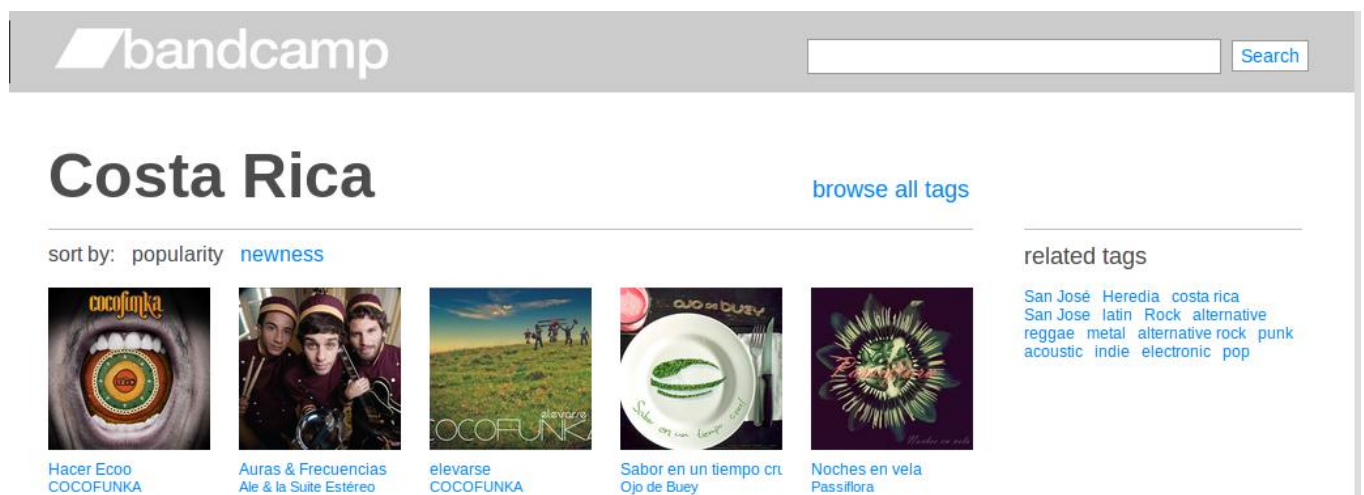
Los cuales tienen contenidos similares a los de Bandcamp, los cuales buscan ayudar a las bandas a darse a conocer y a los usuarios a encontrar nuevas alternativas de música.

La idea es entonces es generar un programa que permita al usuario digitar una etiqueta, la cual podrá ser un género musical o una ciudad y que este haga la consulta respectiva en la página de Bandcamp y obtenga un top 10 de los resultados que retorne dicha página.

Se desea extraer ciertos datos de los resultados que retorne la página, con el objetivo de hacer una publicación o Tweet en Twitter con los datos extraídos.

Por ejemplo si el usuario desea obtener información de bandas un Costa Rica se creará un “query” con el parámetro de búsqueda Costa Rica.

<http://bandcamp.com/tag/costa-rica>



The screenshot shows the Bandcamp website interface. At the top is the Bandcamp logo and a search bar. Below the logo, the search results are displayed for the tag "Costa Rica". The results are sorted by "newness". There are five album covers visible, each with a title and artist name. To the right of the albums, there is a "related tags" section listing various tags like "San José", "Heredia", "costa rica", etc.

bandcamp

Search

Costa Rica

browse all tags

sort by: popularity **newness**

related tags

San José Heredia costa rica
San José latin Rock alternative
reggae metal alternative rock punk
acoustic indie electronic pop

Hacer Ecoo
COCOFUNKA

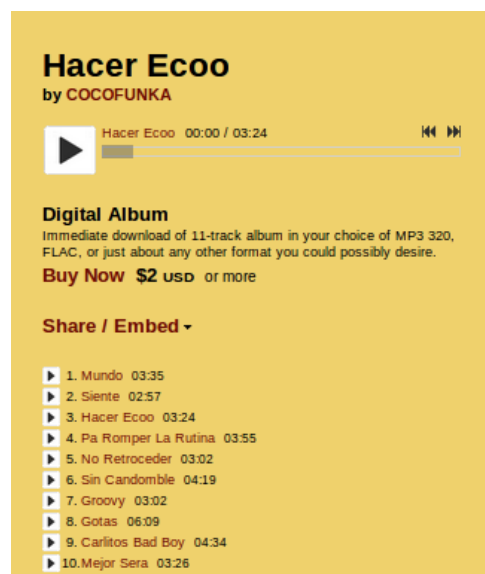
Auras & Frecuencias
Ale & la Suite Estéreo

elevarse
COCOFUNKA

Sabor en un tiempo crt
Ojo de Buey

Noches en vela
Passiflora

Esta consulta retorna entre algunos resultados bandas como Cocofunka, Ojo de Buey entre otras. La idea es que el programa recorra cada uno de los links de esas bandas y retorne información de las mismas la cual será usada para realizar el Tweet.



