



**UNIVERSIDAD DE QUINTANA ROO**  
**DIVISIÓN DE CIENCIAS E INGENIERÍA**

---

**SISTEMA DE ADMINISTRACIÓN DEL ACERVO  
FOTOGRAFICO DE LA UNIVERSIDAD DE  
QUINTANA ROO**

---

**TESIS  
PARA OBTENER EL GRADO DE  
INGENIERO EN REDES**

**PRESENTA  
ALEXIS GERARDO RÁBAGO CARVAJAL**

**DIRECTOR DE TESIS  
DR. JAIME SILVERIO ORTEGÓN AGUILAR**

**ASESORES  
DR. HOMERO TORAL CRUZ  
MSI. RUBÉN ENRIQUE GONZÁLEZ ELIXAVI  
DR. JAVIER VÁZQUEZ CASTILLO  
MTI VLADIMIR VENIAMIN CABAÑAS VICTORIA**



**UNIVERSIDAD DE  
QUINTANA ROO**  
**SERVICIOS ESCOLARES  
TITULACIONES**

CHETUMAL QUINTANA ROO, MÉXICO, NOVIEMBRE DE 2015



**UNIVERSIDAD DE QUINTANA ROO**  
**DIVISIÓN DE CIENCIAS E INGENIERÍA**

**TRABAJO DE TESIS ELABORADO BAJO SUPERVISIÓN DEL  
COMITÉ DE ASESORÍA Y APROBADO COMO REQUISITO  
PARCIAL PARA OBTENER EL GRADO DE:**

**INGENIERO EN REDES**

**COMITÉ DE TRABAJO DE TESIS**

**DIRECTOR:**

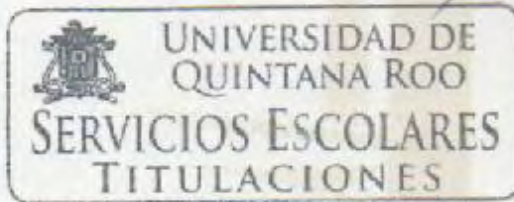
**DR. JAIME SILVERIO ORTEGÓN AGUILAR**

**ASESOR:**

**DR. HOMERO TORAL CRUZ**

**ASESOR:**

**MSL. RUBÉN ENRIQUE GONZÁLEZ ELIXAVIDE**



CHETUMAL QUINTANA ROO, MÉXICO, NOVIEMBRE DE 2015

## Agradecimientos

Durante los 7 años de carrera dentro de la Universidad de Quintana Roo, conocí muchas personas. Personas que se convirtieron en compañeros y amigos. A ellos y muchos mas les agradezco el apoyo brindado desde el comienzo, sobre todo a mis padres y familia quienes me apoyaron y ayudaron durante dichos años por que sin ellos y la voluntad de seguir adelante para conseguir superación y este logro tan grande, no habría llegado tan lejos. Todo este periodo me ha dejado gran aprendizaje no solo técnico si no humano y personal, lograr algo que muchos otros han hecho pero al mismo tiempo saber que me uno y formo parte de los que lo lograron.

Agradezco de igual manera a los profesores ya que sin su paciencia, conocimiento y gran capacidad nada de esto habría sido posible, en especial a mi tutor M.T.I. Vladimir Veniamin Cabañas Victoria, a mi director de tesis Dr. Jaime Silverio Ortegón Aguilar, nuestros maestros de redes M.S.I. Dávalos Castilla Laura Yésica y M.S.I González Elixavide Rubén Enrique, El profesor Dr. Toral Cruz Homero quien me impartió múltiples clases, a la maestra M.T.I. Blanqueto Estrada Melissa ya que sin su guía no habría incursionado en la programación de manera profesional y a todos los profesores de quienes recibí conocimiento con plenitud y dedicación hasta terminar con el proceso de formación profesional.

Agradezco finalmente a la Universidad de Quintana Roo por permitirnos aprender y compartir dentro de sus instalaciones, la biblioteca cuyo espacio siempre nos dio un lugar para desarrollar nuestras capacidades. Los laboratorios, un espacio para realizar prácticas y mejorar nuestro trabajo en equipo. A los apoyos ofrecidos tanto financieros como materiales y a la accesibilidad del personal administrativo.

## Resumen

A la fecha, la Universidad de Quintana Roo cuenta con un recopilado de fotografías tomadas desde su fundación, en 1991. Esto ha generado un problema, ya que tanto el archivo físico, como el digital, no cuentan con una manera fácil y rápida para solicitar copias u originales, todo este proceso conlleva buscar dentro de distintos medios de almacenamiento convirtiéndolo en una tarea larga, tediosa y poco eficiente.

Actualmente, el departamento de difusión es el encargado del manejo del archivo histórico, de uso cotidiano, así como del nuevo material gráfico, constantemente agregado a los medios de almacenamiento, lamentablemente no cuentan con un sistema centralizado para el manejo de este material. El proceso de recuperación de imágenes es lento, enredado y puede incluso derivar en su pérdida accidental o degradado en calidad. Lo anterior debido al factor humano en cada paso desde recibir o enviar fotografías al encargado de la administración hasta la edición y clasificación del mismo en los respectivos medios de almacenamiento a los que van destinados.

En la presente tesis se describe el proceso de desarrollo y documentación básica de una aplicación web desarrollada utilizando el Frameworks “Ruby on Rails” para el manejo y administración de las fotografías del departamento de difusión de la Universidad de Quintana Roo. Todo esto con el fin de lograr un medio centralizado, de fácil uso y acceso tanto local como a través de Internet conectando los encargados de difusión de los diferentes planteles distribuidos dentro de la entidad.

**Contenido**

Agradecimientos.....	3
Resumen.....	4
<b>CAPÍTULO 1. INTRODUCCIÓN .....</b>	<b>8</b>
Justificación .....	10
Objetivo General .....	11
Objetivos Particulares.....	11
Alcance .....	12
<b>CAPÍTULO 2. MARCO TEÓRICO.....</b>	<b>13</b>
MVC (Modelo Vista Controlador) .....	14
Ruby y Ruby on Rails.....	14
SASS (Syntactically Awesome Style Sheets).....	15
Don't Repeat Yourself.....	15
HTML, HAML y ERB (Hypertext Markup Language) .....	16
HAML.....	17
ERB .....	17
C.S.S. y SASS (Cascading Style Sheets) .....	18
SASS.....	18
JavaScript y CoffeeScript .....	19
NGINX.....	20
Unicorn.....	21
ImageMagick .....	22
<b>CAPÍTULO 3. IMPLEMENTACIÓN .....</b>	<b>25</b>
Diagrama de clases .....	25
Controladores.....	30
Disposición .....	31
<b>CAPÍTULO 4 .....</b>	<b>39</b>
Pruebas de funcionamiento .....	39
<b>CAPÍTULO 5 .....</b>	<b>44</b>
CONCLUSIONES .....	44
REFERENCIAS BIBLIOGRÁFICAS.....	46

Anexos .....47

# CAPÍTULO 1

## CAPÍTULO 1. INTRODUCCIÓN

*“El resultado más imponente de la empresa fotográfica es darnos la sensación de que podemos apresar el mundo entero en nuestras cabezas, como una antología de imágenes. Coleccionar fotografías es coleccionar el mundo”.- La fotografía como documento social - Freund, Gisele (1976).*

Tras su fecha de fundación en 1991, con el entonces Rector Enrique Carrillo Barrios Gómez (fecha antes mencionada), la Universidad de Quintana Roo cuenta con aproximadamente veinticuatro años laborando, durante los cuales el Departamento de Difusión desde su fundación ha publicado más de cincuenta números de la Gaceta Universitaria, cuya principal función es difundir y dar a conocer las múltiples actividades de la vida universitaria, con evidencia fotográfica y escritos de calidad los cuales dan a conocer a detalle estas mismas actividades, cabe mencionar que el material fotográfico almacenado va desde negativos, placas y originales únicos hasta contenido digital almacenado en diferentes dispositivos electrónicos algunos datando desde el año 2006.

Sabemos que a la fotografía se le considera una manera de fijar en una placa impresionable a la luz de imágenes obtenidas con ayuda de una cámara. Inicialmente como una habitación, cuya única fuente de luz era un minúsculo orificio en una de las paredes. La luz penetraba por aquel orificio y proyectaba una imagen del exterior en la pared opuesta. Aunque la imagen así formada resultaba invertida y borrosa, los artistas utilizaron esta técnica, mucho antes de que se inventase la película para esbozar escenas proyectadas por la cámara. Con el transcurso de los años la cámara oscura evolucionó y se convirtió en una pequeña caja manejable y al orificio se le instaló un lente óptico para conseguir una imagen más clara y definida, hasta llegar a la cámara digital utilizada en un amplio número de situaciones.

Dentro de los métodos de almacenamiento más utilizados actualmente se encuentran tanto digitales y digitales en “la nube”. Los métodos de almacenamiento digital más comunes son discos duros, memorias USB, Discos DVD, CD y ahora de modo comercial los Blue Ray Disc, lamentablemente estos medios solamente son viables para cantidades de información limitada, tanto por tamaño en megabytes como por cantidad de archivos que fácilmente pueden escalar de manera exponencial cuando se maneja por varias personas. Algunos programas o software ofrecen una solución a este incremento exponencial de archivos proveyendo de una interfaz para catalogar y organizar las fotografías con un mayor detalle utilizando



etiquetas, metadatos, etc. Lamentablemente estos programas o software no solucionan el problema cuando la catalogación se extiende más allá de una computadora, ya que el sistema organiza conforme a las preferencias configuradas por el usuario manejando el dispositivo de manera local. Algunos ejemplos de software para catalogación local son Picasa, Photoscape y Microsoft Photo Story. Para solventar estas limitantes a las cuales nos restringe el software en las computadoras locales existen los métodos de almacenamiento en la nube, estos con ayuda de conexión a red ya sea local o por internet, ofrecen un servicio que centraliza el manejo y catalogación de las fotografías haciendo posible el acceso desde diferentes puntos sin la necesidad de instalar una aplicación específica (en la mayoría de los casos). El único requisito es tener una cuenta y en caso de ser un servicio de paga, comprar dicho servicio, muchos de estos solamente limitando la capacidad en megabytes fácilmente extensible y sin la necesidad de reconfigurar los dispositivos para tener nuestras preferencias de catalogación. Algunos ejemplos de Software de almacenamiento en la nube son One Drive de Microsoft, Google Drive, Picasa Online, Flickr, Dropbox y Everpix.