The screenshot shows the Bandcamp page for the album "Hacer Ecoo" by COCOFUNKA. The page features a yellow background. At the top, the album title and artist name are displayed. Below this is a play button and a progress bar. The "Digital Album" section describes the download options. The "Buy Now" price is listed as \$2 USD or more. The "Share / Embed" section is also visible. At the bottom, there is a list of 10 tracks with their durations.

Hacer Ecoo

by COCOFUNKA

Hacer Ecoo 00:00 / 03:24

Digital Album
Immediate download of 11-track album in your choice of MP3 320, FLAC, or just about any other format you could possibly desire.

Buy Now \$2 USD or more

Share / Embed

1. Mundo 03:35
2. Siente 02:57
3. Hacer Ecoo 03:24
4. Pa Romper La Rutina 03:55
5. No Retroceder 03:02
6. Sin Candomble 04:19
7. Groovy 03:02
8. Gotas 06:09
9. Carlitos Bad Boy 04:34
10. Mejor Sera 03:26

La consulta de la imagen anterior muestra el nombre de la banda, el nombre del álbum, y si la descarga es gratuita o el precio de la misma.

Se requiere que el programa extraiga esta información además del link Bandcamp de la banda, y realice en Tweet con esa información.

Para poder hacer esto es necesario un sistema que permita al programa poder obtener permisos de publicación en una cuenta de Twitter arbitraria, es decir de cualquier persona que use el programa.

Para esto es necesario buscar una herramienta que permite obtener una conexión a Twitter por medio de los credenciales de la persona que utilizara el programa y realizar los Tweets necesarios.

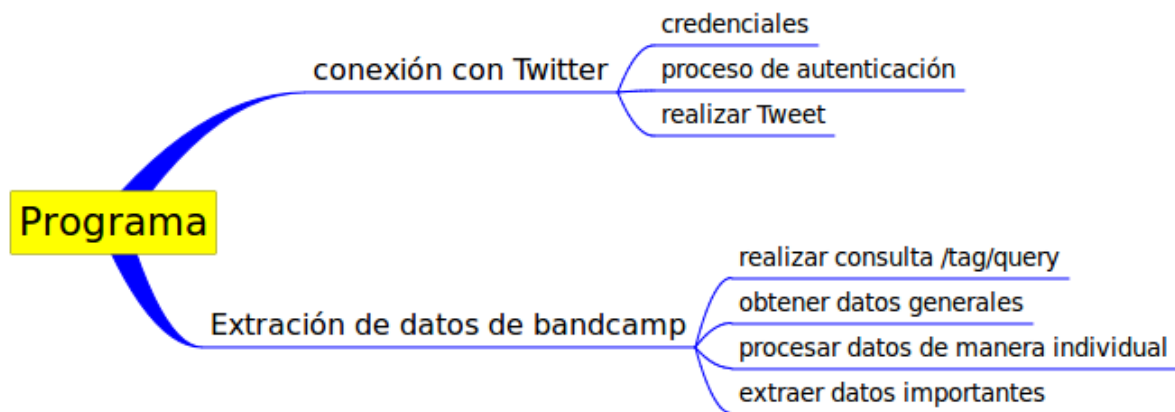
Con esto muchas personas podrán buscar información de sus bandas favoritas o de alguna banda a la cual quieran ayudar para que otras personas la conozcan y hacer Tweets para que sus seguidores pueden observar el Tweet y probablemente les llame la atención y conozcan la banda y en si este importante sitio llamado Bandcamp.

Diseño del programa

El programa se realizará en Ruby 1.9.3p194 (2012-04-20 revisión 35410) [x86_64-linux] el cual es un práctico basado en el lenguaje orientado a objetos; este lenguaje es muy usado en el mercado actual ya que posee herramientas tales como librerías con las cuales se pueden realizar de manera sencilla y práctica múltiples.