Motivados por lo anterior, en la presente tesis se describe el proceso de desarrollo, implementación y prueba de una aplicación web destinada a recibir el material fotográfico desde los diferentes planteles con los que cuenta la Universidad de Quintana Roo (Unidad Académica Cozumel, Unidad Académica Chetumal y Unidad Académica Playa del Carmen), a realizar una catalogación automática (dependiendo del plantel, fecha, etc.), permitir la búsqueda y acceso a dicho material por parte de cualquier usuario con las credenciales pertinentes desde distintas locaciones con acceso a la red de la universidad. Todo esto fue logrado utilizando programación orientada objetos en Ruby on Rails, un Frameworks de tipo MVC (Modelo Vista Controlador).

## **Justificación**

Tomando como base el registro fotográfico con el cual cuenta la Universidad desde su fundación en el año de 1991 podemos acreditar la importancia de la conservación de dicho material. Así mismo, es importante su recopilación, catalogación y almacenamiento, ya que no solamente es depositado tras su captura, si no se accesa y utiliza para diferentes propósitos. Esto nos lleva a la problemática de encontrar un medio versátil y que a su vez de total control sobre dicho material, ya que en la actualidad la Universidad, en específico el Departamento de Difusión encargado de todo este proceso, cuenta solamente con un medio manual y esparcido en diferentes dispositivos de almacenamiento asignados para este fin.

## Objetivo General

El objetivo principal fue desarrollar un sistema de catalogación para el material fotográfico y visual de la Universidad de Quintana Roo, esto permitirá mejorar la eficiencia del trabajo realizado dentro del Departamento de Difusión en los múltiples planteles del estado.

## Objetivos Particulares

- **Catalogar:** Al subir las fotografías se elige una carpeta para una catalogación eficiente.
- **Almacenar:** El sistema almacena las imágenes en un árbol de carpetas idéntico al generado mediante la GUI (Graphic User Interface).
- **Recuperar:** El Sistema cuenta con acceso a descargar las imágenes en su formato original y los sub formatos generados por el post procesamiento.
- **Etiquetar:** Al subir las imágenes el sistema realiza un auto etiquetado predefinido, estas etiquetas son guardadas a nivel de metadatos incrustados en la imagen principal y base de datos.
- **Facilitar la gestión desde las diversas sedes:** El sistema de GUI permite generar carpetas, estas carpetas pueden ser utilizadas por diferentes usuarios, esto permite asignarles áreas de trabajo específicas.
- **Implementar un sistema de mensajes:** Internamente se tiene un sistema de mensajes que permite comunicación entre los diferentes usuarios.

## Alcance

El sistema cumple como principal función, catalogar todo el material digital cargado a través de la interfaz gráfica, este material al subirse o cargarse de manera automática conforme a la estructura de carpetas también definida por los usuarios, es automáticamente guardado dentro de dichas carpetas, se generan diferentes versiones redimensionadas a varios tamaños predefinidos en pixeles y con marca de agua, así mismo como uno de los pasos fundamentales se extraen los metadatos de la imagen original, les son agregadas algunas etiquetas, se guardan en la base de datos y finalmente son entregadas a través del GUI para también ser manejadas, ya sea cortadas, copiadas, pegadas, borradas y también filtradas a través del buscador. También es posible agregar nuevas etiquetas en ambos niveles (metadatos y base de datos).

El sistema está pensado para eliminar todo el proceso de recopilado de material gráfico de la Universidad de Quintana Roo llevado a cabo desde diferentes puntos dentro de la universidad el cual como anteriormente se describió, es propenso a errores humanos o de sistemas no controlados por la Universidad de Quintana Roo. Siendo el primero propenso a errores por factor humano, ya sea el extravió o degradado. Todo esto permitirá el acceso desde diferentes puntos en la red universitaria o incluso planteles dentro del estado.

En el capítulo dos se describe el marco teórico el cual incluye información sobre el Frameworks utilizado MVC y Ruby on Rails, el lenguaje de programación, el software y las diferentes tecnologías involucradas.

En el capítulo tres se describe la implementación del sistema, como fue desarrollado y aplicado dentro de un servidor para que el Departamento de Difusión de la Universidad pudiera utilizarlo.

En el capítulo cuatro se describe la fase de pruebas, es decir, el período de pruebas experimental una vez implementado el sistema dentro de los servidores.

El capítulo cinco consta de las conclusiones sobre el desarrollo de la aplicación, su implementación y el período de pruebas.

# CAPÍTULO 2

## CAPÍTULO 2. MARCO TEÓRICO

El Sistema a continuación fue desarrollado con una variedad de tecnologías bastante amplia, incluyendo desde lenguajes para páginas web hasta servidores tanto de aplicación, como web y librerías para edición de fotografías, todos descritos a continuación.

### MVCi (Modelo Vista Controlador)

MVC es una propuesta de diseño de software para implementar sistemas donde se requiere el uso de interfaces de usuario, surge de la necesidad de crear software más robusto, con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización de código y separación de conceptos. (desarrolloweb, 2015). En la Figura 1 - MVC se presenta las relaciones que se dan entre los modelos, los controladores y las vistas.

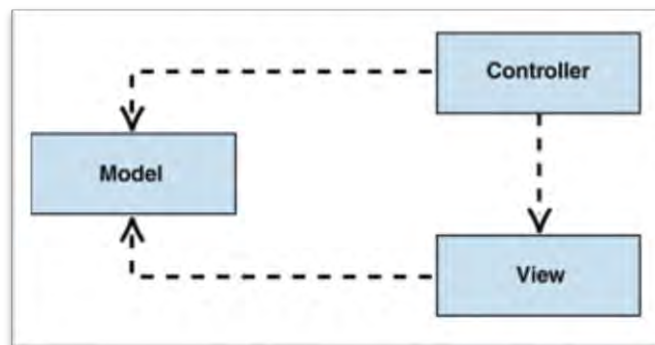


Figura 1 - MVC

### Ruby y Ruby on Rails<sup>ii</sup>

Ruby on Rails es un Frameworks para desarrollo web que utiliza el lenguaje de programación Ruby<sup>iii</sup>, este lenguaje creado en 1993 por el programador japonés Yukihiro Matsumoto es un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad, en la Figura 2 – Ruby se puede observar un ejemplo de código Ruby.

```
1 class Fred
2   def initialize(v)
3     @val = v
4   end
5
6   def set(v)
7     @val = v
8   end
9
10  def get
11    return @val
12  end
13 end
```

Figura 2 – Ruby

Comparte funcionalidad con otros lenguajes como Lisp, Lua, Dylan y CLU, esto lo convierte en un lenguaje de alto nivel muy fácil de aprender y desarrollar con él. El Framework utiliza una variedad de tecnologías por default bastante amplia, estas incluyen:

- JavaScript<sup>iv</sup>
- HTML<sup>v</sup> (HyperText Markup Language)
- CSS<sup>vi</sup> (Cascading Style Sheets)
- CoffeeScript<sup>vii</sup>
- ERB<sup>viii</sup> (Embended Ruby)
- SASS<sup>ix</sup> (Syntactically Awesome Style Sheets)

Como muchos otros Frameworks tienen la posibilidad de expandirse a muchas otras como:

- HAML<sup>x</sup> (HTML abstraction markup language)
- Backbone<sup>xi</sup>
- LESS<sup>xii</sup>

Por la simplicidad que asume trabajar con Ruby on Rails, se trabaja bajo dos principales conceptos:

**Don't Repeat Yourself<sup>xiii</sup>:** DRY (no te repitas) es un principio de desarrollo de software el cual dicta que toda pieza de conocimiento tiene una sola ambigua y autoritaria fuente dentro de un sistema, esto funciona para no repetir código una

cantidad de veces indefinida, esto permite que el código sea más fácil de mantener, extender y sea menos propenso a errores.

**Convention over Configuration:** (convención sobre configuración) Rails cuenta con una variedad de formas predefinidas para hacer todo tipo de cosas, también llamados estándares, estos estándares permiten un desarrollo simple en lugar de realizar multitud de configuraciones a lo largo de una infinidad de archivos múltiples veces (Guides, 2015), la Figura 3 – Ruby on Rails a continuación corresponde a una toma de pantalla de la página oficial de Ruby on Rails.

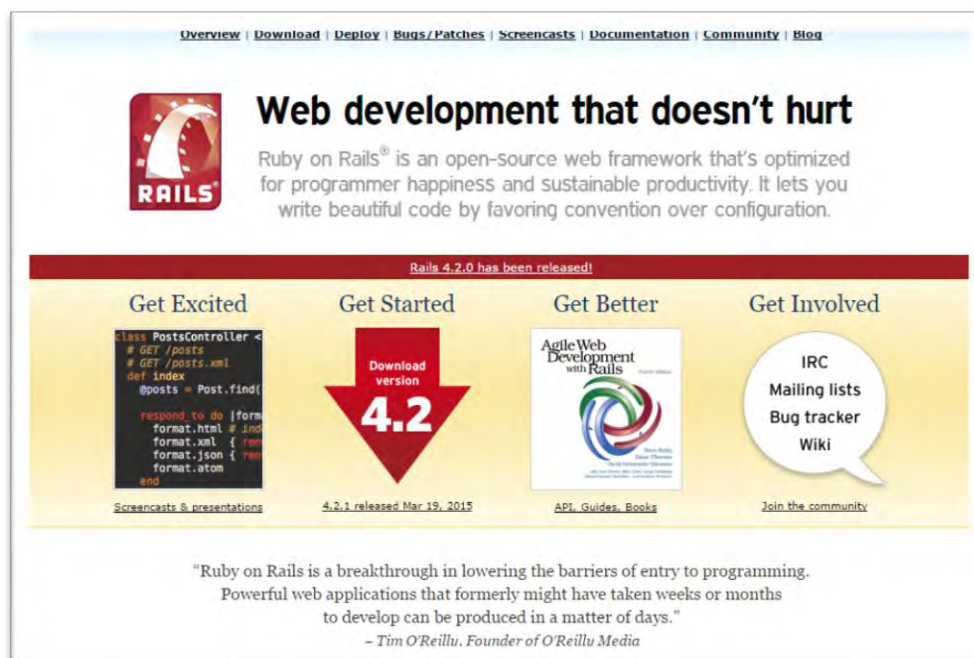


Figura 3 – Ruby on Rails

## HTML, HAML y ERB (Hypertext Markup Language)

HTML es un lenguaje para describir la estructura de las páginas web, este da a los autores la posibilidad de:

- Publicar documentos en línea con encabezados, textos, tablas, listas, fotografías, etc.
- Conseguir información en línea a través de enlaces de hipertexto con un clic.



- Diseñar formas para conducir transacciones con servicios remotos para buscar información.
- Incluir hojas externas, videos, sonido y otras aplicaciones directamente sobre sus documentos.

**HAML** es un lenguaje de marcado que trans-compila hacia HTML, este permite un código más simple a base de indentación jerárquica cuya principal característica es el indentado a 2 espacios y la capacidad de utilizar código Ruby más simple de igual manera. Fue hecho bajo el principio de que el lenguaje de marcado debe ser “bonito” (Clarke, 2015), a continuación en la Figura 4 - HTML/HAML un ejemplo de código HTML traducido a HAML.

```
1 HTML
2 <strong class="code" id="message">Hello, World!</strong>
3
4 HAML
5 %strong{:class => "code", :id => "message"} Hello, World!
```

Figura 4 - HTML/HAML

**ERB** es una característica de Ruby, la cual permite convenientemente generar cualquier clase de texto desde plantillas. Las plantillas combinan texto plano con código Ruby para la sustitución de variables y control de flujo, haciéndolo fácil de escribir y mantener.

A pesar de que ERB es principalmente utilizado para hacer páginas web, puede producir cualquier clase de texto ya sea XML, RSS feeds, código fuente, y otras formas de texto estructurado. Es extremadamente valioso cuando se trata de producir muchas repeticiones de un patrón estándar como pruebas unitarias.

El componente principal de ERB es una librería que puede llamarse dentro de las aplicaciones de Ruby, esta librería acepta cualquier cadena (string) como plantilla y no le impone limitaciones, se puede generar desde el código o un archivo externo, esto significa que se pueden mantener plantillas en archivos, bases de datos o cualquier otra clase de dispositivo de almacenamiento (Ellis, 2015) , en la Figura 5 – ERB se encuentra el ejemplo de un archivo ERB con código Ruby embedido.

```
1 <ul>
2   <% for @item in @shopping_list %>
3     <li><%= @item %></li>
4   <% end %>
5 </ul>
```

Figura 5 – ERB

## C.S.S. y SASS (Cascading Style Sheets)

CSS es el lenguaje para describir la presentación de una página web, incluye colores, marco y fuentes. Le permite al diseñador adaptar la presentación a diferentes tipos de dispositivos como pantallas grandes, pequeñas o impresoras. CSS es independiente de HTML y puede ser usado por cualquier lenguaje de marcado XML. La separación entre HTML y CSS le permite un fácil mantenimiento de sitios y compartir hojas de estilos entre páginas. (w3c, 2015)

**SASS** es un lenguaje que compila directamente hacia CSS, permite utilizar funciones que CSS no tiene como base o no existen como:

- Jerarquías
- Mixins
- Herencia

SASS utiliza muchas funciones comunes para otros lenguajes de programación, las cuales no están disponibles como antes se mencionó en el CSS normal, esto permite un CSS más ordenado, legible y fácil de mantener (Weizenbaum, 2015). En la siguiente Figura 6 - CSS/SASS tenemos presente un ejemplo de código SASS, muy similar a CSS pero con una de sus principales características, el uso de variables.

```
1 $font-stack: Helvetica, sans-serif;
2 $primary-color: #333;
3
4 body {
5   font: 100% $font-stack;
6   color: $primary-color;
7 }
```

Figura 6 - CSS/SASS

## JavaScript y CoffeeScript

JavaScript es un lenguaje de scripting, eso quiere decir que es un lenguaje que no necesita alguna clase de pre procesado, en el contexto de los navegadores scripting usualmente se refiere a programar código escrito en JavaScript el cual es ejecutado por este mismo cuando la página es descargada, o en respuesta a un evento lanzado por un usuario.

JavaScript puede hacer las páginas más dinámicas. Por ejemplo sin recargar una nueva versión de la misma puede permitir modificaciones al contenido, agregarlo o enviarlo desde esta página, esto también llamado DHTML (HTML dinámico) y lo último AJAX (Asynchronous JavaScript and XML). Todo esto funciona como un puente entre la plataforma y el cliente para generar páginas que incorporen información para el ambiente del usuario, en la Figura 7 - JavaScript tenemos un ejemplo de JavaScript.

```
1  var person = {  
2    firstname:"John",  
3    lastname:"Doe",  
4    age:50,  
5    eyecolor:"blue"  
6  };  
7  document.getElementById("demo").innerHTML =  
8  person.firstname + " is " + person.age + " years old.";
```

Figura 7 - JavaScript

La interfaz más básica es el DOM (Document Object Model), este permite la comunicación entre JavaScript y el documento, lo cual tiene como resultado la posibilidad de modificar la estructura, contenido y estilo del documento. (W3C, 2015) JavaScript es un lenguaje muy versátil y con un gran poder para manejar el DOM dentro de una página, lamentablemente puede llegar a ser algo engorroso y terminar en código muy extenso.

CoffeeScript intenta solucionar el problema de la sintaxis compleja agregando una sintaxis más simple ya que es un lenguaje que trans-compila hacia JavaScript, esto quiere decir que en base como su propia página cita:

- *“It’s just JavaScript” (es solo JavaScript)* (Ashkenas, 2015)

El lenguaje compila directamente hacia JavaScript y no hay una interpretación más directa una vez compilado, puede utilizarse cualquier librería de JavaScript sin importar el no estar escrita en CoffeeScript y el resultado compilado es completamente legible por cualquier desarrollador con experiencia en JavaScript (Ashkenas, 2015). En la Figura 8 - CoffeeScript/JavaScript se encuentra un ejemplo de como una función de JavaScript de aproximadamente 9 líneas, se ve traducida a solo 2 en CoffeeScript.

```
1 JavaScript
2
3 var fill;
4
5 fill = function(container, liquid) {
6   if (liquid == null) {
7     liquid = "coffee";
8   }
9   return "Filling the " + container + " with " + liquid + "...";
10 };
11 loadrun: fill("cup")
12
13 Coffeescript
14
15 fill = (container, liquid = "coffee") ->
16   "Filling the #{container} with #{liquid}..."
```

Figura 8 - CoffeeScript/JavaScript

## NGINX

NGINX es un servidor HTTP y “reverse proxy server” (servidor proxy de reversa) escrito por Igor Sysoev. Durante mucho tiempo ha estado corriendo en páginas rusas de carga pesada incluidas Yandex, Mail.Ru, VK y Rambler. De acuerdo con Netcraft, NGINX dio función de servidor o proxy a 21.20% de los sitios con mayor tráfico en febrero 2015. (netcraft, netcraft, 2015). Algunas de las páginas más famosas que utilizan NGINX son:

- Netflix
- Wordpress
- FastMail.FM

NGINX es un servidor basado en eventos, esto significa que cada vez que recibe una petición, cada petición es procesada por separado a diferencia de los servidores web o HTTP basados en procesos, esto permite una escalabilidad bajo altas cargas o concurrencia mayor, de igual manera es utilizado para servir archivos estáticos principalmente por su misma arquitectura basada en eventos (NGINX, 2015) la Figura 9 - NGINX corresponde al logo de NGINX.



Figura 9 - NGINX

## Unicorn

Unicorn es un servidor de aplicación diseñado para servir solamente clientes rápidos en baja latencia, conexiones de alto ancho de banda y kernels tipo UNIX o similares. Los clientes lentos deben ser servidos o se recomienda utilizar un proxy de reversa o “reverse proxy” capaz de atender ambos, la petición y la respuesta entre Unicorn y el cliente, a continuación sus características:

- Diseñado para clientes rápidos de tipo Rack o Unix
- Compatible con Ruby 1.9.3 y posterior
- Manejo de procesos: Unicorn reinicia los workers caídos. No hay necesidad de manejar múltiples procesos o puertos manualmente.
- El balance de la carga es manejado enteramente por el sistema operativo.
- Los workers trabajan bajo su dirección aislada, no importa si es un hilo seguro y solo sirven un cliente a la vez.
- Soporta todas las aplicaciones de Rack.
- Se puede actualizar Unicorn, la aplicación, librerías e incluso el intérprete de Ruby sin tirar clientes.
- Puede ser usado con “copy-on-write-friendly memory management” para ahorrar memoria (la aplicación se precarga en la memoria y se crean copias de los sets a modificar entre aplicaciones cargadas idénticas).
- Puede escuchar múltiples interfaces incluyendo sockets UNIX, cada proceso de worker puede también apegarse a un puerto privado.
- Configuración rápida y fácil de Ruby DSL<sup>xiv</sup>.
- Decodifica transferencias empaquetadas cuando llegan, esto permitiendo que las notificaciones de subida o upload sean implementadas también

permitiendo un túnel arbitrario basado en protocolos de hilo o “stream” sobre HTTP, la Figura 10 - Unicorn corresponde al logo del mismo.



Figura 10 - Unicorn

## ImageMagick

ImageMagick es un set de software que permite crear, editar, componer o convertir imágenes de mapas de bits. Puede leer y escribir imágenes en una variedad de formatos (arriba de 200) incluyendo PNG, JPEG, JPEG-2000, GIF, TIFF, DPX, EXR, WebP, PostScript, PDF, SVG. Puede utilizarse para redimensionar, girar, voltear, rotar, distorsionar, cortar y transformar imágenes, ajustar los colores, aplicar efectos especiales, escribir textos, líneas, polígonos elipses o curvas de Bézier. (Magick, 2015), la Figura 11- ImageMagick pertenece al logo.