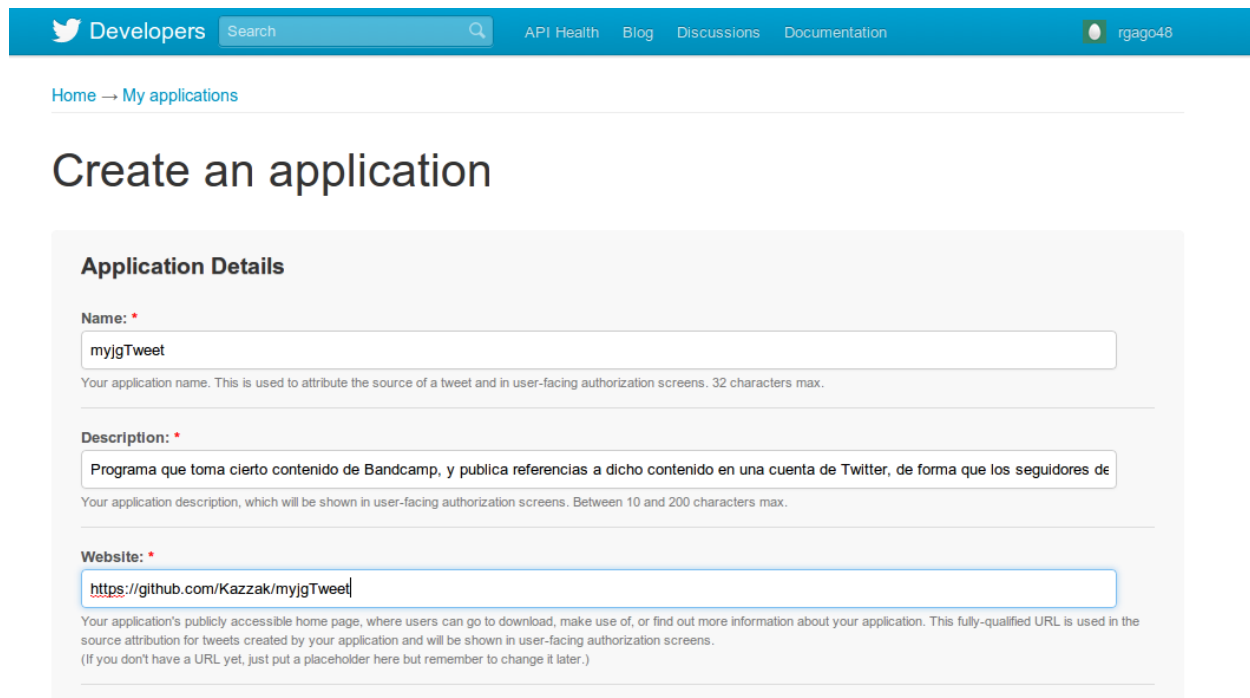
Para obtener más información acerca de este potente lenguaje de programación, puede ingresar la página oficial de Ruby: <http://www.ruby-lang.org/en/>

El programa esta diseñado en dos módulos principales:



Conexión con Twitter

Para poder realizar la conexión con Twitter, son necesarios diversos elementos, entre ellos crear una aplicación en <https://dev.twitter.com/>. Para obtener dicha aplicación, se debe rellenar el formulario que presenta la página.



The screenshot shows the 'Create an application' page on the Twitter Developer portal. The page has a blue header with the Twitter logo, 'Developers' text, a search bar, and links for 'API Health', 'Blog', 'Discussions', and 'Documentation'. The user 'rgago48' is logged in. Below the header, there's a breadcrumb 'Home → My applications' and a large heading 'Create an application'. The main form is titled 'Application Details' and contains three sections: 'Name', 'Description', and 'Website'. Each section has a text input field and a small explanatory text below it. The 'Name' field contains 'myJgTweet'. The 'Description' field contains 'Programa que toma cierto contenido de Bandcamp, y publica referencias a dicho contenido en una cuenta de Twitter, de forma que los seguidores de'. The 'Website' field contains 'https://github.com/Kazzak/myJgTweet'.

Application Details

Name: *

myJgTweet

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description: *

Programa que toma cierto contenido de Bandcamp, y publica referencias a dicho contenido en una cuenta de Twitter, de forma que los seguidores de

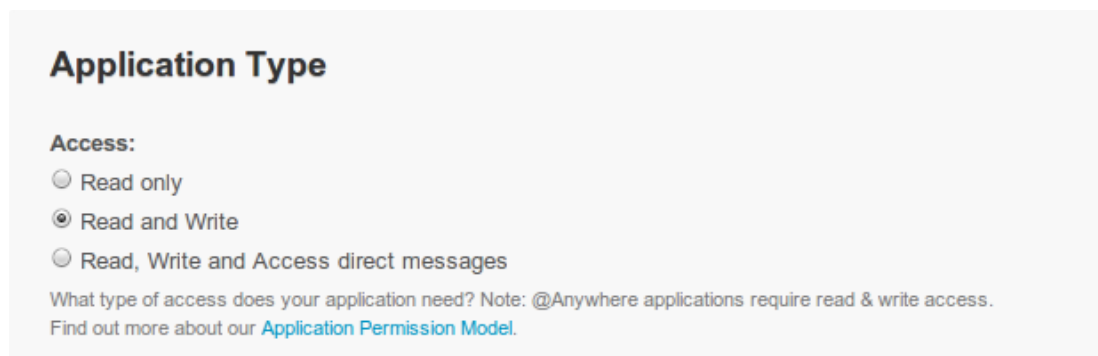
Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website: *

https://github.com/Kazzak/myJgTweet

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Si se rellena el formulario con éxito, la página se re-direccionará a la página de inicio de la aplicación, donde se mostrará datos como *Consumer_key* y *Consumer_secret*, que son necesarios para la conexión. Para finalizar el proceso de creación de la aplicación, se debe ingresar a la pestaña **Settings**, donde es posible modificar la información de la aplicación. En esta pestaña, se debe buscar la sección **Application Type**, y seleccionar *Read and Write*, que corresponden a los permisos de acceso que tendrá la aplicación, y presionar el botón [Update this Twitter Applications settings](#). Luego, se debe volver a la pestaña **Details** y buscar la sección **Your access token**, y presionar el botón para generar el *access token*.



The screenshot shows the 'Application Type' section in the Twitter Developer settings. It has a heading 'Application Type' and a sub-heading 'Access:'. Below this, there are three radio button options: 'Read only', 'Read and Write' (which is selected), and 'Read, Write and Access direct messages'. Below the radio buttons, there is a paragraph of text explaining the importance of the access type and a link to the 'Application Permission Model'.