Figura 11- ImageMagick

La funcionalidad de ImageMagick es típicamente utilizada desde la línea de comandos o pueden utilizarse las características desde programas escritos en el lenguaje de preferencia, se pueden escoger las siguientes interfaces:

- MagickCore (C)
- MagickWand (C)
- ChMagick (Ch)
- ImageMagickObject (COM+)
- Magick++(C++)

- JMagick(Java)
- L-Magick(Lisp)
- Nmagick (Neko/haXe)
- Magick.NET (.Net)
- PascalMagick (Pascal)
- PerlMagick (Perl)
- MagickWand for PHP
- IMagick (PHP)
- PythonMagick (Python)
- RMagick (Ruby)
- TclMagick (Tcl/TK)

ImageMagick es software libre entregado en distribución binaria o como código fuente que puede utilizarse, copiarse, modificarse, distribuirse en ambos tipos de aplicación, abierta y propietaria. (Magick, 2015)

# CAPÍTULO 3



## CAPÍTULO 3. IMPLEMENTACIÓN

### Diagrama de clases

En el siguiente diagrama de clases se describen las clases principales para los procesos internos de la aplicación, todas contienen a excepción de Tag, ImageUploader y Tagging campos en común como created\_at, updated\_at y los ids de relación entre ellas:

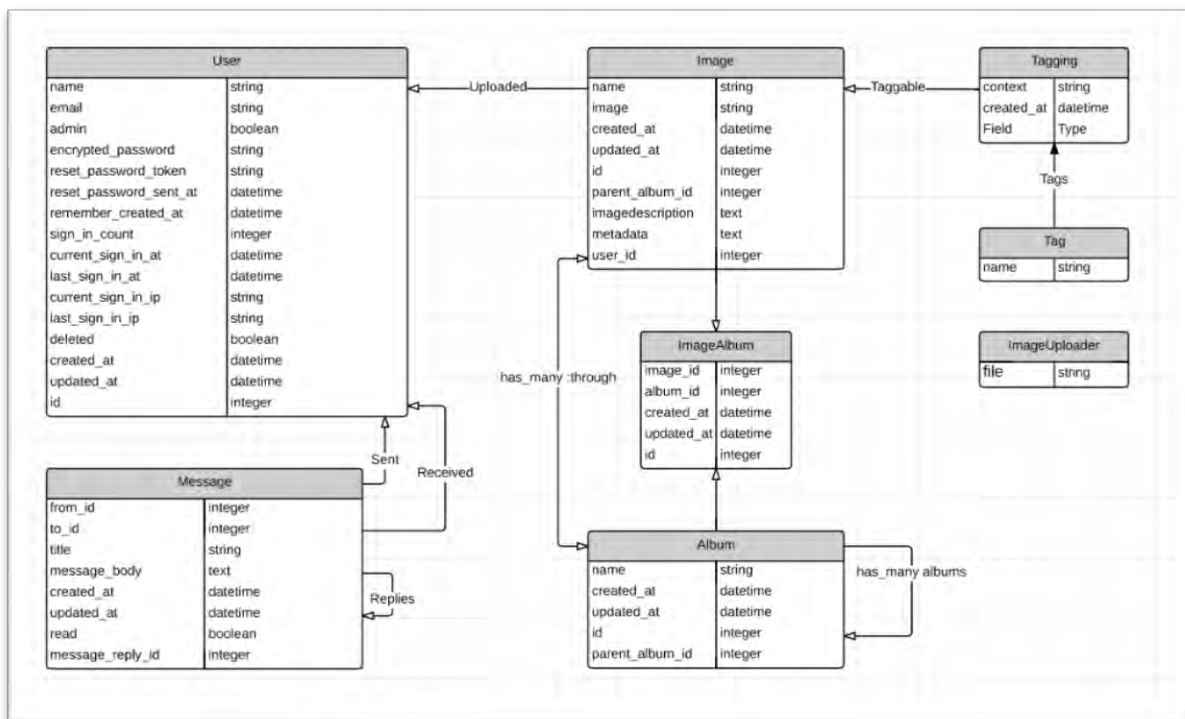


Figura 12 – Diagrama de clases

#### User

Clase o entidad encargada de representar a los usuarios, contiene conexión a la tabla de usuarios o “users” dentro de la base de datos, la mayor parte de sus campos son utilizados para el proceso de registro y sesión dentro de la aplicación, cuenta con las siguientes relaciones:

- Un usuario tiene muchos “sent\_messages”, el campo de relación se encuentra en la tabla “messages”.
- Un usuario tiene muchos “received\_messages”, el campo de relación se encuentra en la tabla “messages”.
- Un usuario tiene muchas “images”, el campo de relación se encuentra en la tabla “images”.

Cuenta con otros campos listados a continuación utilizados para fines específicos:

- **admin:** atributo booleano para identificar a un administrador
- **deleted:** atributo booleano utilizado para identificar a un usuario suspendido

#### Métodos

- **unsigned\_in?:** utilizado para saber si el usuario ha iniciado sesión en el sistema una vez o mas
- **has\_unread\_messages?:** Utilizado para saber si un usuario tiene mensajes sin leer
- **unread\_messages:** Regresa un arreglo con objetos de la clase “Message” que no se han leído, internamente utiliza el método “select\_unread\_messages” que filtra los mensajes no leídos de acuerdo a un listado de condiciones incluyendo si la última respuesta del mensaje o “reply” es propia (del usuario) o no.
- **read\_messages:** Regresa un arreglo con objetos de la clase “Message” que han sido leídos o marcados como leídos.

#### Message

Clase o entidad utilizada para el proceso de mensaje interno en la aplicación, cuenta con los campos básicos de “title” y “message\_body” para designar el contenido del mensaje y contiene 3 principales relaciones:

- Un mensaje pertenece a un usuario como un mensaje recibido o “received” con el campo de relación “to\_id”
- Un mensaje pertenece a un usuario como un mensaje enviado o “sent” con el campo de relación “from\_id”
- Un mensaje pertenece a otro mensaje como una respuesta o “reply” con el campo “message\_reply\_id”

También cuenta con el siguiente campo para propósitos específicos:

- El campo booleano “read” es utilizado para marcar los mensajes leídos

#### Métodos

Los scopes entregan arreglos de objetos de tipo “Message” con diferentes atributos definidos dentro de la petición a la base de datos.

- **unread:** scope que entrega los mensajes marcados como no leídos.
- **read:** scope que entrega los mensajes marcados como leídos.

- **non\_replies**: scope que entrega los mensajes sin contenido en el campo `message_reply_id`.
- **not\_a\_reply?**: utilizado para saber si el mensaje es una respuesta a otro mensaje.
- **last\_reply**: regresa el último mensaje marcado como respuesta hacia el mensaje que llama el método.
- **load\_non\_replies\_from(user)**: regresa un listado de mensajes enviados por o hacia el usuario entregado como argumento al método y que no son respuestas a otros mensajes.

### Image

Clase o entidad utilizada para el almacenamiento de la información sobre las imágenes subidas, esta cuenta con las siguientes relaciones:

- Una imagen pertenece a un usuario como “image”
- Una imagen pertenece a un álbum como “parent\_album”
- Una imagen tiene muchas relaciones con la tabla “image\_albums”.
- Una imagen tiene muchos “albums” a través de la tabla “image\_albums” utilizando la relación “has\_many\_through”, esta facilita las búsquedas ya que se realizan en la tabla “image\_albums” con los correspondientes ids.
- Una imagen tiene muchos “taggings” o es “taggable”, el campo de relación se encuentra en la tabla “taggings”.

También cuenta con los campos siguientes para diferentes propósitos:

- “imagedescription” se utiliza para darle una descripción a la imagen
- “metadata” campo donde se guarda a modo de cadena de caracteres el hash de los metadatos extraídos de la imagen.

### Métodos

- **find\_tags(tags)**: utilizado para encontrar las etiquetas con similitudes a la cadena de caracteres entregada que tiene forma de lista separada con comas y es procesado para regresar las ya existentes en el sistema o guardarlas en caso de no ser así.
- **find\_tags\_like\_string(string)**: Regresa las etiquetas ya existentes similares a la entregada como argumento en forma de cadena de caracteres.
- **find\_tags\_equal\_string(string)**: Regresa las etiquetas ya existentes idénticas a la entregada como argumento en forma de cadena de caracteres.
- **delete\_empty\_upstream\_dirs**: Elimina de manera física el directorio de la imagen.

- **full\_store\_dir**: Regresa una cadena de caracteres con la ruta absoluta del directorio físico de la imagen
- **default\_name**: Asigna un nombre por default a la imagen

### *ImageAlbum*

Clase o entidad utilizada para la relación entre las imágenes y los álbumes, esta contiene los campos de relación “image\_id” y “album\_id”

- Un “ImageAlbum” pertenece a un “Image”
- Un “ImageAlbum” pertenece a un “Album”

### *Album*

Clase o entidad utilizada para representar álbumes dentro de la aplicación, posee un campo especial “name” para nombrar al álbum y las siguientes relaciones:

- Un álbum tiene muchos “images” a través de la relación “has\_many\_through” en la tabla “image\_albums”.
- Un álbum tiene muchos “image\_albums” para realizar la relación “has\_many\_through” hacia “images”.
- Un álbum pertenece a un “parent\_album” utilizando el campo “parent\_album\_id”

### *Métodos*

- **delete\_dir**: elimina el directorio físico.
- **self.root**: regresa el directorio raíz (usado como método de clase).
- **master\_album**: regresa el directorio siguiente al directorio raíz.
- **root?**: identifica si el directorio es raíz o no.
- **make\_album**: genera el directorio a menos de que ya exista.
- **full\_path**: regresa la ruta absoluta del directorio.
- **relative\_path**: regresa la ruta relativa del directorio.
- **set\_name**: agrega nombre al directorio.
- **loop\_name**: agrega el sufijo “nuevo” cuando el directorio en el mismo nivel ya existe con dicho nombre.
- **check\_update**: revisa si el nombre del directorio ya existe al momento de actualizar el nombre de un directorio y regresa un error en caso de ya existir.
- **root\_folder**: regresa el directorio raíz (usado como método de instancia)
- **descendants**: regresa los directorios descendientes
- **ascendants**: regresa los directorios ascendientes en jerarquía directa hacia el directorio raíz

- **self\_and\_ascendants:** regresa el directorio actual y los directorios ascendientes jerárquicamente hacia el directorio raíz.
- **self\_and\_descendants:** regresa el directorio actual y los directorios descendientes.
- **self.descendent\_tree\_for(album):** regresa el directorio y los directorios descendientes (usado como método de clase).
- **self.ascendant\_tree\_for(album):** regresa el directorio y los directorios ascendientes con jerarquía hacia el directorio raíz (usado como método de clase)
- **self.self\_and\_descendants\_tree\_sql(album):** sentencia de SQL para una búsqueda de árbol que regresa los álbumes descendientes a partir del proporcionado como argumento.
- **self.self\_and\_ascendants\_tree\_sql(album):** sentencia de SQL para una búsqueda de árbol que regresa los álbumes ascendientes en jerarquía hasta el álbum raíz a partir del proporcionado como argumento.