Application Type

Access:

☐ Read only

☒ Read and Write

☐ Read, Write and Access direct messages

What type of access does your application need? Note: @Anywhere applications require read & write access.
Find out more about our [Application Permission Model](#).

Si se completa el proceso descrito con anterioridad, ya podrá contar con su propia aplicación de Twitter.

El diseño del programa se basa en la definición de una clase llamada `MyjgTweet`, que posee los métodos de `login` y `twitt` y el uso de gems de Ruby, en este caso **`Twitter`** y **`oauth`**, para realizar la conexión. La forma de trabajo de dicho programa consiste en que en el método `initialize`, el constructor de la clase, se encuentran las variables para uso del programa, como la llave a usar y la llave secreta, y una variable para definir la cuenta a la que se ingresó.

La función `login` realiza la petición de datos y solicita los permisos de la cuenta del usuario. En primer lugar, se realiza la conexión con la API de Twitter, mediante el uso de la gem `oauth`, que posee las llaves de acceso al programa, así como el sitio al que se dirigirá, junto con las direcciones a las que solicitara datos para la autenticación.

```
solConexion=OAuth::Consumer.new(  
  @consumer_key,  
  @consumer_secret,  
  {  
    :site=>"http://twitter.com",  
    :request_token_url=>"https://api.twitter.com/oauth/request_token",  
    :access_token_url =>"https://api.twitter.com/oauth/access_token",  
    :authorize_url    =>"https://api.twitter.com/oauth/authorize"  
  }  
)
```

Luego, se solicita el token de acceso secreto y la dirección para confirmar los datos de la cuenta, o si el usuario posee una cuenta de Twitter. La descripción de las variables utilizadas es la siguiente:

- La variable `request_token` obtiene los datos de acceso de la conexión realizada anteriormente.
- `key2` representa el código generado por Twitter que se mostrará en la dirección que el usuario debe seguir y con la que se accede a la cuenta de instancia de Twitter.
- `secret2` representa la nueva clave de acceso a la cuenta y a la aplicación en Twitter.
- `dir` es la dirección que se muestra en pantalla que corresponde al enlace que el usuario debe seguir.
- `pin` es la variable del PIN ingresado por el usuario.

```
request_token = solConexion.get_request_token  
key2 = request_token.token  
secret2 = request_token.secret  
dir = solConexion.authorize_url + "?oauth_token=" + key2  
puts "  "  
puts ("Por favor ingrese al siguiente enlace. Para acceder de  
forma directa mantén presionado la tecla Ctrl y dale un  
click al enlace")
```



```

puts "    "
puts "Enlace: \"#{dir}\""
puts "    "
print ("Por favor ingrese el PIN que la pagina de Twitter genero
para completar el proceso de autorizacion: ")
pin = STDIN.readline.chomp
puts "    "

```

Después de obtener el PIN, se valida el PIN y los datos ingresados por el usuario. Luego de verificar los datos, configura la variable Twitter con los datos de acceso del usuario y crea una nueva instancia de la cuenta, llamada *\$client*. En caso de error, se envía un mensaje de aviso de error.

```

begin
  OAuth::RequestToken.new(solConexion, key2, secret2)
  access_token=request_token.get_access_token(:oauth_verifier => pin)
  Twitter.configure do |config|
    config.consumer_key = @consumer_key
    config.consumer_secret = @consumer_secret
    config.oauth_token = access_token.token
    config.oauth_token_secret = access_token.secret
  end
  $client = Twitter::Client.new
  $client.verify_credentials
  puts "Usuario autenticado correctamente"

rescue Twitter::Unauthorized
  puts "Error de Autorizacion"
end

```

Como punto final, el método *twitt* toma la variable *\$client* y envía una actualización de estado de Twitter. Esta función se encuentra dentro de un ciclo que posee un catch para controlar las excepciones y errores que puedan suceder.

Extracción de los datos

El proceso de extracción de los datos de la página de Bandcamp consiste en un sistema que recibe un tag del usuario, este debe ser por motivos del alcance de este programa un género o una localización (ciudad, país)

Tag == > Costa Rica

El sistema procesará este Tag y generará un *queryString* que conformará un URL el cual es enviado a la página de Bandcamp mediante una librería de Ruby llamada 'open-uri' la cual permite este tipo de conexiones con archivos alojados en algún servidor.

Esta consulta retorna un conjunto de bandas las cuales tiene relación con el Tag dado por el usuario, el sistema obtendrá un URL respectivo a cada banda la extracción del mismo se logra mediante una librería llamada 'hpricot' la cual permite explorar y recuperar información de Tags de HTML embebidos en determinada página web, este proceso también se usará para la extracción de los datos necesarios de la banda (nombre, álbum, pago o descarga gratuita, url Bandcamp).