### Taggings

Clase o entidad utilizada para el etiquetado de las imágenes, cuenta con las siguientes relaciones:

- Un tagging pertenece a un “Image” a través del campo “taggable\_id”.
- Un tagging tiene un “Tag” a través del campo “tag\_id”

### Métodos

- **remove\_unused\_tags:** borra todas las etiquetas sin relaciones dentro de la tabla taggings.

### Tag

Clase o entidad utilizada para el etiquetado, trabaja en conjunto con “Taggings”, esta entidad guarda la etiqueta como tal. Solamente posee un campo utilizado para guardar el nombre de la etiqueta en “name”.

### Métodos

- **self.named(name):** regresa la etiqueta con el nombre proporcionado como argumento.
- **self.named\_any(name):** regresa etiquetas con el nombre proporcionado en mayúsculas o minúsculas.
- **self.named\_like(name):** regresa la etiqueta con la palabra incluida dentro de sus letras componentes.

- **self.named\_like\_any(name):** regresa etiquetas con la palabra incluida dentro de sus letras componentes minúsculas o mayúsculas.
- **self.find\_or\_create\_with\_like\_by\_name(name):** busca o crea una etiqueta con el nombre similar al proporcionado, con mayúsculas o minúsculas.
- **self.find\_all\_or\_create\_with\_like\_by\_name(name):** busca todas o crea la etiqueta con el nombre similar al proporcionado con mayúsculas o minúsculas.

## Controladores

Los controladores son el puente entre la interfaz o componente de las vistas y los modelos o clases que manejan los datos dentro de la aplicación, estos tienen el papel de generar las peticiones y dependiendo de los parámetros entrantes en las peticiones, solicitan la información y deciden el comportamiento de los flujos.

Los controladores obedecen dentro de Rails al estándar CRUD<sup>xv</sup>, este estándar permite programar de una forma ordenada los procesos o flujos delegando el control de estos a rutas predefinidas:

- **Create:** Crear o generar registros
- **Read:** Mostrar registros
  - Show: Muestra un registro
  - Index: Muestra un listado de registros
- **Update:** Modificar registros
  - Update: Modifica más de un parámetro del registro
  - Patch: Modifica un parámetro del registro
- **Destroy:** Eliminar registros

La aplicación contiene un listado de controladores cada uno auto descriptivo y separando la lógica de la aplicación en tareas concretas, a continuación el listado de controladores:

- **ApplicationController:** El controlador principal, este controlador contiene todos los filtros de autenticación para las diferentes partes de la aplicación haciendo la función de una primera línea de seguridad antes de acceder a los flujos.
- **AlbumsController:** Controlador encargado del manejo de los álbumes.

- **ImageAlbumsController:** Controlador encargado de las peticiones principales del manejador de imágenes.
- **ImagesController:** Controlador encargado del manejo de las imágenes.
- **MessagesController:** Controlador encargado del sistema de mensajería.
- **StaticPagesController:** Controlador encargado de la raíz de la aplicación.
- **UsersController:** Controlador encargado del manejo de los usuarios.

Los controladores cuentan con varias líneas de defensa o seguridad para impedir a usuarios no permitidos acceder a los datos a través de la aplicación:

- **Devise:**<sup>xvi</sup> Librería encargada de la autenticación de los usuarios, esta trabaja dentro de los controladores a modo de filtro para redirigir al formulario de inicio de sesión en caso de haber iniciado sesión.
- **Strong Params:** <sup>xvii</sup> Librería que filtra los parámetros entrantes en cada petición evitando así la inyección de datos a las acciones o secciones de los controladores, principalmente al crear y actualizar registros.
- **SSL:** <sup>xviii</sup> los controladores contienen encriptación a nivel de Rails para evitar que los datos sean interceptados durante su transmisión.
- **CSRF:**<sup>xix</sup> Cada vez que una vista dentro de la aplicación es generada, se envía un token de autenticación, esto permite que solo peticiones provenientes del usuario sean permitidas.
- **Sesión:** Rails cuenta con un sistema de seguridad a través de la sesión, una serie de parámetros enviados desde el servidor y guardados en una “cookie” cada vez que un usuario accede a la aplicación, esto le permite identificar al usuario o hacer a la aplicación del lado del usuario “statefull” manteniendo registro de quienes, cuando acceden y haciendo posible mantener al usuario “conectado” al sistema hasta que esta sesión expire.

## Disposición

La disposición de la aplicación consta de tres vistas principales:

- **Usuarios:** Panel de administración de usuarios
- **Imágenes:** Panel de administración de las imágenes y los mensajes
- **Root o Raíz:** Página de bienvenida o principal (esta página no cuenta con ninguna funcionalidad, solamente funciona como bienvenida)

### *Usuarios*

El panel de usuarios es el panel principal de administración de los usuarios, este cuenta con diferentes secciones para suspender, habilitar o agregar nuevos

usuarios a la aplicación. Muchas de las funcionalidades se ofrecen a través de JavaScript en el archivo “users.js.coffe”, en la Figura 13 - Panel de usuarios se describen las partes del panel de usuarios y sus diferentes vistas.

The screenshot shows the 'Lista de Usuarios' interface. At the top right, there are navigation links: 'Inicio', 'Usuarios', 'Imágenes', and 'Salir'. Below the header, the title 'Lista de Usuarios' is displayed. To the right of the title is a 'Nuevo Usuario' button (2). Below the title, there are search filters for 'Último inicio de sesión' (1) and 'Fecha de registro', each with 'Despues de' and 'Antes de' date pickers. There is also a 'Filtrar' button. Below the filters is a table of users with the following data:

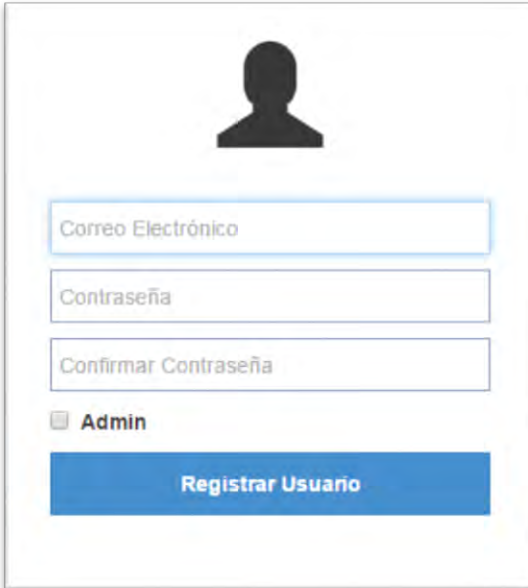
ID	Correo	Inicios de sesión	Último inicio de sesión	Última dirección IP	Creado	Actualizado	Admin	Acciones
4	casfdsa@ygdjgsd.com	1	10-Oct-2014	127.0.0.1	18-Aug-2014	17-Feb-2015	✗	editar activar
5	admin2@admin.com	3	28-Jan-2015	127.0.0.1	28-Jan-2015	17-Feb-2015	✓	editar suspender
1	admin@admin.com	73	17-Feb-2015	127.0.0.1	21-Jun-2014	18-May-2015	✓	editar
6	admin3@admin.com	2	03-Feb-2015	127.0.0.1	03-Feb-2015	03-Feb-2015	✗	editar suspender

Figura 13 - Panel de usuarios

1. Panel de búsqueda para filtrar usuarios.
2. Botón para crear nuevos usuarios.
3. Listado de usuarios con botones de acciones para editar, suspender, activar o eliminarlos.

En la Figura 14 – Formulario de nuevo usuario se observa el formulario para crear un nuevo usuario con los campos de correo electrónico, contraseña, confirmación de contraseña y el campo booleano para identificar a un nuevo administrador.





Formulario de nuevo usuario. El formulario contiene un ícono de perfil de usuario, tres campos de entrada de texto etiquetados como 'Correo Electrónico', 'Contraseña' y 'Confirmar Contraseña', un campo de selección con un botón 'Admin' y un botón azul 'Registrar Usuario'.

Figura 14 – Formulario de nuevo usuario

En la Figura 15 - Editar usuario suspendidose encuentra el panel de edición de usuarios con un usuario suspendido como ejemplo, este cuenta con los controles para activar, borrar o actualizar los datos enviando los mismos campos que un usuario nuevo.



The screenshot shows a web form for editing a suspended user. At the top, there is a black padlock icon. Below it, a pink banner displays the text "Usuario Suspendido". The form contains several input fields: an email field with the placeholder "dsfdsaf@jfgdgjسد.com", a field for "Contraseña Actual", a field for "Contraseña", and a field for "Confirmar Contraseña". Below these fields is a checkbox labeled "Administrador". At the bottom, there are three prominent buttons: a blue button labeled "Actualizar Información", a green button labeled "Reactivar Usuario", and a red button labeled "Borrar Usuario".

Figura 15 - Editar usuario suspendido

### *Imágenes*

El panel de imágenes incluye múltiples secciones con código en módulos separados de JavaScript para hacerla una sección dinámica y que permita al usuario mantenerse en una sola página sin tener que trasladarse entre vista y vista después de cada acción. En la Figura 16 - Panel de Imágenes describen las principales secciones del panel de imágenes.