Extracción de los URL's con respecto a la consulta

Este proceso es llevado a cabo por un módulo creado llamado '*extraeLinks.rb*' el cual recibe la consulta con el Tag dado por el usuario y retorna un array con un máximo de 10 links los cuales fueron dados por el sistema de búsquedas de la página de Bandcamp.

```
#librerias usadas
require 'hpricot'
require 'open-uri'
require 'pp'
```

En la imagen anterior se observa el uso de las librerías mencionadas anteriormente.

```
#proceso de extraccion
var = hp.search("li[@class='item']")
var.each do |x|
  var2 = x.at("a")['href']
  if count < 9
    array+= [var2]
    count+=1
  else
    .
  end
end
```

El módulo Links cuenta con un función llamada recibe la cual toma como parámetro un url, que contiene la consulta con el /tag/ dado por el usuario, Links utiliza algunas utilidades de la librería 'hpricot'

Hpricot(open(url)) > se encarga de abrir el archivo almacenado en la url dada

```
</li><li class="item">

  <a href="http://javierarce.bandcamp.com/album/grabaciones-de-bolsillo" title="grabaciones de bolsillo">
    <div></div>
    <div class="itemtext">grabaciones de bolsillo</div>

    <div class="itemsubtext">Javier Arce</div>

  </a>
</li><li class="item">
```

Como se muestra en la imagen anterior un link que se quiere recuperar se encuentra en los entre los tags

 class = item

que a su vez se encuentra entre el Tag

los cuales se extraen en las siguientes líneas respectivamente.

```
search("li[@class = 'item']") > se encarga de buscar el Tag específico
at("a")["href"]
```

Luego de que se hayan obtenido los links respectivos del Tag dado por el usuario cada link es procesado de manera individual para obtener la información de la banda. Este proceso se lleva a cabo en un módulo llamado 'extraeInfo.rb' el cual se describe a continuación.

Para explicar su funcionamiento bastará solo explicar la extracción de un dato ya que el proceso es homogéneo para los otros.

```
#extracción de requerimiento de pago
if (pago = hp.at("h4[@class='ft compound-button']"))
  pago = pago.at("a")
  pago = pago.search("").inner_html
else
  pago = "unknown"
end
```

Para esto igual que en el procedimiento explicado para obtener los links se utilizar la librería 'hpricot' de la siguiente forma.

```
<h4 class="ft compound-button">

  <a id='noJSDownloadLink' href="/download_tralbum">Buy Now</a>

  &nbsp;$2
  <span class="buyItemExtra secondaryText">USD</span>
  <span class="buyItemExtra buyItemNyp secondaryText">&nbsp;or more</span>

</h4>
```

Como se observa en el código para saber si el álbum es de descarga libre o es de pago, esa información se encuentra entre los Tags

```
<h4 ..... /h4>
```

y dentro de ese Tag se encuentra en Tag

```
< a ..... >
```

para obtener la información en esos Tags se utilizan las siguientes líneas respectivamente

```
at("h4[@class='ft compound-button']"))
pago = pago.at("a")
```

Ya con los procedimientos anteriores se obtiene la información de la banda lista para ser publicada en Twitter.

Para el manejo de la información de las bandas se creó la clase banda.

```
class Banda
  #metodo constructor
  def initialize(nombre, album, pago, link)
    @nombre = nombre
    @album = album
    @pago = pago
    @link = link
  end
```

Para el manejo de la consulta se creó la clase consulta

```
class Consulta
  #metodo constructor
  def initialize(url)
    @info = []
    @url = url
    @links = Links.recibe(url)
    @links.each do |subLink|
      var = Extractor.recibe(subLink)
      temp = Banda.new(var[0], var[1], var[2], var[3])
      @info+= [temp]
    end
  end
end
```

Para el uso de los módulos anteriores se usa la función load 'nameModule'

```
#modulos de trabajo
load 'extraeLinks.rb'
load 'extraeInfo.rb'
```

Ya con esto se crea un array el cual es llenado con objetos de tipo banda, el cual es iterado por una función la cual toma cada banda y genera un Tweet con la información respectiva.

Información de la librería usada en la extracción de los datos

Para la instalación de la misma se utiliza el siguiente comando

```
gem install hpricot
```

Para obtener más información de esta librería puede visitar la página <http://rubygems.org/gems/hpricot>

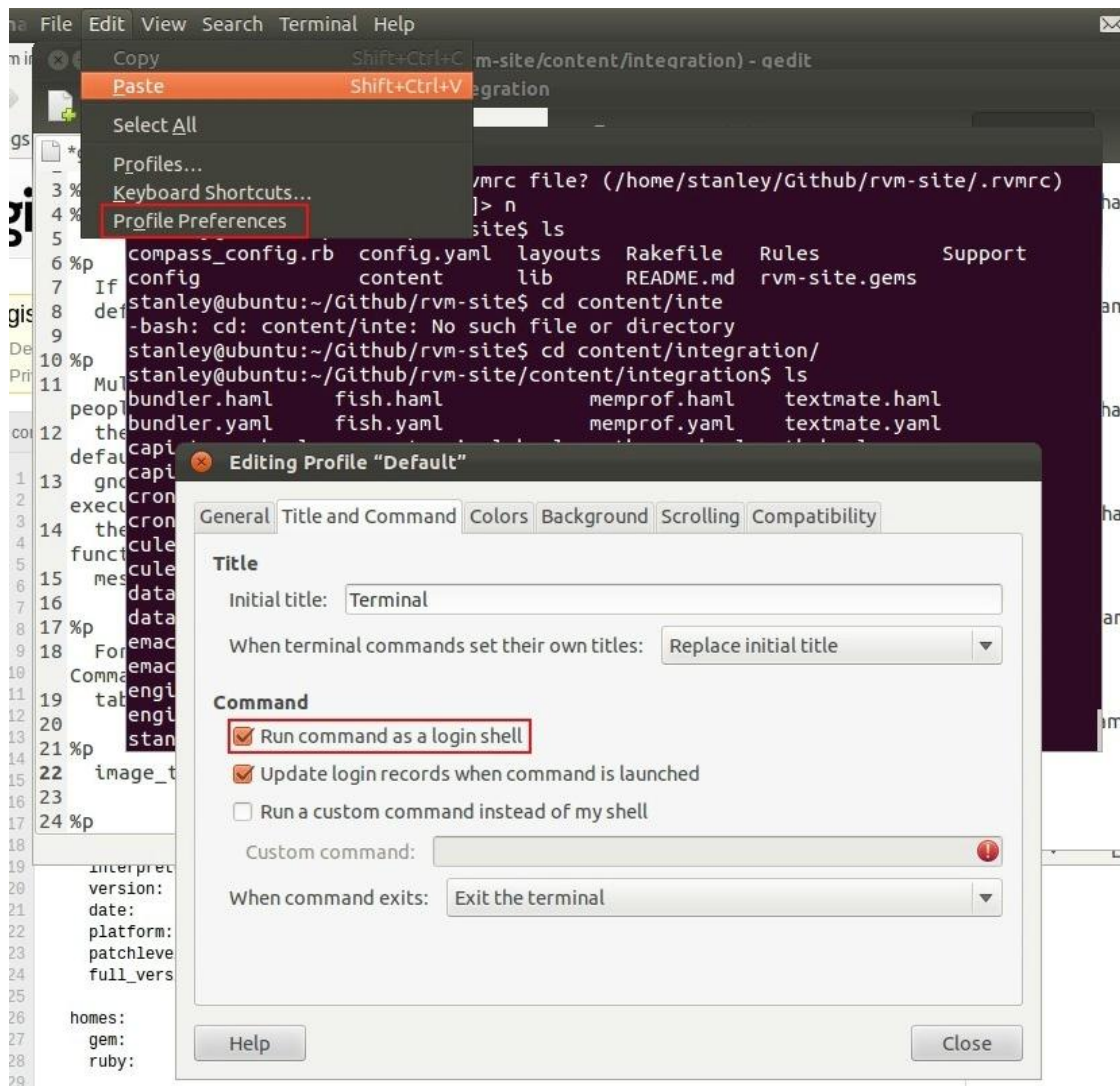
Análisis de resultados

Tarea	Estado	Descripción
Consulta de usuario	Completa	Recibe la consulta del usuario y la realiza en la página de bandcamp.com.
Obtener links de Bandcamp	Completa	Se obtienen como máximo 10 links que retorna Bandcamp a la consulta enviada.
Obtener la información de la banda	Completa	Se obtiene la información de la banda de cada links obtenido <ul style="list-style-type: none">• nombre• álbum• precio o gratis• url de Bandcamp
Realizar tweet en la cuenta de Twitter	Completa	Se realizan un máximo de 10 tweets con la información de las bandas de acuerdo con el Tag ingresado por el usuario.

Manual de usuario

Etapas de instalación

1. En terminal escribir:
 - a. `sudo apt-get install aptitude && sudo apt-get update`
 - b. `sudo aptitude install build-essential libssl-dev libreadline5 libreadline5-dev zlib1g zlib1g-dev`
 - c. Instalar RVM (Ruby enVironment Manager): `curl -L get.rvm.io | bash -s stable --ruby --rails`
2. Para que RVM funcione correctamente, tienes que establecer el comando "Run command as a login shell", en Edit - Profile Preferences – Title and Command, en la terminal de gnome-terminal, de esta manera:



3. Cerrar la terminal y abrir una nueva.
4. Luego abrir una nueva terminal e ingresar
 - a. gem install twitter
 - b. gem install oauth

```
dunix@dunix:~$ gem install oauth
WARNING: Error fetching data: too many connection resets (http://rubygems.org/latest_specs.4.8.gz)
Successfully installed oauth-0.4.6
1 gem installed
Installing ri documentation for oauth-0.4.6...
Installing RDoc documentation for oauth-0.4.6...
dunix@dunix:~$
```

```
dunix@dunix:~$ gem install twitter
WARNING: Error fetching data: too many connection resets (http://rubygems.org/latest_specs.4.8.gz)
WARNING: Error fetching data: too many connection resets (http://rubygems.org/specs.4.8.gz)
Successfully installed twitter-2.5.0
1 gem installed
Installing ri documentation for twitter-2.5.0...
Installing RDoc documentation for twitter-2.5.0...
dunix@dunix:~$
```