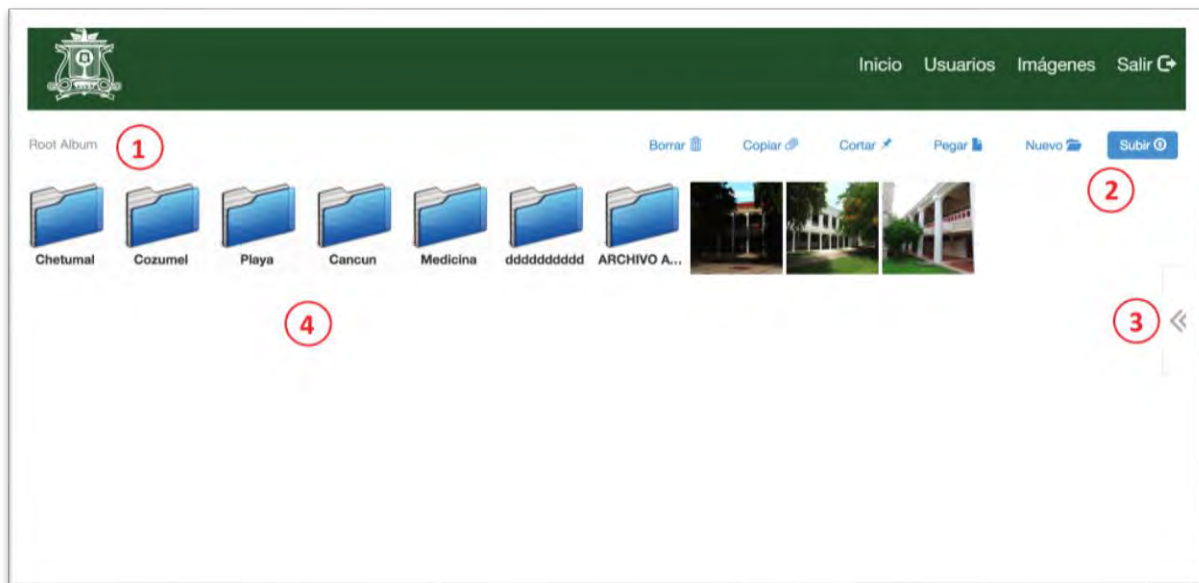


Figura 16 - Panel de Imágenes

1. Breadcrumbs que describen el nivel de carpetas en el que se encuentra actualmente.
2. Lista de botones para el manejo de carpetas e imágenes:
  - Borrar
  - Copiar
  - Cortar
  - Pegar
  - Nueva carpeta
  - Subir imágenes
3. Panel lateral descrito en la Figura 18 - Panel lateral
4. Piscina(Pool) o listado de imágenes y carpetas

En la Figura 17 - Modal de imagen **¡Error! No se encuentra el origen de la referencia.** se presenta el modal de opciones de una imagen, aquí se puede elegir una versión para descargar, cambiar el nombre, agregar una descripción, agregar etiquetas y borrarlas, este modal es accesible al hacer clic sobre un botón que aparece al pasar el cursor sobre el thumbnail de la imagen deseada.

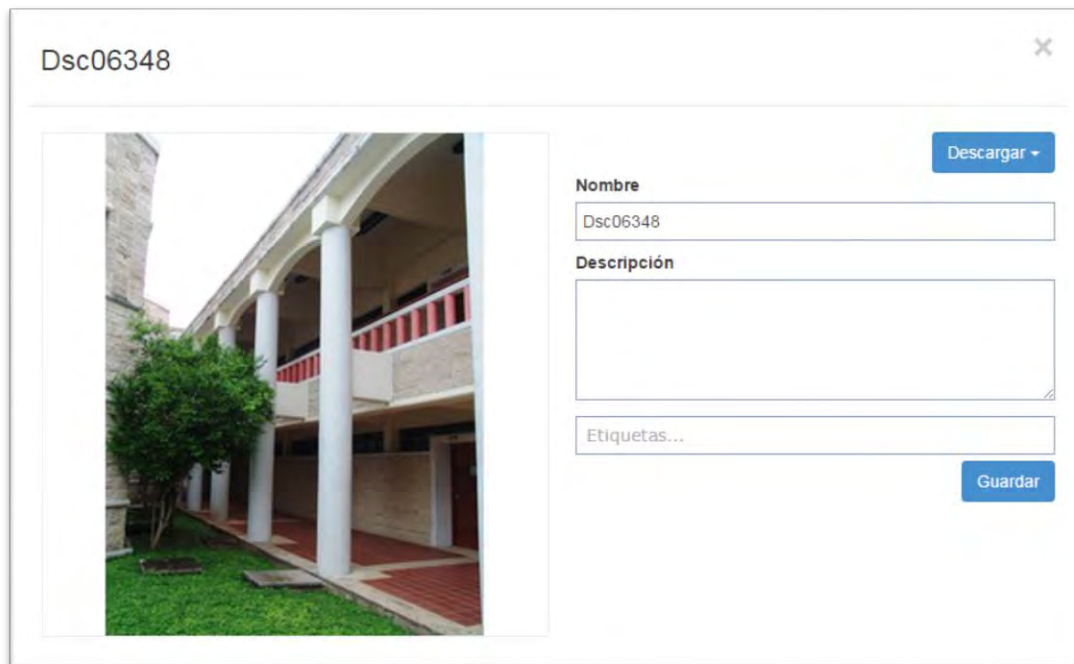


Figura 17 - Modal de imagen

1. Buscador de imágenes.
2. Campo para etiquetar múltiples imágenes seleccionadas previamente.
3. Pestañas del sistema de mensajería



Figura 18 - Panel lateral

En la Figura 19 - Modal de nuevo mensaje **¡Error! No se encuentra el origen de la referencia.** se muestra el modal que aparece al hacer clic en el botón de “Nuevo” mensaje, este se encarga de generar un nuevo mensaje después de elegir un destinatario junto con un título y el contenido del mismo.



Figura 19 - Modal de nuevo mensaje

# CAPÍTULO 4

## CAPÍTULO 4

### Pruebas de funcionamiento

La fase de pruebas se realizó dentro de un servidor en modo de producción con los anteriores mencionados servidores NGINX y Unicorn, Web y de aplicación respectivamente, estos servidores cuentan con 300 Gb de disco duro para el almacenamiento de las imágenes. A continuación, en la Figura 20 - Panel de usuarios se puede observar el panel de usuarios con un grupo de entre administradores y no administradores.

The screenshot shows the 'Lista de Usuarios' interface. At the top, there is a green navigation bar with 'Inicio', 'Usuarios', 'Imágenes', and 'Salir'. Below the navigation bar, the title 'Lista de Usuarios' is displayed. To the right of the title is a 'Nuevo Usuario' button. Below the title, there are search filters for 'Último inicio de sesión' and 'Fecha de registro', each with 'Despues de:' and 'Antes de:' input fields. There is also a 'Filtrar' button. Below the filters is a table with the following data:

ID	Correo	Inicios de sesión	Último inicio de sesión	Última dirección IP	Creado	Actualizado	Admin	Acciones
4	cafdastf0@ggjed.com	1	10-Oct-2014	127.0.0.1	18-Aug-2014	17-Feb-2015	✘	editar activar
5	admin2@amin.com	3	28-Jan-2015	127.0.0.1	28-Jan-2015	17-Feb-2015	✔	editar suspender
1	admin@admin.com	73	17-Feb-2015	127.0.0.1	21-Jun-2014	18-May-2015	✔	editar
6	admin3@admin.com	2	03-Feb-2015	127.0.0.1	03-Feb-2015	03-Feb-2015	✘	editar suspender

Figura 20 - Panel de usuarios

En la Figura 21 - Panel de imágenes en pruebase observa el panel principal de imágenes, con algunas carpetas generadas conforme a los planteles y otras de prueba.

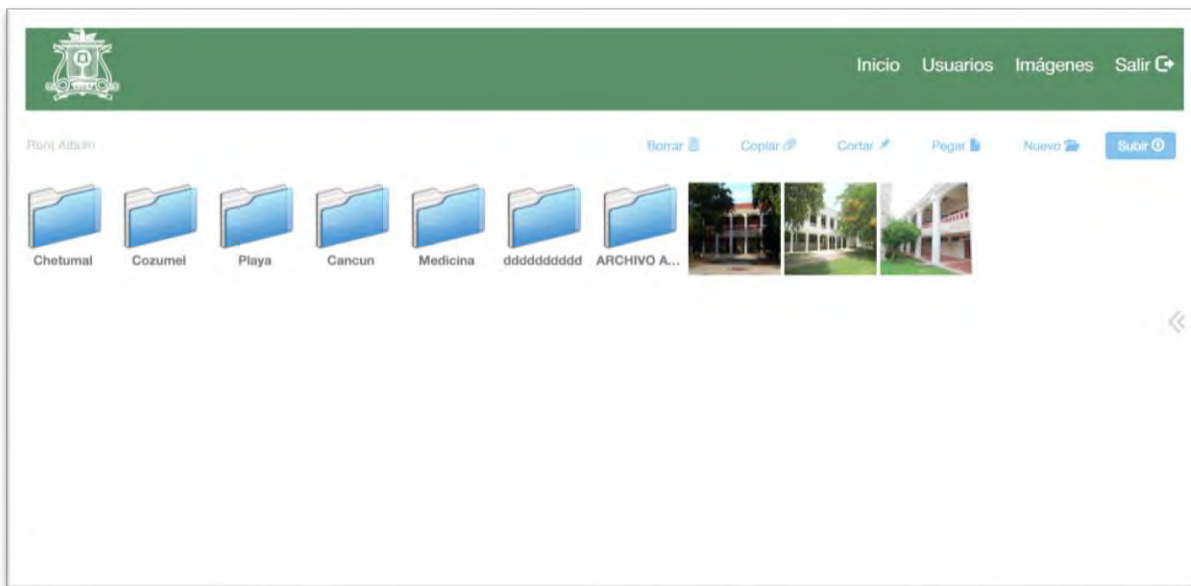


Figura 21 - Panel de imágenes en prueba

En la que se observa el proceso de cargado múltiple de las imágenes, dicho siendo el más importante ya que conlleva múltiples procesos internos e implica un consumo de recursos considerable para el sistema.