Ejecución del programa

1. En primera instancia descargar el programa de github.com.

```
dunix@dunix:~$ git init
Reinitialized existing Git repository in /home/dunix/.git/
dunix@dunix:~$
```

```
dunix@dunix:~$ git clone git@github.com:Kazzak/myjgTweet.git
Initialized empty Git repository in /home/dunix/myjgTweet/.git/
remote: Counting objects: 27, done.
remote: Compressing objects: 100% (24/24), done.
Receiving objects: 100% (27/27), 6.36 KiB, done.
Resolving deltas: 100% (9/9), done.
remote: Total 27 (delta 9), reused 17 (delta 2)
dunix@dunix:~$
```


2. Luego ingresar al directorio del programa que recién se ha creado a la hora de clonar el directorio

```
dunix@dunix:~$ ls
Descargas  Escritorio  Imágenes  myjgTweet  Público  Videos
Documentos examples.desktop Música  Plantillas Ubuntu One
dunix@dunix:~$ cd myjgTweet/
dunix@dunix:~/myjgTweet$ ls
bandaInfo.rb  extraeInfo.rb  extraeLinks.rb  haceTweet.rb  MyjgTweet.rb  principal.rb  README.md
dunix@dunix:~/myjgTweet$
```

3. Luego para ejecutar el programa ingresar en consola *ruby principal.rb* y seguir las instrucciones tal como se muestra en la siguiente imagen.

```
dunix@dunix:~/myjgTweet$ ruby principal.rb
ingrese la consulta: mana

Por favor ingrese al siguiente enlace. Para acceder de
forma directa mantén presionado la tecla Ctrl y dale un
click al enlace

Enlace: "https://api.twitter.com/oauth/authorize?oauth_token=B3bvjmM9DpnLnXhLsl1jH38fLXpzfbl2T2sbrMnNk
"

Por favor ingrese el PIN que la página de Twitter generó
para completar el proceso de autorización:
```

4. El enlace direcciona a una página como esta:



The screenshot shows the Twitter authorization interface for the application 'myjgTweet'. The header includes the Twitter logo and a 'Regístrate' link. The main heading is 'Authorize myjgTweet to use your account?'. Below this, it states 'Esta aplicación será capaz de:' followed by a list of permissions: 'Leer Tweets de tu cronología.', 'Ver a quién sigues y seguir a nuevas personas.', 'Actualizar tu perfil.', and 'Publicar Tweets por ti.' There are input fields for 'Nombre de usuario o correo electrónico' and 'Contraseña', along with a checkbox for 'Recuerda mis datos' and a link for '¿Olvidaste tu contraseña?'. At the bottom, there are two buttons: 'Autorizar la aplicación' (highlighted in blue) and 'No, gracias'. A section at the bottom states 'Esta aplicación no tendrá capacidad para:' followed by 'Acceder a tus mensajes directos.' and 'Ver tu contraseña de Twitter.' On the right side, there is a profile section for 'myjgTweet' with a gear icon, the text 'Por Tecnológico de Costa Rica', the GitHub link 'github.com/Kazzak/myjgTweet', and a description: 'Programa que toma cierto contenido de Bandcamp, y publica referencias a dicho contenido en una cuenta de Twitter, de forma que los seguidores de dicha cuenta puedan ver la información de las bandas.'

5. Luego de ingresar los datos de usuario aparecerá la siguiente página con un código que Twitter generará para validar y continuar con el proceso de autorización. Dicho código o PIN debe ingresarlo en la terminal, tal como se muestra en imágenes anteriores.

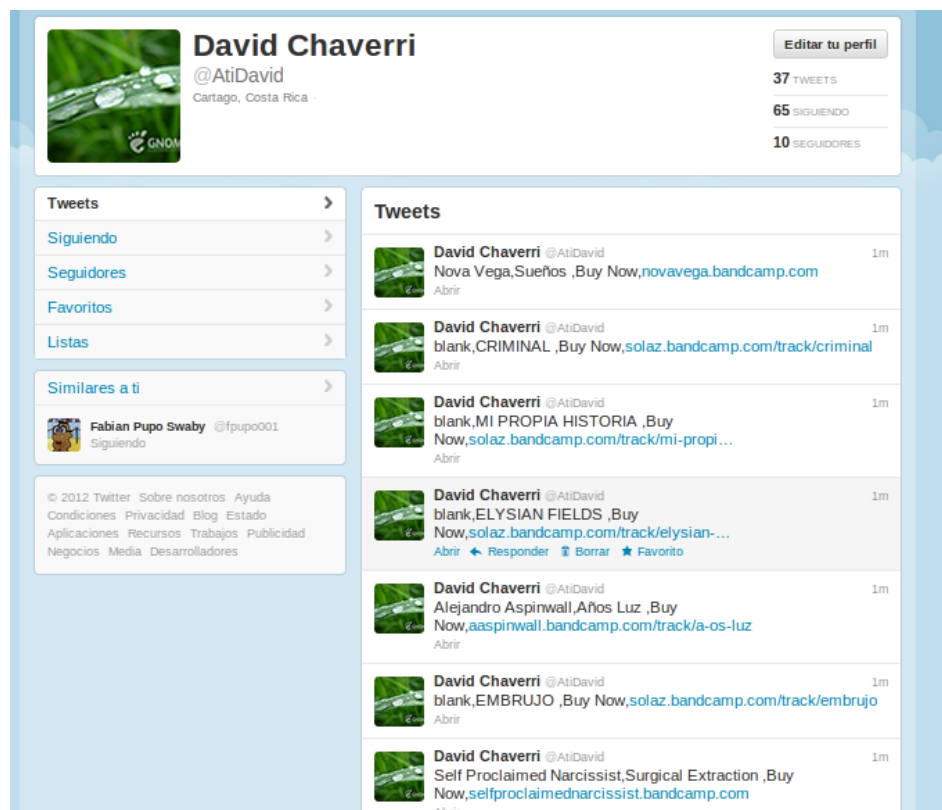


```
Enlace: "https://api.twitter.com/oauth/authorize?oauth_token=B3bvjmM9DpnLnXhLsljH38fLXpzfbl2T2sbrMnNk"
Por favor ingrese el PIN que la pagina de Twitter genero
para completar el proceso de autorizacion: 2039592
```