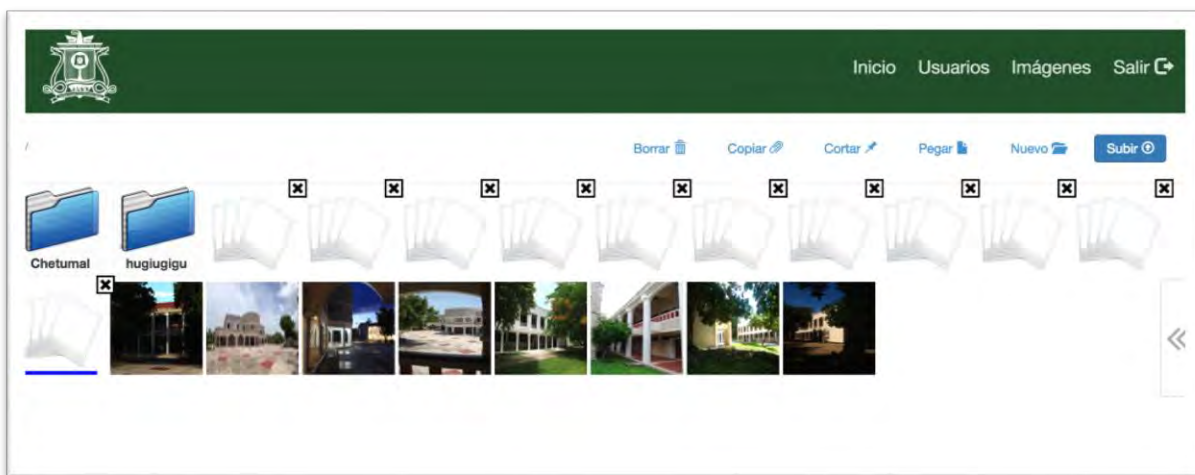


Figura 22 - Cargado múltiple



En la Figura 23 - Servidor subida se puede observar el log del servidor en producción (production.log) con los tiempos de respuesta marcados en rojo para cada solicitud multi-partes (envío de datos binarios) hacia el mismo, estas demoran entre 5 y 9 segundos en total, esto debido a que el servidor auto genera varias versiones de la imagen original redimensionada, con marca de agua y asigna meta datos a la versión original de la fotografía con respecto a la carpeta en la que se subió y otros datos sobre la universidad.

```

I, [2015-06-17T16:57:11.150098 #3496] INFO -- : Started POST "/images" for 127.0.0.1 at 2015-06-17 16:57:11 -0500
I, [2015-06-17T16:57:11.151446 #3496] INFO -- : Processing by ImagesController#create as JSON
I, [2015-06-17T16:57:11.151556 #3496] INFO -- : Parameters: {"utf8"=>"✓", "authenticity_token"=>"NZ11CGwIUZnzRe
#<Tempfile:/tmp/RackMultipart20150617-3496-hg9dq3>,@original_filename="DSCF2871.JPG",@content_type="image/jpeg",
I, [2015-06-17T16:57:17.869816 #3496] INFO -- : Rendered images/create.json.builder (0.5ms)
I, [2015-06-17T16:57:17.870844 #3496] INFO -- : Completed 200 OK in 6718ms (Views: 1.1ms | ActiveRecord: 13.6ms)
I, [2015-06-17T16:57:18.062720 #3496] INFO -- : Started POST "/images" for 127.0.0.1 at 2015-06-17 16:57:18 -0500
I, [2015-06-17T16:57:18.064045 #3496] INFO -- : Processing by ImagesController#create as JSON
I, [2015-06-17T16:57:18.064152 #3496] INFO -- : Parameters: {"utf8"=>"✓", "authenticity_token"=>"NZ11CGwIUZnzRe
#<Tempfile:/tmp/RackMultipart20150617-3496-194vs5k>,@original_filename="DSCF2903.JPG",@content_type="image/jpeg"
I, [2015-06-17T16:57:25.168221 #3496] INFO -- : Rendered images/create.json.builder (0.6ms)
I, [2015-06-17T16:57:25.168473 #3496] INFO -- : Completed 200 OK in 7104ms (Views: 1.3ms | ActiveRecord: 13.8ms)
I, [2015-06-17T16:57:25.353741 #3496] INFO -- : Started POST "/images" for 127.0.0.1 at 2015-06-17 16:57:25 -0500
I, [2015-06-17T16:57:25.355095 #3496] INFO -- : Processing by ImagesController#create as JSON
I, [2015-06-17T16:57:25.355211 #3496] INFO -- : Parameters: {"utf8"=>"✓", "authenticity_token"=>"NZ11CGwIUZnzRe
#<Tempfile:/tmp/RackMultipart20150617-3496-1kpsxc>,@original_filename="IMG_1142.JPG",@content_type="image/jpeg"
I, [2015-06-17T16:57:32.338369 #3496] INFO -- : Rendered images/create.json.builder (0.5ms)
I, [2015-06-17T16:57:32.338592 #3496] INFO -- : Completed 200 OK in 6983ms (Views: 1.1ms | ActiveRecord: 18.5ms)
I, [2015-06-17T16:57:32.890245 #3496] INFO -- : Started POST "/images" for 127.0.0.1 at 2015-06-17 16:57:32 -0500
I, [2015-06-17T16:57:32.891538 #3496] INFO -- : Processing by ImagesController#create as JSON
I, [2015-06-17T16:57:32.891637 #3496] INFO -- : Parameters: {"utf8"=>"✓", "authenticity_token"=>"NZ11CGwIUZnzRe
#<Tempfile:/tmp/RackMultipart20150617-3496-5yf4qx>,@original_filename="IMG_4094.JPG",@content_type="image/jpeg",
I, [2015-06-17T16:57:41.527035 #3496] INFO -- : Rendered images/create.json.builder (0.5ms)
I, [2015-06-17T16:57:41.527263 #3496] INFO -- : Completed 200 OK in 8636ms (Views: 1.2ms | ActiveRecord: 14.2ms)
I, [2015-06-17T16:57:41.875719 #3496] INFO -- : Started POST "/images" for 127.0.0.1 at 2015-06-17 16:57:41 -0500
I, [2015-06-17T16:57:41.877139 #3496] INFO -- : Processing by ImagesController#create as JSON
I, [2015-06-17T16:57:41.877257 #3496] INFO -- : Parameters: {"utf8"=>"✓", "authenticity_token"=>"NZ11CGwIUZnzRe
#<Tempfile:/tmp/RackMultipart20150617-3496-1ry1x7u>,@original_filename="IMG_9325.jpg",@content_type="image/jpeg"
I, [2015-06-17T16:57:49.393050 #3496] INFO -- : Rendered images/create.json.builder (0.5ms)
I, [2015-06-17T16:57:49.393278 #3496] INFO -- : Completed 200 OK in 7516ms (Views: 1.1ms | ActiveRecord: 14.0ms)
I, [2015-06-17T16:57:49.596939 #3496] INFO -- : Started POST "/images" for 127.0.0.1 at 2015-06-17 16:57:49 -0500
I, [2015-06-17T16:57:49.598237 #3496] INFO -- : Processing by ImagesController#create as JSON
I, [2015-06-17T16:57:49.598339 #3496] INFO -- : Parameters: {"utf8"=>"✓", "authenticity_token"=>"NZ11CGwIUZnzRe
#<Tempfile:/tmp/RackMultipart20150617-3496-1a03smb>,@original_filename="IMG_9745.JPG",@content_type="image/jpeg"
I, [2015-06-17T16:57:57.716977 #3496] INFO -- : Rendered images/create.json.builder (0.5ms)
I, [2015-06-17T16:57:57.717208 #3496] INFO -- : Completed 200 OK in 8119ms (Views: 1.2ms | ActiveRecord: 14.4ms)
I, [2015-06-17T16:58:30.310855 #3496] INFO -- : Started GET "/users" for 127.0.0.1 at 2015-06-17 16:58:30 -0500
I, [2015-06-17T16:58:30.315468 #3496] INFO -- : Processing by UsersController#index as HTML
I, [2015-06-17T16:58:30.337948 #3496] INFO -- : Rendered users/_search_form.html.haml (10.5ms)
I, [2015-06-17T16:58:30.348446 #3496] INFO -- : Rendered users/_index_users_table_content.html.haml (9.9ms)
I, [2015-06-17T16:58:30.348584 #3496] INFO -- : Rendered users/index.html.haml within layouts/application (26.3)
I, [2015-06-17T16:58:30.349841 #3496] INFO -- : Rendered shared/_flash.html.haml (0.1ms)
I, [2015-06-17T16:58:30.350090 #3496] INFO -- : Rendered static_pages/_footer.html.haml (0.1ms)
I, [2015-06-17T16:58:30.350634 #3496] INFO -- : Rendered static_pages/_modal.html.haml (0.4ms)
I, [2015-06-17T16:58:30.350831 #3496] INFO -- : Completed 200 OK in 35ms (Views: 29.5ms | ActiveRecord: 1.4ms)

```

Figura 23 - Servidor subida múltiple

Cabe mencionar que el tiempo para la generación de las diferentes versiones de imagen varía dependiendo del tamaño de la imagen original, la cantidad de metadatos originales y el tráfico del servidor al momento (las pruebas fueron hechas con tráfico de un solo usuario).

En la Figura 24 - Mensajes se observa un grupo de mensajes enviados utilizando el sistema de mensajería interno.