6. Luego de ingresar el PIN se validará éste y los datos ingresados. Cuando la aplicación muestre el mensaje “Usuario autenticado correctamente”, ya habrá realizado los tweets en la cuenta del usuario.

```
Por favor ingrese el PIN que la pagina de Twitter genero
para completar el proceso de autorizacion: 2039592

Usuario autenticado correctamente
dunix@dunix:~/myjgTweets
```



Conclusiones

En general el lenguaje Ruby resultó ser muy práctico debido a la gran cantidad de funciones que tiene y a la sintaxis muy sencilla y natural.

Se recomienda a la hora de tener que desarrollar una aplicación en objetos utilizar Ruby.

Se cuenta con una gran cantidad de gemas (librerías) las cuales son fáciles de instalar y otorgan gran cantidad de funcionalidades que pueden ser usadas para hacer múltiples tareas.

Se recomienda a la hora de tener que implementar una tarea un Ruby, antes de todo investigar si hay alguna función que realiza alguna función que nos puede ayudar a la hora de realizar la tarea.

A la hora de utilizar la librería 'oauth', las funciones que trae para autenticación son muy prácticas.

Se recomienda leer la documentación oficial de 'oauth' para aprovechar mejor sus funciones.

La librería '*hpricot*' resultó bastante útil para trabajar con los distintos Tags de los archivos HTML, lo cual facilitó mucho la búsqueda de la información necesaria en la página de <http://bandcamp.com/>.

Se recomienda utilizar esta librería a la hora de tener que trabajar con información como `inner_text` embebido en Tags HTML

Las implementaciones de clases en Ruby son muy prácticas, el manejo de atributos, métodos y constructores resulta bastante sencillo.

Se recomienda leer la documentación oficial acerca de la creación de las clases.

Bibliografía

- Introducción - Ruby Tutorial (s. f.). Recuperado de <http://rubytutorial.wikidot.com/introduccion>
- Módulos - Ruby Tutorial (s. f.). Recuperado de <http://rubytutorial.wikidot.com/modulos>
- Query string - Wikipedia, the free encyclopedia (s. f.). Recuperado de http://en.wikipedia.org/wiki/Query_string
- Ruby - Open an IO stream from a local file or url - Stack Overflow (s. f.). Recuperado de <http://stackoverflow.com/questions/263536/open-an-io-stream-from-a-local-file-or-url>
- Ruby: Net::Http and open-uri - juretta.com (s. f.). Recuperado de http://juretta.com/log/2006/08/13/ruby_net_http_and_open-uri/
- gasman's gist: 524376 ? Gist (s. f.). Recuperado el 16 de Junio del 2012, de <https://gist.github.com/524376>
- RVM: Ruby Version Manager - Using gnome-terminal with rvm (s. f.). Recuperado el 16 de Junio del 2012, de <https://rvm.io/integration/gnome-terminal/>
- Ruby OAuth GEM (s. f.). Recuperado el 16 de Junio del 2012, de <http://oauth.rubyforge.org/>
- 401 error con Ruby OAuth para Twitter (s. f.). Recuperado el 16 de Junio del 2012, de <http://www.stack-es.com/stackoverflow/es/401-error-with-ruby-oauth-for-twitter-3552711.html>
- Keep getting OAuth::Unauthorized error when using oauth and twitter ruby gems - Stack Overflow (s. f.). Recuperado el 16 de Junio del 2012, de <http://stackoverflow.com/questions/1280295/keep-getting-oauthunauthorized-error-when-using-oauth-and-twitter-ruby-gems>
- Stack level too deep (SystemStackError) while using class_eval ruby - Stack Overflow (s. f.). Recuperado el 16 de Junio del 2012, de <http://stackoverflow.com/questions/8112997/stack-level-too-deep-systemstackerror-while-using-class-eval-ruby>
- The Twitter Ruby Gem (s. f.). Recuperado el 16 de Junio del 2012, de <http://twitter.rubyforge.org/>