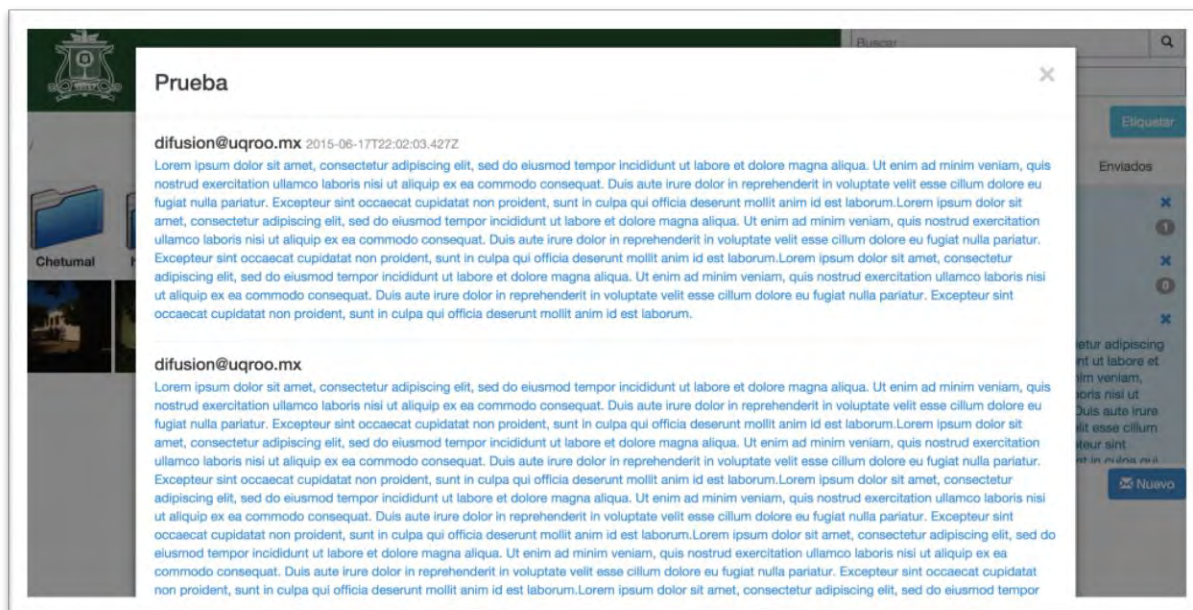


Figura 24 - Mensajes

# CAPÍTULO 5

## CAPÍTULO 5

### CONCLUSIONES

El sistema desarrollado trajo una serie de retos pequeños y grandes, siendo estos los que orillaron en algunos momentos del desarrollo a rediseñar múltiples partes de la aplicación. Algunos de estos retos consistieron en características que se fueron agregando con forme la marcha, tamaño y alcance que adquirió la aplicación, la cual paso de ser un sistema para subir fotografías y obtener metadatos a un completo control de árbol de archivos tanto físico como lógico, catalogación, búsqueda, mensajería interna, post procesamiento de imágenes. Todo esto me orilló aproximadamente la mitad del tiempo de desarrollo a rediseñar el sistema de un método básico RESTfull con recargados de página normales entre operaciones a utilizar AJAX para unificar las diferentes características bajo un solo panel para dar una experiencia más sólida y estable al usuario evitando así que entre el envío de un mensaje y la subida de una fotografía hubiera cambios totales en el navegador.

Otro de los contratiempos encontrados durante el desarrollo de la aplicación fue la localización del servidor donde se establecería la aplicación ya que siendo está desarrollada en Ruby on Rails cuenta con la necesidad de algunas características particulares con las cuales los servidores normales para PHP no cuentan. Sin embargo la aplicación se encuentra en modo de producción instalada temporalmente en un servidor de la Universidad de Quintana Roo, este servidor provisional fue proporcionado por el departamento de cómputo tras la solicitud del departamento de difusión, lamentablemente con retraso por falta de comunicación entre el encargado del proyecto, el departamento de informática y mía.

El proyecto funcionó para cumplir con los requerimientos del departamento de difusión especificados para el mismo, pero como todo software, todavía es posible mejorarlo y de ser posible es recomendable hacerlo.

En la aplicación, para el proceso de desarrollo y planeación, me veo en la necesidad de señalar algunos puntos bastante importantes si se considera seguir desarrollando o mejorando como:

- Mejorar la comunicación entre los departamentos
- Tener un nivel de soporte mayor, independientemente si el proyecto llegaría a adquirir trascendencia ya que en un principio el proceso de planteamiento obtuvo poca atención por parte de las demás secciones de la Universidad y

se tuvo que solicitar el apoyo numerosas veces en múltiples departamentos antes de recibir una respuesta concreta.

- Dar capacitación, mejorar las habilidades u ofrecer algún soporte para la generación de proyectos internos para la Universidad ya que el proceso de planteamiento, desarrollo e implementación de estos fue deficiente y seguramente mejoraría a la Universidad y sus procesos internos enormemente.
- Para el mejoramiento de la aplicación es recomendable implementar sub procesos o “Workers” para los procesos complicados como el redimensionamiento de las imágenes, la inserción de los metadatos, el copiado, cortado, pegado, la implementación de un menú de clic derecho, la habilidad de mover imágenes con el cursor hacia las carpetas arrastrándolas.

De igual manera hago mención de la utilidad y lo indispensable de las clases tomadas a lo largo de los 7 años dentro de la carrera Unix, Programación Web, Programación, Redes 1 – 5 (contando seguridad en redes), Gestión de proyectos, Herramientas de diseño de redes, Base de datos, entre otras, las cuales funcionaron como base para aprender a manejar las diferentes tecnologías utilizadas para el diseño, desarrollo e implementación del proyecto.

## REFERENCIAS BIBLIOGRÁFICAS

- Ashkenas, J. (28 de Marzo de 2015). *coffeescript.org*. Obtenido de <http://coffeescript.org/>:  
<http://coffeescript.org/>
- Clarke, N. (28 de Marzo de 2015). *HAML*. Obtenido de <http://haml.info>: <http://haml.info>
- desarrolloweb. (28 de Marzo de 2015). *www.desarrolloweb.com*. Obtenido de desarrollo web:  
[www.desarrolloweb.com](http://www.desarrolloweb.com)
- Ellis, S. (28 de Marzo de 2015). *StuartEllis.eu*. Obtenido de <http://www.stuartellis.eu/articles/erb/>:  
<http://www.stuartellis.eu/articles/erb/>
- Guides, R. o. (28 de Marzo de 2015). *Getting started with Ruby on Rails*. Obtenido de  
[http://guides.rubyonrails.org/getting\\_started.html](http://guides.rubyonrails.org/getting_started.html):  
[http://guides.rubyonrails.org/getting\\_started.html](http://guides.rubyonrails.org/getting_started.html)
- IETF. (28 de Marzo de 2015). *IETF*. Obtenido de <http://tools.ietf.org/html/rfc7230>:  
<http://tools.ietf.org/html/rfc7230>
- Magick, I. (28 de Marzo de 2015). *ImageMagick*. Obtenido de <http://www.imagemagick.org/>:  
<http://www.imagemagick.org/>
- netcraft. (28 de Marzo de 2015). *netcraft*. Obtenido de  
<http://news.netcraft.com/archives/2015/02/24/february-2015-web-server-survey.html>:  
<http://news.netcraft.com/archives/2015/02/24/february-2015-web-server-survey.html>
- netcraft. (28 de Marzo de 2015). *netcraft*. Obtenido de  
<http://news.netcraft.com/archives/2015/02/24/february-2015-web-server-survey.html>:  
<http://news.netcraft.com/archives/2015/02/24/february-2015-web-server-survey.html>
- NGINX. (28 de Marzo de 2015). *NGINX*. Obtenido de <http://nginx.org/en/>: <http://nginx.org/en/>
- Reyes, M. M. (28 de Marzo de 2015). *Universidad Autonoma de Puebla*. Obtenido de  
[http://h1cu1.dosmildiez.net/marcov/wp-content/uploads/2009/08/63\\_Miriam\\_7Dic05.pdf](http://h1cu1.dosmildiez.net/marcov/wp-content/uploads/2009/08/63_Miriam_7Dic05.pdf):  
[http://h1cu1.dosmildiez.net/marcov/wp-content/uploads/2009/08/63\\_Miriam\\_7Dic05.pdf](http://h1cu1.dosmildiez.net/marcov/wp-content/uploads/2009/08/63_Miriam_7Dic05.pdf)
- unicorn. (28 de Marzo de 2015). <http://unicorn.bogomips.org/>. Obtenido de  
<http://unicorn.bogomips.org/>: <http://unicorn.bogomips.org/>
- w3c. (28 de Marzo de 2015). *W3C*. Obtenido de  
<http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo>:  
<http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo>

W3C. (28 de Marzo de 2015). *w3schools*. Obtenido de <http://www.w3schools.com/js/>:  
<http://www.w3schools.com/js/>

Weizenbaum, H. C. (28 de Marzo de 2015). *sass-lang*. Obtenido de <http://sass-lang.com/>:  
<http://sass-lang.com/>

## Anexos

### *Tabla de ilustraciones*

Figura 1 - MVC.....	14
Figura 2 – Ruby.....	15
Figura 3 – Ruby on Rails .....	16
Figura 4 - HTML/HAML.....	17
Figura 5 – ERB.....	18
Figura 6 - CSS/SASS.....	18
Figura 7 - JavaScript.....	19
Figura 8 - Coffeescript/JavaScript.....	20
Figura 9 - NGINX.....	21
Figura 10 - Unicorn.....	22
Figura 11- ImageMagick .....	22
Figura 12 – Diagrama de clases .....	25
Figura 13 - Panel de usuarios .....	32
Figura 14 – Formulario de nuevo usuario.....	33
Figura 15 - Editar usuario suspendido .....	34
Figura 16 - Panel de Imágenes .....	35
Figura 17 - Modal de imagen.....	36
Figura 18 - Panel lateral.....	37
Figura 19 - Modal de nuevo mensaje .....	37
Figura 20 - Panel de usuarios .....	39
Figura 21 - Panel de imágenes en prueba .....	40
Figura 22 - Cargado múltiple .....	40
Figura 23 - Servidor subida múltiple.....	41
Figura 24 - Mensajes .....	42

<sup>i</sup> MVC - Modelo Vista Controlador (Model View Controller).

<sup>ii</sup> Ruby on Rails – Framework para programación web que utiliza Ruby como lenguaje principal.

<sup>iii</sup> Ruby – Lenguaje de programación orientado a objetos.

- 
- iv JavaScript – Lenguaje de programación utilizado principalmente en navegadores de internet.
  - v HTML – Hypertext Markup Language.
  - vi CSS – Hojas de estilo en cascada (Cascading Style Sheets).
  - vii CoffeeScript – Lenguaje trans compilado hacia JavaScript.
  - viii ERB – Ruby embedido (Embedded Ruby).
  - ix SASS – Syntacticaly Awesome Style Sheets.
  - x HAML – HTML Abstaction Markup Language.
  - xi Backbone – Lenguaje trans-compilado hacia JavaScript.
  - xii Less – Preprocesador de CSS.
  - xiii DRY – No te repitas (Don't Repeat Yourself).
  - xiv Ruby DSL – Estándar para buenas prácticas de Ruby.
  - xv CRUD – Estándar de programación para aplicaciones web (Create, Read, Update, Delete)
  - xvi Devise – Librería para la autenticación de usuarios.
  - xvii Strong Params – Librería para el filtrado de parámetros en los controladores.
  - xviii SSL – Protocolo de encriptación de datos.
  - xix CSRF – Cross Site Request Forgery.